UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA: INGENIERÍA DE SISTEMAS

Tesis previa a la obtención del título de:
INGENIERO DE SISTEMAS

TEMA:
ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN APLICATIVO WEB EN AMBIENTE DE PRUEBAS QUE GESTIONE LA DOCUMENTACIÓN EMPRESARIAL CON REGISTRO DE FIRMA ELECTRÓNICA EN LA EMPRESA BANRED

AUTORES:
PEDRO LUIS CAICEDO LÓPEZ
DIEGO FERNANDO GODOY CEVALLOS

DIRECTOR:
FRANKLIN HURTADO LARREA

Quito, noviembre 2013

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DELTRABAJO DE GRADO

Nosotros Pedro Luis Caicedo López y Diego Fernando Godoy Cevallos, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de grado y su reproducción sin fines de lucro.

Además declaramos que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.

---	---
Pedro Luis Caicedo López	Diego Fernando Godoy Cevallos
1712213709	0923666507

DEDICATORIA

Gracias a Dios, mis padres y mi esposa por impulsarme y su especial colaboración durante el desarrollo de este trabajo. Diego Godoy

A Teresa, Sofía, pasados y nuevos amigos, gracias por recordarme lo esencial del arte de vivir el presente. Pedro Caicedo

AGRADECIMIENTOS

El feliz término y calidad conceptual-técnica dEl proyecto, se debe en gran parte al apoyo de los siguientes actores:

Pablo Narváez, Luis Alejandro Caicedo, Marcelo Balarezo, Franklin Hurtado, Gustavo Raza, Edwin Montenegro y Luis Fernando Merino.

Muchas gracias por su direccionamiento oportuno y sobre todo por su amistad.

Diego Godoy

Pedro Caicedo

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1 2

PLAN DE PROYECTO 2

Antecedentes 2

Planteamiento del problema 2

Objetivo General 3

Objetivos Específicos 3

Justificación del proyecto 3

Alcance del proyecto 4

a) Reorganización del proceso de gestión documental 5

b) Implementación del proceso de gestión documental en una herramienta de software dedicada 5

c) Evaluación de interoperatividad entre ECM y aplicativo web 6

Áreas fuera del alcance 6

Herramientas tecnológicas a utilizar en el proyecto 7

CAPÍTULO 2 8

MARCO TEÓRICO 8

2.2.1 Ingeniería de software 11

2.2.1.1 Proceso de software 11

2.2.1.2 Modelo del proceso de software 12

2.2.2 Lenguaje Unificado de Modelado UML (Unified Modeling Language) 16

2.2.2.1 Diagramas UML 17

2.2.3 El proceso unificado - UP 18

2.2.3.1 Disciplinas y artefactos 18

2.2.3.2 Fases del UP 19

2.2.3.3 Marco de desarrollo UP 19

Descripción de artefactos del UP 21
Modelo de dominio 21
Modelo de casos de uso 21
Casos de uso de negocio 21
Casos de uso de sistema 22
Visión y análisis del negocio 22
Modelo de diseño 22
Modelo de datos 23
Diagramas de paquetes 23
Modelo de clases 23
Modelo de implementación 23
Plan de pruebas 24
2.2.4 Java Enterprise Edition J2EE 24
2.2.4.1 Tecnologías empresariales JAVA EE 25
2.2.4.2 Enterprise Java Bean (EJB) 28
Tipos de EJB 29
Estructura de un EJB 29
Formas de comunicación con un EJB 30
2.2.4.3 Java Persistence Hibernate 30
Clase Entidad y Persistencia en Hibernate 31
2.3.1 Gestión de contenidos empresariales 33
2.3.1.1 Herramientas ECM 33
2.3.2 Gestión de procesos BPM 36
2.3.2.1 Business Process Management System (BPMS) 37
2.3.2.2 Businnes Process Modeling Notation BPMN 37
2.4.1 Dimensiones de la firma electrónica 39
2.4.2 Antecedentes de la firma 40

2.4.3 Criptografía 41
2.4.3.1 Criptografía simétrica 41
2.4.3.2 Criptografía asimétrica 42
2.4.4 Certificados digitales 43
2.4.5 Función HASH 43
2.4.6 Firma electrónica 44
2.4.7 Infraestructura de clave pública PKI 46
2.4.7.1 Tipos de certificados digitales 47
2.4.7.2 Elementos dentro del PKI 48
CAPÍTULO 3 50
ANÁLISIS Y DISEÑO 50
3.2.1 Reorganización del proceso de gestión documental 56
3.2.1.1 Creación de archivos con formato docx 59
Llenar METADATOS desde convocatoria 59
Registrar METADATOS en docx 60
3.2.1.2 Firmar Digitalmente 60
3.2.2 Procesos implementados en herramienta de gestión documental 61
3.2.3 Evaluación de interoperatividad entre ECM y aplicativo web 64
3.2.3.1 Interfaz para administración de perfiles de usuario para acceso al portal web 64
3.2.3.2 Interfaz para parametrización de cadenas de conexión con sistemas externos 65
3.2.3.3 Interfaz para generación de reportes: 65
3.4.1 Autenticación de usuarios 68
3.4.1.1 Diagrama de caso de uso autenticar usuario 69
3.4.1.2 Descripción del caso de uso autenticar usuario 69
3.4.1.3 Diagrama de actividades del caso de uso autenticar usuario 70
3.4.2 Gestión de roles de usuario portal web 70

3.4.2.1 Diagrama de casos de uso gestión de roles usuario portal web..... 71
3.4.2.2 Descripción del caso de uso gestión de roles de usuario portal web 71
3.4.2.3 Diagrama de actividades del caso de uso gestión de roles de usuario en portal web…. 72
3.4.3 Parametrización portal web 72
3.4.3.1 Diagrama de casos de uso parametrización portal web 73
3.4.3.2 Descripción del caso de uso parametrización portal web 73
3.4.3.3 Diagrama de actividades del caso de uso parametrización portal web…… 74
3.4.4 Gestión de reportes 74
3.4.4.1 Diagrama de casos de uso gestión de reportes 75
3.4.4.2 Descripción del caso de uso gestión de reportes 75
3.4.4.3 Diagrama de actividades del caso de uso gestión de reportes 76
3.5.1 Modelo de diseño del portal web 77
3.5.1.1 Arquitectura del portal web 77
3.5.1.2 Diagrama de paquetes portal web 78
3.5.1.3 Diagramas de clases portal web 78
Paquete ec.fin.banred.dto.properties 79
Paquete ec.fin.banred.dto.reportes 79
Paquete ec.fin.banred.entidades 81
Capa de negocio 82
Paquete ec.fin.banred.servicios y ec.fin.banred.servicios.impl…… 82
Capa de aplicación 83
Paquete ec.fin.banred.web.bean.administrador 83
Paquete ec.fin.banred.web.bean.aplicacion 85
Paquete ec.fin.banred.web.bean.funcionario 86
Utilitarios 87

Paquete ec.fin.banred.web.ldapConnection 87
Paquete ec.fin.banred.web.util 88
3.5.1.4 Modelo lógico portal web 89
3.5.1.5 Modelo físico portal web 89
3.5.2 Modelo de diseño del aplicativo de firma electrónica 90
3.5.2.1 Arquitectura aplicativo de firma electrónica 90
3.5.2.2 Diagrama de paquetes aplicativo firma electrónica 90
3.5.2.3 Diagrama de clases aplicativo firma electrónica 91
Paquete Applet 91
Paquete ec.fin.banred.firmaelectrónica.interfaz 91
Paquete ec.fin.banred.firmaelectrónica.interfaz.componentes ……..91
Paquete Aplicación Java 93
Paquete ec.fin.banred.firmaelectrónica.clases.firma 93
Paquete ec.fin.banred.firmaelectrónica.clases.firma.utilitarios ……..94
Paquete ec.fin.banred.firmaelectrónica.clases.utilitarios 96
Paquete Repositorio documental ECM 97
Paquete ec.fin.banred.firmaelectrónica.clases.Alfresco 97
3.5.3 Modelo de diseño de componente para creación de documentos en formato .docx 98
3.5.3.1 Paquete ec.fin.banred.alfresco.javascript 98
3.5.3.2 Paquete ec.fin.banred.office.word.plantillas 99
3.5.3.3 Paquete ec.fin.banred.office.word.utilitarios 100
CAPÍTULO 4 103
CONSTRUCCIÓN 103
4.1.1 Buenas prácticas para manejo de base de datos 103
4.1.2 Convenciones de código para el lenguaje de programación Java 104

4.1.2.1 Extensiones de los archivos 104
4.1.2.2 Archivos fuente Java 105
Sentencias package e import 105
Declaraciones de clases e interfaces 105
Longitud de la línea 106
Rompiendo líneas 106
Comentarios 107
Declaraciones 108
Inicialización 108
Declaraciones de clases e interfaces 109
Convenciones de nombres 109
4.2.1 Descripción de métodos de portal web 112
4.2.2 Descripción de métodos aplicativo de firma electrónica 116
4.2.3 Componente de creación de archivos con formato .docx 118
4.3.1 Modelo de interfaz portal web 120
4.3.2 Modelo de interfaz aplicativo de firma electrónica 121
4.4.1 Construcción del Portal web 122
4.4.1.1 Arquitectura del Portal web 122
Descripción capa de datos 122
Paquete de Clases Entidad: 122
Paquete de clases DTO 123
Descripción capa de servicio o negocio 125
Descripción capa web 129
Utilitario de la aplicación paquete UTIL 133
Archivos de configuración 133
Librerías utilizadas en la aplicación 134
4.4.1.2 Tecnologías Utilizadas 135

4.4.2 Construcción aplicativo de firma electrónica 135
4.4.2.1 Arquitectura del aplicativo de firma electrónica 135
Paquete de clases Applet 135
Paquete de clases aplicación Java 136
Paquete de clases del repositorio documental ECM 140
4.4.3 Construcción de componente Java para generación de documentos en formato .docx 142
4.4.3.1 Paquete ec.fin.banred.alfresco.javascript 142
4.4.3.2 Paquete ec.fin.banred.office.word.plantillas 143
4.4.3.3 Paquete ec.fin.banred.office.word.utilitarios 145
4.4.4 Configuración de plataforma ECM 146
4.4.4.1 Configuración del repositorio documental 146
Sistema de archivos 146
Base de datos 147
Motor de transformación 147
Correo electrónico 148
Motor de búsqueda 148
Configuración autenticación con Directorio Activo 149
Configuración de autenticación 149
Configuración de sincronización de usuarios 149
Configuración de ejecución de tarea programada de sincronización 150
Cadena de autenticación 151
Auditoría 151
METADATOS personalizados sobre documentos del repositorio documental 153
Definir modelos de contenido de la institución 153
Extender el modelo de contenido de la plataforma ECM 155

Especificar etiquetas 156
Crear archivo de contexto que importe archivos de configuración a la plataforma ECM 157
Modelo de procesos 157
4.4.4.2 Despliegue de nuevos procesos 160
Creación de archivos de definición de procesos 160
Archivo de contexto para carga de archivos de configuración de procesos 163
4.4.4.3 Interoperabilidad con otros sistemas mediante exposición de servicios web 165
Verificar usuario dentro del portal 165
Firma electrónica 167
4.4.4.4 Implementación de componente de creación de documentos .docx 168
4.4.4.5 Implementación de aplicativo de firma electrónica dentro del sistema de gestión documental. 169
Creación de interfaz para presentación de aplicativo de firma. 169
Agregar acción personalizada de firma electrónica 171
4.4.4.6 Formularios del aplicativo 172
Formularios de presentación y edición de METADATOS 172
Formularios de tareas de usuarios dentro de los procesos cargados 173
4.5.1 Resumen de Columnas Tabla USUARIO 175
4.5.1.1 Relación entre tablas 175
4.5.1.2 Resumen de columnnas tabla HISTORIAL 176
4.6.1 Prueba de funcionalidad 176
4.6.1.1 Caso de prueba administración de usuarios 177
4.6.1.2 Caso de prueba auditoría base de datos 178

4.6.1.3 Caso de prueba auditoría ECM 178
4.6.1.4 Caso de prueba parametrización de base de datos..... 179
4.6.1.5 Caso de prueba parametrización de ECM 179
4.6.1.6 Caso de prueba parametrización LDAP 180
4.6.2 Prueba de optimización de código 180
4.6.2.1 Portal web..... 181
Media Complejidad Ciclomática (Average Cyclomatic Complexity)…… 182
Número de campos (number of fields)..... 183
4.6.2.2 Aplicativo de firma electrónica 185
CONCLUSIONES 188
RECOMENDACIONES..... 190
LISTA DE REFERENCIAS 192
GLOSARIO 196
ANEXOS EN FORMATO DIGITAL

ÍNDICE DE FIGURAS

Figura 1. Propuesta de solución de proyecto de tesis 4
Figura 2. Marco de desarrollo UP 19
Figura 3. Relación entre los distintos artefactos y su aplicación en las distintas disciplinas reconocidas en el UP 20
Figura 4. Arquitectura multicapas Java 27
Figura 5. Arquitectura multicapas Java 28
Figura 6. Analogía entre el código Java y base de datos 31
Figura 7. Organigrama Estructural de BANRED 51
Figura 8. Captura de pantalla del formulario para registro de trámites de tipo memorando 52
Figura 9. Captura de pantalla de formulario para registro de trámites de tipo oficio 53
Figura 10. Diagrama de actividades del flujo de trabajo para el registro de METADATA y almacenamiento de documentos de tipo *memorando y oficio* 54
Figura 11. BPD proceso de gestión documental 58
Figura 12. BPD creación de archivos con formato docx 59
Figura 13. BPD registro de METADATOS desde convocatoria 60
Figura 14. BPD registro de METADATOS en docx 60
Figura 15. BPD Firma Electrónica 61
Figura 16. Cuadrante mágico de Gartner para ECM 62
Figura 17. Análisis ECM, Q4 2011 The Forrester Wave 63
Figura 18. Modelo de Dominio 68
Figura 19. Caso de uso autenticar usuario 69
Figura 20. Actividades del caso de uso autenticar usuario 70
Figura 21. Casos de uso gestión de roles usuario portal web 71
Figura 22. Diagrama de actividades del caso de uso gestión de roles de usuario en portal web 72
Figura 23. Diagrama de casos de uso parametrización portal web 73
Figura 24. Diagrama de actividades, caso uso parametrización portal web 74
Figura 25. Diagrama de casos de uso gestión de reportes 75
Figura 26. Diagrama de actividades del caso de uso gestión de reportes 76
Figura 27. Arquitectura del portal web 77
Figura 28. Paquetes portal web 78

Figura 29. Clase DTO *ArchivoProperties* 79
Figura 30. Clase DTO *Auditoria* 79
Figura 31. Clase DTO *HojaRuta* 80
Figura 32. Clase DTO *Involucrados* 80
Figura 33. Clase DTO *TareaPendiente* 80
Figura 34. Clase Entidad *Historial* 81
Figura 35. Clase Entidad *Usuario* 81
Figura 36. *Interface Servicio* 82
Figura 37. Clase *BanredServicioImpl* 83
Figura 38. Clase Bean *AuditoriaBean* 84
Figura 39. Clase Bean *ParametrizacionBean* 84
Figura 40. Clase Bean *AdministracionUsuariosBean* 85
Figura 41. Clase Bean *InicioBean* 85
Figura 42. Clase Bean *InvolucradosBean* 86
Figura 43. Clase Bean *HojaRutaBean* 86
Figura 44. Clase Bean *TareasPendientesBean* 87
Figura 45. Clase Bean *EstadoTramiteBean* 87
Figura 46. Clase POJO *Utilldap* 88
Figura 47. Clase POJO *AplicacionUtil* 88
Figura 48. Modelo lógico base de datos *portalBanred* 89
Figura 49. Modelo físico base de datos *portalBanred* 89
Figura 50. Arquitectura del aplicativo de firma electrónica 90
Figura 51. Paquetes del aplicativo de firma electrónica 90
Figura 52. Applet *AppletFirma* con llamada a clase POJO *ParámetrosAlfresco* 91
Figura 53. Clase *InterfazFirma* con paso de parámetros a clases POJO *ParámetrosAlfresco* y *CertificadoToken* 92
Figura 54. Clase POJO *OpcionesFirma* 93
Figura 55. Clase POJO *FirmaPDF* 93
Figura 56. Clase POJO *ParámetrosAlfresco* 94
Figura 57. Clase POJO CertificadoToken 94
Figura 58. Clase POJO LecturaToken 95
Figura 59. Clase POJO ParámetrosFirma 95
Figura 60. Clase POJO *FiltroFormatosArchivo* 96
Figura 61. Clase POJO *RecuperarDocumento* 96

Figura 62. Clase POJO *UtilitariosFecha* 97
Figura 63. Clase POJO *CargaAlfresco* 97
Figura 64. Clase POJO ModificaDocumentoActaAccion 98
Figura No 65. Clase POJO ModificaDocumentoGenericoAccion. 99
Figura 66. Clase POJO *ModificaDocumentoActa* 99
Figura 67. Clase POJO *ModificaDocumentoGenerico* 100
Figura 68. Clase POJO *FormatearTexto* 100
Figura No 69. Clase POJO *Participante.* 101
Figura 70. Componentes modelo de implementación proyecto tesis 102
Figura 71. Modelo de interfaz portal web 120
Figura 72. Modelo de interfaz aplicativo de firma electrónica 121
Figura 73. Paquetes Controlador patrón MVC 129
Figura 74. Paquetes Vista patrón MVC 130
Figura 75. Paquetes Modelo patrón MVC 133
Figura 76. Resumen de incidencias de resultados de código de portal web previo a la depuración 181
Figura 77. Reporte de elementos del portal web con incidencias de media complejidad ciclomática 182
Figura 78. Reporte de evaluación de elementos del portal web de tipo complejidad ciclomática posterior a depuración de código 183
Figura 79. Reporte de elementos del portal web con incidencias de número de campos 183
Figura 80. Reporte de elementos del portal web con incidencias de número de campos posterior a la optimización de código 184
Figura 81. Resumen de incidencias de resultados de código de portal web posterior a la depuración 185
Figura 82. Resumen de incidencias de resultados de código de aplicativo de firma electrónica previo a la depuración 185
Figura 83. Resumen de incidencias de resultados de código de aplicativo de firma electrónica posterior a la depuración 187

ÍNDICE DE TABLAS

Tabla 1. Cuadro comparativo modelos de abstracción de ingeniería de software 13
Tabla 2. Tecnologías Java EE7 25
Tabla 3. Siglas de área de negocio empresa BANRED 50
Tabla 4. Descripción del caso de uso autenticar usuario 69
Tabla 5. Descripción del caso de uso gestión de roles de usuario portal web 71
Tabla 6. Descripción del caso de uso parametrización portal web 73
Tabla 7. Descripción del caso de uso gestión de reportes 75
Tabla 8. Extensiones de archivos Java 104
Tabla 9. Declaraciones de clases e interfaces 105
Tabla 10. Convenciones de nombres para programación 110
Tabla 11. Descripción métodos portal web 112
Tabla 12. Descripción métodos aplicativo de firma electrónica 116
Tabla 13. Descripción métodos de componente de creación de archivos con formato .docx 118
Tabla 14. Tipos de documentos de la institución 153
Tabla 15. Resumen de tablas de base de datos portalbanred 174
Tabla 16. Resumen de columnas tabla USUARIO 175
Tabla 17. Detalles relación RelaciónBD 175
Tabla 18. Resumen de columnas tabla HISTORIAL 176
Tabla 19. Caso de prueba administración de usuarios 177
Tabla 20. Caso de prueba auditoría base de datos 178
Tabla 21. Caso de prueba auditoría ECM 178
Tabla 22. Caso de prueba parametrización de base de datos 179
Tabla 23. Caso de prueba parametrización de ECM 179
Tabla 24. Caso de prueba parametrización LDAP 180

RESUMEN

El proyecto, tiene por objeto proveer a la empresa BANRED de un nuevo modelo de negocio en su gestión documental y la automatización respectiva en una herramienta de gestión de documentos empresariales ECM (Enterprise Content Management, por sus siglas en inglés) de código abierto, en un ambiente de pruebas controlado.

La solución propuesta presentó los siguientes desafíos:

Evaluar la interoperabilidad de la plataforma ECM con la intranet de BANRED: para este fin se desarrolló un portal web en tecnología Java, que monitorea la integración de las tecnologías a través de la exposición y consumo de servicios web. La construcción del portal web se debe a que BANRED gestiona información bancaria de alto riesgo, y sus políticas de seguridad no permiten el acceso de personas ajenas a la institución, a su red corporativa.

Centralizar el directorio de usuarios de BANRED en una única base de datos empresarial: esto se logró a través de la integración del Directorio Activo de Windows al portal web y a la plataforma ECM. Entre las principales ventajas de integración se encuentra la autenticación de usuarios con las credenciales de sistema operativo Windows y la no duplicación de información en las bases de datos de sus sistemas.

El documento está organizado con base en los contenidos mostrados a continuación:

- Formulación de carta de proyecto de la solución propuesta.
- Análisis y diseño conceptual del nuevo modelo de negocio de gestión documental, y establecimiento de arquitectura y diagramación UML para la construcción del software requerido.
- Configuraciones del ECM para el despliegue del nuevo proceso de gestión documental y del protocolo LDAP para la gestión de usuarios con Directorio Activo. Detalle de la construcción de los aplicativos complementarios al ámbito del ECM.

ABSTRACT

The project has the main objective to provide to BANRED Company a new business model about document management and it automation by using an Enterprise Content Management (ECM) Open Source system into a controlled test environment.

The proposed solution to BANRED showed the following challenges:

Evaluate interoperability between ECM platform and BANRED intranet: to achieve this purpose a web portal was developed using Java technology which allows interoperating this two elements through presentation and use of web services. The main reason of the developing a web portal in this project is because BANRED did not allow the access to their production systems because it manages high-risk bank information

Centralize the user directory of BANRED in a unique business data base: this was achieved through integration between Windows Directorio Activo as authentication component of architecture and web portal and ECM platform as font-end for users. The main advantage of this integration is user authentication with Windows OS credentials to avoid duplicity of security credentials in each developed systems databases.

The document is organized based on the following chapters presented below:

- Definition of the project presentation letter with proposed solution.
- Analysis and design of a new document management conceptual business model, specification of project architecture and UML diagramming for software development to achieve the solution of this Project.
- Configuration for ECM plataform for deploy of content management business processes and Active Directory configuration focused on business users management. Development details of complementary software components to ECM platform.

INTRODUCCIÓN

La empresa ecuatoriana BANRED, especializada en el procesamiento electrónico de transacciones financieras, compensación de cobros y pagos e intercambio de información para las instituciones del sector público y privado (BANRED, 2012), ha decidido modernizar su plataforma de gestión documental, automatizar los ciclos de vida de sus trámites administrativos e integrar las herramientas de software relacionadas a esta actividad.

Su modelo de negocio de gestión documental presenta la siguiente problemática:

- Flujos de trabajo del ciclo de vida documental ejecutados manualmente.
- Procesos de gestión documental no definidos ni estandarizados.
- Repositorio de documentos no centralizado.
- Proceso de firma electrónica no vinculado al ciclo de vida documental.
- Herramientas de software relacionadas a este proceso, no integradas.
- Tareas de usuario, concernientes a la gestión documental, ejecutadas manualmente, sin seguimiento ni auditorías de cumplimiento.

BANRED optó por establecer un escenario de pruebas que evalue herramientas de software de código libre orientadas a mitigar la problemática descrita y a través del análisis de resultados, obtener elementos de juicio técnico previo cualquier posible decisión de inversión.

Con la ejecución dEl proyecto de tesis, se busca mitigar las novedades mencionadas y cumplir las hipótesis siguientes:

- Demostrar que el tiempo de ejecución de las tareas relacionadas a la gestión documental, se reduzcan al menos en un 90%.
- Estimar que la reducción del consumo de papel utilizado para gestionar los documentos administrativos de BANRED automatizados en este proyecto sea de al menos en un 90%.

CAPÍTULO 1
PLAN DE PROYECTO

Antecedentes

La empresa BANRED ha previsto en sus líneas de actuación y planificación estratégica informática, proveer a su personal ejecutivo y operativo, en el mediano plazo, de las facilidades necesarias para atender trámites y/o tareas de forma remota, a través de cualquier dispositivo electrónico, todo, conectado entre sí, a través de la nube[1], sin restricciones de horario ni lugar.

Esta iniciativa partirá con la implementación de los flujos de trabajo del proceso de gestión documental en una herramienta de gestión de documentos empresariales - ECM (Enterprise Content Management, por sus siglas en inglés), la cual será desplegada y evaluada en un ambiente de pruebas, previa cualquier decisión de inversión.

El proyecto tendrá por objetivo ejecutar lo último.

Planteamiento del problema

Actualmente la empresa BANRED no gestiona sus trámites documentales mediante una herramienta dedicada, dando a lugar los siguientes inconvenientes:

- Resguardo vulnerable de la información: dado por la no existencia de un repositorio centralizado ni de estándares para el almacenamiento de la información, la cual podría extraviarse sin posibilidad de recuperación.
- Duplicidad de esfuerzos en funciones del personal: provocado por la insuficiente automatización de tareas recurrentes relacionadas a la gestión documental.
- Acceso a documentos empresariales restringido a su red LAN: no se cuenta con herramientas web para el acceso distribuido fuera de BANRED.

[1] Paradigma que permite ofrecer servicios de computación a través de Internet.

- Consumo elevado de papel: todos los trámites de BANRED se diligencian a través de medios físicos.

Objetivo General

Analizar, diseñar e implementar un aplicativo web en ambiente de pruebas, que sirva de apoyo en la gestión de información de la Empresa BANRED, ligada a los procesos de: gestión de documentos empresariales, administración de su plataforma y generación de reportes.

Objetivos Específicos

- Levantar y documentar los procesos relacionados a la gestión documental empresarial.
- Documentar los modelos (diagramas) establecidos dentro de la metodología de desarrollo de software UP.
- Generar una estructura de datos sólida, coherente y actualizada en MYSQL.
- Programar en plataforma Java, las interfaces de usuario de fácil navegabilidad y usabilidad.
- Probar la funcionalidad y navegabilidad de la aplicación desarrollada y su integración con la plataforma ECM establecida.
- Valuar que el tiempo de ejecución de las tareas relacionadas al objeto de este proyecto, se reduzcan al menos en un 90%.
- Estimar que la reducción del consumo de papel utilizado para gestionar los documentos administrativos de BANRED automatizados en este proyecto sea de al menos en un 90%.

Justificación del proyecto

La disponibilidad de la información empresarial, sin restricciones del lugar geográfico ni el horario, así como el registro, control, circulación y organización de sus documentos empresariales, incrementa la efectividad de una organización y la colaboración de sus equipos de trabajo.

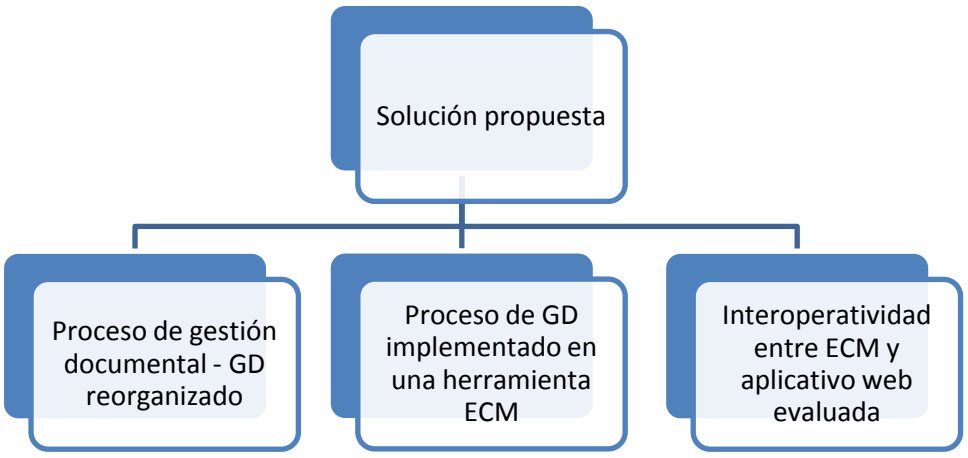
La implementación de aplicativos informáticos destinados a estos fines, presentan las siguientes ventajas:

- Optimización de espacio físico y disminución de costos operativos destinados al almacenamiento de documentos.
- Acceso a los documentos empresariales de forma segura sin restricción del lugar geográfico.
- Búsqueda de documentos por METADATA y contenido a través de varios criterios de selección.
- Eliminación del riesgo en pérdidas de documentos e información a causa de robos, desastres naturales, incendios, negligencia del personal, entre otros.
- Preservación de los documentos a largo plazo.
- Apoyo a la conservación del medio ambiente a través de la disminución de la impresión de documentos.

Alcance del proyecto

Los productos incluidos en el proyecto se organizan de la siguiente forma:

Figura 1. Propuesta de solución de proyecto de tesis



Elaborado por: Pedro Caicedo y Diego Godoy

a) Reorganización del proceso de gestión documental

Se realizará una reingeniería del proceso de gestión documental de BANRED, iniciando por la identificación y modelado del ciclo de vida de sus documentos administrativos tipo: *memorandos, oficios y actas de reuniones, informes Internos* y *convocatorias.*

El ciclo de vida de los documentos, poseerá las siguientes fases genéricas: creación, revisión, aprobación y registro de firma electrónica; dichas fases se personalizarán a cada tipo documental una vez concluido el levantamiento del referido proceso.

En referencia a la fase de registro de firma electrónica, se desarrollará un componente de software; en tecnología Java, que ejecute está tarea y se integre al ciclo de vida propuesto.

También se establecerán roles de involucrados y su participación en el ciclo de vida documental.

b) Implementación del proceso de gestión documental en una herramienta de software dedicada

Se analizará las herramientas de Gestión de Contenido Empresarial ECM de código abierto (open source) existentes en el mercado, y se seleccionará la de mejor desempeño y calidad.

Sobre la herramienta optada se realizará la implementación del nuevo proceso de gestión documental indicado en el punto 1.5.1, logrando así el objetivo propuesto de contar con una correcta administración, almacenamiento, indexación, colaboración y búsqueda de documentos empresariales.

Restricciones de este módulo:

- Este módulo no genera ni digitaliza documentos, solo los almacena en el repositorio de datos destinado en la herramienta ECM.
- El módulo no permitirá la generación de nuevos METADATOS a más de los establecidos por cuanto se deben modificar los archivos de configuración y para que los cambios surtan efecto, se requiere reiniciar el servicio del aplicativo.
- El paso a un servidor de producción de BANRED dependerá de dicha Institución (metodología, pruebas de calidad, migración de la data, etc.).

c) Evaluación de interoperatividad entre ECM y aplicativo web

Se desarrollará un portal web con tecnología Java, para simular la integración de la herramienta ECM a la intranet de BANRED, cuya funcionalidad se centrará en monitorear la interoperabilidad de dichas aplicaciones mediante la exposición y consumo de servicios web.

El portal web consumirá dichos servicios para presentar reportes de gestión de la herramienta ECM.

La gestión de usuarios se la realizará a través del Directorio Activo de Windows, por tanto la autenticación del portal como de la herramienta ECM se la realizará con las credenciales de sistema operativo Windows.

Áreas fuera del alcance

Los servicios o actividades enumeradas a continuación están considerados fuera de la cobertura de los esfuerzos de desarrollo y parametrización incluidos en este proyecto:

- Configuración o pruebas de redes de comunicación y PC's clientes.
- Adiestramiento a personal interno de BANRED en el uso de las herramientas de software utilizadas, tanto en su parametrización como en su desarrollo.

- No es responsabilidad de los tesistas los bugs[2] de producto que se puedan presentar en la versión de la herramienta ECM escogida.

Herramientas tecnológicas a utilizar en el proyecto

Las herramientas a utilizar en el proyecto son las siguientes:

- Sistema operativo Centos OS versión 6.2 (64 bits) servidor ECM y portal web.
- Sistema operativo Windows Server 2003 R2 para servidor de Directorio Activo.
- Servidor de aplicaciones Tomcat v. 7.30 servidor ECM y portal web.
- Navegadores web: Firefox 3.6 o superior; Internet Explorer v. 7.0 u 8.0, Chrome v. 16 o superior.
- Base de datos. MySQL v. 5.5.
- Sistema de Gestión de Contenidos empresariales: ALFRESCO COMMUNITY [3] v. 4.2.
- Máquina Virtual de Java: Java Development Kit v. 1.7.
- Ofimática: Microsoft Office v. 2010.
- Motor BPM: Activiti v. 5.13.
- Transformación de imágenes: ImageMagick v. 6.5.
- Transformación de texto: LibreOffice v. 3.5.
- Vista previa de documentos web en línea: SWFTools v. 0.9.1.
- Java Server Faces v. 2.0.
- Hibernate v. 3.5.
- Motor de base de datos MYSQL v.5.5.
- Servicios web REST
- framework RichFaces 4.2.X
- IDE Eclipse v. Indigo Release
- AJAX

[2] Defecto de software, es el resultado de un fallo o deficiencia durante el proceso de creación de programas de ordenador o computadora.
[3] Sistema de administración de contenidos libre, basado en estándares abiertos y de escala empresarial para sistemas operativos tipo Unix y otros, con licencia LGPL de código abierto.

CAPÍTULO 2
MARCO TEÓRICO

2.1 Legislación Ecuatoriana de Comercio Electrónico y sus Reglamentos

La Ley de Comercio Electrónico, Firmas Digitales y Mensaje de Datos, fue publicada en el "Registro Oficial Suplemento 557", bajo la norma "Ley 67" con fecha de vigencia desde el 17 de abril del 2002, en dicha ley, a manera de síntesis se especifica el tratamiento, reconocimiento jurídico, conservación y protección que tendrán los mensajes de datos electrónicos con relación a los documentos físicos.

La Ley de Comercio Electrónico, Firmas Digitales y Mensaje de Datos se encuentra dividida en cinco secciones, cada una de las cuales a su vez contiene varios capítulos y artículos de ley, dichas secciones globales son resumidas a continuación:

- Objetivo de la Ley de Comercio Electrónico, Firmas Digitales y Mensaje de Datos, Reconocimiento y validez jurídica, confidencialidad, conservación y protección de mensajes de datos electrónicos.
- Especificaciones, aplicación y requisitos sobre firmas electrónicas, certificados de firmas electrónicas, entidades de certificación de información, organismos de promoción de los servicios electrónicos. regulación y control de las entidades de certificación acreditadas.
- Cumplimiento, validez y jurisdicción de los servicios electrónicos, la contratación electrónica y telemática, los derechos de los usuarios, e instrumentos públicos
- Medios, práctica y valoración de las pruebas y los procedimientos sobre las notificaciones electrónicas.
- Definición de infracciones informáticas.

En esta Ley se presentan los principios jurídicos que se aplican sobre las transmisiones de mensajes de datos electrónicos. Donde se concede la misma validez y eficacia jurídica a los mensajes de datos electrónicos, sobre el contenido de su información. La interpretación de la Ley se rige por la legislación ecuatoriana y por los tratados y convenios internacionales incorporados al cuerpo legal ecuatoriano.

Dentro de esta Ley se considera de igual valor tanto el documento físico como el documento electrónico en caso de requerir la presentación de un documento, siempre y cuando exista garantía de su conservación, procedencia e inalterabilidad de su información.

Como punto fundamental dentro de esta Ley, se define la protección de la información creada u obtenida por transmisión electrónica de mensaje de datos, concediendo al titular de dichos datos el poder para autorizar la disposición de su información, sea que dichos datos fueron obtenidos como usuario de un servicio o sea que fueron obtenidos en el intercambio de mensajes de datos.

El documento electrónico será considerado como evidencia con todos sus efectos legales, para lo cual, éste documento electrónico deberá cumplir los principios de integridad[4] e identidad[5], con el fin justificar la voluntad contractual de obligarse por dicho documento. Si la contraparte solicita que se niegue validez sobre un documento electrónico, se deberá comprobar que este no cumple con los requisitos técnicos mencionados anteriormente.

Adicionalmente se establecen varios aspectos para la correcta aplicación como evidencia en estos casos, entre ellos se destacan los siguientes aspectos:

- Presentación de los soportes de verificación necesarios del documento electrónico y los mecanismos para la lectura y verificación de la firma electrónica.
- Dicha firma electrónica deberá ser generada a partir de un certificado validado por un proveedor de servicios de certificación.
- Los mensajes de datos electrónicos deberán presentar la integridad de su contenido. Estas evidencias serán valoradas de acuerdo con la seguridad y fiabilidad con la cual se la verificó, envió, archivó y recibió. Para una mejor

[4] Mantener con exactitud la información tal cual fue generada, sin ser manipulada o alterada por personas o procesos no autorizados.
[5] Estar seguro que un documento es de ese alguien, él que lo ha mandado, y no de una tercera persona haciéndose pasar por el remitente (suplantación de identidad).

apreciación de la evidencia, el juez contará con el asesoramiento de un perito en la materia, en este caso un perito informático.

El organismo facultado para autorizar a las entidades de certificación de información es la Secretaria Nacional de Telecomunicaciones (SENATEL), anteriormente conocida como el Consejo Nacional de Telecomunicaciones, según lo dicta el artículo 37 dispuesto dentro de esta Ley, mediante los Decretos Ejecutivos 3496 (31 de julio del 2002) y 1356 (29 de Septiembre del 2008) en los que se establecen el modelo de Resolución para la Acreditación como Entidad de Certificación e Información y Servicios Relacionados, tal como lo establece el Art. 29 del Capítulo III de esta Ley.

Las funciones y responsabilidades otorgadas por el organismo de autorización del país sobre las entidades que prestan el servicio de certificación de información y otros servicios relacionados, es facultar la generación, gestión, administración, custodia y protección de las claves y los certificados de firma electrónica, así como la validación de la identidad e información de los usuarios o solicitantes de firmas electrónicas, mediante el uso de infraestructura y recurso humano capacitado para operar dicha infraestructura con absoluta pericia y confidencialidad. Uno de los organismos que obtuvo la autorización del Consejo Nacional de Telecomunicaciones como entidad de certificación es el Banco Central del Ecuador para emitir certificados a personas naturales, jurídicas y funcionarios públicos. (Ley de Comercio Electrónico, Firmas Electrónicas y Mensajes de Datos, 2002).

2.2 Ingeniería de software

En esta sección del documento se tratan los temas relacionados a la ingeniería de software, el modelo de desarrollo, el lenguaje de modelado y la tecnología empleada en el proyecto.

2.2.1 Ingeniería de software

La ingeniería de software es una disciplina de ingeniería que ofrece métodos y técnicas para desarrollar y entregar software de calidad, utilizando tecnologías y procedimientos de gestión de proyectos.

La ingeniería de software es aplicable en negocios, investigación científica, medicina, producción, banca, derecho, internet, entre otras disciplinas. (ECURED, 2013).

2.2.1.1 Proceso de software

Un proceso del software es un conjunto de actividades que conducen a la creación de un producto software. Dichas actividades dependen de las decisiones y juicios del personal directivo de una organización.

Pese a existir una gran diversidad de procesos de desarrollo de software no es posible listar todas las actividades del mismo, sin embargo las más frecuentes que pueden destacar son:

- Especificación del software, reúne una lista de requerimientos por parte del cliente sobre la funcionalidad que desea del software.
- Diseño del software, en base a los requerimientos se analiza y diseña la estructura del sistema.
- Implementación del software, en base a lo diseñado se procede a desarrollar el software solicitado.
- Validación del software, verificar con el cliente el funcionamiento esperado del software y realizar las correcciones necesarias
- Evolución del software, debido a la constante actualización de las tecnologías no es posible tener un software estático, este debe buscar siempre un enfoque de mejoras y evolución en base a las tecnologías actuales del mercado.

Aunque no existe un proceso del software “ideal”, la aplicación de estas prácticas ayudarán a mejorar el proceso desarrollo de software dentro de las organizaciones (Pressman, 1993).

2.2.1.2 Modelo del proceso de software

Un modelo del proceso del software es una simplificación o representación abstracta de un proceso real. (UNAM, 2012)

Los modelos, paradigmas y filosofías de desarrollo, en los cuales se apoya la ingeniería de software para su construcción son:

- Modelo en cascada o clásico (modelo tradicional)
- Modelo de prototipos
- Modelo en espiral
- Desarrollo por etapas
- Desarrollo iterativo y creciente o iterativo e incremental
- RAD (Rapid Application Development)
- Desarrollo concurrente
- Proceso unificado
- RUP (Proceso Unificado de Rational)
- Metodología XP (extreming programming)

(Ecotec, 2008).

A continuación se presentará el cuadro comparativo de cinco modelos de abstracción los cuales ejemplifican una abstracción inicial, una reciente y otra actual.

Tabla 1. Cuadro comparativo modelos de abstracción de ingeniería de software

	MODELO EN CASADA	**MODELO ESPIRAL**	**MODELO INCREMENTAL**	**MODELO DRA (desarrollo rápido de aplicaciones)**	**Metodología XP (extreming programming)**
QUE ES	Es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.	Consiste en una serie de ciclos que se repiten en forma de espiral, comenzando desde el centro. El Espiral puede verse como un modelo evolutivo que conjuga la naturaleza iterativa del modelo MCP con los aspectos controlados y sistemáticos del modelo cascada.	El incremental es un modelo de tipo evolutivo que está basado en varios ciclos Cascada realimentados aplicados repetidamente, con una filosofía iterativa	Es un modelo de proceso de desarrollo de software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto.	Es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación, y realimentación del código desarrollado. Fue desarrollado por Kent Beck.
FASES DEL MODELO	**1. Análisis de requerimientos:** Contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos. **2. Diseño del Sistema** Contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras. **3. Diseño del Programa** Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario. **4. Codificación**	1. Establecer la comunicación entre el cliente y el desarrollador. 2. Definición de los recursos, tiempo y otra información relacionada con el proyecto. 3. Evaluar los riesgos técnicos y de gestión del proyecto. 4. Construir una o más *representaciones* de la aplicación software. 5. Construir la aplicación, instalarla, probarla y proporcionar soporte al usuario o cliente 6. Obtener la reacción del cliente, según la evaluación de lo creado e instalado en los ciclos anteriores.	Dentro de modelo incremental podemos encontrar el modelo **DRA.**	**1. Modelado de gestión:** Flujo de información entre las funciones de gestión responde las siguientes preguntas: ¿qué información conduce al proceso de gestión?, ¿A dónde va la información?, ¿Quién la procesa? **2. Modelado de datos:** flujo de información definido como parte de la fase del modelado de gestión se refina como un conjunto de objetos y datos necesarios para apoyar la empresa. **3. Modelado de procesos:** los objetos de datos definidos en la fase de modelado quedan transformados para lograr el fin deseado.	**1. Planificación del proyecto.** Definir las historias de usuario con el cliente, las historias de usuario tienen la misma finalidad que los casos de uso, pero con algunas diferencias, constan de 3 o 4 líneas escritas por el cliente en un lenguaje no técnico, sin profundizar mucho en los detalles **2. Diseño** **3. Codificación.** Debe hacerse atendiendo estándares de codificación ya creados, para facilitar su comprensión y escalabilidad. **4. Pruebas**. Test de comprobación de código.

	MODELO EN CASADA	MODELO ESPIRAL	MODELO INCREMENTAL	MODELO DRA (desarrollo rápido de aplicaciones)	Metodología XP (extreming programming)
	5. Pruebas **6. Implantación**				
VENTAJAS	· Se tiene todo bien organizado y no se mezclan las fases. · Es perfecto para proyectos que son rígidos, y además donde se especifiquen muy bien los requerimientos y se conozca muy bien la herramienta a utilizar.	· Reduce riesgos del proyecto. · Incorpora objetivos de calidad. · Integra el desarrollo con el mantenimiento, etc. · Además es posible tener en cuenta mejoras y nuevos requerimientos sin romper con la metodología, ya que este ciclo de vida no es rígido ni estático.	· Se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial. · Proporciona todas las ventajas del modelo en cascada realimentado, reduciendo sus desventajas sólo al ámbito de cada incremento. · Más rápido en comparación del modelo de cascada. · Resulta más sencillo acomodar cambios al acotar el tamaño de los incrementos.	· Permiten que los ingenieros de SW desarrollen versiones cada vez más completas del SW. · Producen una versión completa en forma incremental con cada iteración.	· Programación organizada. · Menor taza de errores. · Satisfacción del programador.
DESVENTAJAS	· Un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso. · El proceso de creación del software tarda mucho tiempo ya que debe pasar por el proceso de prueba y hasta que el software no esté completo no se opera.	· Genera mucho tiempo en el desarrollo del sistema. · Modelo costoso. · Requiere experiencia en la identificación de riesgos. .	· El modelo Incremental no es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido, y/o de alto índice de riesgos. · Requiere de mucha planeación, tanto administrativa como técnica. · Requiere de metas claras para conocer el estado del proyecto.	· Para proyectos grandes, necesita suficientes recursos humanos para crear el número correcto de equipos DRA. · Si los desarrolladores y clientes no se comprometen con las actividades rápidas necesarias para completar un sistema en un marco de tiempo muy breve, los proyectos fallarán. · Si un sistema no se	· Es recomendable emplearlo solo en proyectos a corto plazo. · Altas comisiones en caso de fallar.

	MODELO EN CASADA	MODELO ESPIRAL	MODELO INCREMENTAL	MODELO DRA (desarrollo rápido de aplicaciones)	Metodología XP (extreming programming)
				puede modular en forma apropiada, la construcción de los componentes necesarios será problemática. ▪ Inapropiado cuando los riesgos técnicos son altos… cuando se aplican muchas nuevas tecnologías.	
USOS	Conviene el uso de este modelo cuando los proyectos poseen requisitos claros y cuando se necesita un rápido desarrollo.	El modelo en espiral es beneficioso en proyectos que necesitan reducción de riesgos.	El modelo incremental es útil sobre todo cuando el personal necesario para una implementación completa no está disponible.	El modelo DRA es utilizado para ciclos de vida del software cortos.	Es utilizado para la creación y desarrollo práctico de software, es utilizado mucho últimamente ya que es una metodología ágil para el desarrollo.

Fuente: (WINTEXTIL, 2012).

2.2.2 Lenguaje Unificado de Modelado UML (Unified Modeling Language)

UML es el lenguaje más conocido y utilizado en la actualidad para modelar la construcción de software; el cual está respaldado por el consorcio internacional Object Managment Group (OMG). Este lenguaje está orientado a presentar notación gráfica expresiva que permite representar de manera detallada y ordenada todas las fases de un proyecto informático. Las fases que descritas en el UML parten desde el análisis con la ayuda de diagramas de casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la representación de la implementación y configuración del software con los diagramas de despliegue.

UML como método formal de modelado presenta las siguientes características:

- Verificar y validar del modelado realizado en cada una de las fases de la construcción del software.
- Especificación de la construcción del software con mayor rigor.
- Posibilidad de automatizar determinados procesos de la construcción y la posibilidad de la generación de código a partir de los modelos especificados y viceversa, es decir, generar los modelos del software a partir del código fuente del sistema.
- Presenta un vocabulario y reglas para permitir una correcta comunicación, mediante el enfoque sobre la representación gráfica de un sistema.

Con respecto a los objetivos de UML, como se comentó anteriormente representa toda la construcción del software, por lo tanto presenta diversos objetivos, sin embargo, se pueden sintetizar estos objetivos en base a sus funciones:

- Representar: Expresa mediante una notación gráfica el software a construir, de forma que otra persona ajena al proyecto pueda entenderlo.
- Definir: Define las características del software antes de su construcción.
- Construir: mediante los modelos especificados se construye el software.
- Documentar: Los elementos gráficos generados durante la construcción sirven como documentación del software desarrollado.

Los componentes que integran a un modelo UML son:

- Elementos: representan abstracciones de elementos reales o ficticios.
- Relaciones: vinculan los elementos de la construcción entre ellos.
- Diagramas: conjunto de colecciones de los elementos con sus respectivas relaciones.

2.2.2.1 Diagramas UML

Como se especificó anteriormente, un diagrama es la representación gráfica de un conjunto de elementos con sus respectivas relaciones, por lo tanto, el diagrama brinda visión de modelo del software y así representar claramente los componentes, actores y características de dicho software a desarrollar. (Hernandez, 2012).

El lenguaje UML modela el software mediante los siguientes diagramas:

- Casos de uso, diagrama que representa el comportamiento del software y define una notación gráfica para representarlo.
- Clases, Muestra las diferentes relaciones entre las clases que componen el sistema, diferenciándolas a su vez en clases asociativas, clases de herencia, clases de uso y clases de comportamiento.
- Objetos, destacan la relación existente entre las instancias de las clases en un espacio de tiempo. Reflejan a su vez esencia, multiplicidad y roles.
- Secuencia, Modela la interacción entre los objetos del software.
- Colaboración, representa las interacciones organizadas alrededor de los roles de los actores del software y muestran explícitamente las relaciones de dichos roles, siendo, la principal diferencia con los diagramas de secuencia.
- Estados, especifican el conjunto de estados por los cuales pasa un objeto durante su ciclo de vida dentro del software en respuesta los diferentes eventos de dicho software.
- Actividades, Muestra los flujos de trabajo del negocio y operacionales de los componentes.

- Componentes, representan las piezas del software, controladores, etc., los cuales conforman el software. Poseen varios niveles de abstracción dependiendo de la característica individual.
- Despliegue, se utiliza para modelar el hardware utilizado en la implementación del software y relacionarlos con sus componentes

2.2.3 El proceso unificado - UP

El UP, (Unified Process, por sus siglas en inglés), es un proceso de desarrollo de software iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura.

La perspectiva iterativa e incremental de un proyecto se organiza en una serie de mini proyectos de corta duración llamada iteración, que establece unos requerimientos iniciales, los diseña, implementa y prueba. El resultado de cada iteración es un incremento al producto final.

El enfoque de caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante y define la línea base para la arquitectura del sistema. (slideshare, 2012).

La arquitectura es una vista de diseño completo del software que resalta las características más importantes del software, dejando de lado los detalles (Jacobson, Booch y Rumbaugh, 2000).

UP establece los siguientes componentes:

2.2.3.1 Disciplinas y artefactos

El UP se organiza en disciplinas o flujos de trabajo, que son un conjunto de actividades que se ejecutan en un área específica.

Estas actividades producen artefactos, que corresponde a un resultado del trabajo, es decir un entregable de cualquier tipo que sea verificable (físico).

2.2.3.2 Fases del UP

Inicio: Análisis del quehacer de la empresa cliente (visión y análisis del negocio), estimación aproximada del alcance del proyecto (costo, tiempo y calidad). Aquí se define la viabilidad del proyecto.

Elaboración: Implementación iterativa de las funcionalidades gruesas de la aplicación, resolución de los riesgos más altos (gestión de riesgos), identificación de nuevos requerimientos y nuevos alcances más ajustados.

Construcción: Implementación iterativa de los requisitos restantes de menor riesgo y elementos más sencillos, preparación para el despliegue (entrega, instalación y configuración).

Transición: pruebas beta, despliegue (IIE Instituto de Ingeniería Eléctrica, 2009).

2.2.3.3 Marco de desarrollo UP

El diagrama siguiente especifica el estado de desarrollo de los artefactos especificados en la metodología UP y su relación con sus disciplinas y fases:

Figura 2. Marco de desarrollo UP

Componentes del UP		Fases del UP			
Disciplina	Artefacto Iteraciones:	Inicio I1	Elaboración E1...En	Construcción C1...Cn	Transición T1...Tn
Modelado del negocio	Modelo del dominio		c		
Requerimientos	Modelo de Casos de Uso Visión y Análisis del Negocio Especificación Complementaria Glosario	c c c c	r r r r		
Diseño	Modelo de Diseño Documento de Arquitectura Modelo de Datos		c c c	r r	
Implementación	Modelo de implementación		c	r	r
Gestión del proyecto	Plan de desarrollo	c	r	r	r
Pruebas	Modelo de Pruebas		c	r	
Entorno	Marco de desarrollo	c	r		

Fuente: (IIE Instituto de Ingeniería Eléctrica, 2009)

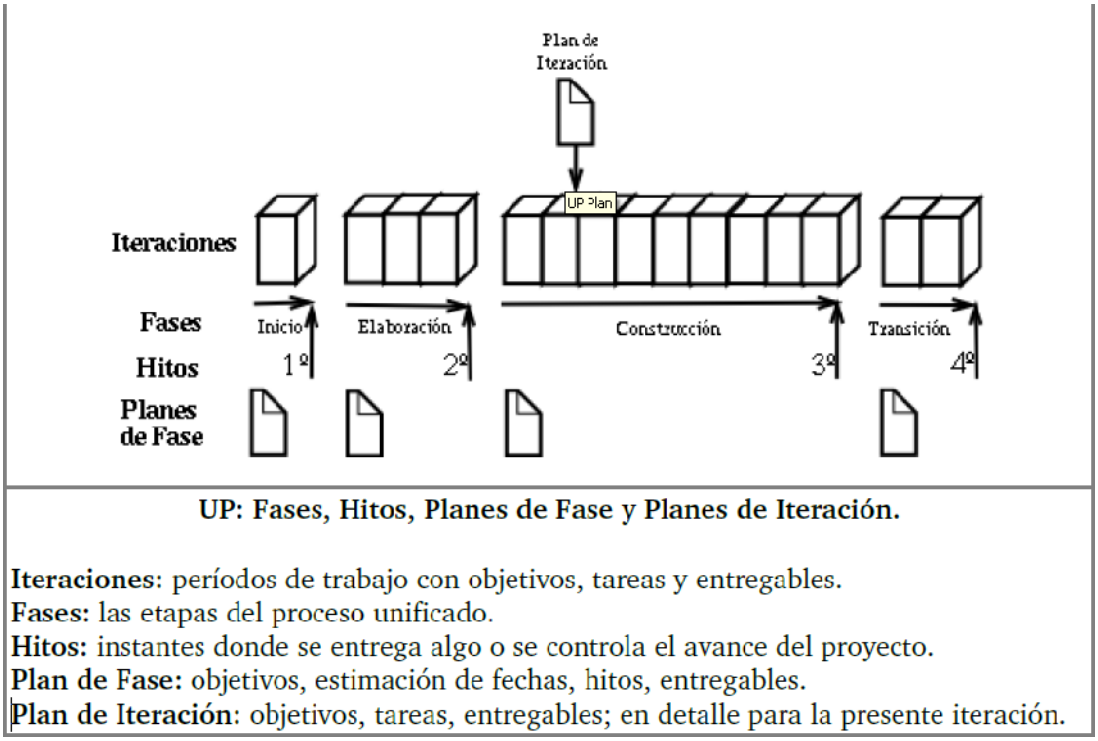
Donde: c, es el comienzo de la construcción del artefacto, y r es el refinamiento (ampliación o mejora) de dicho artefacto.

Cada una de las fases del UP cumple una o más iteraciones. Dado que el desarrollo es iterativo e incremental; en cada iteración se adiciona algo más, y el producto software está en continuo crecimiento hasta su entrega.

En cada iteración hay análisis de requerimientos, diseño, implementación y verificación, así como puesta a punto y coordinación de todos los artefactos.

El diagrama siguiente muestra la relación entre los distintos artefactos y su aplicación en las distintas disciplinas reconocidas en el UP. Las flechas indican algunos aportes significativos de unos artefactos hacia otros.

Figura 3. Relación entre los distintos artefactos y su aplicación en las distintas disciplinas reconocidas en el UP



UP: Fases, Hitos, Planes de Fase y Planes de Iteración.

Iteraciones: períodos de trabajo con objetivos, tareas y entregables.
Fases: las etapas del proceso unificado.
Hitos: instantes donde se entrega algo o se controla el avance del proyecto.
Plan de Fase: objetivos, estimación de fechas, hitos, entregables.
Plan de Iteración: objetivos, tareas, entregables; en detalle para la presente iteración.

Fuente: (IIE Instituto de Ingeniería Eléctrica, 2009).

- **Descripción de artefactos del UP**

A continuación se detallan los artefactos o entregables del software, definidos por la metodolotía UP.

- **Modelo de dominio**

El modelo de dominio o modelo conceptual de alto nivel, muestra todos los objetos físicos o abstractos de un área de interés y sus relaciones. Se emplea para documentar clases conceptuales, cuyo foco de atención es el concepto de un grupo de cosas en lugar de clases que definen un objeto de programación.

Un modelo de dominio muestra las unidades físicas, la organización de dominio, relaciones entre estas unidades y la multiplicidad de las relaciones. (DICAMPUS, 2012).

- **Modelo de casos de uso**

Los casos de uso pueden ser de negocio y/o de sistema, a continuación se detalla el ámbito de cada uno:

- **Casos de uso de negocio**

Describe los procesos de negocio[6] de una compañía vinculados a su campo de acción, así como la manera en que se benefician e interactúan los socios y clientes en estos procesos.

Para identificar los objetivos de negocio se debe partir de los objetivos estratégicos, analizar los sub objetivos de estos y bajar de nivel hacia los procesos de negocio.

[6] Conjunto de tareas relacionadas que se ejecutan en secuencia y que emplean recursos de una organización para apoyar sus objetivos.

Toda interacción con el ambiente de negocio se modela con actores, los cuales poseen un rol específico.

- **Casos de uso de sistema**

Describe la funcionalidad del sistema de forma independiente a la implementación, su principal objetivo es descubrir lo que se quiere construir.

El actor representa a un usuario u otro programa que interactúa con el caso de uso, el caso de uso es una funcionalidad de negocio o sistema y el límite del sistema separa a los actores de los casos de uso y delimita un escenario para los casos de uso de negocio o sistema para los casos de uso de sistema.

- **Visión y análisis del negocio**

Se realiza a partir del modelo de procesos de negocio actual de una compañía, donde se muestra el comportamiento y los flujos de información dentro de una asociación o un sistema.

En éste se captura los eventos más relevantes como: entradas, recursos, procesos y salidas asociados con el proceso de negocio.

El modelo de procesos de negocio se clasifica en los siguientes componentes:

- Contexto de negocio
- Objetos de negocio
- Flujos de trabajo de negocio

- **Modelo de diseño**

El modelo de diseño consta de los siguientes elementos:

- **Modelo de datos**

Conocido también como modelo de base de datos, define la estructura de datos para su almacenamiento y recuperación, que se usa en el sistema bajo desarrollo. Generalmente constituyen modelos de base de datos relacionales los cuales definen las tablas y datos en detalle, permitiendo la generación de scripts y/o rutinas de código para crear, instalar y/o configurar bases de datos.

- **Diagramas de paquetes**

Se usan para reflejar la organización de los paquetes y sus elementos, y para proveer una visualización de sus correspondientes nombres de espacio.

- **Modelo de clases**

El diagrama de clases captura la estructura lógica del sistema (las clases y cosas que constituyen el modelo). Es un modelo estático, describiendo lo que existe y qué atributos y comportamiento tiene, más que cómo se hace algo. Los diagramas de Clases son los más útiles para ilustrar las relaciones entre las clases e interfaces. Las generalizaciones, las agregaciones y las asociaciones son todas valiosas para reflejar la herencia, la composición o el uso y las conexiones respectivamente.

- **Modelo de implementación**

Precisa cómo las clases, artefactos y otros elementos de bajo nivel, se agrupan en componentes de alto nivel así como las interfaces y conexiones entre estos.

A continuación sus elementos:

- Los componentes, son elementos de software compilados que proveen el comportamiento requerido considerando las restricciones definidas en el modelo de requisitos. Incluyen interfaces expuestas adicionales, puertos y otras entradas o componentes estructurales internos.

- Las estructuras internas, muestran como los componentes se conectan internamente para proveer servicios internos y externos, así como sus dependencias, usualmente entre clases (objetos o partes) e interfaces soportadas (servicios).
- Las conexiones ilustran mediante diagramas detallados las dependencias y conectividad entre varios componentes en tiempo de ejecución y como el sistema, como un todo, es capaz de realizar el trabajo requerido. Los componentes suelen exponer interfaces y API's, los cuales se usan por otros componentes (Schmuller, 2000).

- **Plan de pruebas**

Es un documento donde se explica los propósitos y enfoques del proceso de prueba de un sistema, como son: el plan de trabajo, los procedimientos operacionales, las herramientas requeridas y ámbito de responsabilidades.

Seguidamente se muestra un marco de referencia del contenido de un plan de pruebas de proyectos para PYMES[7]:

- Establecimiento del plan de pruebas y del software a evaluar.
- Elementos a probar: módulos, clases y casos de uso a probar.
- Enfoque: estrategia de prueba.
- Calendario: tiempos e hitos en el proceso (Sumano, Fernández y Cortés, 2011).

2.2.4 Java Enterprise Edition J2EE

La plataforma Java 2, Enterprise Edition (J2EE) define el estándar para el desarrollo de aplicaciones empresariales multinivel. Esta plataforma J2EE simplifica las aplicaciones empresariales basándolas en componentes estandarizados, modulares, con el fin de proporcionar un conjunto completo de servicios a esos componentes, y manejando un conjunto de detalles del comportamiento de la aplicación automáticamente, sin necesidad de agregar complejidad al código de programación.

[7] Acrónimo de pequeña y mediana empresa.

Las aplicaciones empresariales desarrolladas con esta plataforma establecen las reglas de negocio del sistema a desarrollar, además de brindar conectividad a los distintos tipos de clientes ofreciendo una solución integral a las necesidades de sistemas de información a la medida.

La plataforma J2EE es un conjunto de APIs enfocadas a brindar servicios empresariales que toda aplicación necesita como: transaccionalidad (JTA – Java Transaction API), seguridad, interoperabilidad, persistencia de objetos (Hibernate - Java Persistance API), manejo de objetos distribuidos (RMI – Remote Method Invocation), entre otros.
Los principales beneficios que presenta la plataforma J2EE son los siguientes:

- Permite a los desarrolladores la creación de aplicaciones empresariales portátiles entre plataformas.
- Las aplicaciones empresariales desarrolladas con esta plataforma gozan una fácil integración con tecnologías anteriores.
- Los desarrolladores poseen un mayor enfoque sobre la lógica del negocio.
- Define una arquitectura separada de tres capas:
  - Presentación.
  - Lógica del negocio
  - Datos.

2.2.4.1 Tecnologías empresariales JAVA EE

En el diagrama siguiente se muestra las tecnologías Java para aplicaciones tipo escritorio y web:

Tabla 2. Tecnologías Java EE7

Technologies
Java EE Platform
Java Platform, Enterprise Edition 7 (Java EE 7)JSR 342Download spec
Web Application Technologies
Expression Language 3.0
Java API for JSON Processing

Technologies
Java API for webSocket
Java Servlet 3.1
JavaServer Faces 2.2
JavaServer Pages 2.3
Standard Tag Library for JavaServer Pages (JSTL) 1.2
Enterprise Application Technologies
Batch Applications for the Java Platform
Bean Validation 1.1
Concurrency Utilities for Java EE 1.0
Contexts and Dependency Injection for Java 1.1
Dependency Injection for Java 1.0
Enterprise JavaBeans 3.2
Interceptors 1.2
(Maintenance Release covered under JSR 318)
Java EE Connector Architecture 1.7
Java Persistence 2.1
Common Annotations for the Java Platform 1.2
Java Message Service API 2.0
Java Transaction API (JTA) 1.2
JavaMail 1.5
web Services Technologies
Java API for RESTful web Services (JAX-RS) 1.1
Implementing Enterprise web Services 1.3
Java API for XML-Based web Services (JAX-WS) 2.2
web Services Metadata for the Java Platform
Java API for XML-Based RPC (JAX-RPC) 1.1 (Optional)
Java APIs for XML Messaging 1.3
Java API for XML Registries (JAXR) 1.0
Management and Security Technologies
Java Authentication Service Provider Interface for Containers 1.1
Java Authorization Contract for Containers 1.5
Java EE Application Deployment 1.2 (Optional)
J2EE Management 1.1
Debugging Support for Other Languages 1.0
Java EE-related Specs in Java SE
Java API for XML Processing (JAXP) 1.3
Java Database Connectivity 4.0
Java Management Extensions (JMX) 2.0

Technologies
JavaBeans Activation framework (JAF) 1.1
Streaming API for XML (StAX) 1.0

Fuente: (Oracle, 2012)

A continuación se muestra la extensión de la versión estándar de Java para implementación del portal web con tecnologías: JSF, EJB, Hibernate y web services.

Figura 4. Arquitectura multicapas Java

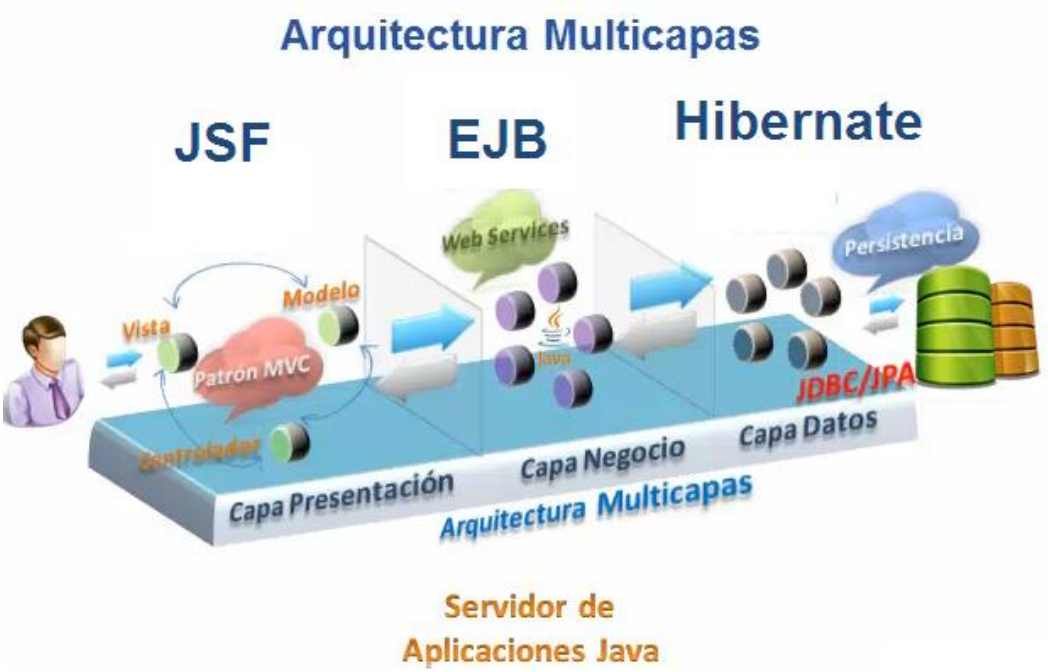
Elaborado por: Pedro Caicedo y Diego Godoy

Una aplicación empresarial Java se compone de distintas capas (presentación, negocio y datos) para cumplir una función específica, dicha división de la aplicación en capas presenta las siguientes ventajas:

- Separación de responsabilidades
- Mejor mantenimiento a la aplicación
- Especialización a los programadores Java en cada una de las capas

La versión empresarial de Java brinda una o varias APIs para cada una de las capas, cuya arquitectura se muestra de forma esquemática en la gráfica siguiente:

Figura 5. Arquitectura multicapas Java



Fuente:(GlobalMentoring, 2012).

La capa de presentación se implementa con el framework JSF, el cual usa el patrón Modelo Vista Controlador-MVC.

Posteriormente se tiene la capa de negocio la cual se implementa con tecnología de EJBs, estos tienen las reglas de negocio de la aplicación Java.

Finalmente se puede observar que Hibernate aplica directamente en la capa de datos.

2.2.4.2 Enterprise Java Bean (EJB)

Los Enterprise Java Beans (EJB) representan una arquitectura mediante el uso de la plataforma Java que permiten a los desarrolladores crear de forma rápida y sencilla aplicaciones de negocio, mediante un desarrollo simplificado de aplicaciones distribuidas con la tecnología Java.

Los EJB son representados mediante clases Java con características especiales, que las hacen más potentes y robustas comparadas con las clases POJO (Plain old Java Object), presentando las siguientes ventajas:

- Los métodos dentro de un EJB pueden ser transaccionales.
- Los métodos pueden ser ejecutados de manera remota.
- Facilidad de comunicación con la base de datos.
- Los métodos pueden ser seguros.

Los EJBs contienen la lógica de negocio de la aplicación Java, por lo tanto cubren el rol de la capa de negocios o servicios dentro de las aplicaciones empresariales mediante la plataforma J2EE.

Con EJB, el desarrollador se enfoca solamente por resolver los problemas que aparecen con el desarrollo del proceso de negocio de la organización, sin importar el procesamiento que tienen estos datos de forma interna, es decir, el desarrollador se enfoca por los problemas generales que conllevan una aplicación empresarial como concurrencia, seguridad, transacción, entre otros.

- **Tipos de EJB**

Los tipos de EJB se presentan mediante el rol que se les va a asignar dentro de la solución, los cuales se listan a continuación.

- EJB de sesión, estos EJB contienen funcionalidad relacionada con lógica de manejo de sesiones de un usuario dentro de la aplicación.
- EJB de Entidad, estos EJB contienen la funcionalidad relacionada con el manejo de datos desde y hacia la base de datos que utiliza el aplicativo desarrollado.
- EJB de Mensaje, estos EJB presentan la funcionalidad necesaria para recibir y publicar mensajes del aplicativo hacia otros módulos del mismo o mediante la interacción hacia otros aplicativos externos.

- **Estructura de un EJB**

Como se mencionó anteriormente, los EJB son representados mediante clases Java, de manera general la estructura de estos EJB la componen los siguientes elementos:

- **POJO,** es la clase Java que contiene los atributos y métodos que contiene el EJB.
- **Anotación** (annotation), conjunto de METADATOS que agregan características y funcionalidades especificas de la plataforma J2EE al objeto POJO anteriormente mencionado. Estos METADATOS en resumen brinda información adicional al código del EJB la cual no es ejecutable.

- **Formas de comunicación con un EJB**

Las diferentes formas de comunicación con un EJB son las siguientes:

- Interfaz local: Se utiliza cuando un cliente se encuentra dentro del mismo servidor Java.
- Interfaz remota: Se usa cuando un cliente está fuera del servidor Java.
- No Interface: Variante de los EJBs locales, es decir dentro del mismo servidor Java, pero que no define una interfaz para comunicarse con el EJB sino que hace uso de los métodos de negocio de la clase java.

2.2.4.3 Java Persistence Hibernate

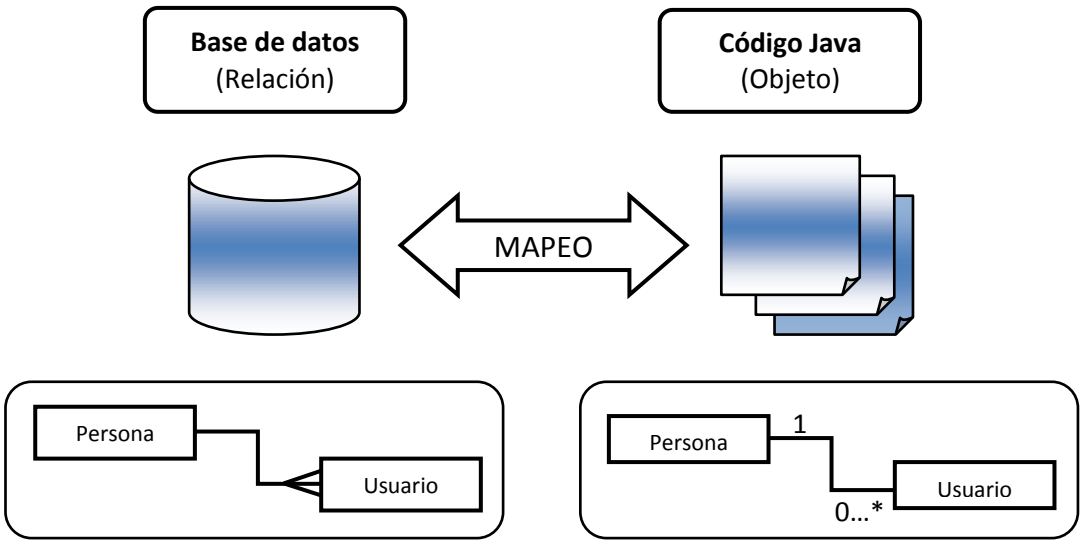
La mayoría de información empresarial que generan los aplicativos desarrollados es almacenada en motores de bases de datos relacionales. Hibernate, es el estándar de persistencia para la plataforma Java para realizar estas tareas.

Hibernate es una herramienta de mapeo entre los objetos (aplicativo) y las relaciones (base de datos) que contiene un aplicativo, facilitando el mapeo de los atributos tanto para establecimiento como para recuperación de información a la base de datos hacia el modelo de objetos de la aplicación. Este mapeo se lo puede realizar mediante archivos con formato XML o mediante anotaciones sobre un EJB de la aplicación.

Hibernate implementa conceptos del framework de Mapeo Relacional de Objetos - ORM (Object Relational Mapping, por sus siglas en inglés) para realizar operaciones sobre un motor de base de datos relacional de tipo select, insert, update y delete. En

el siguiente diagramase puede observar el tipo de tecnología descrita donde se tiene una base de datos con sus tablas, y código y clases Java del lado del servidor:

Figura 6. Analogía entre el código Java y base de datos



Elaborado por: Pedro Caicedo y Diego Godoy

El framework ORM mapea las clases Java con las tablas de bases de datos relacionales. El proceso de ingeniería inversa permite generar código Java a partir de la base de datos relacional. Por ejemplo, al crear un objeto en memoria de la clase Persona, al establecer un valor sobre este objeto, de inmediato se almacena un valor directamente en la tabla Persona. Como se puede visualizar, el uso de esta tecnología ha simplificado el desarrollo sobre la capa de datos de una aplicación empresarial.

- **Clase Entidad y Persistencia en Hibernate**

Una clase de tipo Entidad contiene propiedad o atributos que representan los campos de una tabla de una base de datos relacional.

Hibernate es un framework de persistencia, el cual convierte una clase Java POJO en una clase Entidad simplemente agregando Anotaciones.

Los tipos más comunes de Anotaciones son los siguientes:

- @Entity: especifica que una clase es entidad
- @Transient: indica que el atributo es un valor no persistente
- @Table: especifica el nombre de la tabla que representa en la base de datos
- @Id: indica que el atributo es el ID de la entidad
- @GeneratedValue: indica que el atributo es un valor autosecuencial
- @JoinColumn: indica que el atributo tiene relación con otra tabla
- @ManyToOne: indica que existe una relación de varios a uno con el atributo especificado
- @OneToMany: indica una relación de uno a varios con el atributo especificado

Gracias a esto se podrá consultar los objetos de tipo entidad disminuyendo el desarrollo de la creación de la capa de datos de una aplicación empresarial.

Las características principales de Hibernate son las siguientes:

- **No intrusivo**, Hibernate es una capa separada de los objetos a persistir por ello las clases Java de Entidad no requieren extender ninguna funcionalidad en particular ni saber de la existencia del API Hibernate.
- **Consultas**, utilizando objetos Java, Hibernate ejecuta consultas sobre el motor de base de datos expresados en términos de objetos Java y sus relaciones sin utilizar el lenguaje SQL en la aplicación, luego estas consultas son traducidos por el API Hibernate al código SQL equivalente.
- **Configuración simplificada**, mediante anotaciones o archivos xml se puede realizar personalizaciones.
- **Integración**, Hibernate integra con las demás capas de la aplicación Java de manera simple
- **Testing**, Hibernate realiza pruebas unitarias o utilizar cualquier clase con un método main() fuera del servidor Java.

Cabe resaltar que las tareas de persistencia la realiza el patrón de diseño DTO (Data Transfer Object) el cual define una clase Entidad, la cual es transmitida a las capas de Negocio y Presentación, inclusive llegan a presentarse al usuario a través de la vista del patrón MVC.

2.3 Gestión de documentos empresariales

En este apartado se tratarán los temas más relevantes relacionados a la gestión de documentos empresariales como son: la metodología de desarrollo de software orientado a procesos y los programas informáticos para automatización de procesos y gestión de contenido empresarial.

2.3.1 Gestión de contenidos empresariales

La gestión de contenidos empresariales – ECM (Enterprise Content Management, por sus siglas en inglés), es solo uno de los muchos términos utilizados en el contexto del manejo de documentos que contiene una organización. Definida como una visión o estrategia que contiene un grupo de normas, técnicas y buenas prácticas usadas bajo los principios de racionalización y economía, con el fin administrar el flujo de documentos de cualquier tipo dentro de una empresa.

La importancia sobre la organización de contenido es permitir la recuperación de información de documentos desde repositorios documentales. Además determina el tiempo de conservación de los diferentes tipos de documentos y proceder a categorizarlos como pasivos en caso que éstos posean valiosa información o proceder a eliminarlos cuando ya no sea necesario.

2.3.1.1 Herramientas ECM

Las herramientas que ayudan a la gestión documental presentan una constante evolución, desde los libros de registro, carpetas, archivadores, cajas y estanterías en que se guardan los documentos de papel; más adelante se incluyeron los equipos audiovisuales, posteriormente comenzó el almacenamiento de los documentos en soportes magnéticos u ópticos, los archivos o kárdex los cuales permitieron y facilitaron realizar referencias cruzadas. Todas estas herramientas presentan una larga lista de técnicas de recuperación de información mediante sistemas de codificación y clasificación de la documentación.

Actualmente se ha sumando a ellos los recursos informáticos, los cuales son cada vez más necesarios debido al nivel de sofisticación que han alcanzando los sistemas computacionales y la complejidad que ha llegado a tener la documentación por factores como cantidad, diversificación, confidencialidad y seguridad que se debe aplicar sobre ellos para el apoyo de la actividad administrativa.

Cabe resaltar un factor que cambió el manejo de la documentación mencionada anteriormente, con la aparición del correo electrónico, dentro de las organizaciones surgió la necesidad de capturar y conservar de igual manera los documentos que nacen, viven y mueren en formato electrónico. A razón de este factor surgió como derivado la digitalización, presentando una solución para aquellos procesos críticos que implican un riesgo para la organización, cuando esta excede su capacidad de almacenamiento físico de la documentación o se decide a trabajar bajo la cultura de cero papel[8].

Las herramientas de gestión de contenidos empresariales (ECM) permiten la gestión integral de toda la información de una organización.

A continuación se puntualizan diferentes enfoques sobre el uso de sistemas ECM:

- Estratégico o conceptual, ayuda a las empresas a normar y controlar el contenido generado, de este modo, permite incrementar la eficacia, fomentar la colaboración en la organización y hacer que la información sea más fácil de compartir dentro y fuera de ésta.

- Mediante un conjunto de herramientas de software, se brinda un conjunto de capacidades y/o aplicaciones que permiten organizar, controlar y brindar seguridad a la gestión del ciclo de vida de los contenidos que circulan dentro de la organización.

Los componentes básicos que confirman una solución ECM son:

[8] Cultura de implementación que busca reducir el consumo de papel dentro de una organización.

- **Gestión de documentos,** Como características principales cuenta con servicios de almacenamiento, organización, seguridad, control de versionamiento y la automatización de procesos que se centran en la gestión de dichos documentos.
- **Aplicaciones de procesamiento de imágenes** para la captura, transformación y gestión de imágenes obtenidas a partir de documentos físicos. Este componente requiere la presentación de la siguiente funcionalidad:

  - Captura de documentos mediante hardware de escaneo y su correcto procesamiento mediante software especializado que utilice las tecnologías de reconocimiento óptico de caracteres (OCR) e inteligente (ICR), y la tecnología de procesamiento de marcas sobre formularios (OMR). Esta captura puede realizarse mediante el uso de las capacidades nativas o mediante una asociación formal con un tercer proveedor de soluciones de otros fabricantes, como Kofax, Abbyy Data Capture, Ephesoft, entre otros.
  - Capacidad de almacenar el contenido documentos digitalizados en el repositorio documental diferenciándolo como "otro" tipo de contenido dentro de esta estructura documental, con el fin de organizarlos mediante la ejecución de un proceso electrónico.

- **Ejecución de procesos o flujos de trabajo colaborativos,** con el fin de soportar los procesos documentales y generales de la organización, debe presentar la opción de enrutar contenidos, asignación de tareas a usuarios del sistema, especificación estados de dichas tareas, y la registro de pistas de auditoría.
- **Gestión de expedientes,** debe permitir la retención a largo plazo de documentos a través de la automatización de las políticas de almacenamiento de la organización, lo que garantiza el cumplimiento legal, reglamentario y de la archivística. Se debe permitir como mínimo la capacidad para hacer cumplir la retención de documentos críticos de negocio, basado en un programa de retención de registros.
- **De gestión de contenidos web (WCM),** otra de las funcionalidades que deben presentar estas soluciones es el control de contenidos web las cuales influyen sobre las interacciones de una experiencia web a través del uso de herramientas de gestión específicas basadas en un repositorio central. Esto incluye funciones

de creación de contenidos, tales como plantillas de páginas web, flujo de trabajo de publicación de información y gestión del cambio.

- **Contenido social,** uno de los puntos más solicitados actualmente es la colaboración de los usuarios sobre el sistema, mediante funciones de compartir documentos, gestión del conocimiento. Se añaden a esta funcionalidad blogs, wikis, calendarios de actividades y el apoyo a otras interacciones en línea como la notificación de actividades entre usuarios con roles similares en la organización. Se presenta adicionalmente un enfoque social, la conexión entre usuarios, la presentación de estados sobre el sistema, el seguimiento de documentos y la inclusión de video y audio sobre el repositorio documental.
- **Búsqueda exhaustiva,** uno de los puntos más importantes que prestan estas soluciones es la recuperación de la documentación mediante eficaces prestaciones de búsqueda, las cuales permiten a los usuarios encontrar fácilmente los documentos que son importantes para ellos. Estas búsquedas se pueden realizar mediante un término en el contenido o los atributos o índices clasificadores del documento. (Gilbert, Shegda, Chin, Tay y Koehler-Kruener, 2012).

En base a los factores descritos, un sistema ECM requiere de sistemas de Gestión de Procesos de Negocio - BPMS (Business Process Management System) para la gestión de sus contenidos documentales.

2.3.2 Gestión de procesos BPM

La metodología BPM (Bussiness Process Management, por sus siglas en inglés) presenta un enfoque de mejora de los procesos y flujos de trabajo empresariales.

Bajo el contexto de este proyecto, la gestión de procesos de negocio está orientada a la gestión documental.

El modelo de procesos de negocio se lo realiza en notación BPMN y su automatización con sistemas BPMS.

2.3.2.1 Business Process Management System (BPMS)

Un BPMS o sistema BPM se define como un conjunto de utilidades de software que definen, implementan y mejoran procesos de negocio mediante el cumplimiento del conjunto de especificaciones técnicas necesarias para aplicar la metodología BPM.

Estos sistemas BPM están integrados por un grupo de módulos que permiten crear, transformar y agilizar los procesos y tareas tanto internas como externas, generando una capa de procesos independiente con el fin de diferenciar el entorno tecnológico y el de datos subyacentes.

Mediante esta capa, se pueden visualizar todos los procesos operacionales de la organización y ver su relación con el riesgo estratégico, además indica cómo funciona la organización, la gestión de operaciones más de cerca y ofrece información sobre procesos críticos útiles para perfeccionar los mecanismos de control.

Los sistemas BPM complementan la estrategia de riesgo operacional de la organización, mediante el cumplimiento de dos requerimientos: la automatización y gestión de procesos de negocio, obteniendo como resultado la información necesaria para la toma de decisiones en el tiempo oportuno.

Por medio de la automatización se obtiene un completo seguimiento de las actividades de la organización, además permite identificar rápidamente en qué punto fallan los procesos con más frecuencia o dónde son habituales los errores humanos, por ejemplo, en tareas de ingreso de datos en los sistemas informáticos dentro de los sistemas informáticos.

2.3.2.2 Businnes Process Modeling Notation BPMN

BPMN es un estándar basado en diagramas de flujo, el cual provee una notación gráfica con el fin de representar los procesos de manera legible y de fácil entendimiento para todos los usuarios de una organización, desde los analistas que

diseñan los procesos, los desarrolladores del software que ejecutará dichos procesos, hasta los encargados del negocio que administran y monitorean estos procesos.

Este estándar fue creado por la organización Business Process Management Initiative – BPMI, en el año 2004 (OMG[9], 2012).

Antes de la aparición de este estándar, el diseño de los modelos de procesos de la organización estaba separado de la representación requerida para la implementación y ejecución, por lo que eventualmente fue creciendo la necesidad de traducir manualmente el modelo original diseñado al modelo final en ejecución, al ser esta tarea manual, está sujeta a errores.

Por tanto un objetivo clave que se logra con el uso de este estándar es, cubrir la brecha entre la especificación de reglas que permiten la generación de un código en lenguaje de ejecución de procesos del negocio o BPEL (por sus siglas en inglés) y, la lectura o mapeo entre los elementos gráficos del estándar BPMN.

Cabe mencionar que el estándar BPMN es análogo al UML, el cuál estandarizó el mundo de la ingeniería del software en el pasado.

2.4 Gestión de firma electrónica

El vocablo firma proviene del latín "firmare", que significa afirmar o dar fuerza, también conocida como rúbrica. La firma es una escritura gráfica o grafo manuscrito que representa el la identificación de una persona mediante la especificación de sus nombres, apellidos y/o título, donde dicha persona los escribe de su propia mano.

Esta firma es utilizada con fines de identificación, jurídicos, representativos y diplomáticos, es decir, su fin es identificar, asegurar o autentificar la identidad de un autor o remitente, también permite ser usado como una prueba del consentimiento y/o de verificación de la integridad y aprobación de la información contenida dentro de un documento o similar, el cual tiene carácter legal. (Revista de Cultura Ñ, 2008).

[9]Object Management Group

La firma electrónica surge debido a la necesidad de satisfacer un mundo globalizado, en donde las transacciones y la interacción entre individuos son impersonales y sin vínculos físicos, haciendo de la identificación un problema y requerimiento de primera necesidad. Debido a esta realidad, los medios cotidianos de identificación pierden validez dentro el mundo electrónico, surgiendo así los medios digitales de identificación.

Sin embargo, antes de profundizar en el tema de la firma electrónica resulta necesario delimitar su dimensión y ámbito de aplicación sobre el entorno actual.

2.4.1 Dimensiones de la firma electrónica

La firma electrónica es una herramienta más que permite la adaptación tecnológica a este nuevo paradigma cultural, económico y social, el cual facilita la expansión del comercio dentro de la nueva economía electrónica global, redefiniendo las relaciones e interacción humana, con el fin de optimizar la eficiencia mediante un bajo costo.

El objetivo básico de la firma electrónica es aportar al mundo de los documentos electrónicos la misma funcionalidad que aporta la firma tradicional a un documento impreso. Donde presenta un enfoque cultural mediante el reemplazo de eventos como concertar una reunión para firmar las diferentes copias de un documento y satisfacer la necesidad de firmar un documento por parte de personas que pueden encontrarse a miles de kilómetros de distancia los cuales realizarán la firma sin coincidir en el tiempo ni tener ningún contacto personal.

Por otra parte, la firma electrónica presenta un enfoque social permitiendo la identificación segura y veraz de un individuo o una organización sobre toda la población presente en la Internet con fin de prestar una comunicación entre las partes que intervienen en un trámite. Para finalizar, el enfoque económico se presenta en las diversas circunstancias y aplicaciones que tiene sobre el comercio actual, dando lugar a una gran variedad de nuevos servicios y productos relacionados con ella.

2.4.2 Antecedentes de la firma

A lo largo de la historia, la firma ha sido el símbolo de un elemento esencial en todo acuerdo suscrito entre personas, sin embargo, no siempre la firma ha existido como tal.

Antiguamente, en Roma, los documentos no eran firmados, en esta época se celebraba una ceremonia denominada "manufirmatio", que consistía en la lectura del documento mediante el autor del mismo o por un funcionario del gobierno. Luego se extendía el documento sobre la mesa del escribano (el notario de entonces) y después de pasar la mano sobre el pergamino, se realizaba un juramento solemne en signo de aceptación entre los involucrados. Después de realizada esta ceremonia era cuando se estampaba el nombre del autor o autores del documento a manera de una resolución (Serrano, 2012).

Durante la edad media, se utilizaban sellos, marcas y signos, estos últimos los compuestos por una cruz a la que se le agregaban diversas letras y rasgos de forma entrelazada. Siendo estos signos los cuales han continuado evolucionando y llegando a ser la representación de las firmas hasta nuestros días. (Ruiz, 2010).

Como se detalló, las firmas cumplen una función, de "identificación", así como de avalar derechos y obligaciones convenidas por su autor, sin embargo este método no es totalmente fiable puesto que el mismo podría ser falsificado y su autoría debería ser comprobada por un perito.

Estas firmas, tradicionalmente han incluido elementos tácitos que aportan autenticidad a la misma y que cumplen la función de identificar a la persona, o de proporcionar certidumbre de la persona que participa, sin embargo el aspecto más importante es la vinculación de la persona con el contenido del documento.

La firma, además cumple con una función específica como ya se señalaba en líneas anteriores, de ser un medio de identificación, también agrega un medio de declaración de conocimiento del contenido del documento por el autor de la firma. Además poseen un carácter probatorio, permitiendo identificar si el autor de la firma

es efectivamente aquél que ha sido identificado como tal en el acto de la propia firma.

2.4.3 Criptografía

Es la ciencia de conservar en secreto los mensajes, donde el texto original es convertido en un texto equivalente codificado, el cual es conocido como criptotexto[10] mediante el uso de un algoritmo de encriptación.

Posteriormente este criptotexto es decodificado al momento de su llegar al receptor del mensaje, donde este texto vuelve a su estado original (Stallings, 2010).

La criptografía busca la ocultación, disimulación o cifrado de la información, mediante el diseño de técnicas que realicen estas funciones. Esta ciencia abarca tanto a la criptografía tradicional (datos, texto, e imágenes), la criptofonía [11] y al criptoanálisis[12].

Cifrar por tanto consiste en transformar una información (texto claro) en otra ininteligible (texto cifrado o cripto) según un procedimiento y usando una clave determinada. La seguridad de un sistema criptográfico reside en el secreto de la llave más que en el secreto del algoritmo que encripta el texto. (LEFISPedia, 2012).

2.4.3.1 Criptografía simétrica

Es aquel uso de criptografía en la que la clave de encriptación es la misma que decriptación, por lo tanto todo criptosistema es llamado simétrico cuando las claves para cifrar y descifrar la información son idénticas. Cabe recalcar que estos sistemas son mucho muy útiles para el cifrado de grandes volúmenes de datos.

El mecanismo de este tipo de criptografía implementa técnicas que usan una clave llamada clave privada, debido a que es la clave que le pertenece al remitente de los

[10]Texto que contiene información encriptada de un mensaje.
[11] Rama de la criptografía encargada de la ocultación de mensajes de audio.
[12] Ciencia que estudia los pasos y operaciones orientados a transformar un criptotexto al mensaje original sin la necesidad de conocer el sistema de cifrado utilizado en la encriptación inicial.

mensajes y es compartida al receptor con el fin de cifrar y descifrar respectivamente el mensaje. Para mantener la seguridad del cifrado, tanto el emisor como el receptor deben mantener esta clave en secreto, siendo este su principal inconveniente.

El sistema de cifrado de este tipo más extendido es Data Encryption Standard (DES), desarrollado por IBM y adoptado por las oficinas gubernamentales estadounidenses para protección de datos desde 1977 (Schneier y Banisar, 1997).

Para finalizar se presenta a continuación las características principales de las claves privadas:

- Solo existe una clave privada por individuo.
- El uso la clave es único del propietario.
- La clave es secreta.
- La clave se usa también para desencriptar información y generar firmas electrónicas.

2.4.3.2 Criptografía asimétrica

Llamado también cifrado de clave pública, donde cada persona tiene un par de claves, una pública conocida por todos y otra privada que sólo su propietario conoce. El uso dado a este par de claves es el siguiente: la clave privada, es usada por el propietario para encriptar los mensajes, mientras que la otra, llamada clave pública, es usada para desencriptar el mensaje cifrado.

Por lo tanto si el individuo A y B usan la criptografía asimétrica, en este caso, el individuo A envía un mensaje al individuo B. El individuo A, usa su clave privada para encriptar el mensaje. Al recibir el mensaje el individuo B, usa la clave pública del individuo A para desencriptar dicho mensaje y verificar la identidad del individuo A.

La firma electrónica, para cumplir con los requisitos de autenticación, fiabilidad e inalterabilidad requiere de métodos de encriptación, como el llamado asimétrico o de clave pública. En la práctica, debido a que los algoritmos de clave pública requieren

mucho tiempo para cifrar documentos largos, los protocolos de firma electrónica se implementan junto con funciones unidireccionales de resumen de esta información (funciones Hash), de manera que en vez de firmar todo el documento, se firma un resumen del mismo.

Uno de los primeros esquemas de clave pública fue desarrollado por R. Rivest, A. Shamir y L. Adleman. El esquema Rivest-Shamir-Adleman (RSA) ha sido desde la fecha de su publicación el único sistema ampliamente aceptado para la implementación de encriptación mediante clave pública (Schneier y Banisar, 1997).

2.4.4 Certificados digitales

Un certificado digital es el único medio válido que permite avalar tanto técnica como legalmente la identidad de un individuo u organización dentro de Internet. Por otra parte, estos certificados digitales son un requisito indispensable para que las organizaciones puedan brindar servicios seguros a través de Internet.

Adicionalmente una de las funcionalidades más utilizadas de los certificados digitales es permitir la firma electrónica de documentos, estos certificados permiten además cifrar las comunicaciones donde solamente el emisor y el destinatario de la información podrán conocer el contenido de la misma.

En definitiva, la principal ventaja que presentan los certificados digitales es el ahorro de tiempo y dinero al realizar trámites administrativos o burocráticos sobre Internet, a cualquier hora y desde cualquier lugar.

Un certificado digital consta de una pareja de claves criptográficas, una pública y una privada, creadas con un algoritmo matemático, las cuales serán detalladas más adelante.

2.4.5 Función HASH

Una manera fácil e intuitiva de entender el concepto de Hash es referirse al concepto de "huella digital" o "huella dactilar" sobre la información.

El Hash se puede entender como una "huella" única e irrepetible que se calcula a partir de aplicar algoritmos matemáticos complejos como SHA1[13] o SHA2[14] a la información que se quiere aplicar en conjunto a la clave privada o a la validación mediante la ayuda de la clave pública.

De esta "fusión" algorítmica, resulta una huella, que es una cadena de dígitos alfanuméricos, con una extensión de 2024Kb.

El Hash se conoce normalmente con el nombre de firma electrónica.

2.4.6 Firma electrónica

La firma electrónica consiste en la utilización de un método de encriptación llamado asimétrico o de clave pública. Dicho método consiste en establecer un par de claves asociadas a un sujeto, una pública, conocida por todos los sujetos intervinientes en la transición y la otra privada, sólo conocida por el sujeto en emisor del mensaje. (Peñaranda, 2011)

Actualmente existen muchas definiciones de firma electrónica, sin embargo la más completa con respecto al enfoque del presente trabajo es la siguiente: "Por firma electrónica se entenderán los datos en forma electrónica consignados en un mensaje de datos, adjuntados o lógicamente asociados al mismo, que puedan ser utilizados para identificar al firmante en relación con el mensaje de datos e indicar que el firmante aprueba la información recolectada" (CNUDMI[15], 2003).

Según Contreras (2009), "El concepto de firma electrónica aspira a abarcar todos los usos tradicionales de una firma manuscrita con consecuencias jurídicas", dado el hecho que en la actualidad existen un flujo de transacciones en donde las partes ya no tienen contacto "físico" y se presentan las siguientes dudas como:

[13] SHA1, algoritmo criptográfico que genera una cadena de 160 bits en base a la información a encriptar.
[14] SHA2, versión mejorada de SHA1 actualmente utilizada por tecnologías de criptografía asimétrica, llega a generar una cadena de 512 bits sobre la información a encriptar.
[15] Comisión de las Naciones Unidas para el derecho mercantil internacional

- ¿Cómo se puede asegurar e la identidad del individuo con quien se están realizando una operación?
- ¿Cómo se tiene la certeza de que la información intercambiada no ha sido robada, alterada o conocida por personas ajenas?

Por lo tanto, los retos que debe atender la firma electrónica son garantizar la identidad del firmante y garantizar que el documento no ha sido modificado tras ser firmado. El fin de validar la integridad del documento no se relaciona con el hecho de validar el contenido, sino el de garantizar que este documento no ha sido modificado tras su firma.

Para presentar estas garantías no es necesario que un tercero custodie una copia del documento, esto se realiza mediante la generación de un código único del documento a partir de la información que contiene en el momento de ser firmado. Donde cualquier alteración del contenido del documento provocará el rompimiento de la integridad que posee dicho documento (Cano, 2007).

Dicho concepto de identidad debe ser complementado con el concepto de no repudio, que jurídicamente implica que el firmante no pueda negar haber firmado. Entre otros, los factores principales que garantizan el no repudio son:

- La clave privada vinculada al certificado y que confiere unicidad a los documentos firmados, se encuentre en posesión del firmante desde el mismo momento que se generan dichas claves.
- El certificado y los dispositivos de firma empleados deben basarse en tecnologías y procesos seguros que eviten el uso o sustracción de la clave por parte de terceros y que se encuentren homologados por una autoridad de certificación reconocida.
- Que el certificado esté activo en el momento de ser utilizado..
- Los receptores de documentos firmados deben disponer de una herramienta de verificación segura que no permita suplantar identidades del firmante.

2.4.7 Infraestructura de clave pública PKI

La infraestructura de clave pública o PKI por sus siglas en inglés Public Key Infrastructure, representa una tecnología que contiene conjunto combinado de hardware y software, el cual es gestionado mediante políticas y procedimientos de seguridad, los cuales permiten la correcta ejecución con garantías necesarias sobre las diferentes operaciones criptográficas (cifrado, firma electrónica o el no repudio) en transacciones electrónicas (Carlisle y Lloyd, 2003).

Esta tecnología permite a sus usuarios autenticarse frente a otros usuarios que comparten esta infraestructura, mediante el uso de la información de sus certificados de identidad con el fin de cifrar y descifrar mensajes, firmar electrónicamente información, garantizar el no repudio de información, entre otras aplicaciones. Por lo general dentro de una operación criptográfica que ejecuta el PKI, intervienen como base los siguientes elementos:

- El usuario que inicia la transacción electrónica.
- Los sistemas servidores que registran la ocurrencia de la transacción y garantizan la validez de los certificados que intervienen en esta transacción.
- El usuario destinatario de los datos cifrados, firmados, enviados por parte del usuario iniciador de la operación, los cuales están garantizados por los sistemas anteriormente mencionados.

Debido a que dentro de las operaciones criptográficas asimétricas se utilizan algoritmos de cifrado conocidos y estos son de acceso público, la custodia y privacidad de la llamada clave privada es responsabilidad del usuario de la infraestructura y los procedimientos operacionales o políticas de seguridad que aplica el PKI en la organización.

Por tanto se enfatiza la importancia de aplicar estas las políticas de seguridad sobre esta tecnología, puesto que ni los dispositivos más seguros ni los algoritmos de cifrado más fuerte sirven de nada si por ejemplo un tercero accede a una copia de la clave privada protegida del PKI o de alguno de sus usuarios.

Con respecto a los servicios que brinda la tecnología PKI se detallan los siguientes:

- Autenticación de usuarios.
- Autenticación sistemas.
- Identificación de usuarios dentro a una transacción electrónica.
- Cifrado de información.
- Firma electrónica de información.
- Garantizar la ejecución de transacciones electrónicas.
- No repudio.

2.4.7.1 Tipos de certificados digitales

Dentro de una PKI existen diferentes tipos de certificados digitales, los cuales varían en función de la información que contiene cada uno y a nombre de quién se emite el certificado, a continuación se detalla cada uno de estos certificados:

- Certificado personal, dentro de la PKI, un individuo, al igual que posee su cédula de identificación, puede tener este certificado el cual acredita la identidad del titular. Cabe recalcar que al igual que la cédula, esta persona podrá representarse como persona natural.
- Certificado de pertenencia a empresa, similar al certificado anterior, con la diferencia de que el mismo individuo acredita que la identidad que presenta posee una vinculación con la entidad para la que trabaja.
- Certificado de representante, en base al certificado anterior, este además de la pertenencia a una empresa acredita también los poderes de representación que el titular tiene sobre esta empresa.
- Certificado de persona jurídica, este certificado identifica a una empresa o sociedad como tal a la hora de realizar trámites electrónicos ante otras administraciones o instituciones.
- Certificado de cualidades, permite identificar una cualidad que defina a un individuo, por ejemplo su estado civil, situación, entre otros. Este tipo de certificado complementa al certificado personal. (p.ej. ingeniero, arquitecto, soltero, encargado, entre otros).

Cabe resaltar, que existen otros tipos de certificados digitales utilizados en entornos automatizados, los cuales son presentados a continuación:

- Certificado de servidor seguro, certificado utilizado en los servidores web que desean quieren identificarse en el Internet y además proteger ante terceros el intercambio de información con los usuarios al prestar sus servicios.
- Certificado de firma de software, permite garantizar la autoría y la no modificación del código que contienen las aplicaciones informáticas. (Carlisle y Lloyd, 2003).

2.4.7.2 Elementos dentro del PKI

Los elementos que contiene una infraestructura de clave pública son presentados a continuación:

- Certificate Authority (CA), o autoridad de certificación, se encarga de emitir y revocar todos los certificados digitales solicitados a la PKI. Adicionalmente es la entidad de confianza que brinda legitimidad a la relación entre la clave pública con la identidad de un usuario o servicio. Adicionalmente dependiendo del tamaño del PKI generalmente se crean autoridades de certificación secundarias las cuales se distribuyen su información, por ejemplo, una CA para Quito y otra CA para Guayaquíl, donde la CA de Quito contiene información únicamente de los usuarios de la ciudad de Quito y lo propio para la CA de Guayaquíl.
- Registration Authority (RA), o autoridad de registro, es la encargada de verificar el enlace entre los certificados (concretamente, entre la clave pública del certificado) y la identidad de sus respectivos titulares.
- Repositories, o repositorios son las estructuras encargadas de almacenar la información relativa a la PKI. Los dos repositorios más relevantes dentro de la infraestructura son el repositorio de certificados y el repositorio de listas de revocación de certificados (certificados no válidos o inhabilitados), los cuales son administrados mediante una lista de revocación de certificados o CRL por sus siglas en inglés de Certificate Revocation List (Carlisle y Lloyd, 2003).
- Validation Authority (VA), o autoridad de validación, su funcionalidad es comprobar la validez de los certificados digitales, es decir, verificar si existe

algún todos los factores de un certificado sean válidos, por ejemplo la fecha de vigencia del certificado esté dentro del tiempo permitido, entre otros.

- TimeStamp Authority (TSA), o autoridad de sellado de tiempo es la encargada de adicionar un sello de tiempo validado por el PKI a una firma electrónica en reemplazo del estampado de tiempo que lo hace el equipo donde se genera dicha firma.
- Usuarios y organizaciones, son los elementos dentro del PKI que poseen un par de claves (una pública y otra privada) y un certificado asociado a su clave pública. Estos elementos por lo general usan un conjunto de software que hacen uso de la tecnología PKI, por ejemplo, la generación de firmas digitales. (Adams y Lloyd, 2002).

CAPÍTULO 3
ANÁLISIS Y DISEÑO

3.1 Modelo del negocio (procesos)

Seguidamente se muestra el resultado del dimensionamiento del proceso de gestión documental en uso en BANRED, levantado mediante entrevistas a su personal (ver anexo: Plan de entrevista para dimensionamiento del proceso gestión documental):

- BANRED diligencia los trámites[16] documentales siguientes: la solicitud, el *memorando, el informe, el contrato, el currículum vitae, el acta, la convocatoria y el oficio*.

  El estándar de nominación de los documentos es el siguiente:

  "Siglas del área de negocio" – "número de documento" – "año vigente" – "iniciales del funcionario que creó el documento"

  Donde,

  - "Siglas del área de negocio": Esta dado en función del organigrama estructural de BANRED vigente (ver figura No 7), cuyas siglas son:

  Tabla 3. Siglas de área de negocio empresa BANRED

DESCRIPCIÓN	SIGLAS
GERENCIA GENERAL	GG
PROCESO DE OPERACIONES	PO
GESTIÓN COMERCIAL	PGC
PROCESO DE TECNOLOGÍA	PT
PROCESO DE ADMINISTRACIÓN Y FINANZAS	PAF
PROCESO DE PROYECTOS	PP
PROCESO DE REVISIÓN Y CONTROL	PRC
PROCESO DE INGENIERÍA DE PROCESOS	PGP
PROCESO DE GESTIÓN DE RIESGOS	PGR

Fuente: (BANRED, 2012).

[16] Trámite es la gestión realizada para obtener un resultado, en pos de algo, o los formulismos requeridos para resolver un asunto.

Figura 7. Organigrama Estructural de BANRED

Fuente:(BANRED, 2012).

- "Número de documento": Identificador único secuencial de trámites de hasta 4 dígitos.
- "Año vigente": periodo fiscal en el que se crea el documento.
- "Iniciales del funcionario que creó el documento": Iniciales del primer nombre y primer apellido del personal de BANRED. En caso de existir un homónimo, se añade la sigla del segundo nombre y del segundo apellido.

Por tanto, del ejemplo siguiente: PO-0388-2013-MB, se concluye que, PO corresponde a la oficina de Proceso de Operaciones, 0388 es el secuencial del número de trámite, 2013 el año en que se creó el documento, y MB las iniciales del funcionario que creó el documento, en este caso, Marcelo Balarezo.

- No se cuenta con flujos de aprobación de documentos, ni versionado de éstos.

- Existe un repositorio centralizado para el almacenamiento histórico de documentos de tipo *memorando y oficio*, no así para los documentos de tipo solicitud, *informe*, contrato, currículum vitae, *convocatoria y acta de reuniones*, cuyos archivos se encuentran distribuidos en la LAN de BANRED en el computador de cada usuario.

- Se cuenta con dos formularios para el registro de atributos o METADATA de los archivos tipo memorando y oficio, cuyo fin es contar con criterios de búsqueda de dichos documentos.

  Estos formularios y su base de datos se encuentran desarrollados en tecnología Domino Lotus de IBM. Para su acceso el usuario debe proporcionar sus credenciales de autenticación.

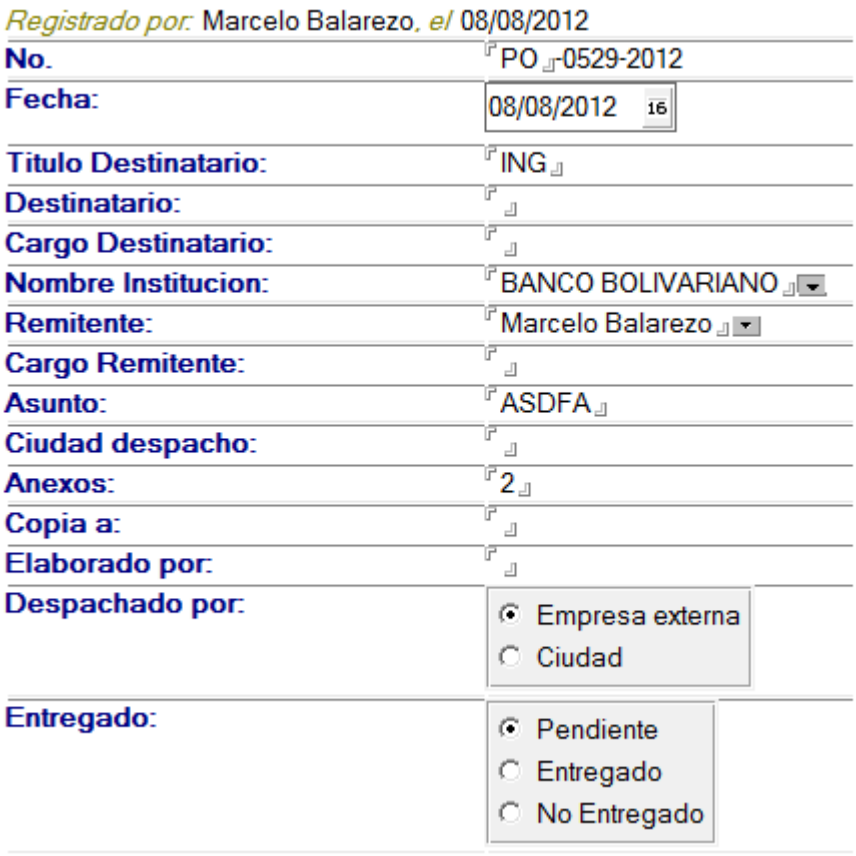
  Cabe mencionar que no existen roles de usuario para esta actividad.

  A través de un par de capturas de pantalla de los formularios descritos, se establecen los atributos de los documentos tipo *memorando y oficio*:

Figura 8. Captura de pantalla del formulario para registro de trámites de tipo memorando

Fuente:(BANRED, 2012).

Figura 9. Captura de pantalla de formulario para registro de trámites de tipo oficio



Fuente:(BANRED, 2012).

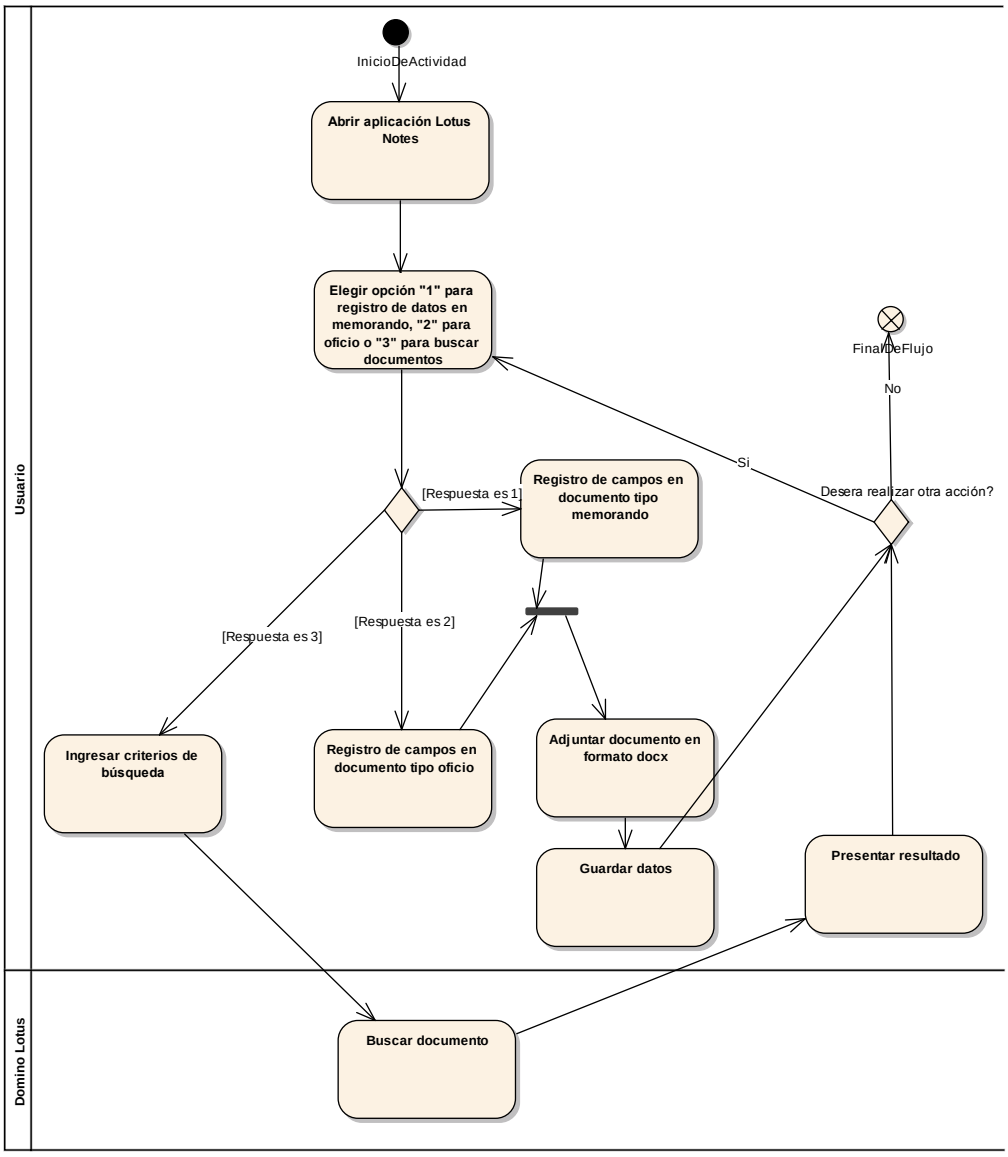
Dicho aplicativo está en capacidad de realizar la búsqueda de documentos en el repositorio documental, pero no posee la funcionalidad de generar reportes de gestión ni de auditoría.

En resumen, el aplicativo Lotus posee las siguientes funcionalidades:

- Registra METADATA de documentos tipo memorando y oficio y adjunta los respaldos digitales seleccionando el documento y almacenándolo en un repositorio centralizado.
- Posee credenciales de autenticación de usuario.
- Realiza búsqueda de documentos tipo *memorando y oficio* por los campos de la METADATA almacenada.

- En seguida se muestra el diagrama de actividades del flujo de trabajo instaurado en BANRED, para el registro de METADATA y almacenamiento de documentos de tipo *memorando y oficio*:

Figura 10. Diagrama de actividades del flujo de trabajo para el registro de METADATA y almacenamiento de documentos de tipo *memorando y oficio*



Elaborado por: Pedro Caicedo y Diego Godoy

Nota: Este flujo usa el patrón de interacción entre diagramas de actividades de UML 2.0, y los procesos del negocio modelados y automatizados bajo

tecnologías workflow (EIA[17], 2008), para representar las operaciones del sistema descrito.

- Respecto al proceso de firma electrónica, se indicó que BANRED posee un aplicativo cliente instalado en el computador de los usuarios autorizados para tal fin, el cual solicita al usuario la selección del certificado digital (token o archivo digital) emitido por Banco Central del Ecuador y el documento a firmar en formato *.docx.

  Una vez realizado lo anterior, el aplicativo descrito genera un nuevo documento en formato *.pdf con la impresión del Hash de firma electrónica.

  Se aclara que este software no se integra a ninguna herramienta ni plataforma de BANRED, ni interviene en flujos de trabajo.

- En tanto que la gestión de recursos de la red LAN de BANRED, se la realiza mediante el Directorio Activo de Microsoft Windows, pero dicha herramienta no se encuentra integrada a la plataforma Domino Lotus, por ende no es posible realizar la autenticación de usuarios con una única credencial de sistema operativo Microsoft Windows hasta el momento.

3.2 Visión y análisis del negocio

Una vez analizada la situación actual del proceso de gestión documental de BANRED, a continuación se detalla la problemática que deberá ser resuelta con este producto:

- Flujos de trabajo del ciclo de vida documental ejecutados manualmente.
- Procesos de gestión documental no definidos ni estandarizados.
- Repositorio de documentos no centralizado.
- Proceso de firma electrónica no vinculado al ciclo de vida documental.
- Herramientas de software relacionadas a este proceso, no integradas.

[17] Escuela de Ingeniería de Antioquia

- Tareas de usuario, concernientes a la gestión documental, ejecutadas manualmente, sin seguimiento ni auditorías de cumplimiento.

Considerando lo expuesto, así como los objetivos de este proyecto, a continuación se detalla la propuesta de solución correspondiente:

3.2.1 Reorganización del proceso de gestión documental

La reingeniería del proceso de gestión documental, partió de la identificación y modelado del ciclo de vida de los documentos[18] tipo memorandos, oficios y actas de reuniones, informes Internos y convocatorias.

Con esto se normó el tratamiento secuencial y coherente de los documentos mencionados que se van desde la creación hasta el momento en que son eliminados o archivados,

A continuación se especifican las fases de los ciclos de vida por tipo de documento definidos:

- Documentos tipo memorandos, oficios y actas de reuniones: Las fases son creación, revisión, aprobación y registro de firma electrónica.
- Documento tipo informes Internos: Las fases son creación y registro de firma electrónica.
- Documento tipo *convocatorias para reuniones*: Solo contempla la fase de creación.

Los siguientes roles de usuario poseerán los privilegios de operación sobre las fases del ciclo documental en la plataforma ECM:

- Gerente: Encargado de aprobar y estamparla firma electrónica en los documentos. Posee privilegios de visualización de todos los documentos.

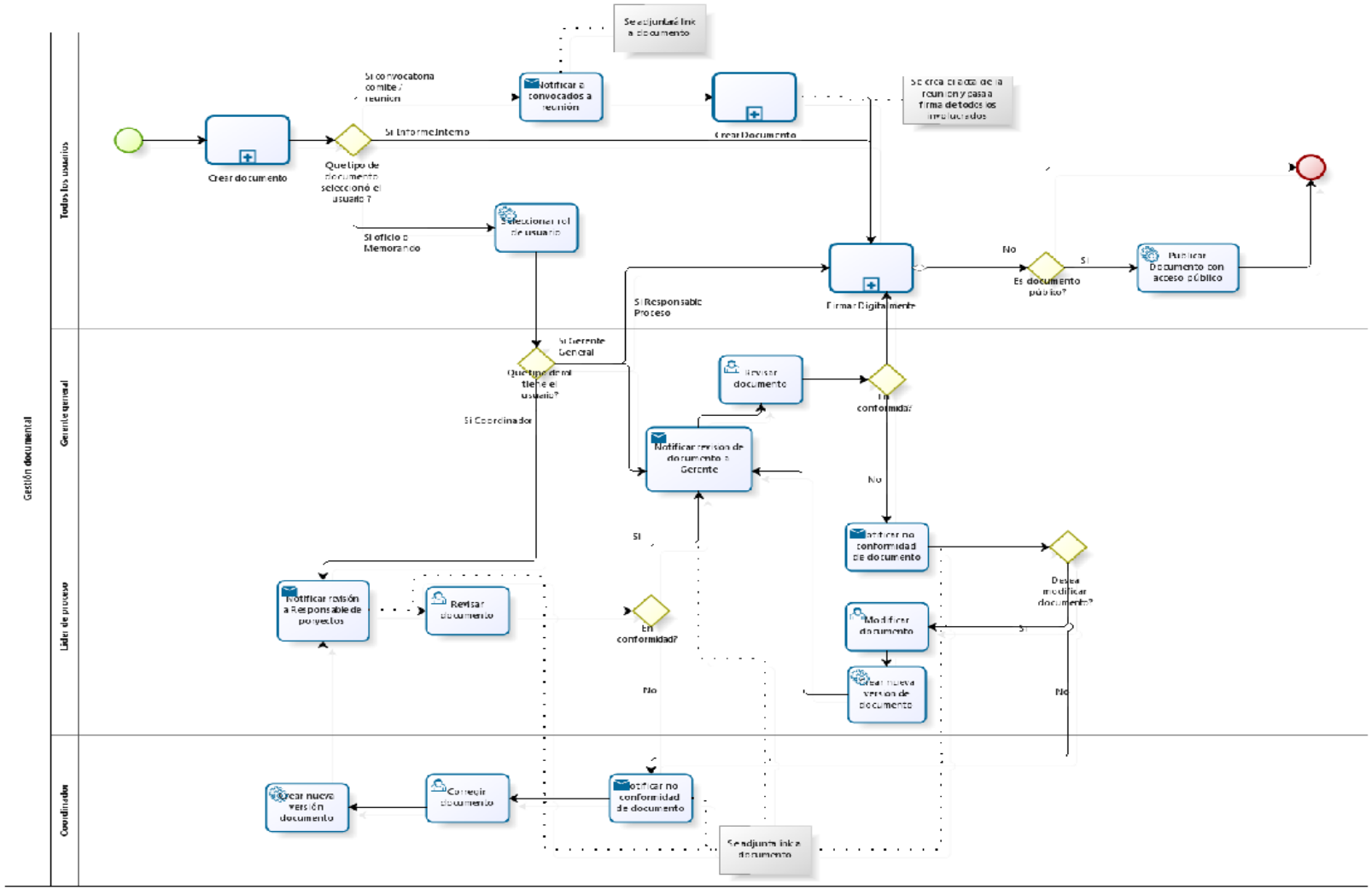
[18]El ciclo de vida de los documentos es la agrupación de fases por las que atraviesa un documento y las tareas relacionadas con dicha fase.

- Líder de proceso: Responsable de la revisión y edición de los documentos. Dicho rol podrá visualizar los documentos que corresponden a su posición jerárquica en relación al organigrama estructural de BANRED.
- Coordinador: Encargado de crear los documentos de negocio, solo poseerá permiso de visualización de los documentos creados por éste.
- Administrador: Responsable de la creación y mantenimiento de flujos los de trabajo y de las plantillas para documentos administrativos, así como de la asignación y mantenimiento de los roles y usuarios en los flujos de trabajo.

A fin de cubrir la brecha entre el diseño de los procesos de negocio y su posterior implementación, se ha optado por el uso del estándar BPMN como notación para el modelado de los flujos de trabajo de este proyecto.

En seguida se muestra el diagrama de proceso de negocio (BPD, por sus siglas en inglés) bajo el estándar referido y la descripción de sus componentes:

Figura 11. BPD proceso de gestión documental

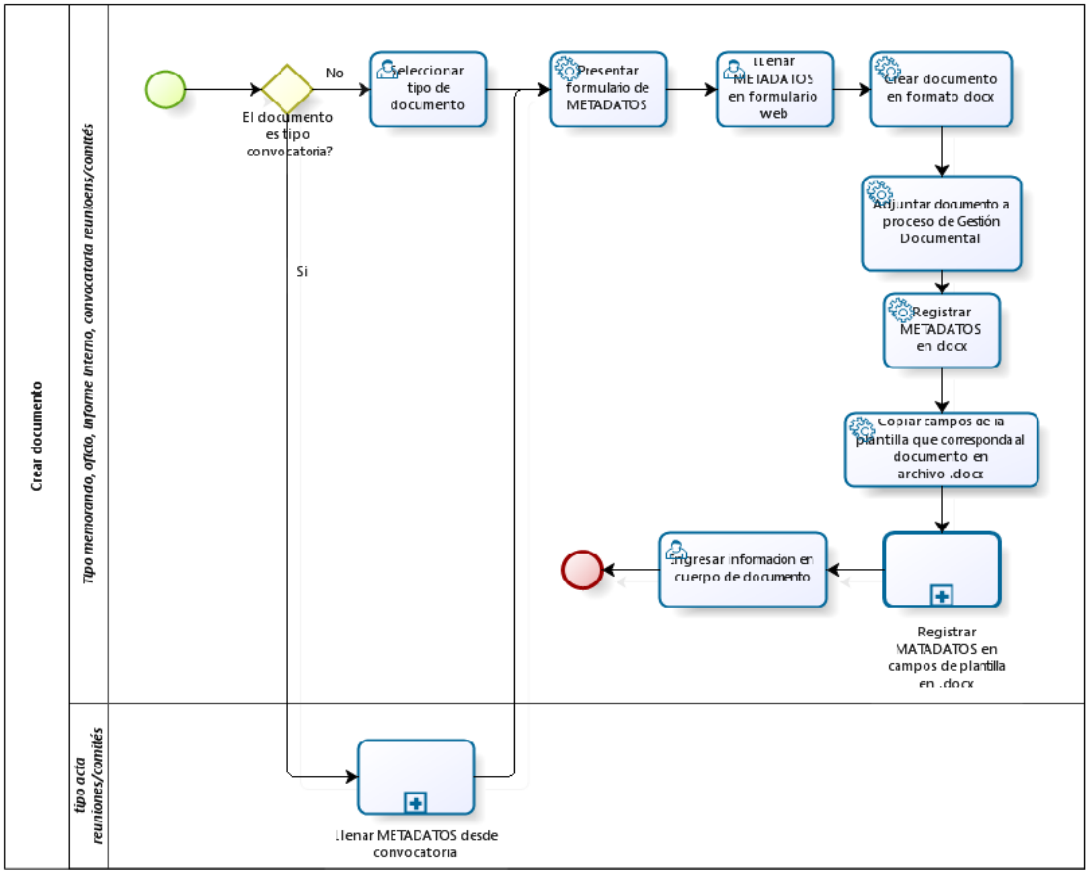


Elaborado por: Pedro Caicedo y Diego Godoy

3.2.1.1 Creación de archivos con formato docx

Este proceso se encarga de crear los documentos inmersos en los flujos de trabajo en formato docx. Inicia con la petición del tipo de documento al usuario, y continúa con la solicitud de registro de información en la cabecera (METADATA) y cuerpo de dicho documento, según muestra el diagrama siguiente:

Figura 12. BPD creación de archivos con formato docx

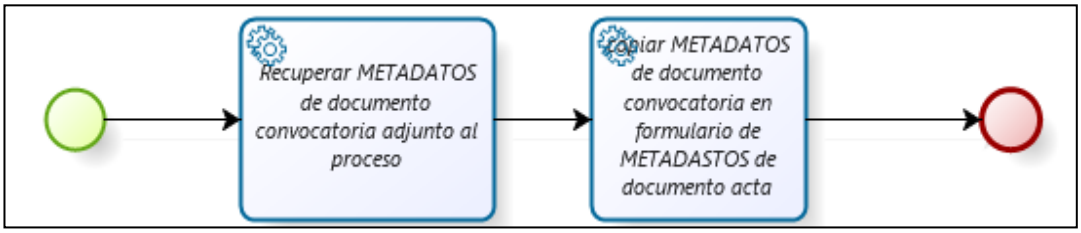


Elaborado por: Pedro Caicedo y Diego Godoy

- **Llenar METADATOS desde convocatoria**

El documento tipo *acta reuniones/comités*, registra ciertos campos de METADATA a partir de la información del documento tipo *convocatoria* y posteriormente sigue el curso del flujo. A continuación se muestra el diagrama:

Figura 13. BPD registro de METADATOS desde convocatoria

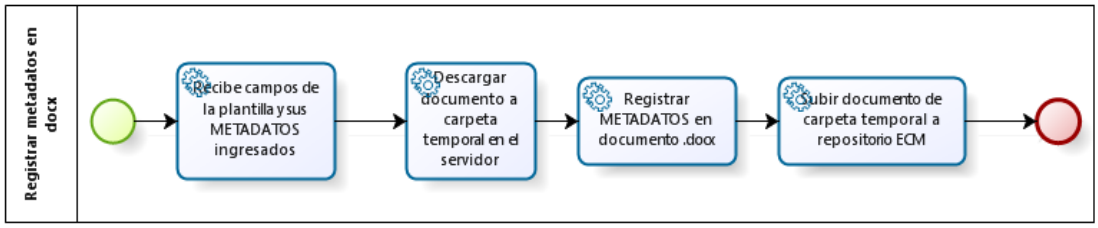


Elaborado por: Pedro Caicedo y Diego Godoy

- **Registrar METADATOS en docx**

Este proceso es el responsable de escribir los METADATOS ingresados por los usuarios a través de las plantillas documentales en el encabezado de los archivos *.docx.

Figura 14. BPD registro de METADATOS en docx



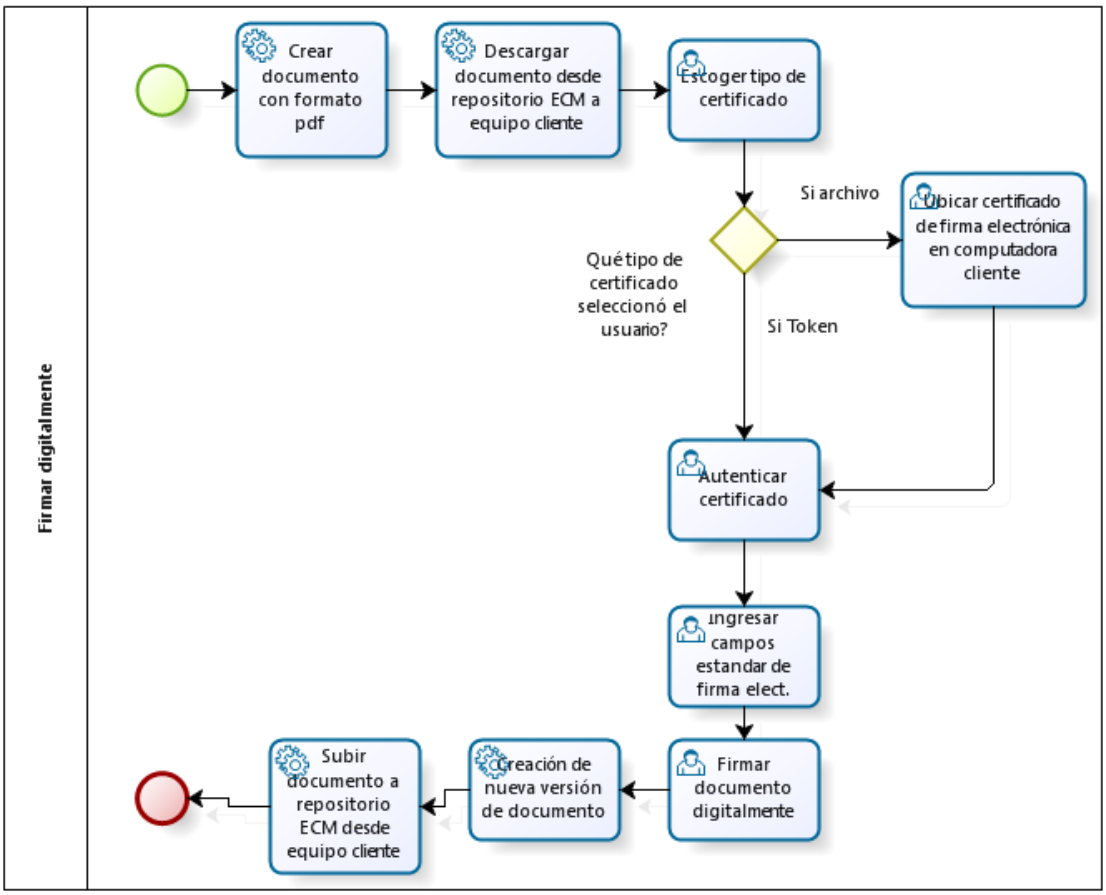
Elaborado por: Pedro Caicedo y Diego Godoy

3.2.1.2 Firmar Digitalmente

Este proceso se encarga de registrar la firma electrónica en los documentos creados por el usuario, para lo cual, inicia con una transformación de formato de archivo de *.docx a *.pdf, continuando con la selección del tipo de certificado de firma y finalizando con el registro de la firma electrónica, otorgando de esta manera validez jurídica al documento.

A continuación se muestra el diagrama del proceso en mención:

Figura 15. BPD Firma Electrónica



Elaborado por: Pedro Caicedo y Diego Godoy

Todas las funcionalidades descritas serán implementadas en un motor BPMS a excepción del proceso de firma electrónica, el cual dado su complejidad, será desarrollado en un componente de software Java, y será invocada su funcionalidad cuando el proceso de gestión documental lo requiera.

Posteriormente se despegarán todos los procesos incluido el de firma electrónica en una herramienta empresarial de gestión documental ECM.

3.2.2 Procesos implementados en herramienta de gestión documental

La primera tarea destinada a la implementación del nuevo proceso de gestión documental de BANRED; indicado en el punto anterior, consistió en seleccionar una herramienta ECM; de tipo open source disponible en el mercado, que proporcione el

rendimiento y escalabilidad empresarial necesaria para que BANRED gestione sus documentos empresariales.

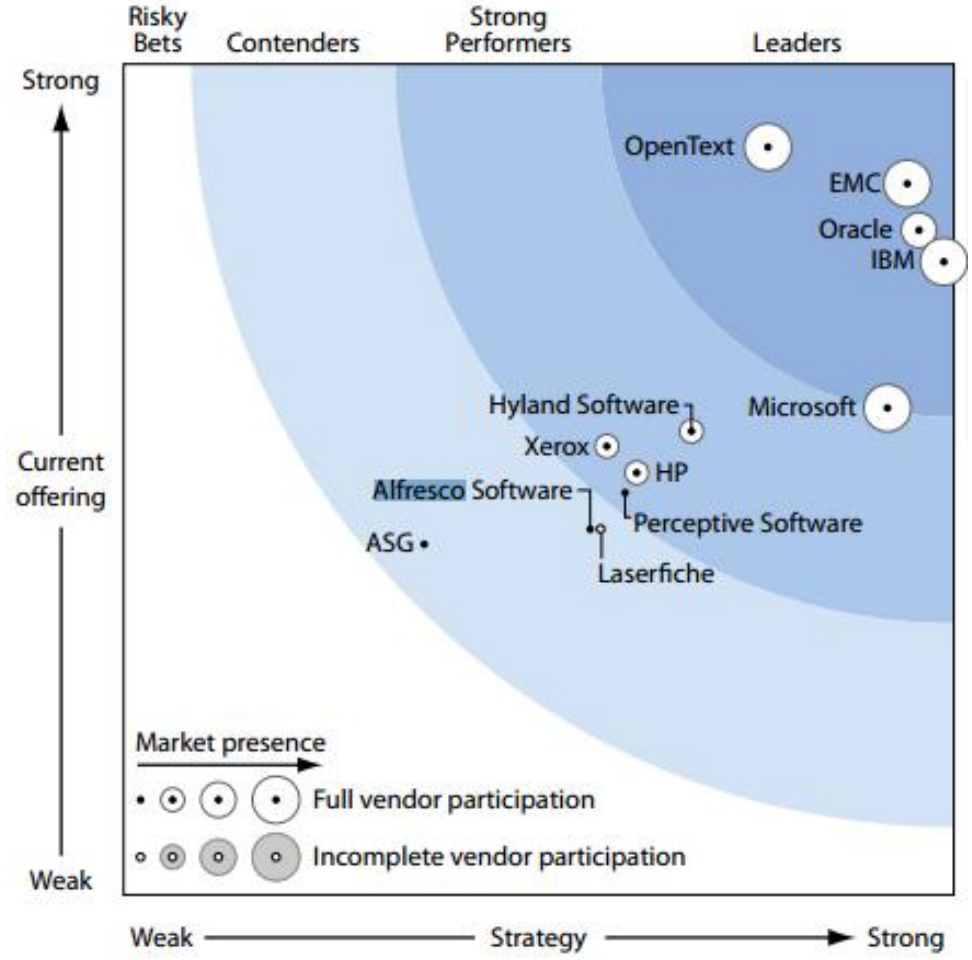
Para tal fin, se tomó como marco de referencia el análisis de dos de las principales compañías de asesoramiento e investigación en tecnología de la información del mundo, como son Gartner Inc. y The Forrester Wave.

De cuyos resultados se desprendió que el ECM ALFRESCO es la única herramienta de código libre que cumple los estándares de calidad evaluados, según lo muestran las figuras 5 y 6.

Figura 16. Cuadrante mágico de Gartner para ECM

Fuente: Gilbert, Shegda, Chin, Tay, Koehler-Kruener, 2012)

Figura 17. Análisis ECM, Q4 2011 The Forrester Wave



Fuente:(Forrester, 2012)

Los componentes evaluados[19] fueron los siguientes:

- La gestión de documentos
- Aplicaciones de procesamiento de imágenes
- Workflow / BPM
- Gestión de contenidos web (WCM)
- Contenido social

Por tanto, se utiliza la plataforma ECM ALFRESCO como medio tecnológico para desplegar el proceso de gestión documental de BANRED, cuya implementación se explicará en el capítulo de construcción.

[19]La descripción de cada componente se la puede revisar en el punto 2.3.1.1 del marco teórico de este proyecto.

3.2.3 Evaluación de interoperatividad entre ECM y aplicativo web

Por pedido expreso de BANRED se desarrollará un portal web con tecnología Java, para simular la integración de la herramienta ECM propuesta, con su intranet. Cuya funcionalidad se centrará en monitorear la interoperabilidad de dichas aplicaciones mediante la exposición y consumo de servicios web.

La herramienta ECM publicará servicios web con información relacionada a la gestión documental y auditorías, y el portal web consumirá dichos servicios para presentar los reportes de gestión solicitados.

Por otra parte, la gestión de usuarios se la realizará con el aplicativo Directorio Activo de Microsoft Windows, cuyas credenciales de autenticación de sistema operativo será utilizada por el portal web y la herramienta ECM mediante el uso del Protocolo Ligero de Acceso a Directorios - LDAP (Lightweight Directory Access Protocol, por sus siglas en inglés).

En resumen, el portal web contará con las siguientes funcionalidades:

3.2.3.1 Interfaz para administración de perfiles de usuario para acceso al portal web

El portal web proporcionará una interfaz que permita al *usuario administrador* del portal, adicionar roles de acceso a los usuarios existentes en el Directorio Activo empresarial.

Dicho rol será de tipo *funcionario y/o administrador, el cual* se almacenará en una base de datos MYSQL.
El rol de tipo *funcionario* únicamente podrá consultar el estado de tareas y trámites gestionados por el portal web.

El rol de tipo *administrador* realizará las siguientes acciones:

- Gestión de roles de acceso al portal

- Parametrización de cadenas de conexión con sistemas externos
- Auditorías sobre la herramienta ECM y portal web

Los ámbitos o contextos de roles de usuario de la plataforma ECM se muestran en el punto 3.2.1 REORGANIZACIÓN DEL PROCESO DE GESTIÓN DOCUMENAL.

3.2.3.2 Interfaz para parametrización de cadenas de conexión con sistemas externos

Dado que la gestión de usuarios de la solución propuesta se realizará a través del Directorio Activo de Windows, El portal web proporcionará una interfaz que permita al usuario *administrador del portal,* registrar los parámetros de conexión entre el portal web y el Directorio Activo. Ésta información se almacenará en un archivo de tipo *properties.*

Dicha interfaz también proporcionará la opción de editar los parámetros de conexión con la herramienta ECM y la base de datos MYSQL, ésta última contendrá información referente a los perfiles de usuario del portal web. La información descrita se almacenará también en un archivo de tipo *properties.*

3.2.3.3 Interfaz para generación de reportes:

Se proporcionará a los usuarios del portal web de una opción para generación de reportes relacionados a la gestión documental para el usuario de tipo *funcionario* y auditorías de la herramienta ECM y portal web para el usuario de tipo *administrador*.

Seguidamente se describen los tipos de reportes existentes por tipo de usuario

Usuario tipo *funcionario:*

Los reportes disponibles para este tipo de usuario estarán vinculados a la gestión de trámites, siendo estos los siguientes:

- Estado del trámite

- Estado del trámite por usuario
- Involucrados en el trámite

El criterio de búsqueda de estos reportes será el campo número de trámite.

También se contará con un reporte de las tareas pendientes de usuario asociadas a los flujos del ciclo de vida de los documentos, las cuales se mostrarán a manera de listado, filtrado por el usuario autenticado en el sistema.

Usuario de tipo *administrador*

Los reportes disponibles para este tipo de usuario son: auditorías del portal web y de la herramienta ECM:

La auditoría del portal web estará asociada a las acciones de creación, modificación y eliminación de usuarios en el portal web.

Mientras que el ECM registrará auditorías de copia, creación, eliminación, modificación y lectura, y actualización de METADATA de documentos.

Los filtros de búsqueda de estos reportes serán: usuario, fecha de inicio y fecha de fin de auditoría.

El detalle de las funcionalidades descritas será mostrado en los casos de uso correspondientes.

3.3 Modelo de dominio

Como se mencionó, la problemática del proceso de gestión documental de BANRED se solventará con la implementación del nuevo proceso de negocio en la plataforma ECM ALFRESCO, herramienta que como se sabe, posee una arquitectura estándar que no puede alterarse, sino que se ajusta a las funcionalidades particulares empresariales, presentando desafíos distintos a los de programación tradicional,

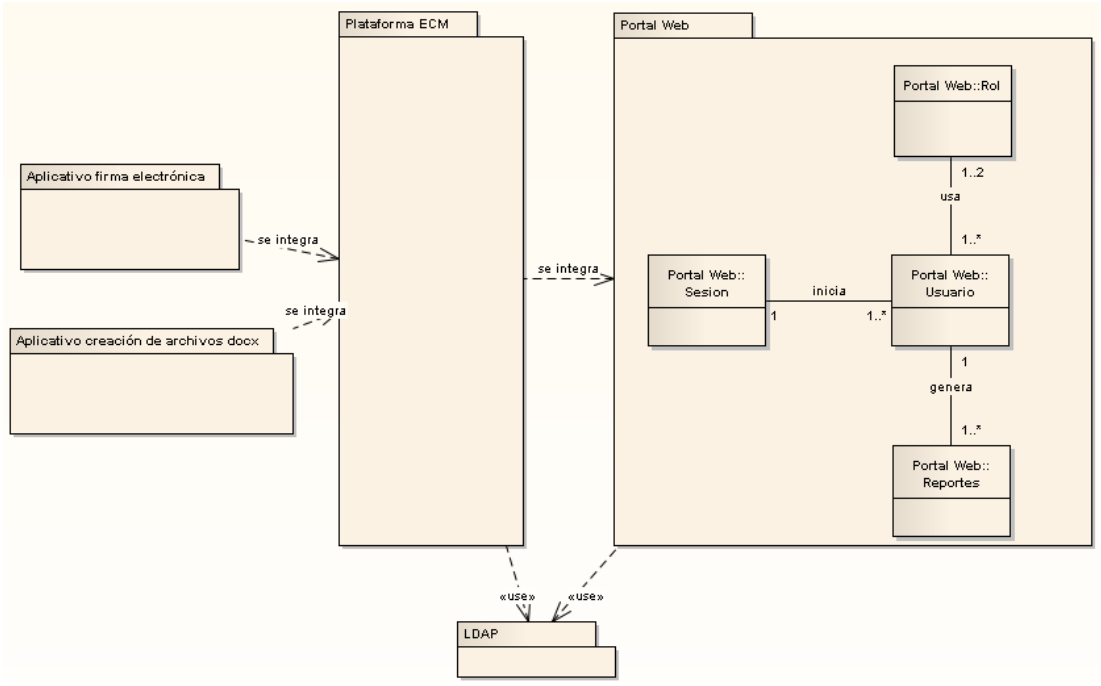
mismos que se explicarán a detalle en el capítulo de construcción y que se indican de forma resumida a continuación:

- Personalización de sus parámetros de configuración (Despliegue de modelo de contenidos[20], motor de transformación y gestión de datos, conexión con aplicaciones externas "portal web y Directorio Activo" y configuración del motor de búsqueda documental).
- Despliegue de los nuevos procesos de negocio.
- Interoperabilidad con otros sistemas mediante la exposición y consumo de servicios web.
- Implementación de nuevas funcionalidades Java en flujos de trabajo, como fueron para este proyecto, el aplicativo de firma electrónica y el componente java para creación de documentos con formato .docx incluido en el proceso de creación de documentos.

Por los motivos expuestos, en los contenidos restantes de este capítulo, se expondrá los modelos del software que ha sido construido; es decir, del portal web, del aplicativo de firma electrónica y del componente de creación de documentos en formato docx, iniciando con el modelo de dominio que se esquematiza a continuación:

[20] Establecimiento de plantillas documentales, para este proyecto se definió las siguientes*: memorandos, oficios, informes, convocatorias y actas de trabajo*.

Figura 18. Modelo de Dominio



Elaborado por: Pedro Caicedo y Diego Godoy

En el anexo (Modelo de Dominio ALFRESCO) se muestra el modelo de dominio a través del cual los tesistas comprendieron el funcionamiento de la plataforma ALFRESCO.

3.4 Modelo de casos de uso

Delante se muestra el modelo de casos de uso del portal web.

3.4.1 Autenticación de usuarios

Como se citó, BANRED utiliza el aplicativo Directorio Activo para normar los recursos de su red LAN, por tanto, la administración de roles de usuario y de unidades organizativas, se seguirán gestionando con esta herramienta.

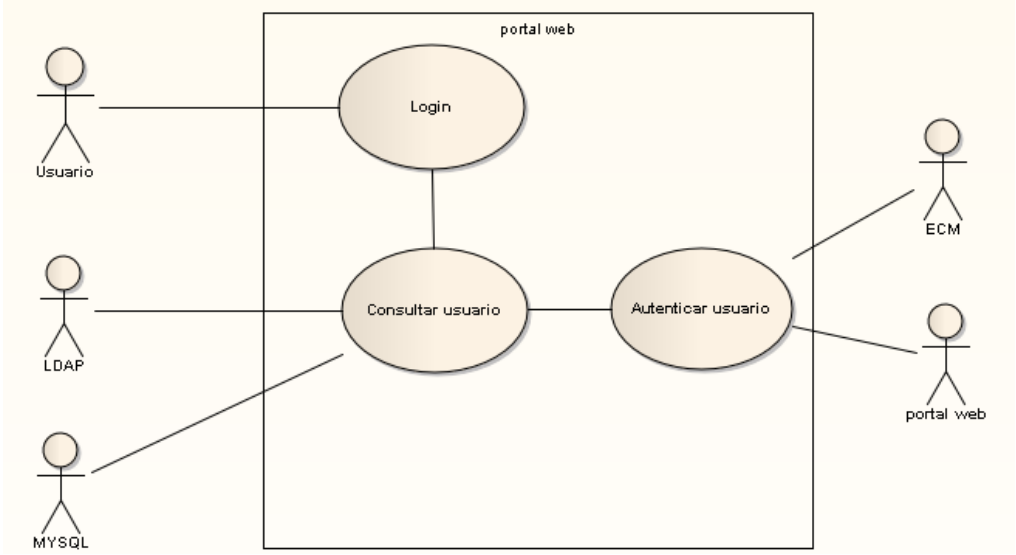
Por consiguiente, se empleará el protocolo LDAP para obtener las credenciales de autenticación de usuario en sistemas operativos Windows y con éstas se realizará el acceso al portal web y a la herramienta ECM.

Cabe destacar que no se ha implementado un proceso Single Sign On[21] para autenticación de usuarios en este proyecto.

El modelo de caso de uso siguiente muestra lo antedicho.

3.4.1.1 Diagrama de caso de uso autenticar usuario

Figura 19. Caso de uso autenticar usuario



Elaborado por: Pedro Caicedo y Diego Godoy

3.4.1.2 Descripción del caso de uso autenticar usuario

Tabla 4. Descripción del caso de uso autenticar usuario

Actor que inicia	Usuario *funcionario o administrador* del portal web
Condiciones previas al CU	La cuenta de usuario del portal web debe poseer el permiso de acceso correspondiente.
Pasos	1) El usuario ingresa las credenciales de autenticación (usuario y contraseña) en el portal web 2) Mediante script, se verifica la existencia de este usuario en Directorio Activo y del permiso de acceso en el portal web 3) El sistema realizará el proceso de autenticación de la cuenta de usuario en portal web y mediante script, también en el EMC
Condiciones posteriores	Usuario con sesión activa en portal web y herramienta ECM
El actor que se beneficia del CU	Usuarios

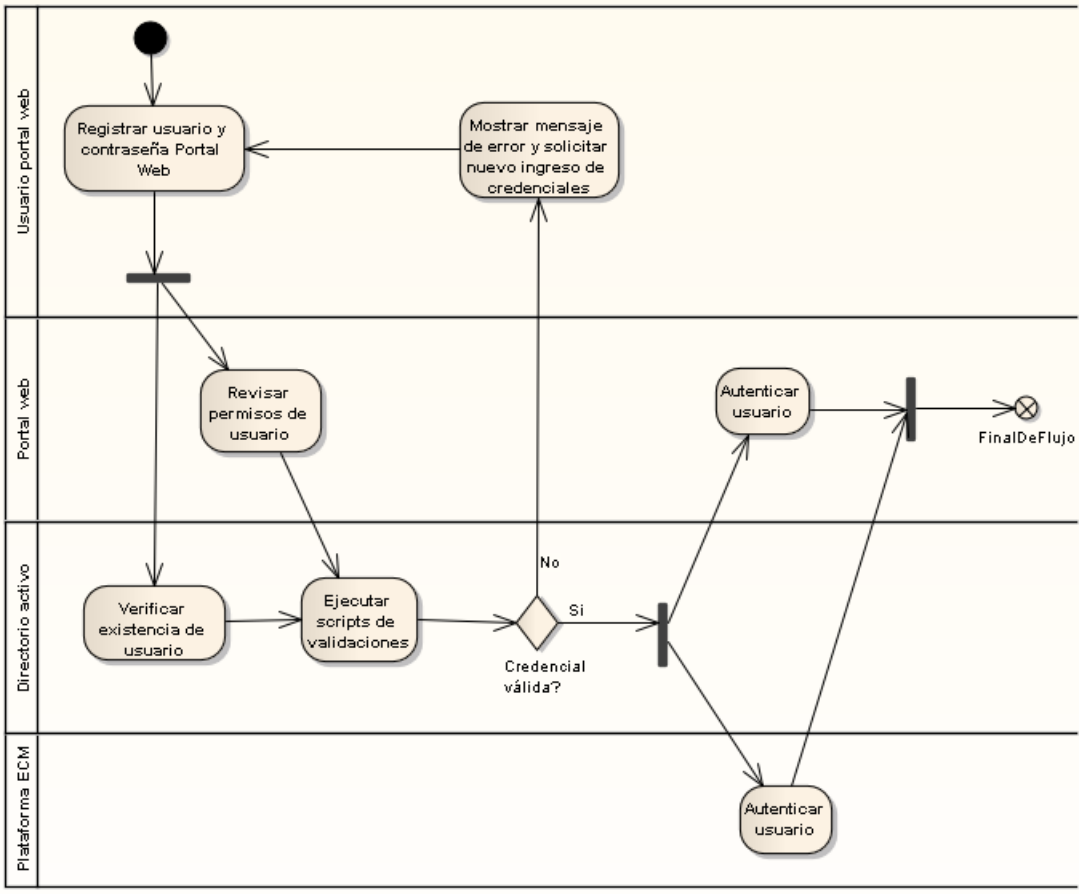
[21] Procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación. Su traducción literal sería "sistema centralizado de autenticación y autorización".

Excepción	- El sistema mostrará un mensaje de error al usuario en caso que las credenciales de autenticación ingresadas sean inválidas o incorrectas y negará su acceso al portal - El sistema mostrará un mensaje de notificación al usuario en caso que éste no posea permisos de acceso del portal web y negará su acceso al portal

Elaborado por: Pedro Caicedo y Diego Godoy

3.4.1.3 Diagrama de actividades del caso de uso autenticar usuario

Figura 20. Actividades del caso de uso autenticar usuario



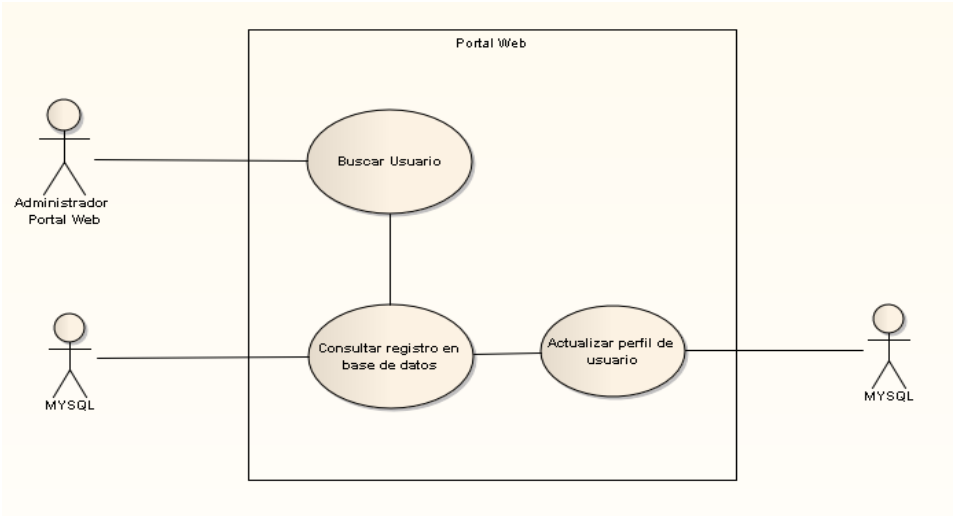
Elaborado por: Pedro Caicedo y Diego Godoy

3.4.2 Gestión de roles de usuario portal web

Seguidamente se muestran los casos de uso necesarios para la gestión de roles de usuarios del portal web.

3.4.2.1 Diagrama de casos de uso gestión de roles usuario portal web

Figura 21. Casos de uso gestión de roles usuario portal web



Elaborado por: Pedro Caicedo y Diego Godoy

3.4.2.2 Descripción del caso de uso gestión de roles de usuario portal web

Tabla 5. Descripción del caso de uso gestión de roles de usuario portal web

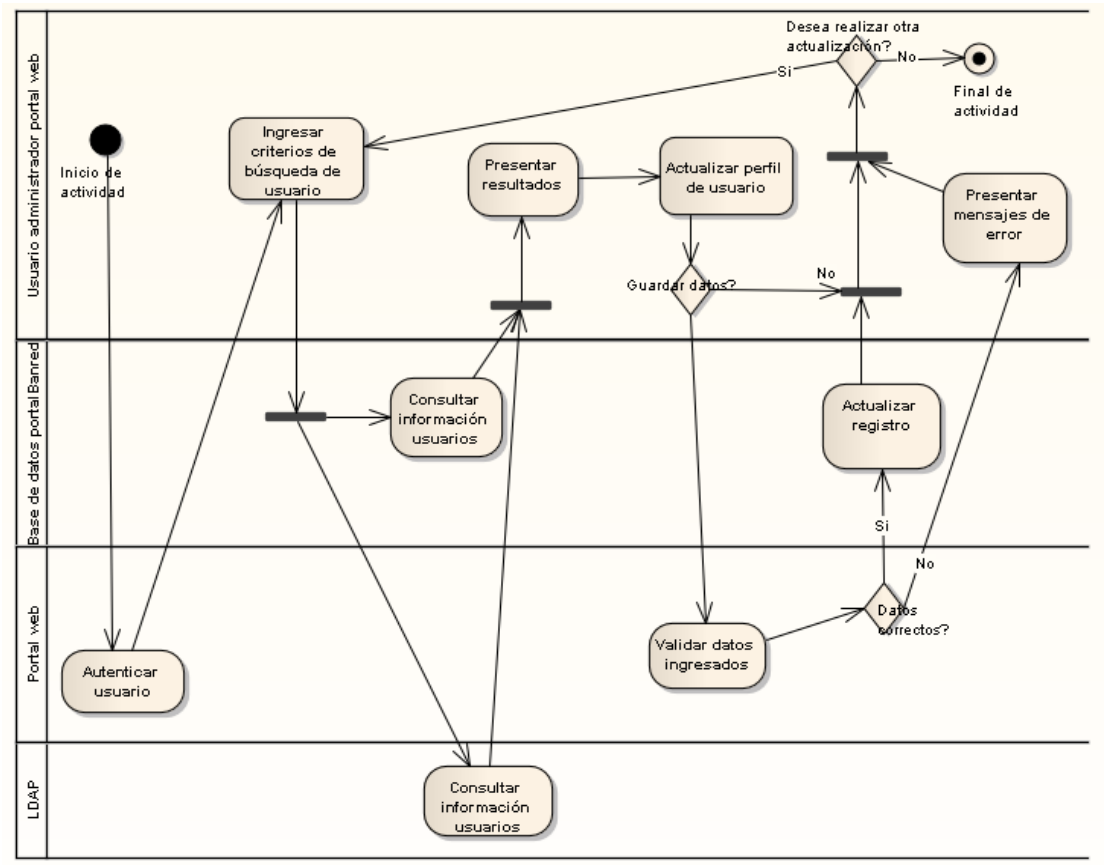
Actor que inicia	Usuario *administrador* del portal web
Condiciones previas al CU	-El usuario *administrador* del portal deberá tener una sesión activa en el portal web - La edición de perfiles de usuario se realizará sobre usuarios registrados previamente en Directorio Activo de Windows - El usuario que ejecuta los cambios deberá poseer un perfil de *administrador* de portal web
Pasos	1) El *administrador* escoge la opción "Administración de usuarios del portal" 2) El *administrador* ingresa el criterio de búsqueda del usuario requerido por nombre, por cédula o por defecto (muestra todos los usuarios existentes en LDAP), la búsqueda se realiza en LDAP y base datos portalBanred 2.1) LDAP devuelve reporte con los campos siguientes: nombre, cédula, nombre LDAP (clave primaria), estado (activo o inactivo) y nacionalidad 2.2) Se ejecuta verificación de la cédula de usuario almacenada en LDAP y muestra de forma gráfica el resultado (correcto o incorrecto) 2.3) La base de datos portalBanred devuelve el dato del rol de usuario *administrador o funcionario* 3) Se presenta al usuario un reporte unificado tanto de la base de datos portalBanred como del LDAP 4)El *administrador* podrá realizar edición sobre los siguientes campos: cédula y tipo de usuario (*funcionario o administrador*) 5) Previo a la actualización del registro en la base de datos, se realizará una validación de los campos nacionalidad y cédula y nombre (primera letra capital) 6) Se almacena la nueva información en la base de datos portalBanred
Condiciones posteriores	Cuenta de usuario del portal web con perfil de acceso actualizado

El actor que se beneficia del CU	Base de datos portalBanred
Excepción	- En caso que el campo nacionalidad no sea “ecuatoriano”, no se ejecutará la validación de dígito verificar aplicado a los números de cédula ecuatorianos - Si el número de cédula es incorrecto no se desplegará un mensaje el usuario y no se registrará o actualizará el registro en la base de datos portalBanred

Elaborado por: Pedro Caicedo y Diego Godoy

3.4.2.3 Diagrama de actividades del caso de uso gestión de roles de usuario en portal **web**

Figura 22. Diagrama de actividades del caso de uso gestión de roles de usuario en portal web



Elaborado por: Pedro Caicedo y Diego Godoy

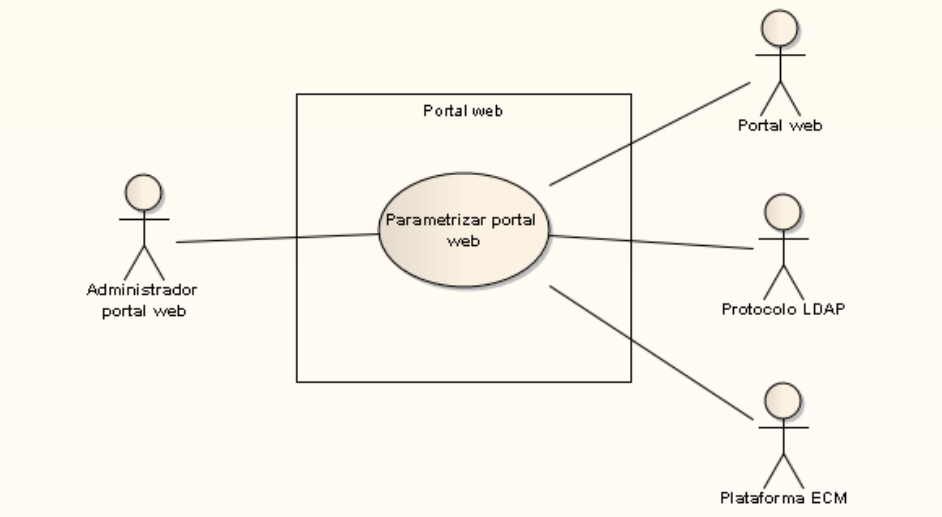
3.4.3 Parametrización portal web

La parametrización del portal web se realizará mediante archivos de tipo *properties*, los cuales permitirán al usuario administrador configurar lo siguiente: conexión a la base de datos del portal, al protocolo LDAP para administración de usuarios

mediante DIRECTORIO ACTIVO y con la herramienta ECM para la gestión documental.

3.4.3.1 Diagrama de casos de uso parametrización portal web

Figura 23. Diagrama de casos de uso parametrización portal web



Elaborado por: Pedro Caicedo y Diego Godoy

3.4.3.2 Descripción del caso de uso parametrización portal web

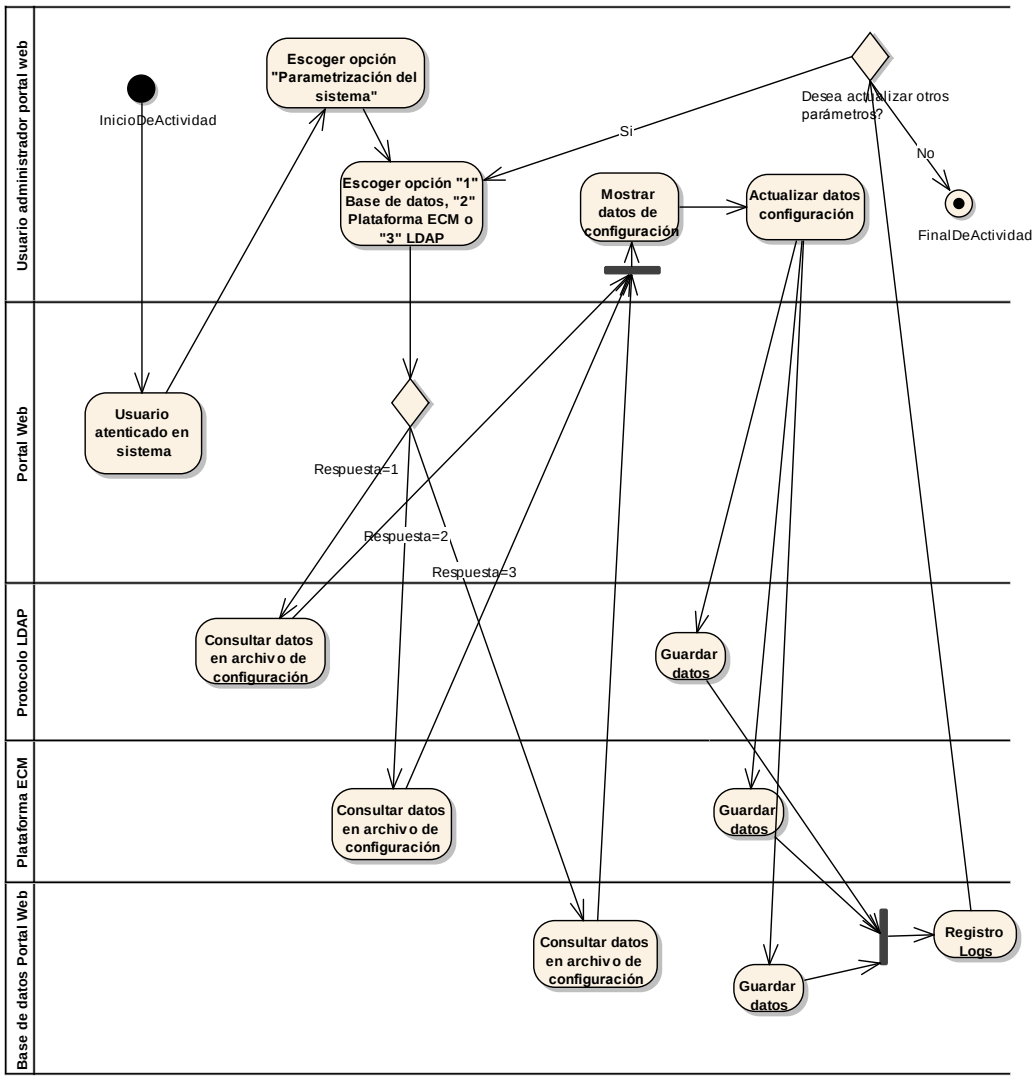
Tabla 6. Descripción del caso de uso parametrización portal web

Actor que inicia	Usuario *administrador* del portal web
Condiciones previas al CU	- El usuario *administrador* del portal deberá tener una sesión activa en el portal web - El usuario que ejecuta los cambios deberá poseer un perfil de *administrador* de portal web
Pasos	1)El *administrador* escoge la opción "Parametrización del sistema" 2) El *administrador* selecciona una de las opciones de parametrización siguientes: "Base de datos", "plataforma ECM" o "LDAP" 3) El usuario realiza la actualización de datos, según corresponda 4) El sistema actualiza los cambios respectivos en los archivos *properties* de la plataforma ECM, el portal web y LDAP 4) El sistema graba los cambios en logs
Condiciones posteriores	Parámetros de configuración del portal web actualizados
El actor que se beneficia del CU	portal web, ECM y LPAP
Excepción	En caso que el usuario administrador ingrese datos erróneos en los archivos *properties*, al reiniciar las aplicaciones surtirán efecto los cambios y no se conecatarán los aplicativos.

Elaborado por: Pedro Caicedo y Diego Godoy

3.4.3.3 Diagrama de actividades del caso de uso parametrización portal web

Figura 24. Diagrama de actividades, caso uso parametrización portal web



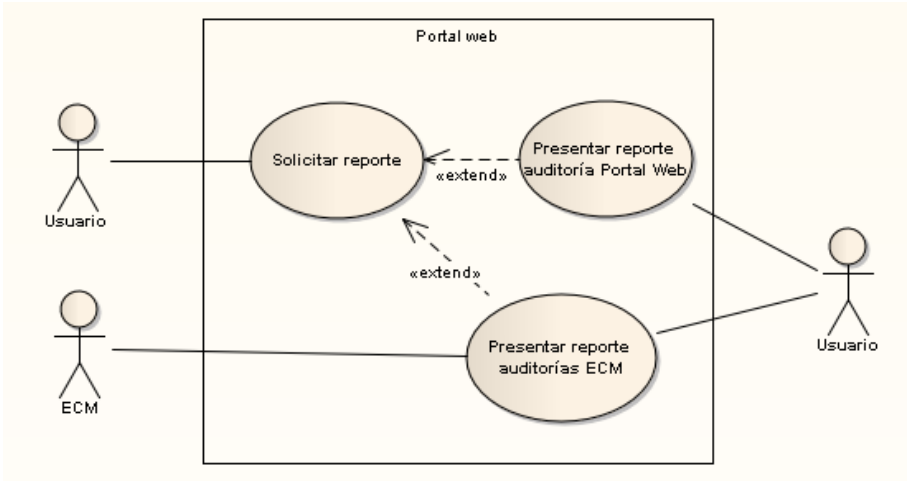
Elaborado por: Pedro Caicedo y Diego Godoy

3.4.4 Gestión de reportes

La usabilidad requerida para los reportes de la herramienta ECM mostrada a través del portal web, así como ciertas acciones de auditoría es mostrada a continuación:

3.4.4.1 Diagrama de casos de uso gestión de reportes

Figura 25. Diagrama de casos de uso gestión de reportes



Fuente: Tesistas.

3.4.4.2 Descripción del caso de uso gestión de reportes

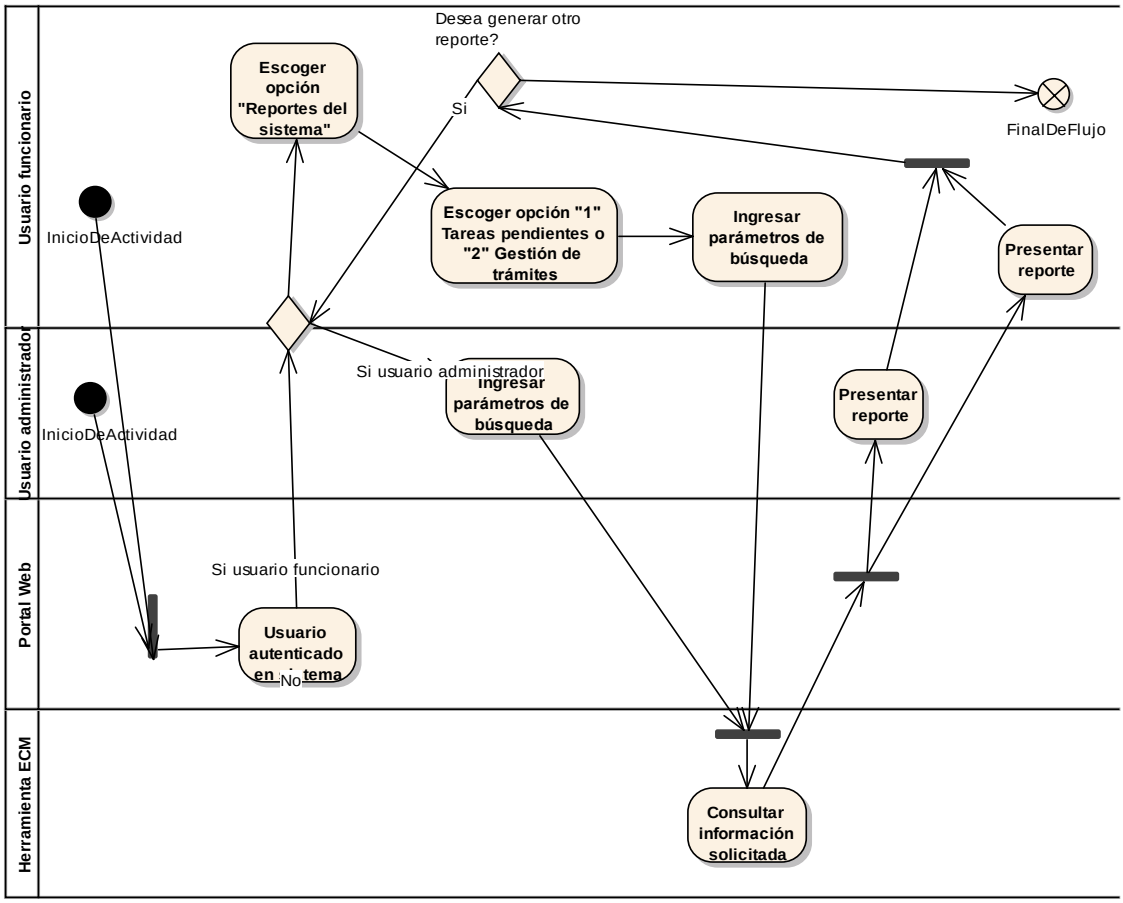
Tabla 7. Descripción del caso de uso gestión de reportes

Actor que inicia	Usuario *funcionario* o *administrador* portal web
Condiciones previas al CU	El usuario del portal deberá tener una sesión activa en el portal web
Pasos	1) El usuario escoge la opción "reportes del Sistema" 2) El usuario escoge una de las siguientes opciones de reporte: "Tareas pendientes" o "Gestión de trámites" 3) El usuario ingresa criterios de búsqueda 4) El portal web envía petición de información a la herramienta ECM. 5) La herramienta ECM envía datos solicitados 6)El sistema presenta el reporte solicitado
Condiciones posteriores	El usuario podrá visualizar el reporte solicitado
El actor que se beneficia del CU	Todos los usuarios del portal web
Excepción	Los tipo de reporte estarán asociados al tipo de usuario

Elaborado por: Pedro Caicedo y Diego Godoy

3.4.4.3 Diagrama de actividades del caso de uso gestión de reportes

Figura 26. Diagrama de actividades del caso de uso gestión de reportes



Elaborado por: Pedro Caicedo y Diego Godoy

Respecto al proceso de negocio del componente de creación de documentos en formato .docx y de firma electrónica, estos pueden ser vistos en las referencias siguientes:

- Punto 3.2.1.1: Creación de archivos con formato docx
- Punto 3.2.1.1.1: Llenar METADATOS desde convocatoria
- Punto 3.2.1.1.2: Registrar METADATOS en docx
- Punto 3.2.1.2: Firmar Digitalmente

3.5 Modelo de diseño

Seguidamente se muestra el modelo de diseño del portal web, el aplicativo de firma electrónica y el componente de creación de archivos en formato .docx.

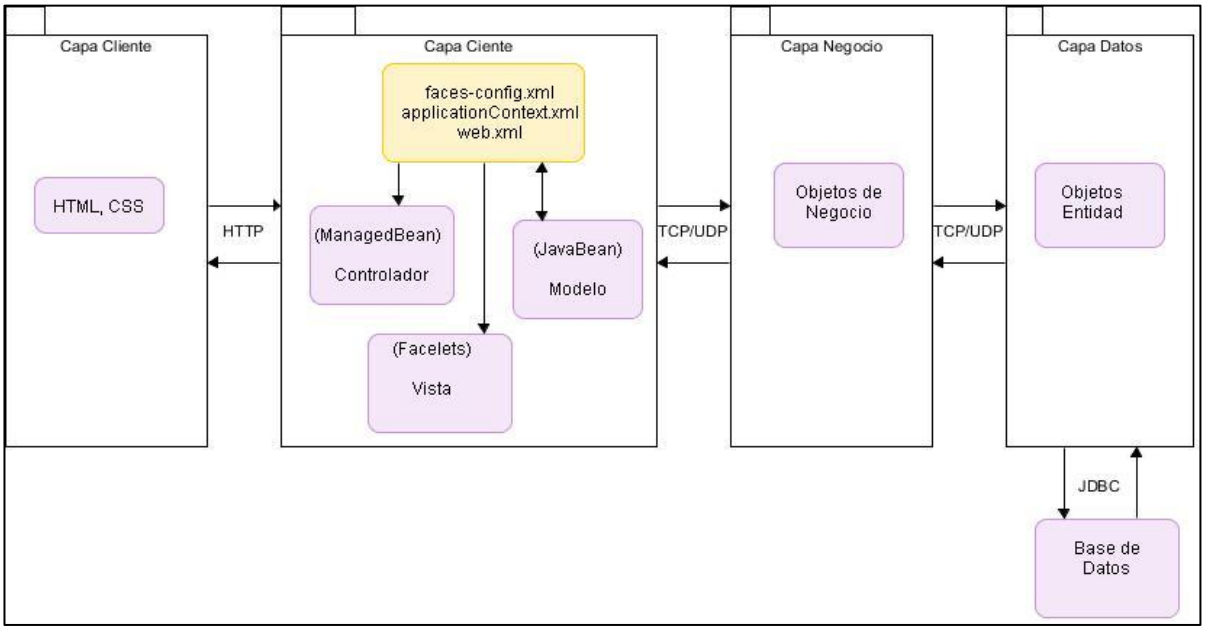
3.5.1 Modelo de diseño del portal web

Este modelo consta de los siguientes elementos: arquitectura, diagrama de paquetes, diagrama de clases, y, modelo lógico y físico de la base de datos del portal web, cuyo detalle es mostado a continuación:

3.5.1.1 Arquitectura del portal web

En la figura No 28, mostrada a continuación se detalla la arquitectura en capas definida para el portal web:

Figura 27. Arquitectura del portal web

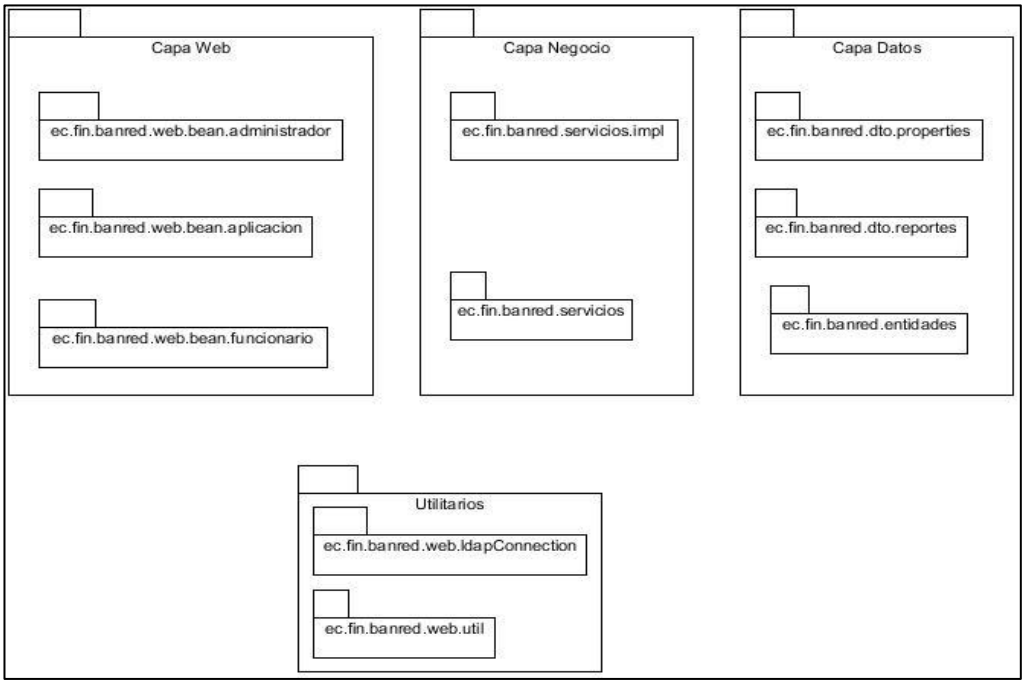


Elaborado por: Pedro Caicedo y Diego Godoy

3.5.1.2 Diagrama de paquetes portal web

En seguida se detalla los paquetes de la lógica de las capas de la aplicación web y sus clases utilitarias.

Figura 28. Paquetes portal web



Elaborado por: Pedro Caicedo y Diego Godoy

3.5.1.3 Diagramas de clases portal web

En función a lo indicado en la figura No. 29, a continuación se describen las clases correpondientes a las capa de la aplicación web, así como de las clases utilitarias.

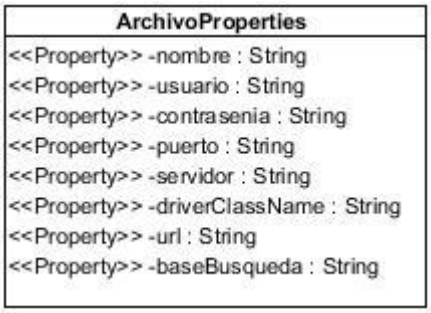
- **Capa de datos**

En esta capa se encuentran las clases tipo entidad Hibernate y DTO's "Data Transfer Object", donde las clases entidad representan tablas de una base de datos relacional y las DTO's se utilizan como plantillas para el manejo de los servicios web tipo REST.

- **Paquete ec.fin.banred.dto.properties**

La clase *ArchivoProperties* sirve como plantilla para el manejo de los archivos properties: ecm.properties, ldap.properties

Figura 29. Clase DTO *ArchivoProperties*

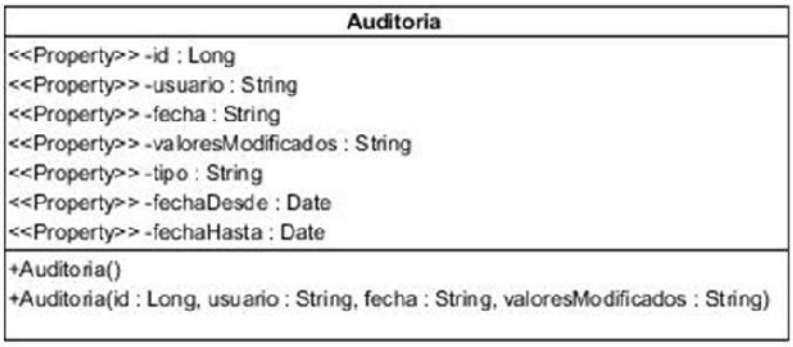


Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete ec.fin.banred.dto.reportes**

La clase *Auditoria* sirve de plantilla para el manejo de las auditorías del ECM que se obtiene por el servicio web REST

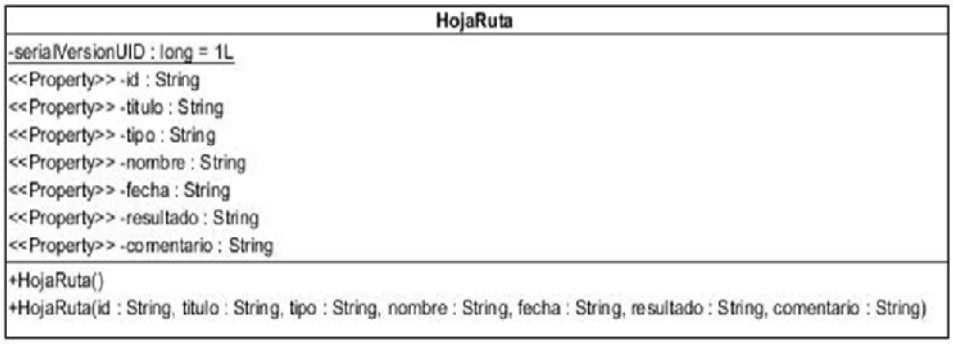
Figura 30. Clase DTO *Auditoria*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *HojaRuta* sirve como plantilla para el manejo de los antecedentes de un trámite del ECM que se obtiene por el servicio web REST

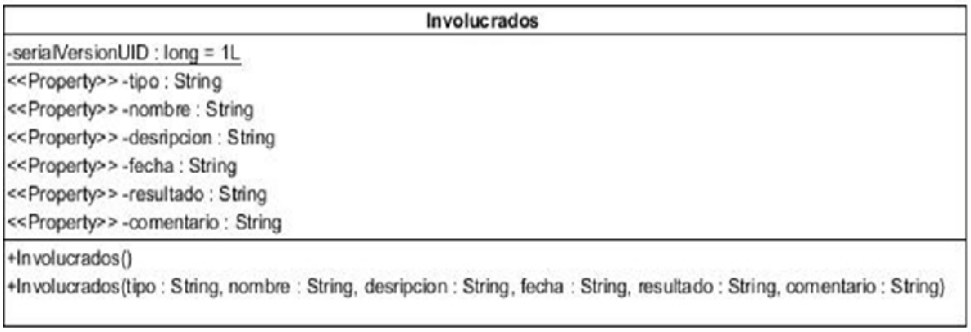
Figura 31. Clase DTO *HojaRuta*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *Involucrados* sirve de plantilla para la gestión de los involucrados de un trámite del ECM que se obtiene por el servicio web REST

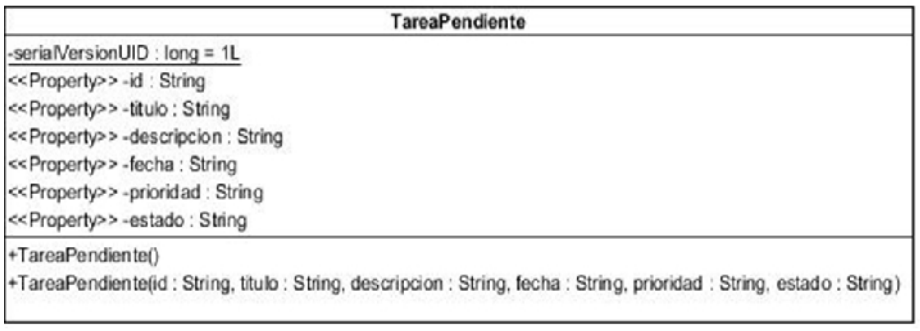
Figura 32. Clase DTO *Involucrados*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *TareaPendiente* sirve de plantilla para el manejo de tareas pendientes de un usuario en el ECM que se obtiene por el servicio web REST

Figura 33. Clase DTO *TareaPendiente*

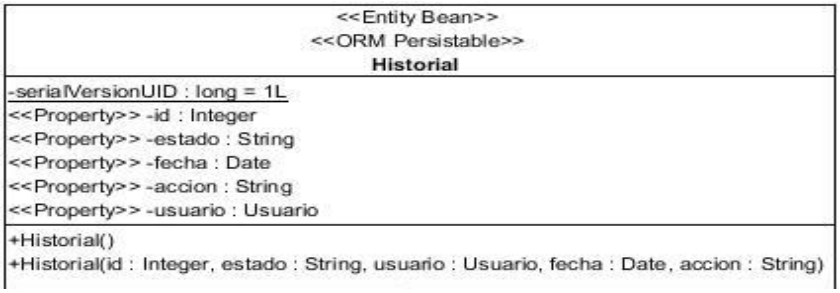


Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete ec.fin.banred.entidades**

La clase *Historial* es una entidad que representa la tabla Historial de la base de datos, sirve para el manejo de las transacciones con la base de datos

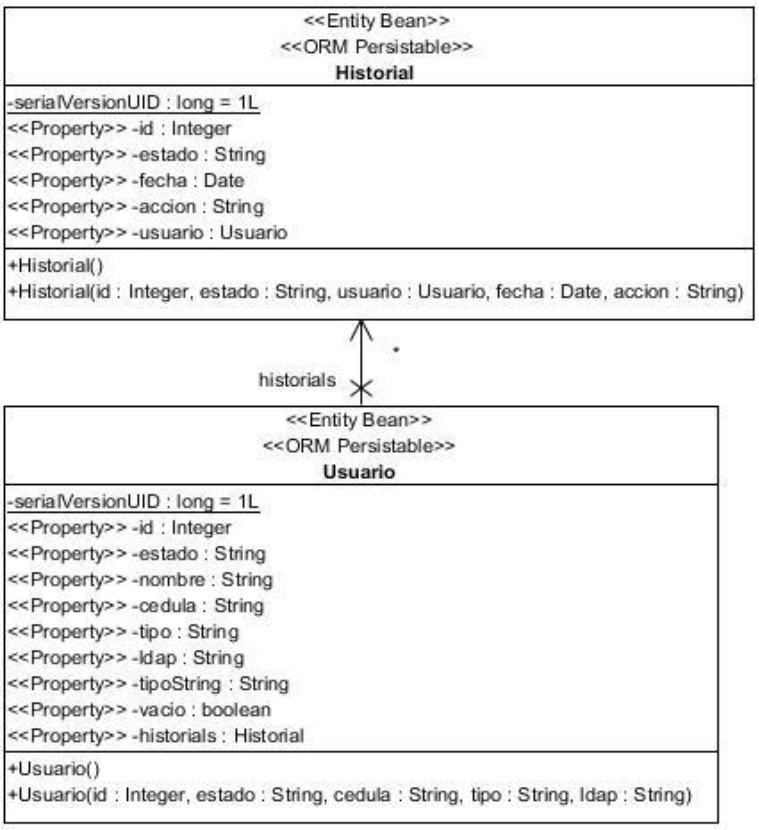
Figura 34. Clase Entidad *Historial*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *Usuario* es una entidad que representa la tabla *Usuario* de la base de datos, sirve para el manejo de las transacciones con la base de datos

Figura 35. Clase Entidad *Usuario*



Elaborado por: Pedro Caicedo y Diego Godoy

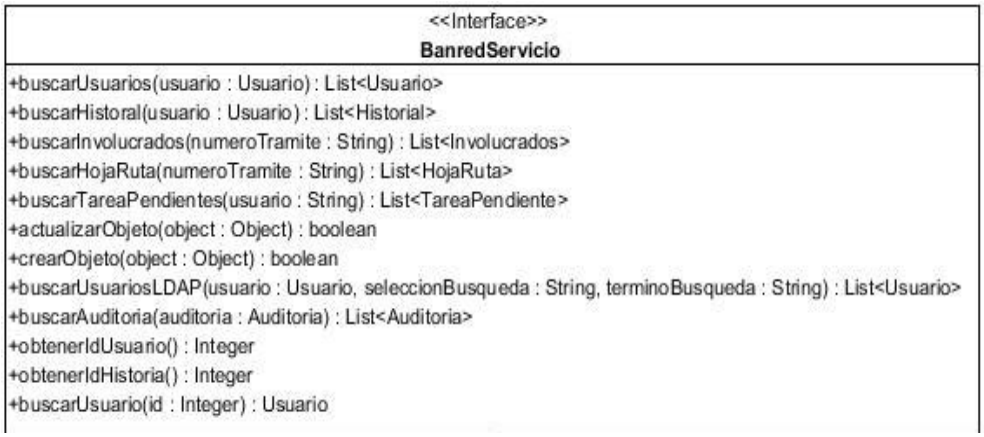
- **Capa de negocio**

En esta capa se describe la lógica del negocio, en general encontraremos operaciones con los orígenes de datos (bases de datos y servicios web).

- **Paquete ec.fin.banred.servicios y ec.fin.banred.servicios.impl**

La interface *BanredServicio* sirve para publicar los métodos que serán vistos de la capa de servicio.

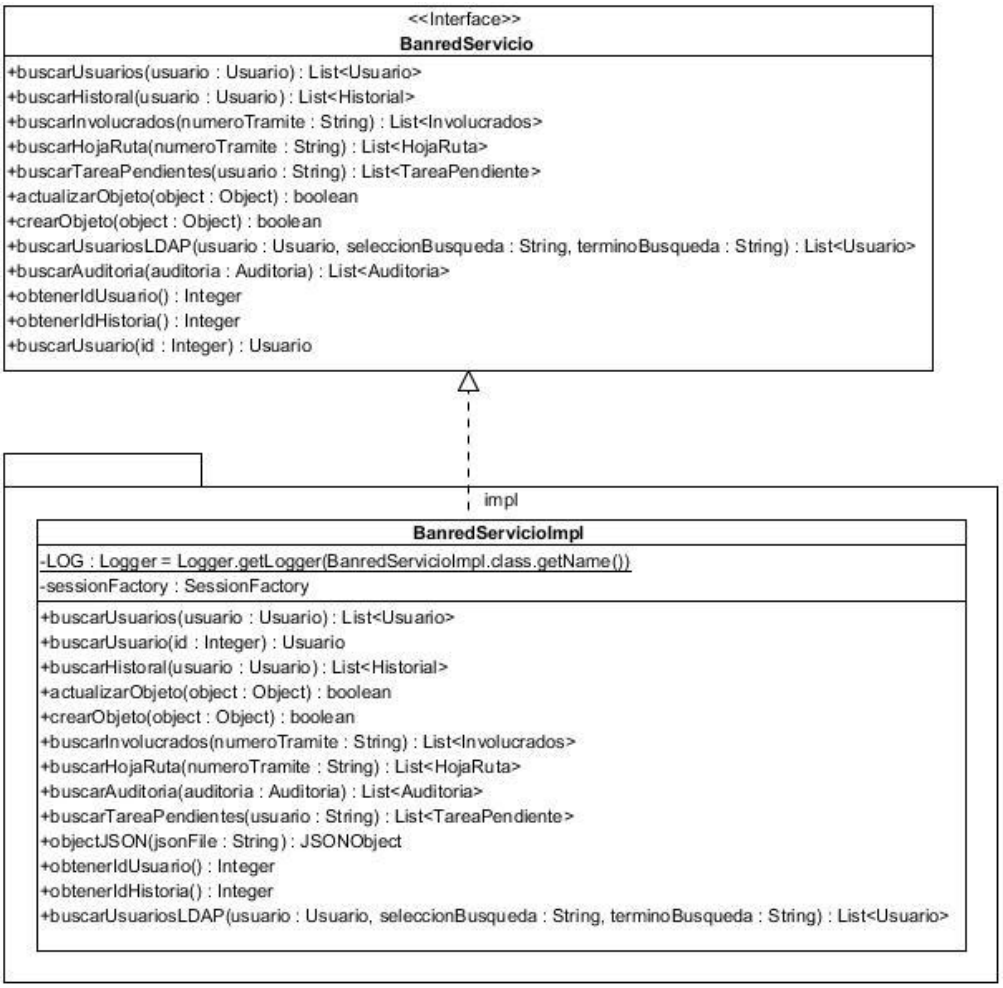
Figura 36. *Interface Servicio*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *BanredServicioImpl* implementa la interface *BanredServicio* y contiene toda la implementación de los métodos publicados en la interface.

Figura 37. Clase *BanredServicioImpl*



Elaborado por: Pedro Caicedo y Diego Godoy

- **Capa de aplicación**

Esta capa se divide en dos secciones, la parte del controlador y la de presentación (páginas web), en el capítulo de construcción se especificará de forma más detallada este tema.

- **Paquete ec.fin.banred.web.bean.administrador**

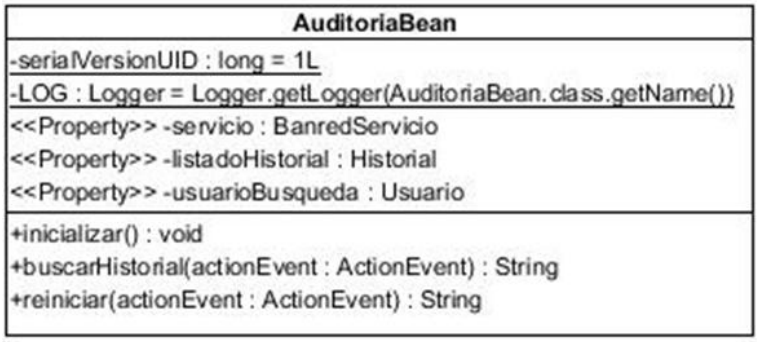
El bean *AuditoriaECMBean* es el controlador de la página que administra la información sobre auditoría del ECM.

Figura 38. Clase Bean *AuditoriaECMBean*

Elaborado por: Pedro Caicedo y Diego Godoy

El bean *AuditoriaBean* es el controlador de la página que administra la información sobre auditoría de la base de datos local.

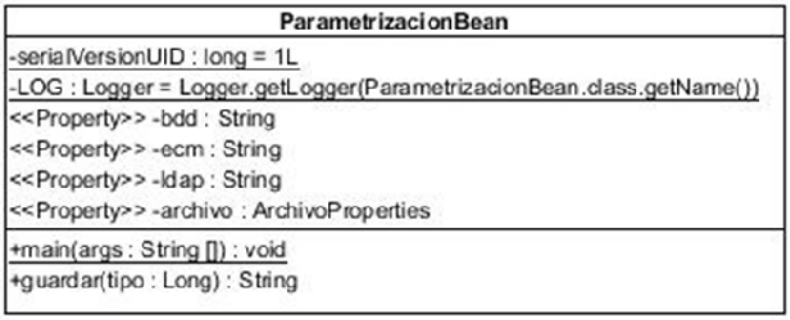
Figura 38. Clase Bean *AuditoriaBean*



Elaborado por: Pedro Caicedo y Diego Godoy

El bean *ParametrizacionBean* es el controlador de la página que administra la información sobre la edición de los campos de los properties: ecm.properties, ldap.properties.

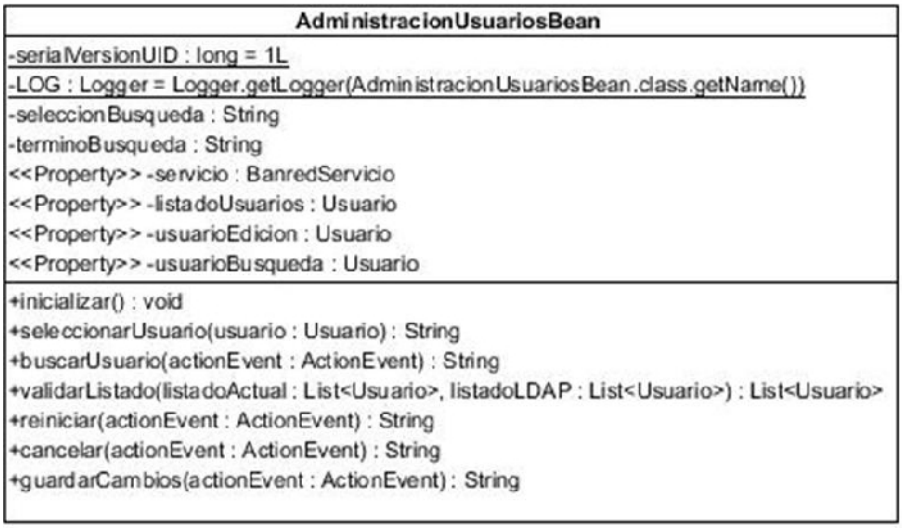
Figura 39. Clase Bean *ParametrizacionBean*



Elaborado por: Pedro Caicedo y Diego Godoy

El bean *AdministracionUsuariosBean* es el controlador de la página que administra la asignación de un rol para un usuario del sistema en su totalidad.

Figura 40. Clase Bean *AdministracionUsuariosBean*

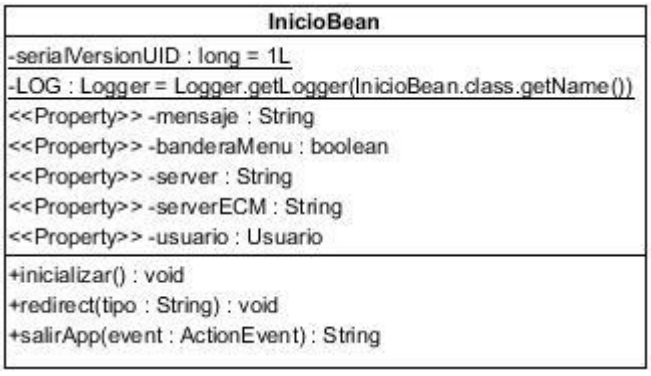


Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete ec.fin.banred.web.bean.aplicacion**

El bean *InicioBean* es el controlador de la página que administra la información sobre el manejo de la sesión de la aplicación, login, logout.

Figura 41. Clase Bean *InicioBean*

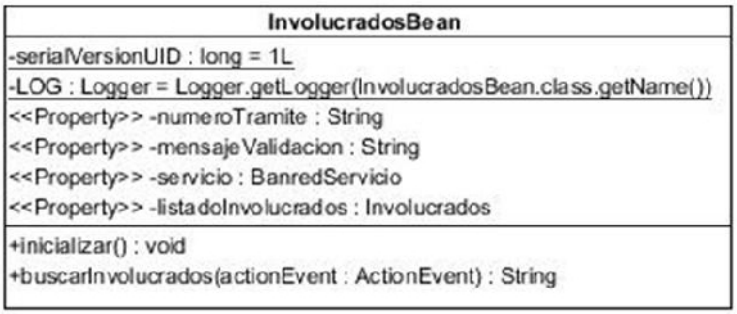


Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete ec.fin.banred.web.bean.funcionario**

El bean *InvolucradosBean* es el controlador de la página que administra la información sobre Involucrados de un trámite que se encuentra en el ECM.

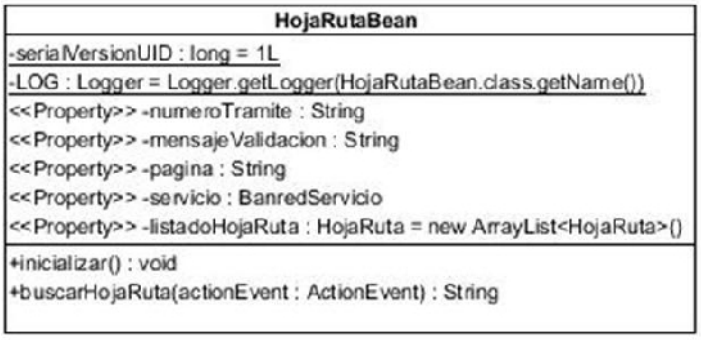
Figura 42. Clase Bean *InvolucradosBean*



Elaborado por: Pedro Caicedo y Diego Godoy

El bean *HojaRutaBean* es el controlador de la página que administra la información sobre la hoja ruta de un trámite que se encuentra en el ECM.

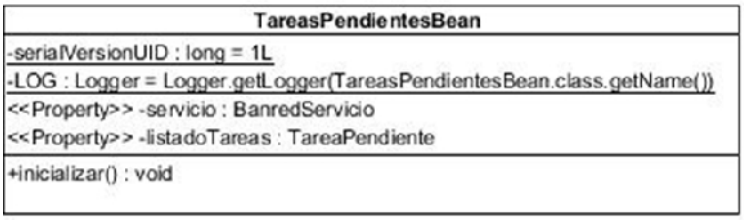
Figura 43. Clase Bean *HojaRutaBean*



Elaborado por: Pedro Caicedo y Diego Godoy

El bean *TareasPendientesBean* es el controlador de la página que administra la información sobre las tareas pendientes de un funcionario.

Figura 44. Clase Bean *TareasPendientesBean*



Elaborado por: Pedro Caicedo y Diego Godoy

El bean *EstadoTramiteBean* es el controlador de la página que administra la visualización del estado de un trámite específico que se encuentra en el ECM

Figura 45. Clase Bean *EstadoTramiteBean*

Elaborado por: Pedro Caicedo y Diego Godoy

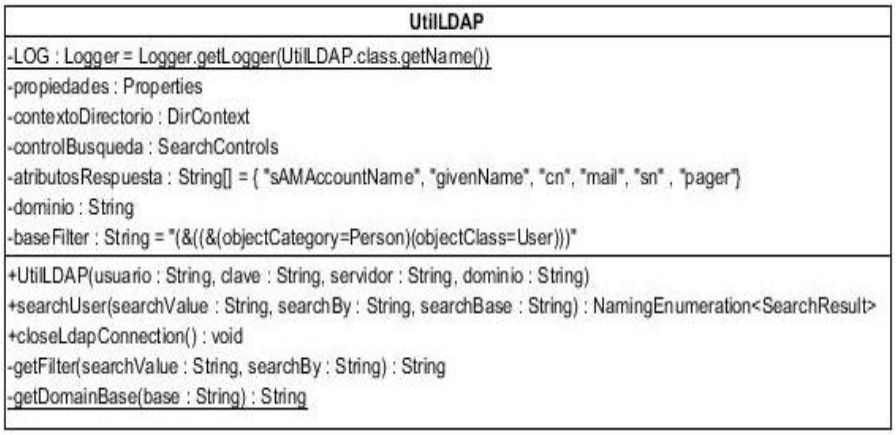
- **Utilitarios**

En esta sección se considerará los métodos y constantes que se utilizan en toda la aplicación, como son configuración de las llamadas hacia los servicios web, credenciales de conexión al LDAP, credenciales de conexión al ECM, URL's que especifican los servicios web y path's o direcciones de los archivos properties para la parametrización.

- **Paquete ec.fin.banred.web.ldapConnection**

La clase *Utilldap* contiene métodos que permiten la conexión con el Directorio Activo, así como parámetros de configuración con éste.

Figura 46. Clase POJO *Utilldap*

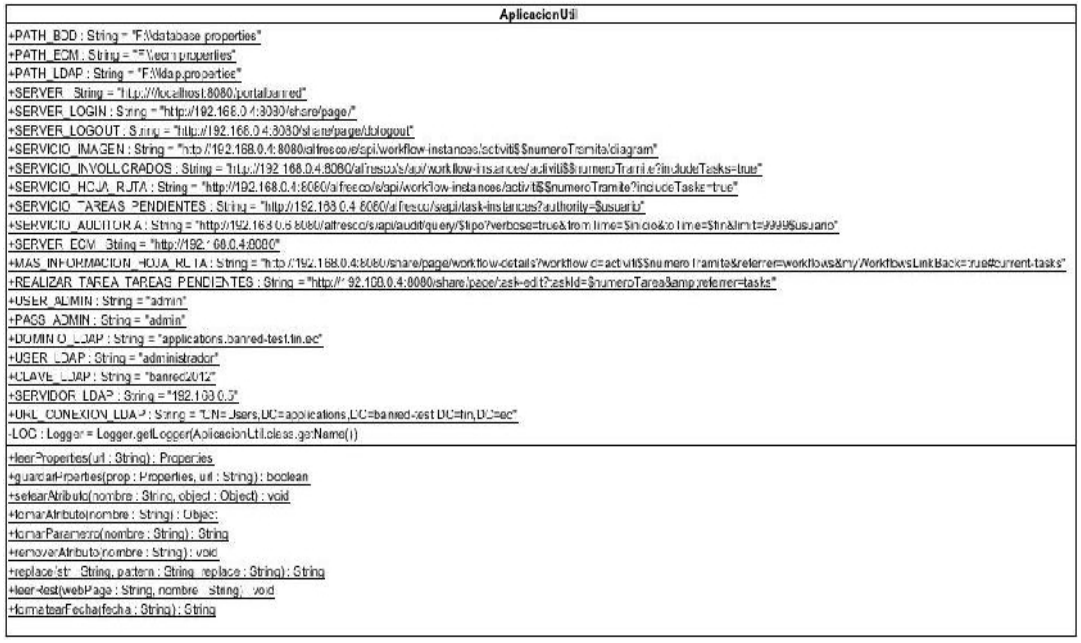


Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete ec.fin.banred.web.util**

La clase *AplicacionUtil* contiene métodos utilitarios que se requieren en la aplicación web para un correcto funcionamiento, así como parámetros de configuración.

Figura 47. Clase POJO *AplicacionUtil*



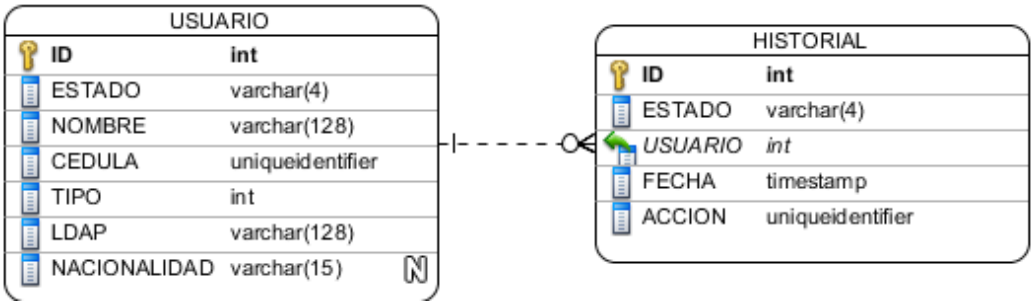
Elaborado por: Pedro Caicedo y Diego Godoy

3.5.1.4 Modelo lógico portal web

El modelo de base de datos se basa en el modelo de dominio del sistema (ver punto 3.3). Los campos establecidos se relacionan al rol de usuarios del portal web y las auditorías sobre dichos roles.

Los campos corresponden a los campos de Directorio Activo a excepción de TIPO, que almacena precisamente el tipo de rol del portal web. Donde 1= administrador y 2= funcionario. En la figura siguiente se muestra el modelo lógico de base de datos del portal:

Figura 48. Modelo lógico base de datos *portalBanred*



Elaborado por: Pedro Caicedo y Diego Godoy

3.5.1.5 Modelo físico portal web

En el diagrama siguiente se muestra el modelo físico obtenido:

Figura 49. Modelo físico base de datos *portalBanred*

```
drop table if exists USUARIO;
create table USUARIO (
        ID      int(10) not null auto_increment,
        ESTADO varchar(4) not null,
        NOMBRE varchar(128) not null,
key (ID));
alter table HISTORIAL drop foreign key FKHISTORIAL396145;
drop table if exists HISTORIAL;
create table HISTORIAL (
        ID      int(10) not null auto_increment,
        ESTADO  varchar(4) not null,
        USUARIO int(10) not null,
         FECHA   timestamp not null,
        ACCION  varchar(16) not null,
        primary key (ID));
alter table HISTORIAL add index FKHISTORIAL396145 (USUARIO),
add constraint FKHISTORIAL396145
foreign key (USUARIO) references USUARIO (ID);
```

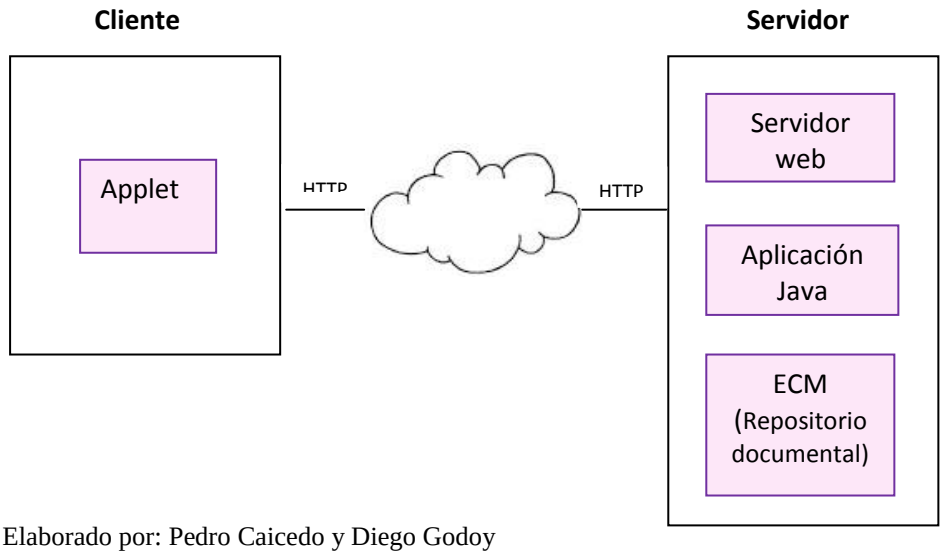
Elaborado por: Pedro Caicedo y Diego Godoy

3.5.2 Modelo de diseño del aplicativo de firma electrónica

Este modelo consta de los siguientes elementos: arquitectura, diagrama de paquetes y diagrama de clases, cuyo detalle es mostado a continuación:

3.5.2.1 Arquitectura aplicativo de firma electrónica

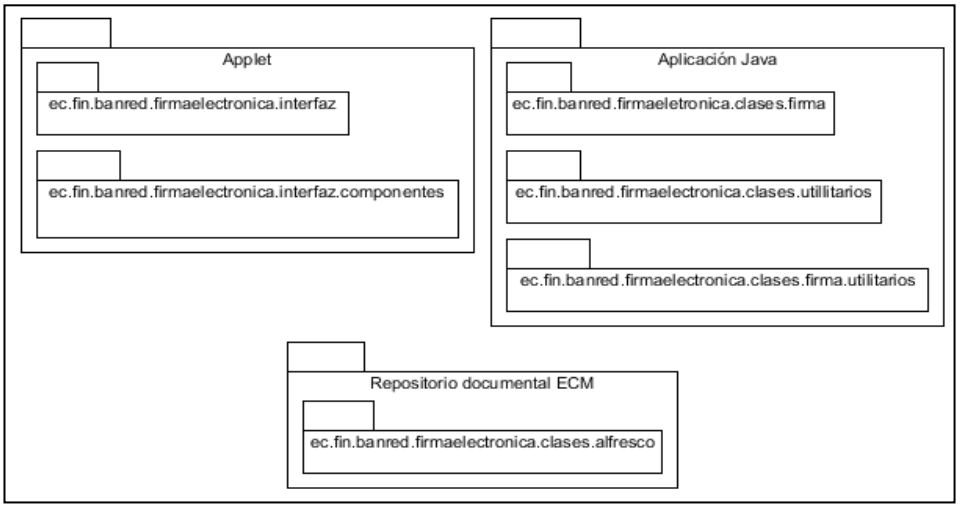
Figura 50. Arquitectura del aplicativo de firma electrónica



Elaborado por: Pedro Caicedo y Diego Godoy

3.5.2.2 Diagrama de paquetes aplicativo firma electrónica

Figura 51. Paquetes del aplicativo de firma electrónica



Elaborado por: Pedro Caicedo y Diego Godoy

3.5.2.3 Diagrama de clases aplicativo firma electrónica

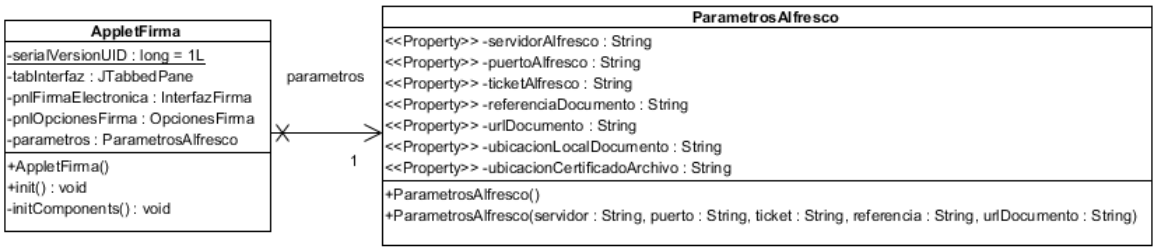
A continuación se describen las clases correpondientes a cada uno de los paquetes referidos en la figura No. 51.

- **Paquete Applet**

- **Paquete ec.fin.banred.firmaelectrónica.interfaz**

La clase *AppletFirma* contiene la ventana principal que contiene los diferentes componentes de la interfaz visual del aplicativo de firma.

Figura 52. Applet *AppletFirma* con llamada a clase POJO *ParámetrosAlfresco*

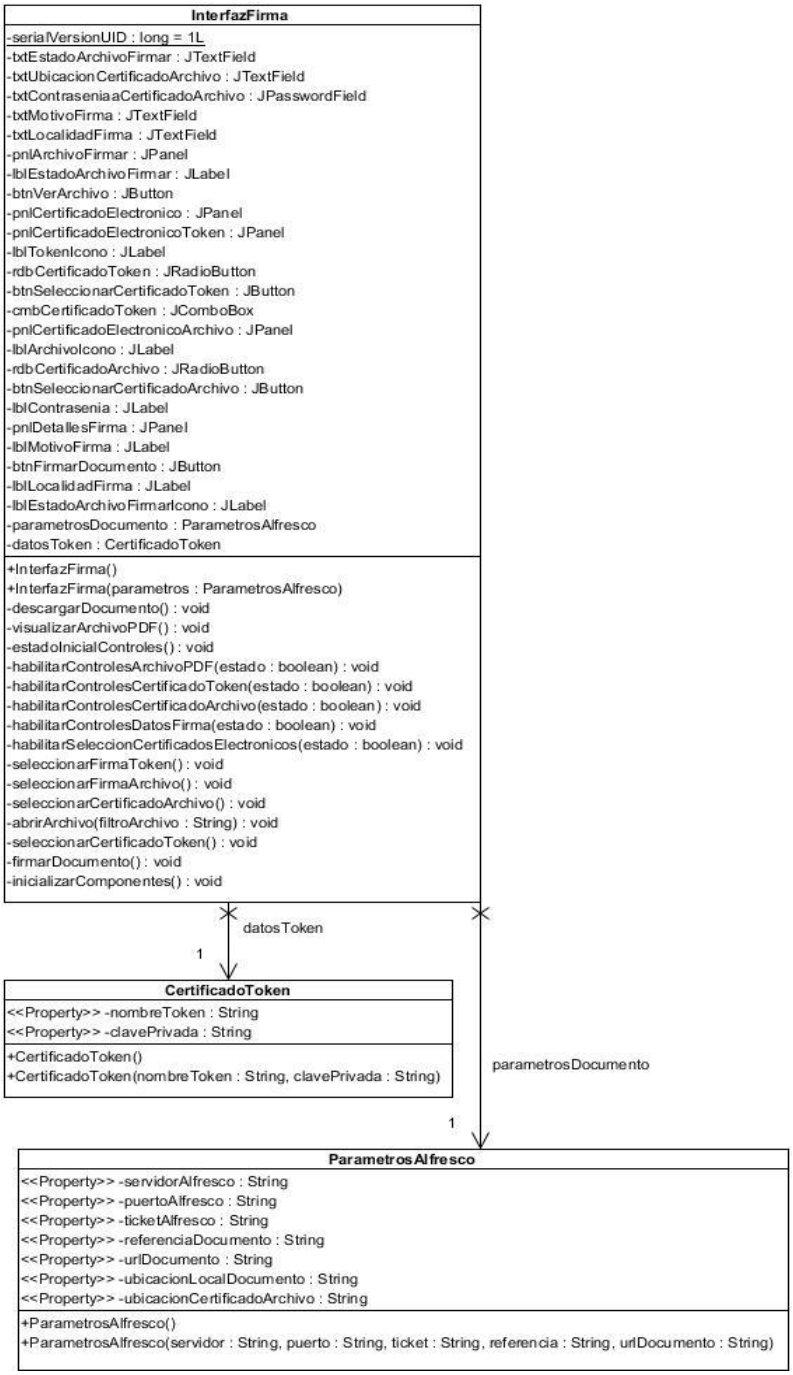


Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete ec.fin.banred.firmaelectrónica.interfaz.componentes**

La clase *InterfazFirma* contiene el panel con los elementos necesarios para recuperar la información necesaria para la firma electrónica. Entre los elementos se encuentran cajas de texto, botones, lista de datos, etc.

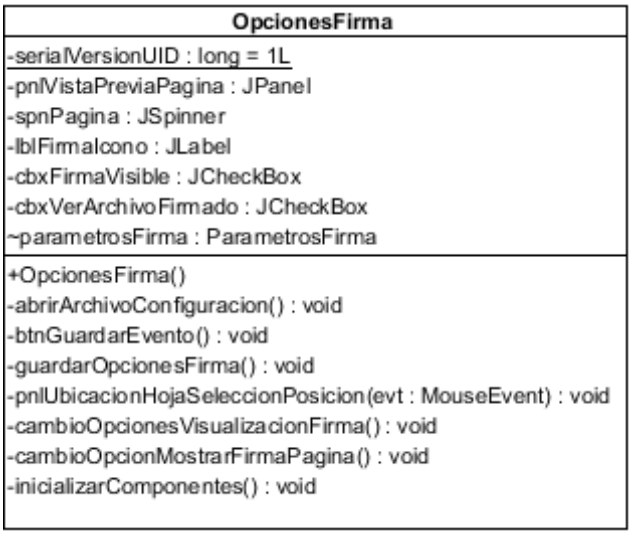
Figura 53. Clase *InterfazFirma* con paso de parámetros a clases POJO *ParámetrosAlfresco* y *CertificadoToken*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *OpcionesFirma* presenta la interfaz de usuario que captura los parámetros de firma que especifique el usuario dentro del Applet. Estos parámetros son reflejados a la clase *ParametrosFirma.*

Figura 54. Clase POJO *OpcionesFirma*



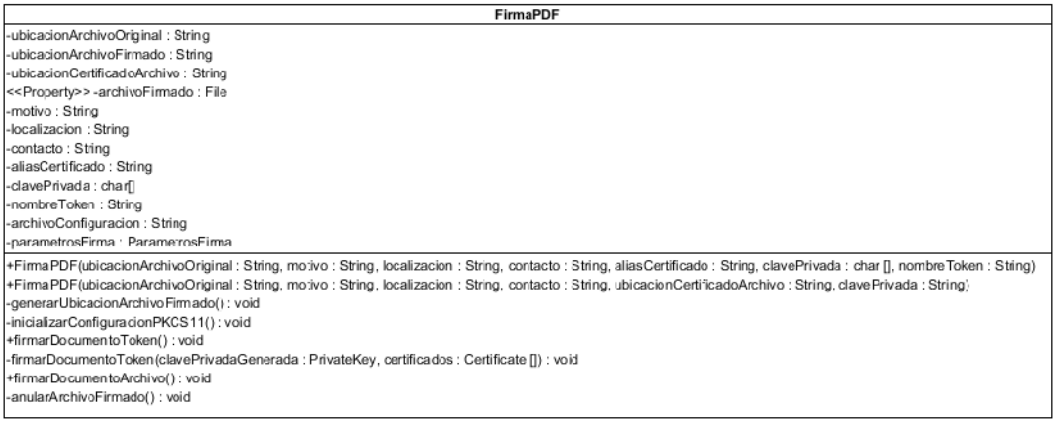
Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete Aplicación Java**

- **Paquete ec.fin.banred.firmaelectrónica.clases.firma**

La clase *FirmaPDF* se encarga de ejecutar la firma electrónica dentro del aplicativo, esta clase recupera los diferentes parámetros de firma, ubicación en el documento electrónico y los datos del documento a actualizar dentro del repositorio documental. También especifica si se firmará con certificado digital tipo *archivo* o tipo *token*.

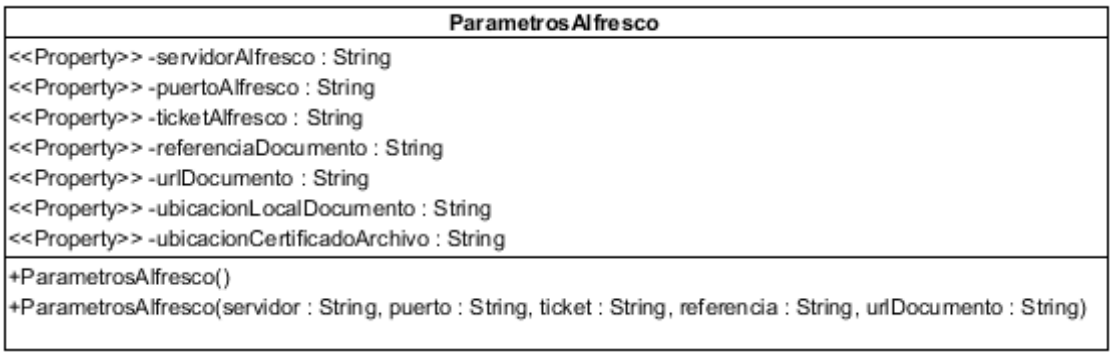
Figura 55. Clase POJO *FirmaPDF*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *ParámetrosAlfresco* guarda la persistencia de los parámetros de conexión al sistema ECM, adicionalmente guarda la ruta en el repositorio documental del documento a firmar electrónicamente y la ruta de la carpeta temporal donde se descargará el documento. En el caso de utilizar la firma electrónica con un archivo local de certificado electrónico se guardará la ruta del mismo.

Figura 56. Clase POJO *ParámetrosAlfresco*



Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete ec.fin.banred.firmaelectrónica.clases.firma.utilitarios**

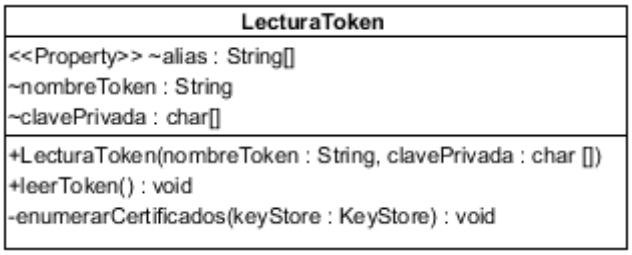
La clase CertificadoToken guarda la persistencia de los datos del certificado electrónico de tipo token que se usará en el proceso de firma electrónica.

Figura 57. Clase POJO CertificadoToken

Elaborado por: Pedro Caicedo y Diego Godoy

La clase LecturaToken permite la lectura del dispositivo token para la firma electrónica.

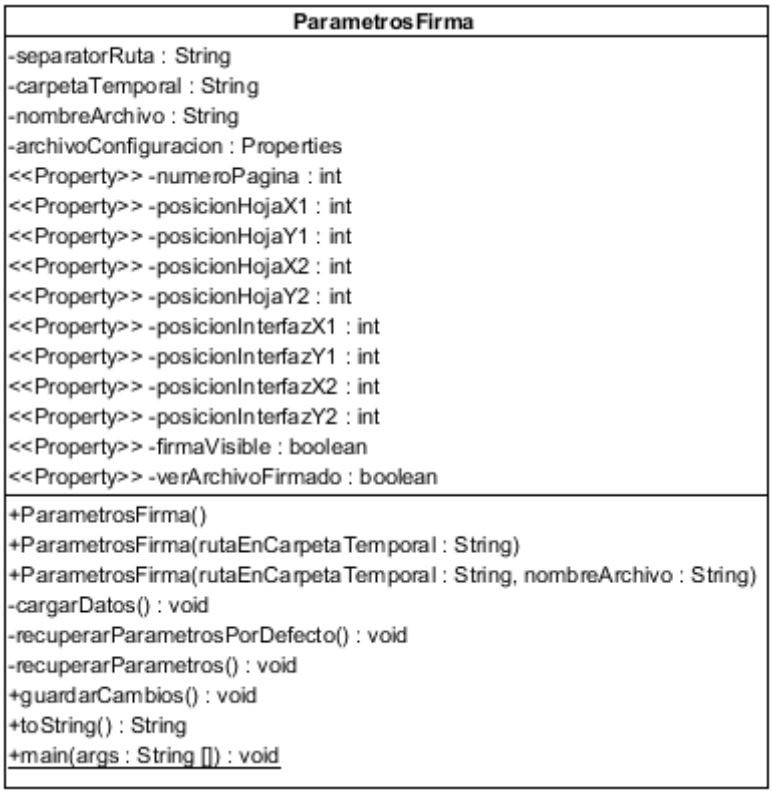
Figura 58. Clase POJO LecturaToken



Elaborado por: Pedro Caicedo y Diego Godoy

La clase ParámetrosFirma guarda los parámetros de firma electrónica que el aplicativo de firma reflejará sobre los documentos PDF. Estos parámetros se encargan de guardar la posición de la firma, visibilidad de la firma, ubicación del archivo que guardará estos parámetros en la carpeta temporal y si se desea ver el documentó una vez se haya firmado.

Figura 59. Clase POJO ParámetrosFirma

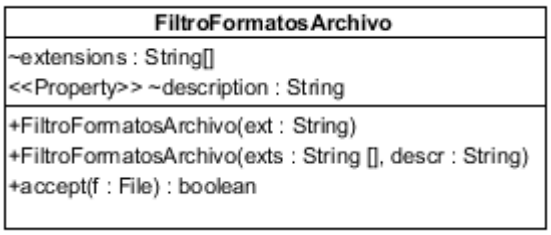


Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete ec.fin.banred.firmaelectrónica.clases.utilitarios**

La clase *FiltroFormatosArchivo* es una clase utilitaria que filtra el formato de archivos a visualizar durante la apertura de un cuadro de dialogo de tipo "Abrir archivo".

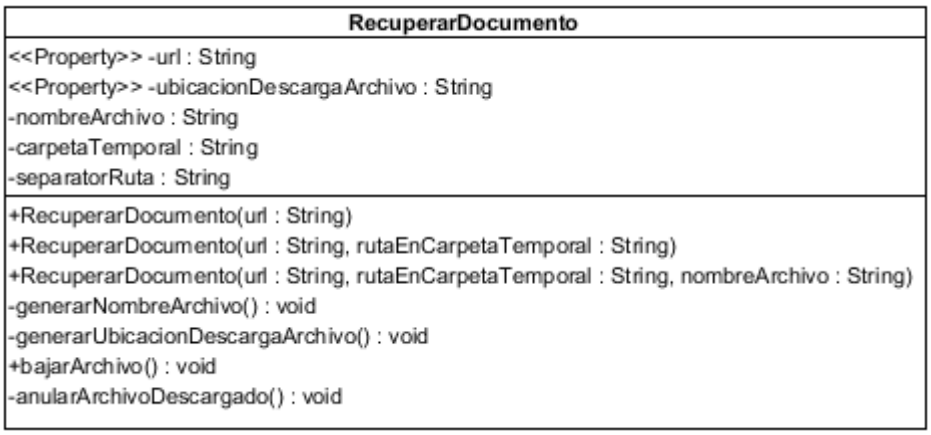
Figura 60. Clase POJO *FiltroFormatosArchivo*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *RecuperarDocumento* descarga el documento que se desea firmar electrónicamente desde el repositorio documental del ECM hacia una ubicación temporal del equipo cliente.

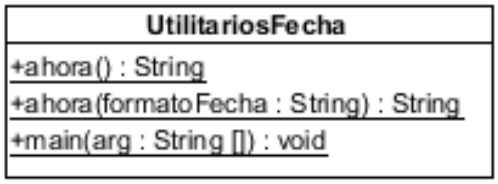
Figura 61. Clase POJO *RecuperarDocumento*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase UtilitariosFecha se encarga de formatear la fecha con el estándar ISO 8601, el cual es el formato que recibe el sistema ECM. Adicionalmente se puede especificar cualquier formato de fecha que se desee recuperar.

Figura 62. Clase POJO *UtilitariosFecha*



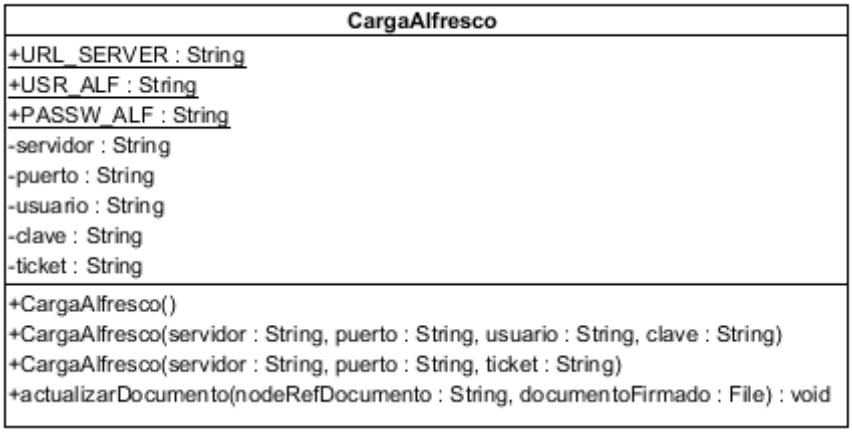
Elaborado por: Pedro Caicedo y Diego Godoy

- **Paquete Repositorio documental ECM**

- **Paquete ec.fin.banred.firmaelectrónica.clases.Alfresco**

La clase CargaAlfresco está enfocada a la actualización del documento que se va a firmar electrónicamente dentro del repositorio documental del ECM. Los parámetros que necesita esta clase son la ruta del ECM y las credenciales de acceso para realizar la actualización.

Figura 63. Clase POJO *CargaAlfresco*



Elaborado por: Pedro Caicedo y Diego Godoy

Observación: Cabe recalcar que MYSQL es el motor de base de datos del ECM ALFRESCO establecido en este proyecto, en el cual se almacenará los documentos en formato PDF generados por el aplicativo de firma electrónica.

3.5.3 Modelo de diseño de componente para creación de documentos en formato .docx

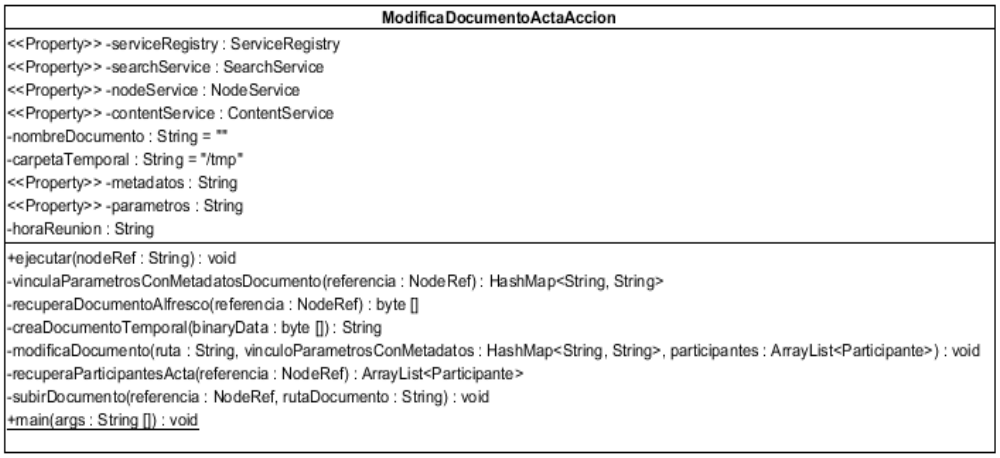
Este modelo consta de los siguientes paquetes y clases.

3.5.3.1 Paquete ec.fin.banred.alfresco.javascript

ModificaDocumentoActaAccion es una clase extendida del API interno de ALFRESCO, presenta el método ejecutar(),que recibe el identificador del documento tipo acta de reunión/comité a modificar.

En esta clase se recuperan los METADATOS y participantes del acta que se han especificado dentro del proceso documental con el fin de ser aplicados dentro del contenido del documento que hasta el momento contiene los datos genéricos de la plantilla. También permite la descarga del documento en una ruta temporal dentro del servidor. Una vez modificado el documento procede a actualizarse en el repositorio documental.

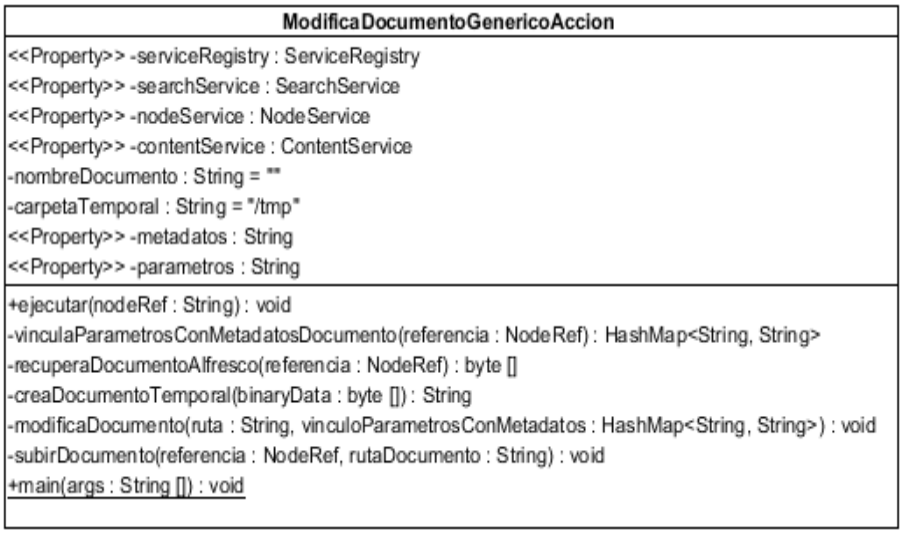
Figura 64. Clase POJO ModificaDocumentoActaAccion



Elaborado por: Pedro Caicedo y Diego Godoy

ModificaDocumentoGenericoAccion es una clase con funcionalidad similar a la clase ModificaDocumdentoActaAccion, cuyo método ejecutar() recibe en este caso, el identificador de un documento de tipo oficio, memorando o informe que se va a modificar.

Figura No 65. Clase POJO ModificaDocumentoGenericoAccion.

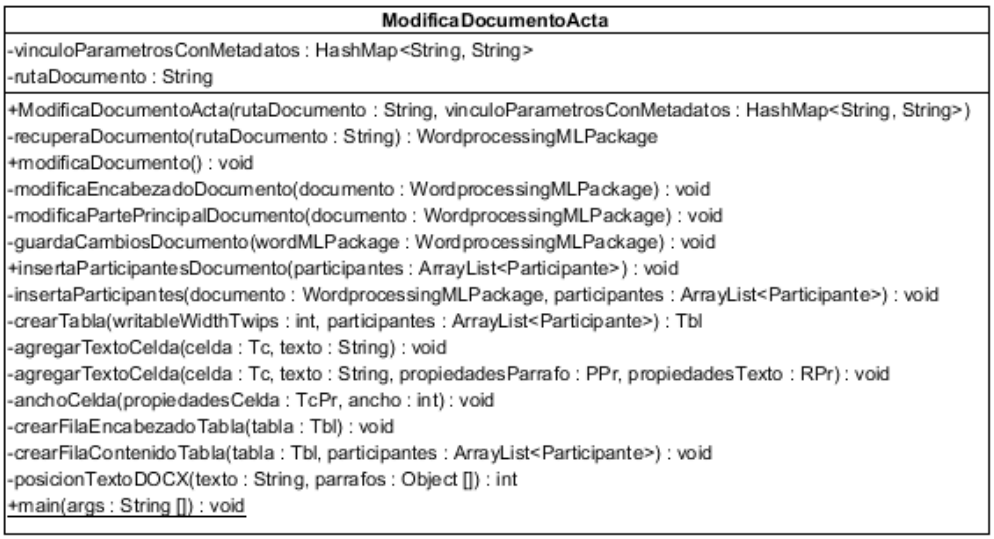


Elaborado por: Pedro Caicedo y Diego Godoy

3.5.3.2 Paquete ec.fin.banred.office.word.plantillas

La clase ModificaDocumentoActarecibe un documento para su modificación, de tipo acta, con formato .docx, así como los METADATOS que se van a escribir en él, genera una tabla dinámica que lista los datos de los participantes de un evento de tipo reunión de trabajo, y finalmente devuelve el documento modificado a la clase ModificaDocumentoGenericoAccion.

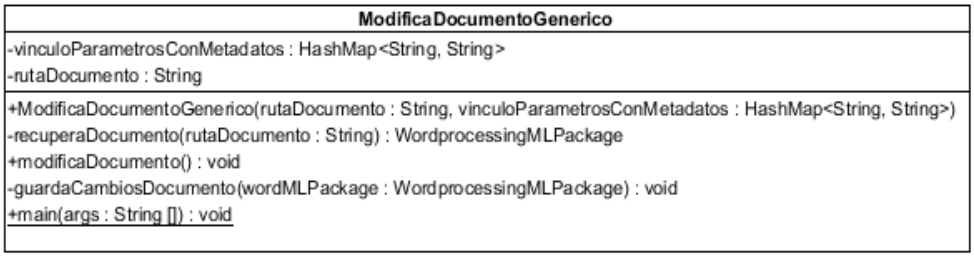
Figura 66. Clase POJO *ModificaDocumentoActa*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *ModificaDocumentoGenerico* posee la misma funcionalidad de *ModificaDocumentoActa*, pero en este caso recibe un documento de tipo *informe*, *memorando* u *oficio* de formato .docx.

Figura 67. Clase POJO *ModificaDocumentoGenerico*

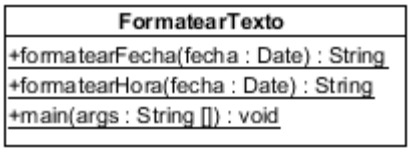


Elaborado por: Pedro Caicedo y Diego Godoy

3.5.3.3 Paquete ec.fin.banred.office.word.utilitarios

FormatearTexto es una clase utilitaria que da formato a la fecha que se ha especificado en el documento, para que sea entendible por el usuario, el formato de la fecha es el siguiente: "DD de MMM del YYYY". Adicionalmente se puede especificar una estampa de tiempo para que sea formateada adecuadamente, por ejemplo, "HH:MM"

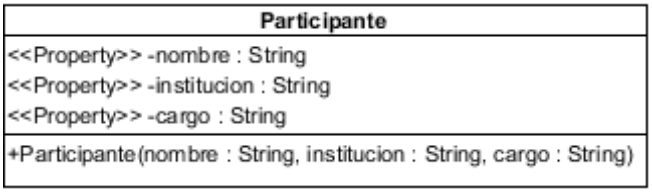
Figura 68. Clase POJO *FormatearTexto*



Elaborado por: Pedro Caicedo y Diego Godoy

La clase *Participante* contiene la persistencia de un objeto de tipo *participante del acta*, el cual contiene los datos que son reflejados dentro de la clase *ModificaDocumentoActa*.

Figura No 69. Clase POJO *Participante.*



Elaborado por: Pedro Caicedo y Diego Godoy

3.6 Modelo de implementación

El modelo presenta los siguientes elementos:

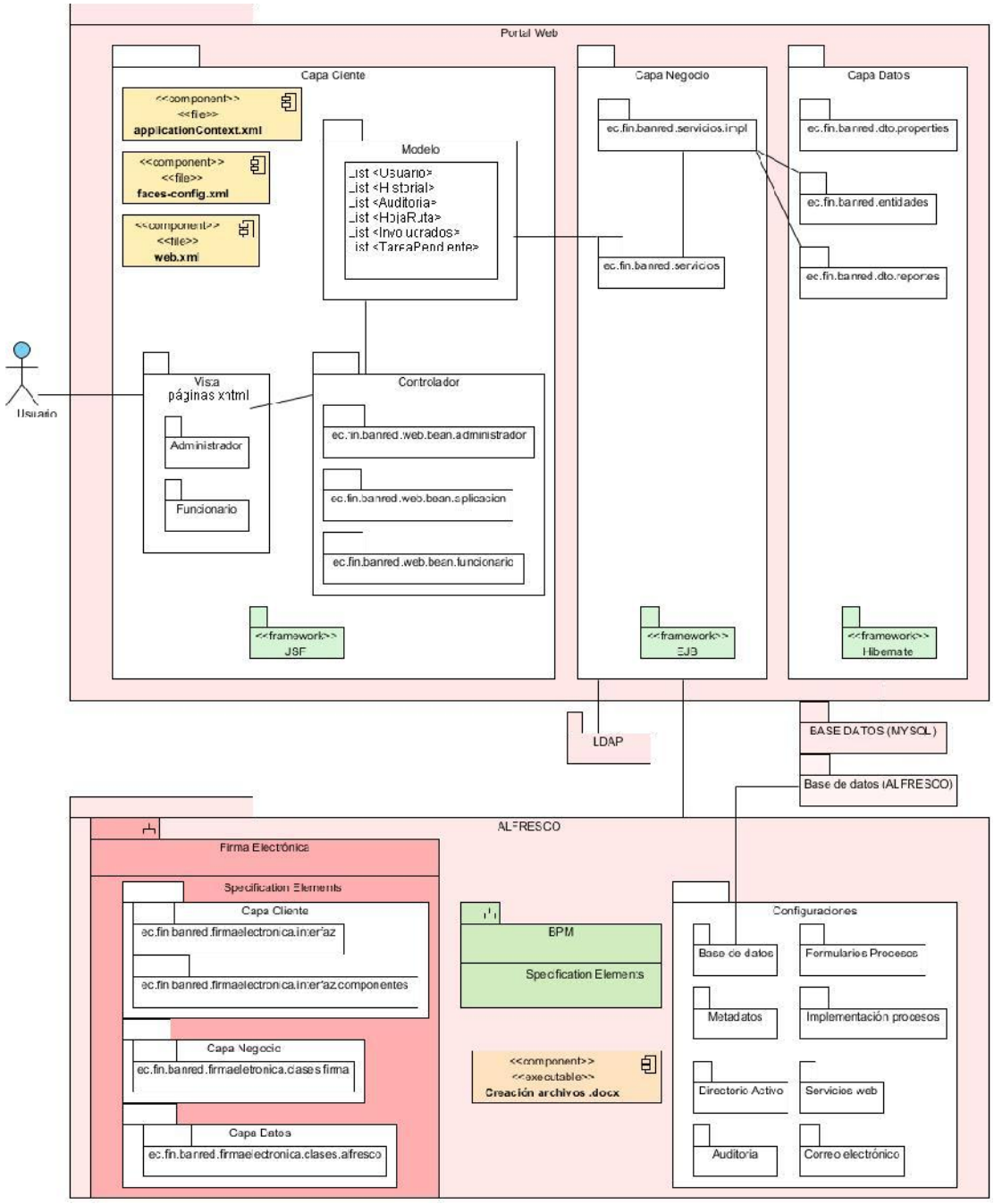
- Portal web: Considera el uso del patrón MVC en una arquitectura de tres capas en su desarrollo y una base de datos en motor MYSQL para almacenar información relacionada a los usuarios del portal web y la auditoría sobre la gestión de dichos usuarios.

- Motor BPM: El que despliegan los procesos de negocio de la gestión documental y el ECM Alfresco los ejecuta.

- Aplicativo de firma electrónica y el componente de creación de documentos con formato docx: Interactúan con los flujos de trabajo de la gestión documental de la herramienta ECM.

En seguida se muestra el modelo de implementación de la solución propuesta, donde se precisa los sistemas involucrados, las clases, artefactos, API’s y patrón MVC del proyecto web así como los componentes del ECM Alfresco.

Figura 70. Componentes modelo de implementación proyecto tesis



Elaborado por: Pedro Caicedo y Diego Godoy

CAPÍTULO 4
CONSTRUCCIÓN

4.1 Estándares o convenios para la programación

En seguida se detallan los estándares y convenios para programación establecidos para este proyecto.

4.1.1 Buenas prácticas para manejo de base de datos

- Usar nombres consistentes y bien definidos para tablas y columnas (ejemplo: Escuela, CursoEstudiante, etc).
- Usar nombres en singular para las tablas (Estudiante en lugar de Estudiantes). La tabla representa una colección de entidades, pero no es necesario usar nombres en plural.
- No incluir espacios en los nombres de las tablas.
- Usar prefijos claros, evitar TblEscuela, o EscuelaTabla, etc.
- Mantener los passwords encriptados por seguridad. Desencriptarlos en la aplicación si es necesario.
- Usar tipo de datos enteros como identificadores para todas las tablas.
- Elegir columnas con tipos enteros (o sus variantes) para indexar. Una columna con tipo varchar puede causar problemas de rendimiento.
- Usar campos de tipo bit para almacenar valores booleanos. Usar enteros o varchar repercute en un consumo innecesario de almacenamiento. Incluso los nombres de esas columnas se podrá poner con el prefijo "Is" o "Es" en español.
- Proveer siempre procedimientos de autenticación para el acceso a base de datos. Administrar de forma eficiente los roles de usuario de la base de datos, en especial el rol de administrador.
- No usar querys del tipo "select * " a menos de que sea necesario, extraer solo las columnas necesarias para un mejor rendimiento.
- Usa un framework o marco de trabajo ORM como Hibernate, oBatis, etc, si el código de la aplicación es lo suficientemente grande.
- Particionar la base de datos separando las tablas que se usan mucho de las que no se usan tanto para un mejor desempeño del motor de base de datos.

- Para bases de datos grandes, sensibles y sistemas de misión crítica, usar los servicios de recuperación de desastres y servicios de seguridad como el failover clustering, respaldos automáticos, replicación, etc.
- Usar constraints (llaves foraneas, checks, valores no nulos, etc) para la integridad de datos. No controlar todo desde el código de la aplicación.
- Documentar el diseño de base de datos con esquemas de entidad relacionales (ER) e instrucciones. Escribir líneas de comentarios en los trigers, procedimientos almacenados y otros scripts.
- Usar índices para scripts frecuentemente usados en tablas grandes.
- Un servidor de base de datos y un servidor web en ambientes de producción deben estar en máquinas diferentes. Esto provee de más seguridad y separan la carga de trabajo en dos CPUs y memoria diferentes.
- Prestar especial atención al diseño de la base de datos.
- La normalización debe ser usada cuando sea requerida para optimizar el performance de la base de datos. Una baja normalización puede repercutir en una repetición de datos, mientras que una sobre normalización puede tener efectos en el rendimiento a causa de las excesivas uniones entre tablas para extraer datos. Se deberá mantener un equilibro.

4.1.2 Convenciones de código para el lenguaje de programación Java

4.1.2.1 Extensiones de los archivos

El software Java usa las siguientes extensiones para los archivos:

Tabla 8. Extensiones de archivos Java

Tipo de archivo	Extensión
Fuente Java	.java
Bytecode de Java	.class

Elaborado por: Pedro Caicedo y Diego Godoy

4.1.2.2 Archivos fuente Java

Cada archivo fuente Java contiene una única clase o interface pública.

Los archivos fuentes Java tienen la siguiente ordenación:

- Sentencias package e import
- Declaraciones de clases e interfaces

- **Sentencias package e import**

La primera línea no-comentario de los archivos fuente Java es la sentencia package. Después de esta, pueden seguir varias sentencias import. Por ejemplo:

package java.awt;
import java.awt.peer.CanvasPeer;

- **Declaraciones de clases e interfaces**

La siguiente tabla describe las partes de la declaración de una clase o interface, en el orden en que deberían aparecer.

Tabla 9. Declaraciones de clases e interfaces

No.	Partes de la declaración de una clase o interface	Notas
1	Comentario de documentación de la clase o interface (/**...*/)	N/A
2	Sentencia class o interface	N/A
3	Comentario de implementación de la clase o interface si fuera necesario (/*...*/)	Este comentario debe contener cualquier información aplicable a toda la clase o interface que no era apropiada para estar en los comentarios de documentación de la clase o interface.

No.	Partes de la declaración de una clase o interface	Notas
4	Variables de clase (static)	Primero las variables de clase public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
5	Variables de instancia	Primero las public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
6	Constructores	
7	Métodos	Estos métodos se deben agrupar por funcionalidad más que por visión o accesibilidad. Por ejemplo, un método de clase privado puede estar entre dos métodos públicos de instancia. El objetivo es hacer el código más legible y comprensible.

Fuente. (Oracle, 2013).

- **Longitud de la línea**

Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

- **Rompiendo líneas**

Cuando una expresión no entre en una línea, habrá que romperla de acuerdo con estos principios:

- Romper después de una coma.
- Romper antes de un operador.
- Preferir roturas de alto nivel (más a la derecha que el "padre") que de bajo nivel (más a la izquierda que el "padre").
- Alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea anterior.

Ejemplo de cómo romper la llamada a un método:

```
unMétodo(expresionLarga1, expresionLarga2, expresionLarga3,
        expresionLarga4, expresionLarga5);

var = unMétodo1(expresionLarga1,
                    unMétodo2(expresionLarga2,
                    expresionLarga3));
```

Ahora otro ejemplo de ruptura de líneas en expresiones aritméticas.

```
nombreLargo1 = nombreLargo2 * (nombreLargo3 + nombreLargo4
- nombreLargo5) + 4 * nombreLargo6;
```

Ahora un ejemplo de identación de declaraciones de métodos.

```
unMétodo(int anArg, Object anotherArg, String yetAnotherArg,Object andStillAnother) {
        …
}
```

Ejemplo de Salto de líneas por sentencias if .

```
if ((condicion1 && condicion2)
        || (condicion3 && condicion4)
        ||!(condicion5 && condicion6)) {
hacerAlgo();
}
```

- **Comentarios**

Los programas Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación.

Los comentarios de implementación son aquellos que también se encuentran en C++, delimitados por /*...*/, y //.

Los comentarios de documentación (conocidos como "doc comments") existen sólo en Java, y se limitan por /**...*/, estos se pueden exportar a archivos HTML con la herramienta javadoc.

Los comentarios deben contener sólo información que es relevante para la lectura y entendimiento del programa.

- **Declaraciones**

Se recomienda una declaración por línea, ya que facilita los comentarios, como por ejemplo:

```
int nivel; // nivel de identación
```

No poner varios en la misma línea, ejemplo:

```
int foo, fooarray; //ERROR!
```

- **Inicialización**

Intentar inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

Evitar las declaraciones locales que ocultan declaraciones de niveles superiores. Por ejemplo, no declarar la misma variable en un bloque interno:

```
int cuenta;
...
miMétodo() {
        if (condicion) {
        int cuenta = 0; // EVITAR!
            ...
                    }
         ...
                }
```

- **Declaraciones de clases e interfaces**

Al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:

- Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.
- La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración
- La llave de cierre "}" se debe ajustar a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{"

```
class Ejemplo extends Object {
        int ivar1;
        int ivar2;

        Ejemplo(int i, int j) {
                ivar1 = i;
                ivar2 = j;
        }
        intmétodoVacio() {} //Los métodos se separan con una línea en blanco
        ...
}
```

- **Convenciones de nombres**

Las convenciones de nombres hacen los programas más entendibles haciéndolos más fácil de leer. También pueden dar información sobre la función de un identificador, por ejemplo, cuando es una constante, un paquete, o una clase, que puede ser útil para entender el código.

Tabla 10. Convenciones de nombres para programación

Tipos de identificadores	Reglas para nombres	Ejemplos
Paquetes	El primer componente del nombre de un paquete único se escribe siempre en minúsculas con caracteres ASCII y debe ser uno de los nombres de dominio de último nivel, actualmente com, edu, gob, mil, net, org, o uno de los códigos de dos letras que especifican el país como se define en el ISO Standard 3166, 1981. Los subsecuentes componentes del nombre del paquete variarán de a cuerdo a las convenciones de nombres internas de cada organización como es la especificación de algunos nombres de los directorios que correspondan a divisiones, departamentos, proyectos o máquinas.	com.sun.eng com.apple.quicktime.v2 edu.cmu.cs.bovik.cheese
Clases	Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).	class Cliente; class ImagenAnimada;
Interfaces	Los nombres de las interfaces siguen la misma regla que las clases.	interface Objeto Persistente; interface Almacen;

Métodos	Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.	ejecutar();
		ejecutarRapido();
		cogerFondo();

Tipos de identificadores	Reglas para nombres	Ejemplos
Variables	Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje. Para variables temporales son i, j, k, m, y n para enteros; c, d, y e para caracteres.	int i; char c; float miAnchura;

Constantes	Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas separando las palabras con un subguión ("_"). (Las constantes ANSI se deben evitar, para facilitar su depuración.)	static final int ANCHURA_MINIMA =4;
		static final int ANCHURA_MAXIMA =999;
		static final int COGER_LA_CPU = 1;

Fuente: (Oracle, 2013).

4.2 Descripción de métodos

A continuación se detallan los métodos definidos en el portal web, el aplicativo de firma electrónica y el componente de creación de archivos en formato docx

4.2.1 Descripción de métodos de portal web

Tabla 11. Descripción métodos portal web

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
buscarUsuarios	ec.fin.banred.servicios.BanredServicio	Método que busca usuarios en la base de datos local	public List<Usuario> buscarUsuarios(Usuario usuario)
buscarHistoral	ec.fin.banred.servicios.BanredServicio	Método que busca registros históricos en la base de datos local	public List<Historial> buscarHistoral(Usuario usuario)
buscarInvolucrados	ec.fin.banred.servicios.BanredServicio	Método que busca involucrados de un trámite específico	public List<Involucrados> buscarInvolucrados (String numeroTramite)
buscarHojaRuta	ec.fin.banred.servicios.BanredServicio	Método que busca la hoja ruta de un trámite específico	public List<HojaRuta> buscarHojaRuta(String numeroTramite)
buscarTareaPendientes	ec.fin.banred.servicios.BanredServicio	Método que buscar tareas pendientes de un funcionario	public List<TareaPendiente> buscarTareaPendientes(String usuario)
actualizarObjeto	ec.fin.banred.servicios.BanredServicio	Método que actualiza un objeto en la base de datos	public boolean actualizarObjeto(Object object)
crearObjeto	ec.fin.banred.servicios.BanredServicio	Método que crea un objeto en la base de datos	public boolean crearObjeto(Object object)
buscarUsuariosldap	ec.fin.banred.servicios.BanredServicio	Método que busca usuarios del Directorio Activo	public List<Usuario> buscarUsuariosldap(Usuario usuario,String seleccionBusqueda, String terminoBusqueda)
buscarAuditoria	ec.fin.banred.servicios.BanredServicio	Método que busca auditoría del portal web	public List<Auditoria> buscarAuditoria(Auditoria auditoría)

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
obtenerIdUsuario	ec.fin.banred.servicios.BanredServicio	Método que busca ID para registro usuario	public Integer obtenerIdUsuario()
obtenerIdHistoria	ec.fin.banred.servicios.BanredServicio	Método que busca ID para registro historial	public Integer obtenerIdHistoria()
buscarUsuario	ec.fin.banred.servicios.BanredServicio	Método que busca registro usuario por ID	public Usuario buscarUsuario(Integer id)
Inicializar	ec.fin.banred.web.bean.administrador.AdministracionUsuariosBean	Método que inicializa la página	public void inicializar()
seleccionarUsuario	ec.fin.banred.web.bean.administrador.AdministracionUsuariosBean	Método que selecciona un usuario del listado	public String seleccionarUsuario(Usuario usuario)
buscarUsuario	ec.fin.banred.web.bean.administrador.AdministracionUsuariosBean	Método que busca usuarios	public String buscarUsuario(ActionEvent actionEvent)
validarListado	ec.fin.banred.web.bean.administrador.AdministracionUsuariosBean	Método que valida listado entre Directorio Activo y BDD	public List<Usuario> validarListado(List<Usuario> listadoActual, List<Usuario> listadoldap)
Reiniciar	ec.fin.banred.web.bean.administrador.AdministracionUsuariosBean	Método que reinicia la página	public String reiniciar(ActionEvent actionEvent)
cancelar	ec.fin.banred.web.bean.administrador.AdministracionUsuariosBean	Método que cancele edición de usuario	public String cancelar(ActionEvent actionEvent)
guardarCambios	ec.fin.banred.web.bean.administrador.AdministracionUsuariosBean	Método que guardar cambios realizados en usuario	public String guardarCambios(ActionEvent actionEvent)
inicializar	ec.fin.banred.web.bean.administrador.AuditoriaBean	Método que inicializa la página	public void inicializar()
buscarHistorial	ec.fin.banred.web.bean.administrador.AuditoriaBean	Método que busca registros de auditoría de BDD local	public String buscarHistorial(ActionEvent actionEvent)
reiniciar	ec.fin.banred.web.bean.administrado.AuditoriaBeanr	Método que reinicia la página	public String reiniciar(ActionEvent actionEvent)
inicializar	ec.fin.banred.web.bean.administrador.AuditoriaECMBean	Método que inicializa la página	public void inicializar()
buscarHistorial	ec.fin.banred.web.bean.administrador.AuditoriaECMBean	Método que busca registros de auditoría de BDD local	public String buscarHistorial(ActionEvent actionEvent)
reiniciar	ec.fin.banred.web.bean.administrador.AuditoriaECMBean	Método que reinicia la página	public String reiniciar(ActionEvent actionEvent)

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
guardar	ec.fin.banred.web.bean.administrador.ParametrizacionBean	Método que guarda un registro properties modificado	public String guardar(Long tipo)
getBdd	ec.fin.banred.web.bean.administrador.ParametrizacionBean	Método que obtiene properties de base de datos	public String getBdd()
getEcm	ec.fin.banred.web.bean.administrador.ParametrizacionBean	Método que obtiene properties de ECM	public String getEcm()
getLdap	ec.fin.banred.web.bean.administrador.ParametrizacionBean	Método que obtiene properties de LDAP	public String getLdap()
inicializar	ec.fin.banred.web.bean.aplicacion.InicioBean	Método que inicializa parámetros en la aplicación	public void inicializar()
redirect	ec.fin.banred.web.bean.aplicacion.InicioBean	Método que re direcciona a la página del user autenticado	public void redirect(String tipo)
salirApp	ec.fin.banred.web.bean.aplicacion.InicioBean	Método que sale y elimina sesión de usuario	public String salirApp(ActionEvent event)
inicializar	ec.fin.banred.web.bean.funcionario.EstadoTramiteBean	Método que inicializa la página	public void inicializar()
buscarEstadoTramite	ec.fin.banred.web.bean.funcionario.EstadoTramiteBean	Método que busca el estado de trámite	public String buscarEstadoTramite(ActionEvent actionEvent)
inicializar	ec.fin.banred.web.bean.funcionario.HojaRutaBean	Método que inicializa la página	public void inicializar()
buscarHojaRuta	ec.fin.banred.web.bean.funcionario.HojaRutaBean	Método que busca la hoja ruta de un trámite	public String buscarHojaRuta(ActionEvent actionEvent)
inicializar	ec.fin.banred.web.bean.funcionario.InvolucradosBean	Método que inicializa la página	public void inicializar()
buscarInvolucrados	ec.fin.banred.web.bean.funcionario.InvolucradosBean	Método que busca involucrados de un trámite	public String buscarInvolucrados(ActionEvent actionEvent)
inicializar	ec.fin.banred.web.bean.funcionario.TareasPendientesBean	Método que inicializa la página con las tareas pendientes	public void inicializar()
Utilldap	ec.fin.banred.web.ldapConnection.Utilldap	constructor de la clase que inicializa variables	public Utilldap(String usuario, String clave, String servidor, String dominio)

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
searchUser	ec.fin.banred.web.ldapConnection.Utilldap	Método que busca usuario en Directorio Activo	public NamingEnumeration<SearchResult> searchUser(String searchValue, String searchBy, String searchBase)
closeLdapConnection	ec.fin.banred.web.ldapConnection.Utilldap	Método que cierra la conexión	public void closeLdapConnection()
getFilter	ec.fin.banred.web.ldapConnection.Utilldap	Método que identifica el filtro para la búsqueda en Directorio Activo	private String getFilter(String searchValue, String searchBy)
getDomainBase	ec.fin.banred.web.ldapConnection.Utilldap	Método que identifica el dominio de base del Directorio Activo	private static String getDomainBase(String base)
leerProperties	ec.fin.banred.web.util.AplicacionUtil	Método que lee un archivo properties	public static Properties leerProperties (String URL)
guardarPrperties	ec.fin.banred.web.util.AplicacionUtil	Método que guarda un archivo properties	public static boolean guardarPrperties(Properties prop, String URL)
setearAtributo	ec.fin.banred.web.util.AplicacionUtil	Método que escribe un atributo	public static void setearAtributo(String nombre, Object object)
tomarAtributo	ec.fin.banred.web.util.AplicacionUtil	Método que lee un atributo	public static Object tomarAtributo(String nombre)
tomarParámetro	ec.fin.banred.web.util.AplicacionUtil	Método que lee un parámetro	public static String tomarParámetro(String nombre)
removerAtributo	ec.fin.banred.web.util.AplicacionUtil	Método que remueve un atributo	public static void removerAtributo(String nombre)
replace	ec.fin.banred.web.util.AplicacionUtil	Método que reemplaza una cadena	public static String replace(String str, String pattern, String replace)
leerRest	ec.fin.banred.web.util.AplicacionUtil	Método que lee un servicio rest	public static void leerRest(String webPage, String nombre)
formatearFecha	ec.fin.banred.web.util.AplicacionUtil	Método que formatea una fecha	public static String formatearFecha(String fecha)

Elaborado por: Pedro Caicedo y Diego Godoy

4.2.2 Descripción de métodos aplicativo de firma electrónica

Tabla 12. Descripción métodos aplicativo de firma electrónica

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
descargarDocumento	InterfazFirma	Descargar documento del repositorio documental a firmar	descargarDocumento()
visualizarArchivoPDF	InterfazFirma	Abre el aplicativo para visualización de documentos PDF en equipo del cliente	visualizarArchivoPDF()
abrirArchivo	InterfazFirma	Presenta un cuadro de dialogo para abrir certificados electrónicos con formato P12 para firma electrónica	abrirArchivo(String rutaBuscar)
seleccionarCertificadoToken	InterfazFirma	Permite seleccionar el certificado electrónico de tipo Token para firmar documentos.	seleccionarCertificadoToken()
firmarDocumento	InterfazFirma	Firma el documento con el certificado electrónico seleccionado	firmarDocumento()
abrirArchivoConfiguración	OpcionesFirma	Leer el archivo de configuración con los parámetros para firma electrónica	abrirArchivoConfiguración()
actualizarDocumento	CargaAlfresco	Actualizar documento PDF firmado en el equipo local hacia el documento dentro del repositorio	actualizarDocumento(String nodeRef, File archivoFirmado)
generarUbicaciónArchivoFirmado	FirmaPDF	Genera la ruta del archivo PDF firmado por el aplicativo	generarUbicaciónArchivoFirmado()
FirmaPDF	FirmaPDF	Constructor de la clase el cual recibe los parámetros a reflejar en la firma electrónica del documento a firmar	FirmaPDF(String ubicacionArchivoOriginal, String motivo, String localizacion, String contacto, String aliasCertificado, char[] clavePrivada, String nombreToken)
inicializarConfiguraciónPKCS11	FirmaPDF	Genera la configuración PKCS11 para que el equipo del cliente reconozca el certificado digital de tipo token	inicializarConfiguraciónPKCS11()

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
firmarDocumentoToken	FirmaPDF	Firmar electrónicamente un documento con un certificado digital de tipo token	firmarDocumentoToken(PrivateKey clavePrivadaGenerada, Certificate[] certificados)
firmarDocumentoArchivo	FirmaPDF	Firmar electrónicamente un documento con un certificado digital de tipo archivo	firmarDocumentoArchivo()
getArchivoFirmado	FirmaPDF	Recuperar el archivo firmado electrónicamente	getArchivoFirmado()
ParámetrosAlfresco	ParámetrosAlfresco	Constructor que recupera los parámetros de conexión del aplicativo con el sistema de gestión documental	ParámetrosAlfresco(String servidor, String puerto, String ticket, String referencia, String urlDocumento)
getServidorAlfresco	ParámetrosAlfresco	Devolver la dirección IP o hostname del servidor de gestión documental	getServidorAlfresco()
getPuertoAlfresco	ParámetrosAlfresco	Devolver el puerto del servidor de gestión documental	getPuertoAlfresco()
getTicketAlfresco	ParámetrosAlfresco	Devolver el ticket de autenticación del servidor de gestión documental	getTicketAlfresco()
getReferenciaDocumento	ParámetrosAlfresco	Devolver la referencia del documento recuperado del gestor documental	getReferenciaDocumento()
getUrlDocumento	ParámetrosAlfresco	Recupera el URL de acceso al documento	getUrlDocumento()
LecturaToken	LecturaToken	Constructor con parámetros de lectura de un certificado electrónico de tipo token	LecturaToken(String nombreToken, char[] clavePrivada)
getAlias	LecturaToken	Recuperar el alias del token	getAlias()
leerToken	LecturaToken	Permite acceder al token y recuperar sus funcionalidades de firma y encriptación	leerToken()
ParámetrosFirma	ParámetrosFirma	Constructor para recuperar los parámetros de firma almacenados por la interfaz de firma	ParámetrosFirma(String rutaEnCarpetaTemporal, String nombreArchivo)
cargarDatos	ParámetrosFirma	Cargar los datos almacenados en el archivo de parámetros de firma	cargarDatos()
recuperarParámetrosPorDefecto	ParámetrosFirma	Si no existe el archivo de parámetros de firma, establece parámetros por defecto	recuperarParámetrosPorDefecto()
recuperarParámetros	ParámetrosFirma	Recupera los parámetros al sistema de firma electrónica	recuperarParámetros()

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
Ahora	UtilitariosFecha	Recupera la hora actual del sistema	ahora(String formatoFecha)
RecuperarDocumento	RecuperarDocumento	Constructor de la clase para recuperar un documento del sistema de gestión documental	RecuperarDocumento(String URL, String rutaEnCarpetaTemporal, String nombreArchivo)
generarNombreArchivo	RecuperarDocumento	Genera un nombre para el documento temporal almacenado por el aplicativo	generarNombreArchivo()
bajarArchivo	RecuperarDocumento	Recupera el archivo desde el repositorio documental hacia el equipo del cliente	bajarArchivo()

Elaborado por: Pedro Caicedo y Diego Godoy

4.2.3 Componente de creación de archivos con formato .docx

Tabla 13. Descripción métodos de componente de creación de archivos con formato .docx

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
Ejecutar	ModificaDocumentoActaAccion.java	Ejecuta la inserción de los METADATOS de un documento del repositorio documental hacia un documento tipo plantilla con formato .DOCX	ejecutar(String nodeRef)
vinculaParámetrosConMetadatosDocumento	ModificaDocumentoActaAccion.java	Vincula los METADATOS recuperados del documento del sistema ECM y los parámetros a reemplazar dentro de la plantilla .DOCX	vinculaParámetrosConMetadatosDocumento(String nodeRef)
recuperaDocumentoAlfresco	ModificaDocumentoActaAccion.java	Recupera los bytes del documento en el gestor documental	recuperaDocumentoAlfresco(String nodeRef)
creaDocumentoTemporal	ModificaDocumentoActaAccion.java	Guarda los bytes recuperados de un documento del sistema ECM a un archivo temporal del servidor	creaDocumentoTemporal(byte[] documento)

NOMBRE	UBICACIÓN	UTILIDAD	DECLARACIÓN
modificaDocumento	ModificaDocumentoActaAccion.java	Refleja los índices del documento dentro de la plantilla .DOCX	modificaDocumento(String ruta, HashMap<String, String> vinculoParmetrosConMetadatos)
recuperaParticipantesActa	ModificaDocumentoActaAccion.java	Recupera los participantes del documento tipo acta	recuperaParticipantesActa(String nodeRef)
subirDocumento	ModificaDocumentoActaAccion.java	Envía los bytes del documento modificado al sistema de gestión documental	subirDocumento(String nodeRef, String rutaDocumento)
formatearFecha	FormatearDocumento.java	Formatea la fecha de forma entendible para el usuario	formatearFecha(Date fecha)
formatearHora	FormatearDocumento.java	Formatea la hora de forma entendible para el usuario	formatearHora(Date fecha)
ejecutar	ModificaDocumentoGenericoAccion.java	Ejecuta la inserción de los METADATOS de un documento del repositorio documental hacia un documento tipo plantilla con formato .DOCX	ejecutar(String nodeRef)
vinculaParámetrosConMetadatosDocumento	ModificaDocumentoGenericoAccion.java	Vincula los METADATOS recuperados del documento del sistema ECM y los parámetros a reemplazar dentro de la plantilla .DOCX	vinculaParámetrosConMetadatosDocumento(String nodeRef)
recuperaDocumentoAlfresco	ModificaDocumentoGenericoAccion.java	Recupera los bytes del documento en el gestor documental	recuperaDocumentoAlfresco(String nodeRef)
creaDocumentoTemporal	ModificaDocumentoGenericoAccion.java	Guarda los bytes recuperados de un documento del sistema ECM a un archivo temporal del servidor	creaDocumentoTemporal(byte[] documento)
modificaDocumento	ModificaDocumentoGenericoAccion.java	Refleja los índices del documento dentro de la plantilla .DOCX	modificaDocumento(String ruta, HashMap<String, String> vinculoParmetrosConMetadatos)
subirDocumento	ModificaDocumentoGenericoAccion.java	Envía los bytes del documento modificado al sistema de gestión documental	subirDocumento(String nodeRef, String rutaDocumento)

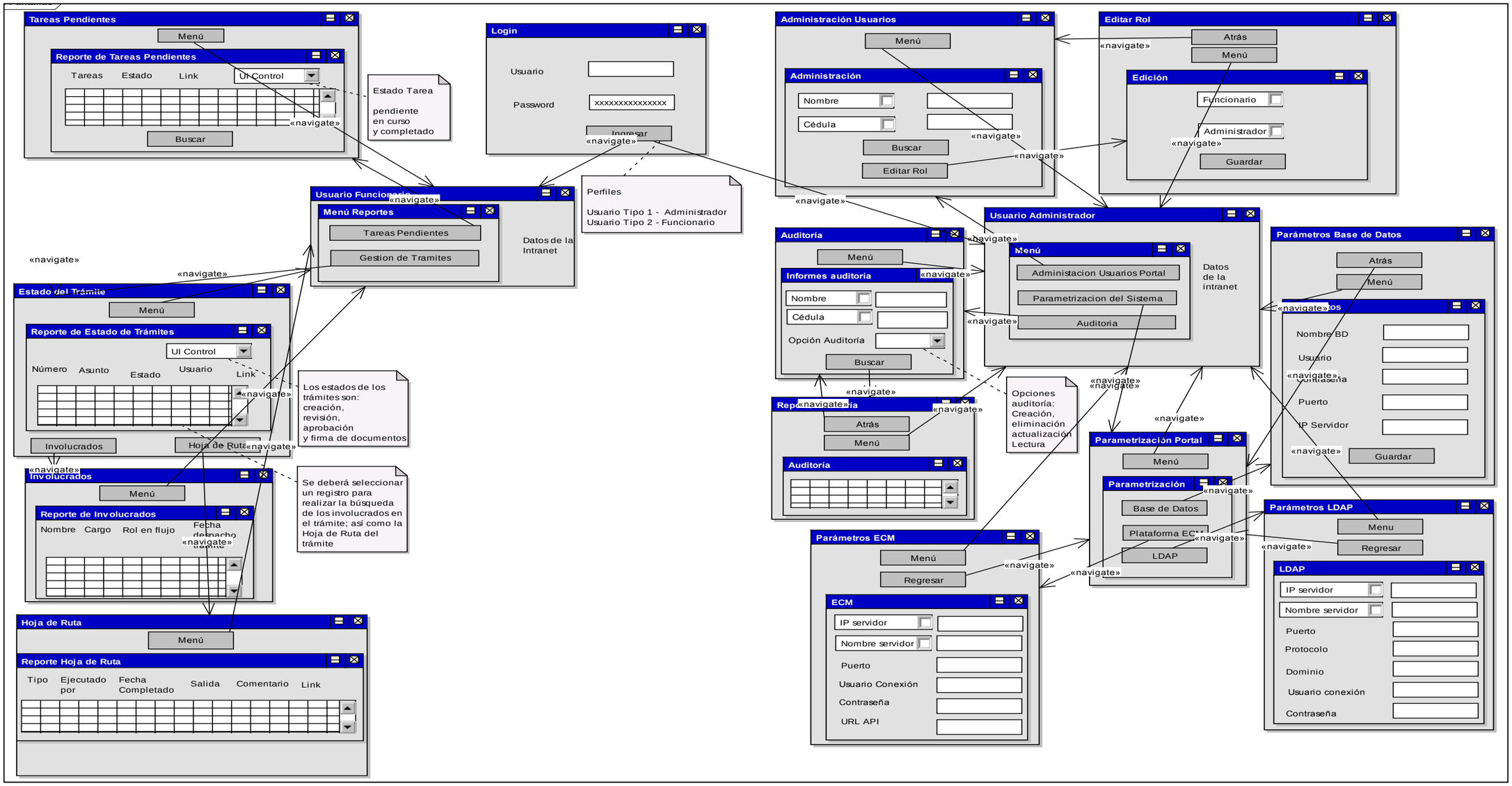
Fuente: Predro Caicedo y Diego Godoy.

4.3 Descripción de interfaces

Seguidamente se esquematiza el modelo de interfaz de usuario del portal web y del aplicativo de firma electrónica.

4.3.1 Modelo de interfaz portal web

Figura 71. Modelo de interfaz portal web

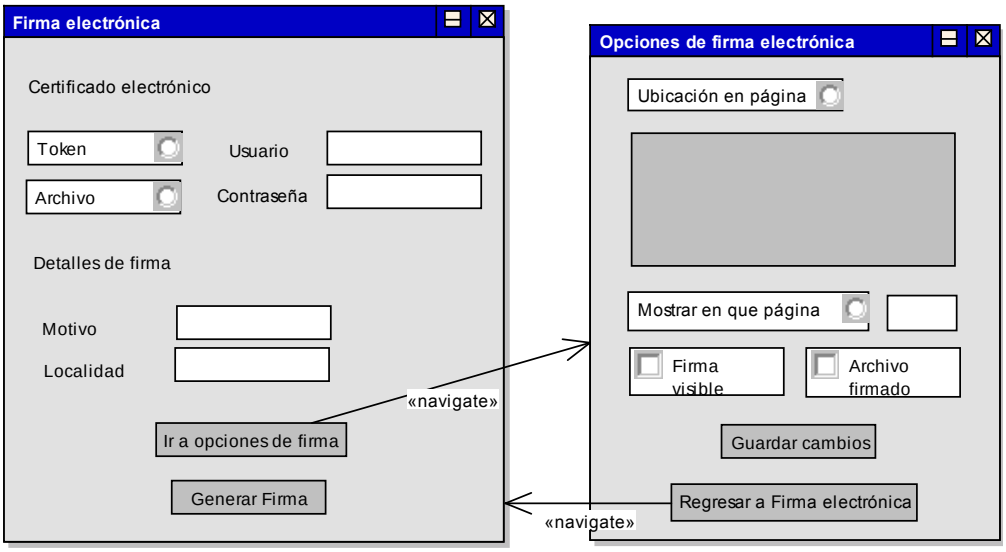


Elaborado por: Pedro Caicedo y Diego Godoy

Las interfaces descritas, atenderán los requerimientos mencionados en el punto 3.2.3 “Evaluación de interoperatividad entre ECM y aplicativo web”. Para mayor detalle, ver anexo DETALLE DE INTERFAZ DE USUARIO.

4.3.2 Modelo de interfaz aplicativo de firma electrónica

Figura 72. Modelo de interfaz aplicativo de firma electrónica



Elaborado por: Pedro Caicedo y Diego Godoy

El detalle funcional pude verse en el anexo MANUAL DE USUARIO INTEROPERABILIDAD DE LA PLATAFORMA ECM CON LA SIMULACIÓN DE INTRANET DE BANRED.

4.4 Código relevante

En seguida se detalla el código fuente obtenido de la construcción del portal web, aplicativo de firma electrónica y componente java para creación de documentos en formato .docx, así como los parámetros de configuración de la plataforma ECM.

4.4.1 Construcción del Portal web

4.4.1.1 Arquitectura del Portal web

A continuación se describen los componentes de cada uno de los paquetes de la aplicación web, por las capas del proyecto. Referencia diagrama No 20. Arquitectura del portal web,

- **Descripción capa de datos**

Las clases tipo Entidad y DTO's "Data Transfer Object" son especificadas a continuación:

- **Paquete de Clases Entidad:**

El paquete *ec.fin.banred.entidades* contiene las clases Entidad Historial.java y Usuario.java. A continuación se transcribe el código fuente de la clase Entidad Historial.java:

```
@Entity // anotación de la clase abstracta "persistence", que convierte en entidad a una tabla
@Table(name="historial") // nombre de la tabla de la base de datos "historial"

    public class Historial implements Serializable{ // Definición clase "Historial" que //hace uso de la interface "Serializable"
                                                     //para manejar atributos de la clase de forma nativa para varios lenguajes de programación

        private static final long serialVersionUID = 1L; // Serialización Java usada para controlar las diferentes //versiones
                                                         //de objetos en un escenario de serialización.

        @Id //anotación para identificar el id de la entidad, // la primera línea de sintaxis java después de la anotación es
                                                             //afectada por la dicha
                                                            //anotación
        @GeneratedValue(strategy=GenerationType.IDENTITY) // valor autogenerado
        private Integer id;
        private String estado;
        private Date fecha;
        private String accion;
        @JoinColumn(name="USUARIO")//apunta a campo usuario de tabla Usuario
        @ManyToOne(fetch=FetchType.EAGER)//relación de la base de datos varios a uno, donde  //varios historiales
                                                                                            //tienen un usuario
        private Usuario usuario;
```

```
    public Historial(){ // Definición de constructor de la clase
    }
    public Historial(Integer id, String estado, Usuario usuario, Date fecha, String accion) {  // Definición de constructor
                                                                                    //de la clase e inicialización de atributos del objeto
            super();
            this.id = id;
            this.estado = estado;
            this.usuario = usuario;
            this.fecha = fecha;
            this.accion = accion;
    }

    public Integer getId() { // método para obtener valor del atributo "id" del objeto
            return id;
    }
    public void setId(Integer id) { //métodо para actualizar el valor en el atributo "id" del objeto
            this.id = id;
    }

} // fin clase
```

- **Paquete de clases DTO**

Las clases DTO's se utiliza para mostrar la información relacionada a la gestión documental, manejada en la plataforma Alfresco, a manera de tablas en las páginas de la aplicación web.

Dichos DTO's servirán como plantillas para la administración de los servicios web tipo REST.

El paquete *ec.fin.banred.dto.reportes* contiene las clases DTO siguientes:

- Involucrados.java
- Auditoria.java
- HojaRuta.java
- TareaPendiente.java

A continuación se transcribe el código más importante de la clase DTO "Involucrados.java" y se explica su funcionamiento:

```
public class Involucrados implements Serializable { //Definición clase “Involucrados” que hace uso de la interface
                                                    //“Serializable” para manejar atributos de la clase de forma nativa para
                                                    //lenguajes de programación

        private static final long serialVersionUID = 1L; // Serialización Java usada para controlar las diferentes versiones de
        objetos en un escenario de serialización.

        private String tipo;
        private String nombre;
        private String desripcion;
        private String fecha;
        private String resultado;
        private String comentario;

        public Involucrados(){ //Declaración de constructor sin parámetros
        }
        public Involucrados(String tipo, String nombre, String desripcion, String fecha, String resultado, String comentario)
        {
                //Declaración de constructor con parámetros
                        super();
                        this.tipo = tipo;
                        this.nombre = nombre;
                        this.desripcion = desripcion;
                        this.fecha = fecha;
                        this.resultado = resultado;
                        this.comentario = comentario;
                }
                public String getTipo() { // Método para obtener valor de atributo tipo
                        return tipo;
                }
                public void setTipo(String tipo) { //Método para asignar valor a atributo tipo
                        this.tipo = tipo;
…..}
```

El otro paquete DTO *ec.fin.banred.dto.properties,* contiene la clase DTO ArchivoProperties.java

Este DTO maneja y muestra los valores almacenados en los archivos planos “properties” del ECM, LDAP y Portal web de la sección parametrización del sistema en el aplicativo web

A continuación se transcribe el código más relevante de esta clase:

```
public class ArchivoProperties {

        private String nombre;
        private String usuario;
        private String contrasenia;
        private String puerto;
```

```
private String servidor;
private String driverClassName;
private String url;
private String baseBusqueda;

public String getNombre() {
        return nombre;
}
public void setNombre(String nombre) {
        this.nombre = nombre;
}
public String getUsuario() {
        return usuario;
……..
```

Las librerías del API de persistencia Hibernate (capa de datos), se encuentra en el directorio WEB-INF/lib/ de la aplicación web, y estas son:

- Hibernate-3.2.4.ga.jar
- Hibernate-annotations-3.3.1GA.jar
- Hibernate-commons-annotations-3.0.0.ga.jar
- Hibernate-entitymanager-3.3.2.GA.jar

- **Descripción capa de servicio o negocio**

En esta capa se describe la lógica del negocio, en general encontraremos operaciones con los orígenes de datos (bases de datos y servicios web).

Esta capa la constituye, la interface *BanredServicio*que presenta todos los métodos que van a estar publicados para consumo de las otras capas de la aplicaciónweb y la clase *BanredServicioImpl* que implementa dicha interface con sus métodos.

A continuación se muestra el código fuente de la interface *BanredServicio*

```
public interface BanredServicio {
        public List<Usuario> buscarUsuarios(Usuario usuario);
        public List<Historial> buscarHistoral(Usuario usuario);
        public List<Involucrados> buscarInvolucrados (String numeroTramite);
        public List<HojaRuta> buscarHojaRuta(String numeroTramite);
        public List<TareaPendiente> buscarTareaPendientes(String usuario);
        public boolean actualizarObjeto(Object object);
```

```
	public boolean crearObjeto(Object object);
	public List<Usuario> buscarUsuariosldap(Usuario usuario,String seleccionBusqueda, String
                                                                terminoBusqueda) ;
	public List<Auditoria> buscarAuditoria(Auditoria auditoría);
	public Integer obtenerIdUsuario() ;
	public Integer obtenerIdHistoria() ;
	public Usuario buscarUsuario(Integer id);
}
```

A continuación se transcribe la porción más relevante del código de la clase *BanredServicioImpl*:

```
public List<Involucrados> buscarInvolucrados(String numeroTramite) {
		List<Involucrados> listado = new ArrayList<Involucrados>();
		try {
			String jsonFile = "involucrados.json";
			String url = AplicacionUtil.SERVICIO_INVOLUCRADOS;
			url = AplicacionUtil.replace(url, "$numeroTramite", numeroTramite);
			AplicacionUtil.leerRest(url, jsonFile);
			JSONObject jsonObjectInit = objectJSON(jsonFile); //buscar jar de jason
			JSONObject jsonObjectData = (JSONObject) jsonObjectInit.get("data");
			JSONArray list = (JSONArray) jsonObjectData.get("tasks");
			for (Object element : list) {
				JSONObject jsonObject = (JSONObject) element;
				String tipo = (String) jsonObject.get("title");
				String desripcion = (String) jsonObject.get("description");
				String resultado = (String) jsonObject.get("state");
				String comentario = (String) jsonObject.get("outcome");
				JSONObject jsonObjectProperties = (JSONObject) j
			sonObject.get("properties");
				String fecha = AplicacionUtil.formatearFecha((String)
				jsonObjectProperties.get("bpm_startDate"));
				JSONObject jsonObjectOwner = (JSONObject)
				jsonObject.get("owner");
				String nombre = ((String) jsonObjectOwner.get("firstName"))+"
					"+((String) jsonObjectOwner.get("lastName"));
				listado.add(new Involucrados(tipo, nombre, desripcion, fecha,
					resultado, comentario));
			}
		} catch (Exception e) {
			LOG.severe("Error al buscar Involucrados: "+e.getLocalizedMessage());
		}
		return listado;
	}
public JSONObject objectJSON(String jsonFile) { // método para crear objeto json
		try {
			JSONParser parser = new JSONParser();
			Object obj = parser.parse(new FileReader(jsonFile));
			JSONObject jsonObjectInit = (JSONObject) obj;
			return jsonObjectInit;
```

```
		} catch (Exception e) {
			e.printStackTrace();
			return null;
		}
	}
```

Los métodos que maneja el API de persistencia Hibernate para realizar diferentes operaciones sobre la base de datos son:

- Find: mediante un query sql recupera datos de la base de datos a manera de lista de objetos, aplicado en método: *buscarHistorial().*
- Persist: con este método se crea registros en la base de datos, lo que se envía como parámetro es un objeto tipo entidad, aplicado en método: *crearObjeto()*.
- Update: con este método se actualiza registros en la base de datos, lo que se envía como parámetro es un objeto tipo entidad, aplicado en método: *actualizarObjeto().*

Cabe recalcar que dentro de los métodos se puede utilizar lógica de base de datos, por ejemplo en los métodos *crearRegistro()* o *actualizarRegistro()* se maneja transacciones que tienen el mismo funcionamiento de una transacción en base de datos, como lo muestra el ejemplo siguiente:

```
public boolean actualizarObjeto(Object object) { // aquí se maneja la transacción o sesión
sessionFactory = super.getSessionFactory(); // se toma los datos de conexión a la base de datos
		Session session = sessionFactory.openSession(); // se abre sesión de usuario root
		Transaction tx = null; // se inicializan variables
		boolean valor=false;
		try {
			tx = session.beginTransaction(); //inicio de transacción
			session.update(object);
			tx.commit();
			valor = true;
		} catch (RuntimeException e) {
			LOG.severe("Error al actualizar objeto: "+e.getLocalizedMessage());
			if (tx != null)
				tx.rollback(); //deja la base de datos tal cual antes de la
					transacción
			throw e;
		} finally {
			session.close();
		}
		return valor;
	}
```

Otro origen de datos con el que consta la aplicación son servicios web, específicamente del tipo REST, que para nuestra aplicación nos retorna archivos JSON.

Una vez obtenido el archivo JSON del servicio web se procede a realizar una conversión en objetos, para lo cual se utilizará las plantillas o clases DTO antes especificadas. Se podría decir que los servicios se apoyan en las entidades y en los DTO's para el manejo de las estructuras de datos.

Para realizar esta conversión se utiliza la librería json_simple.jarque junto al método *getJasonObject()*obtiene uno a uno los objetos requeridos. Por ejemplo, para la búsqueda de involucrados se tiene el siguiente método:

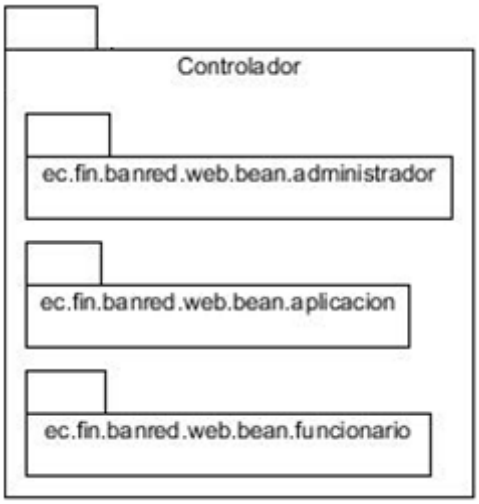
```
publicList<Involucrados> buscarInvolucrados(String numeroTramite) {
        List<Involucrados> listado = new ArrayList<Involucrados>();
                try {
                        String jsonFile = "involucrados.json";
                        String url = AplicacionUtil.SERVICIO_INVOLUCRADOS;
                        url = AplicacionUtil.replace(url, "$numeroTramite", numeroTramite);
                        AplicacionUtil.leerRest(url, jsonFile);
                        JSONObject jsonObjectInit = objectJSON(jsonFile); //buscar jar dejason
                        JSONObject jsonObjectData = (JSONObject) jsonObjectInit.get("data");
                        JSONArray list = (JSONArray) jsonObjectData.get("tasks");
                        for (Object element : list) {
                                JSONObject jsonObject = (JSONObject) element;
                                String tipo = (String) jsonObject.get("title");
                                String desripcion = (String) jsonObject.get("description");
                                String resultado = (String) jsonObject.get("state");
                                String comentario = (String) jsonObject.get("outcome");
                                JSONObject jsonObjectProperties = (JSONObject)
                                    jsonObject.get("properties");
                                String fecha = AplicacionUtil.formatearFecha((String)
                                    jsonObjectProperties.get("bpm_startDate"));
                                JSONObject jsonObjectOwner = (JSONObject) jsonObject.get("owner");
                                String nombre = ((String) jsonObjectOwner.get("firstName"))+" "+((String)
                                                        jsonObjectOwner.get("lastName"));
                                listado.add(new Involucrados(tipo, nombre, desripcion, fecha, resultado,
                                comentario));
                        }
                } catch (Exception e) {
                        LOG.severe("Error al buscar Involucrados: "+e.getLocalizedMessage());
                }
                return listado;
        }
```

- **Descripción capa web**

El patrón MVC de esta capa considera la siguiente estructura en el proyecto:

1. Los controladores o beans son clases que contienen los atributos y métodos que manejará una página web (la vista) en la presentación, estos utilizarán los servicios de la aplicación publicados en la capa anterior y también se apoyarán en los objetos de la capa de datos (modelo) para el manejo más ágil en la programación.

Figura 73. Paquetes Controlador patrón MVC



Elaborado por: Pedro Caicedo y Diego Godoy

Como ejemplo se muestra a continuación un fragmento de código de la clase I*nvolucradosBean* que se encuentra en el paquete *ec.fin.banred.web.bean.funcionario*

```
@ManagedBean(name = "involucradosBean")
@ViewScoped
public class InvolucradosBean implements Serializable{

        private static final long serialVersionUID = 1L;
        private static final Logger LOG = Logger.getLogger(InvolucradosBean.class.getName());

        @ManagedProperty(value="#{servicio}")
        private BanredServicio servicio;

        private String numeroTramite;

        private List<Involucrados> listadoInvolucrados;
```

```
private String mensajeValidacion;

@PostConstruct
public void inicializar() {
        LOG.info("INICIO DEL BEAN");
        numeroTramite="";
        mensajeValidacion = "";
}
public String buscarInvolucrados(ActionEvent actionEvent) {
        mensajeValidacion = "";
        listadoInvolucrados = new ArrayList<Involucrados>();
        if ("".equalsIgnoreCase(numeroTramite.trim())) {
                mensajeValidacion = "El campo número de trámite es un campo requerido.";
        }else {
                listadoInvolucrados = servicio.buscarInvolucrados(numeroTramite);
        }
        return null;
}
```

2. Vista: En esta sección se encuentra los xhtml (páginas web), y toda la lógica que manejan actualmente este estándar como son CSS, imágenes entre otros.

Figura 74. Paquetes Vista patrón MVC

Elaborado por: Pedro Caicedo y Diego Godoy

La tecnología utilizada para la realización de esta sección de la capa fue JSF 2 (Java Server Faces) apoyado en el framework de JBossRichFaces en la versión 4.x.

A continuación se transcribe el código de la página xhtml involucrados.xhtml.

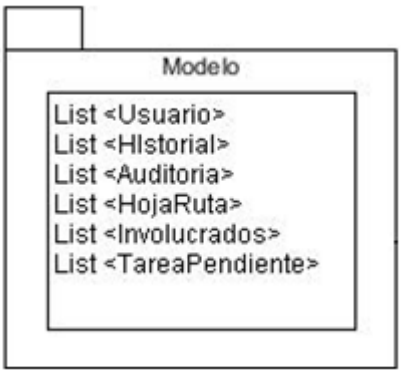
```
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<htmlxmlns="http://www.w3.org/1999/xhtml"
```

```
        xmlns:ui="http://java.sun.com/jsf/facelets"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:a4j="http://richfaces.org/a4j"
xmlns:rich="http://richfaces.org/rich">

        <ui:compositiontemplate="/paginas/plantilla.xhtml">
                <ui:definename="title"><h:outputTextvalue="Involucrados"/></ui:define>
                <ui:definename="subtitle"><h:outputTextvalue="Involucrados"/></ui:define>
                <ui:definename="contenidoPagina">
                        <h:form>
                        <rich:panel>

                        <div>
                                <h:outputTextvalue="Ingrese número de trámite: "/>
                                <h:inputTextvalue="#{involucradosBean.numeroTramite}"/>
                                <a4j:commandButtonactionListener="#{involucradosBean.buscarInvolucrados}"
                                        render="panelResultadosHistorial"value="Buscar"/>

                        </div>
                        <br/>
                                <divalign="center">
                                <h:panelGridwidth="100%"id="panelResultadosHistorial">
                                        <h:outputTextvalue="#{involucradosBean.mensajeValidacion}"
                                style="color:red;" />
                                        <rich:dataTablestyle="width:100%;"  var="element"
                                value="#{involucradosBean.listadoInvolucrados}"rowKeyVar="count"rows="5">
                                <f:facetname="header">
                                <rich:columnGroup>
                                <rich:column>
                                <h:outputTextvalue="N°"/>
                                </rich:column>
                                <rich:column>
                                <h:outputTextvalue="Tipo de tarea"/>
                                </rich:column>
                                <rich:column>
                                <h:outputTextvalue="Nombre del Involucrado"/>
                                </rich:column>
                                <rich:column>
                                <h:outputTextvalue="Descripción"/>
                                </rich:column>
                                <rich:column>
                                <h:outputTextvalue="Fecha de finalización"/>
                                </rich:column>
                                <rich:column>
                                <h:outputTextvalue="Resultado"/>
                                </rich:column>
                                <rich:column>
                                <h:outputTextvalue="Comentario"/>
                                </rich:column>
```

```
                    </rich:columnGroup>
                    </f:facet>

                    <rich:column>
                        <h:outputTextvalue="#{count+1}"/>
    </rich:column>

    <rich:column>
        <h:outputTextvalue="#{element.tipo}"/>
    </rich:column>
    <rich:column>
        <h:outputTextvalue="#{element.nombre}"/>
    </rich:column>
    <rich:column>
        <h:outputTextvalue="#{element.desripcion}"/>
    </rich:column>
    <rich:column>
        <h:outputTextvalue="#{element.fecha}"/>
    </rich:column>
    <rich:column>
        <h:outputTextvalue="#{element.resultado}"/>
    </rich:column>
    <rich:column>
        <h:outputTextvalue="#{element.comentario}"/>
    </rich:column>
    <f:facetname="footer">
        <rich:dataScrollerpage="1"/>
        </f:facet>
    </rich:dataTable>

                    </h:panelGrid>

                    </div>
        </rich:panel>
        </h:form>
    </ui:define>
</ui:composition>

</html>
```

3. Modelo: Aquí se encuentran todos los paquetes de la capa de datos, mencionados en el punto 4.4.1.1.2 .1, cuyas clases se muestran en el diagramasiguiente:

Figura 75. Paquetes Modelo patrón MVC



Elaborado por: Pedro Caicedo y Diego Godoy

- **Utilitario de la aplicación paquete UTIL**

En esta clase se encuentran métodos, constantes que se utilizan en toda la aplicación, como por ejemplo toda la configuración de las llamadas hacia los servicios web, credenciales de conexión al LDAP, credenciales de conexión al ECM, URL's que especifican los servicios web, path's o direcciones de los archivos properties para la parametrización, entre algunos métodos se encuentran, métodos para administrar properties, para manejo de la sesión de usuario, para formatear fecha, para leer un servicio REST.

- **Archivos de configuración**

- Paquete WEB-INF:
  - applicationtext.xml (conexión base de datos y declaración servicios)
  - faces-config.xml (reglas de navegación)
  - web.xml (configuración de instancias del proyecto)

- Paquete resources:
  - database.properties (propiedades conexión base de datos mysql)

El uso de estos archivos de configuración es el siguiente:

- facesConfig.xml: Contiene las reglas de navegación de la aplicación.
- web.xml: Configuración e instancias del proyecto.

- applicationContext.xml: Configuración de servicios, Hibernate y de entityManagerdatabase.properties: contiene las credenciales para conexión a la base de datos.

- **Librerías utilizadas en la aplicación**

Seguidamente, se indican las librerías necesarias para el funcionamiento de las tecnologías utilizadas en la aplicación web. Es necesario recalcar que algunas librerías dependen de otras.

RichFaces: Librerías necesarias para crear aplicaciones web con Ajax.

- richfaces-components-api-4.0.0.Final.jar
- richfaces-components-ui-4.0.0.Fina.jar
- richfaces-core-api-4.0.0.Final.jar
- richfaces-core-impl-4.0.0.Final.jar

JSF: Librerías requeridas para creación de aplicaciones java basadas en web.

- Jsf-api-2.0.2-b09.jar
- Jsf-facelets-1.1.14.jar
- Jsf-impl-2.0.4-b09.jar

Hibernate: Librerías necesarias para el mapeo objeto-relacional con la base de datos.

- hibernate-3.2.6.ga.jar
- hibernate-annotations-3.3.1.GA.jar
- hibernate-commons-annotations-3.0.0.ga.jar
- hibernate-entitymanager-3.3.2.GA.jar

Además se está utilizando la tecnología java con el jdk 1.7.

4.4.1.2 Tecnologías Utilizadas

- Capa Cliente: html, CSS
- Capa Web: JSF 2.0, framework RichFaces
- Capa Negocio: Servicios web REST, EJB
- Capa Datos: Hibernate v.3.5
- Base de Datos: MYSQL v.5.5
- Servidor de Apliaciónes:Tomcat v 7.0.30

4.4.2 Construcción aplicativo de firma electrónica

A continuación se describen las características de este Applet Java.

4.4.2.1 Arquitectura del aplicativo de firma electrónica

A continuación se describen los componentes que conforman el aplicativo de firma electrónica por cada uno de los paquetes que contiene este aplicativo. Ver figura No 53. “Paquetes aplicativo firma electrónica”, como referencia de esta arquitectura.

- **Paquete de clases Applet**

Con el fin de utilizar el certificado digital de tipo token en la firma electrónica de documentos dentro del aplicativo, se presentó la necesidad de exponer una aplicación que se ejecute desde el equipo del cliente, que recupere la información del hardware criptográfico (token) y lo envíe hacia el servidor web.

Mediante estos requerimientos se decidió utilizar un Applet que provea esta funcionalidad, dichos Applets son aplicaciones Java tipo cliente que se son ejecutados en el contexto de un navegador web y presenta toda la funcionalidad de la suite de Java.

El paquete *ec.fin.banred.firmaelectrónica.interfaz* contiene la clase *AppletFirma*, el cual se encarga de presentar el Applet con la interfaz de usuario para presentar la

aplicación. Cabe recalcar que esta clase es el contenedor principal, su interfaz hace una llamada a las clases *InterfazFirma* y *OpcionesFirma* del paquete *ec.fin.banred.firmaelectrónica.interfaz.componentes* las cuales contienen los controles necesarios para la interacción entre el aplicativo y el usuario. A continuación se presenta el código relevante de la clase *AppletFirma*:

```
                …
public final class AppletFirma extends JApplet
{
        …
        //Componentes del applet
        private javax.swing.JTabbedPane tabInterfaz;
        private InterfazFirma pnlFirmaElectronica;
        …

        public void init()
        {
                //Recupera parametros del navegador web
                String servidor = this.getParameter("servidor");
                String puerto = this.getParameter("puerto");
                String ticket = getParameter("ticket");
                String referenciaNodo  = this.getParameter("nodeRef");
                String urlDocumento =getParameter("pathsinfirmar");
                //Guardar parametros para su uso posterior
                this.parametros = new ParametrosAlfresco(servidor, puerto, ticket,
                                        referenciaNodo, urlDocumento);
                …
        }
        …
}
```

- **Paquete de clases aplicación Java**

Las clases de este paquete son utilizadas para recibir los parámetros recuperados del dispositivo criptográfico por parte del Applet y aplicar esta información sobre el documento a firmar electrónicamente.

El paquete *ec.fin.banred.firmaelectrónica.clases.firma* contiene las siguientes clases:

- FirmaPDF: Esta clase representa el núcleo del aplicativo, mediante el uso de librerías de seguridad propias de Java, imprime la firma electrónica con los datos recuperados sobre el documento PDF especificado. La modificación de este

documento se lo hace con la ayuda de la librería *iText* para manejo de archivos PDF.

- Parámetros ALFRESCO: Contiene los datos de conexión hacia el repositorio documental y el documento que va a ser actualizado con la firma electrónica.

Las librerías del API de seguridad utilizadas en esta capa son las siguientes:

- sunpkcs11.jar.
- Paquete java.security.* propio de la JDK de Java.

Las librerías del API para manipulación de archivos PDF utilizadas en este paquete son las siguientes:

- iText-2.1.0.jar.

A continuación se presenta el código más relevante de la clase FirmaPDF.java:

```
                …
public class FirmaPDF
{
        …
        public void firmarDocumentoToken()
        {
                …
                Provider proveedorCertificadosPKCS11 = new sun.security.pkcs11.SunPKCS11
                  (streamArchivoConfiguración);
                Security.addProvider(proveedorCertificadosPKCS11);
                KeyStore almacenCertificados = null;
                try {
                        // Inicializar el almacen de certificados  - Initialize Key Store
                        almacenCertificados = KeyStore.getInstance("PKCS11");
                        //Cargar el almacen de certificados PKCS11 con el PIN del token
                        almacenCertificados.load(null, this.clavePrivada);
                        //Inicializarla clave privada del Token - Initialize private key object
                        PrivateKey clavePrivada = (PrivateKey) almacenCertificados.getKey
                 (this.aliasCertificado, this.clavePrivada);
                        //Crear el certificado X509
                        X509Certificate certificadoX509 =  (X509Certificate)
                                almacenCertificados.getCertificate(this.aliasCertificado);
                        //Crear el arreglo de certificados para enviar al firma de PDF
                        Certificate[] certificados = new Certificate[1];
                        certificados[0] = (Certificate) certificadoX509;
                        this.firmarDocumentoToken(clavePrivada, certificados);
                } catch (…) { … }
                finally{
```

```
                        //Cerrar el proveedor PKCS11
                        Security.removeProvider(proveedorCertificadosPKCS11.getName());
                }
        }
        //Estampa la firma en el archivo PDF
        private void firmarDocumentoToken(PrivateKey clavePrivadaGenerada, Certificate[] certificados )
        {
                try
                {
                        //Abrir un PDFReader para modificar el archivo a firmar
                        PdfReader lectorPDF = new PdfReader(this.ubicacionArchivoOriginal);
                        //Crear un archivo como salida del proceso TODO: Crearlo antes!
                        this.archivoFirmado = new File(this.ubicacionArchivoFirmado);
                        //Crear un estampado para el PDF
                        PdfStamper selladorPDF;
                        selladorPDF = PdfStamper.createSignature(lectorPDF, null,
                                '\0', archivoFirmado, true);
                        //Crear una apriencia para la firma
                        PdfSignatureAppearance aparienciaFirmaPDF =
        selladorPDF.getSignatureAppearance();
                        ….
                                //Enviar parametro final de firma y cerrar para aplicar la firma
                                selladorPDF.setFormFlattening(false);
                                selladorPDF.close();
                }catch (…) { … }
        }
…
}
```

El paquete *ec.fin.banred.firmaelectrónica.clases.firma.utilitarios* contiene las clases necesarias para guardar los datos recuperados por el dispositivo criptográfico además de los datos de ubicación de la firma electrónica sobre el documento. Las clases dentro de este paquete son:

- *CertificadoToken:* clase que guarda los datos que se recuperan del dispositivo criptográfico.
- *LecturaToken: clase que se encarga de la lectura y recuperación de los datos del dispositivo criptográfico.*
- *ParametrosFirma*: clase que contiene los parámetros de ubicación de la firma sobre el documento PDF.

A continuación se presenta el código relevante de la clase *LecturaToken*:

```
…
```

```
public class LecturaToken {
        …
        public void leerToken()
        {
                //PKCS11Library = "/Library/OpenSC/lib/" + pkcs11LibName; //Llamada de libreria en Linux
                //Ubicacion de la libreria en Windows
                String ubicacionLibreriaPKCS11 = "C:\\Windows\\system32\\etpkcs11.dll";
                // Define SunPKCS#11 parameters
                String parametrosPKSC11 = "attributes(*,*,*)=\n{\nCKA_TOKEN=true\nCKA_LOCAL=true\n}";
                // Crear un archivo de configuración de token dentro de un string
                archivoConfiguración = "name = " + this.nombreToken.replace(' ', '_') +
                                        //Nombre del token, reemplazar espacios por guion bajo
                                        "\n" +
                                        "library = " + ubicacionLibreriaPKCS11 + //Ubicacion de la lib PKCS11
                                       "\n "+ "slot = 0 " + "\n "+
                                        "attributes = compatibility \n"+ parametrosPKSC11;
        }
}
```

Dentro de este paquete de clases se puede encontrar también el paquete *ec.fin.banred.firmaelectrónica.clases.utilitarios* contiene las clases necesarias para recuperar el documento a firmar desde el repositorio documental, entre otras. Las clases dentro de este paquete son:

- *FiltroFormatosArchivo:* clase que permite filtrar el formato especificado de archivo sobre un cuadro de diálogo de tipo “Abrir documento”.
- *RecuperaDocumento*: clase que se encarga de recuperar el contenido del documento desde el repositorio documental mediante el uso de servicios web de tipo REST.
- *UtilitariosFecha*: clase que convierte la fecha actual del sistema en el formato especificado.

Las librerías del API de comunicación con servicios web tipo REST utilizadas en esta capa es parte del proyecto “Apache HTTP Commons” la cual está compuesta por los siguientes elementos:

- httpcore.4.2.1.jar.
- httpmime-4.2.1.jar.
- httpclient-4.2.1.jar.
- httpclient-cache-4.2.1.jar.

A continuación se presenta el código relevante de la clase *RecuperaDocumento*:

```
…
public class RecuperarDocumento {
        …
        public void bajarArchivo() {
            // crear directorio de destino en caso de que no exista
        File directorioTemporal = new File(this.carpetaTemporal);
        if (!directorioTemporal.exists())
            {
                if (!directorioTemporal.mkdir()){
                        throw new SecurityException("No se pudo crear la ruta " + this.carpetaTemporal);
                }
            }
            // Crea el archivo destino
            File archivoDescarga = new File(this.ubicacionDescargaArchivo);
        try {
                // establece la conexion con la url
                URLConnection conexion = new URL(this.url).openConnection();
                conexion.connect();
                // Abre los streams
                InputStream entrada = conexion.getInputStream();
                OutputStream salida = new FileOutputStream(archivoDescarga);
                int bytesLeidos = 0;
                while (bytesLeidos != -1) {
                        bytesLeidos = entrada.read();
                        if (bytesLeidos != -1) salida.write(bytesLeidos);
                }
                // Cierra los streams
                salida.close();
                entrada.close();
            } catch (…) { …}
        }
        …
}
```

- **Paquete de clases del repositorio documental ECM**

En este paquete se encuentran las clases encargadas de realizar el almacenamiento de la información generada por el aplicativo de firma electrónica, en este caso esta información no es guardada dentro de una base de datos como se lo hace tradicionalmente, dicha información es almacenada dentro del repositorio documental del sistema ECM debido a que el aplicativo genera documentos. Dentro de este paquete se encuentra la clase *CargaAlfresco* la cual se encarga de ejecutar el almacenamiento de datos del aplicativo mediante llamadas a servicios web de tipo REST.

Esta clase es la encargada de guardar el documento firmado electrónicamente generado por la aplicación dentro del repositorio documental, cabe recalcar que el aplicativo sube una nueva versión del documento con la firma electrónica. A continuación se presenta el código relevante de la clase CargaAlfresco.java:

```
public class CargaAlfresco {
	public void actualizarDocumento(String nodeRefDocumento, File documentoFirmado)
				throws ClientProtocolException, IOException{
		//ClienteHTTP
		DefaultHttpClient httpclient =  new DefaultHttpClient();
		//Autenticarse
		httpclient.getCredentialsProvider().setCredentials(
					new AuthScope(AuthScope.ANY_HOST, AuthScope.ANY_PORT),
					new UsernamePasswordCredentials(this.usuario, this.clave));
		//Llamada servicio POST
		String URL = "http://"+ this.servidor +":"+ this.puerto +"/Alfresco/service/api/upload";
		//Apuntar ticket de autenticacion
		if(this.ticket!=null) URL+="?alf_ticket=" + this.ticket;
		HttpPost httppost = new HttpPost(URL);
		//Definirlo como multiparte
		MultipartEntity reqEntity = new MultipartEntity();
		//Parametros a enviar
		…
		reqEntity.addPart("updatenoderef", new StringBody(nodeRefDocumento));
		//Archivo a subir
		reqEntity.addPart("filedata", new FileBody(documentoFirmado));
		httppost.setEntity(reqEntity);
		System.out.println("Ejecutando peticion: " + httppost.getRequestLine());
		//Ejecutar clienteHTTP con peticion POST
		HttpResponse response = httpclient.execute(httppost);
		//Recuperar la respuesta
		HttpEntity resEntity = response.getEntity();
		if (resEntity != null) {
			String page = EntityUtils.toString(resEntity);
			System.out.println("Respuesta :" + page);
			System.out.println("Codigo: " + response.getStatusLine().getStatusCode());
		}
	}
}
```

4.4.3 Construcción de componente Java para generación de documentos en formato .docx

A continuación se especifican los elementos que conforman el componente enfocado a la creación, modificación y actualización de documentos con formato .docx con desde el repositorio documental.

4.4.3.1 Paquete ec.fin.banred.alfresco.javascript

Este paquete contiene las clases necesarias para la recuperación de un documento con formato .docx desde el repositorio documental y la recuperación de los METADATOS de este documento. Luego de realizar esta recuperación se realiza una llamada a las clases dentro del paquete *ec.fin.banred.office.word.plantillas* para realizar la modificación del documento .docx. Una vez que se ha modificado el documento se lo envía hacia documento al repositorio documental. Este paquete contiene las siguientes clases:

- *ModificaDocumentoActaAccion: clase encargada de recuperar la información de los documentos de tipo "acta".*
- *ModificaDocumentoGenericoAccion: clase encargada de* recuperar la información de los documentos de tipo "oficio", "informe" y "memorando".

Estas clases mencionadas heredan de la clase *BaseProcessorExtension*, una de las clases que componen el "Foundation API" de Alfresco. El objetivo de la clase BaseProcessorExtension es presentar la funcionalidad de esta clase o sus subclases en un framework de trabajo basado en JavaScript dentro del sistema ECM, con el fin de facilitar o simplificar la llamada de funcionalidades dentro del repositorio documental.

A continuación se presenta el código relevante de la clase ModificaDocumentoActaAccion.

```
public class ModificaDocumentoActaAccion extends BaseProcessorExtension {
        …
        public void ejecutar(String nodeRef)
        {
                //Recuperar parametros del  JS script
                String referencia = nodeRef;
                NodeRef nodo = new NodeRef(referencia);
                //Mapa de parametros del contenido y valores de los metadatos para pasar a la
                //clase ModificaDocumento
                HashMap<String, String> valoresReemplazoDocumento;
                //Recuperar metadatos del documento vinculados con los parametros a reemplazar del
                //contenido del documento
                //String []metadatos = this.vinculaParametrosConMetadatosDocumento(nodo);
                valoresReemplazoDocumento = this.vinculaParametrosConMetadatosDocumento(nodo);
                //Llamar y leer al nodo solicitado dentro del repositorio
                byte[] binaryData = this.recuperaDocumentoAlfresco(nodo);
                //Crear el documento en la carpeta temporal del documento leido en Alfresco.
                String rutaDocumento = this.creaDocumentoTemporal(binaryData);
                //Recuperar asociaciones del documento **
                ArrayList <Participante> participantes = this.recuperaParticipantesActa(nodo);
                //Modificar archivos Office Word
                try {
                        this.modificaDocumento(rutaDocumento,
                                        valoresReemplazoDocumento, participantes);
                } catch (…) { … }
                this.subirDocumento(nodo, rutaDocumento);

        }
        private void modificaDocumento(String ruta, HashMap<String, String> vinculoParametrosConMetadatos,
                                ArrayList<Participante> participantes)
                                throws IOException, Docx4JException, JAXBException{
                //Llamara a clase para modificar documento memorando
                ModificaDocumentoActa doc =
                        new ModificaDocumentoActa(ruta, vinculoParametrosConMetadatos);
                //Primero insertar pq despues los ${} se remueven
                doc.insertaParticipantesDocumento(participantes);
                //Luego cambiar metadatos
                doc.modificaDocumento();
        }
}
```

4.4.3.2 Paquete ec.fin.banred.office.word.plantillas

Este paquete contiene las clases necesarias para la modificación de documentos con formato .docx, los cuales insertan los METADATOS del documento establecidos en el repositorio documental y reflejan esta información dentro del contenido del

documento. Para realizar esta modificación se utiliza la librería *DOCX4J* la cual presenta un API para Java que permite modificar documentos .docx.

Dentro de este paquete se presentan las siguientes clases:

- ModificaDocumentoActa, clase para modificar documentos de tipo acta.
- ModificaDocumentoGenerico, clase para modificar documentos de tipo oficio, informe y memorando.

A continuación se presenta el código relevante dentro de la clase ModificaDocumentoGenerico:

```
public class ModificaDocumentoGenerico{
	/** Recupera documento de ruta temporal */
	private WordprocessingMLPackage recuperaDocumento(String rutaDocumento)
                                                    throws IOException, Docx4JException{
	String inputfilepath = rutaDocumento;
		// Open a document from the file system
		// 1. Load the Package
		WordprocessingMLPackage wordMLPackage =
                                WordprocessingMLPackage.load(new java.io.File(inputfilepath));
		return wordMLPackage;
	}
	/** Reemplaza valores estaticos por metadatos de documento */
	public void modificaDocumento() throws IOException, Docx4JException, JAXBException
	{
		WordprocessingMLPackage wordMLPackage = this.recuperaDocumento(this.rutaDocumento);
		// 2. Fetch the document part
		MainDocumentPart documentPart = wordMLPackage.getMainDocumentPart();
		org.docx4j.wml.Document wmlDocumentEl =
                                (org.docx4j.wml.Document) documentPart.getJaxbElement();
		//xml --> string
		String xml = XmlUtils.marshaltoString(wmlDocumentEl, true);
		//valorize template
		Object obj = XmlUtils.unmarshallFromTemplate(xml, this.vinculoParametrosConMetadatos);
		//change  JaxbElement
		documentPart.setJaxbElement((Document) obj);
		this.guardaCambiosDocumento(wordMLPackage);

	}
	/** Guarda cambios realizados en el documento*/
	private void guardaCambiosDocumento(WordprocessingMLPackage wordMLPackage)  throws IOException,
		Docx4JException{
			SaveToZipFile saver = new SaveToZipFile(wordMLPackage);
			saver.save(this.rutaDocumento);
		}
	}
```

4.4.3.3 Paquete ec.fin.banred.office.word.utilitarios

Este paquete contiene clases utilitarias para soportar las funcionalidades de este componente. Dentro de este paquete se encuentran las clases FormatearTexto y Participante.

A continuación se presenta el código relevante de la clase FormatearTexto la cual devuelve la fecha de especificada en un formato entendible por el usuario el cual es reflejado dentro del documento.

```
public class FormatearTexto
{
	public static String formatearFecha(Date fecha)
	{
		//recuperar dia de la semana
		String diaSemana = "";
		switch (fecha.getDay())
		{
			case 0: diaSemana = "domingo"; break;
			case 1: diaSemana = "lunes"; break;
			case 2: diaSemana = "martes"; break;
			case 3: diaSemana = "miercoles"; break;
			case 4: diaSemana = "jueves"; break;
			case 5: diaSemana = "viernes"; break;
			case 6: diaSemana = "sabado"; break;
		}
		//recuperar dia del mes
		int diaMes = fecha.getDate(); //calendario.get(Calendar.DAY_OF_MONTH);
		String mes = "";
		switch(fecha.getMonth())
		{
			case 0: mes = "enero"; break;
			case 1: mes = "febrero"; break;
			case 2: mes = "marzo"; break;
			case 3: mes = "abril"; break;
			case 4: mes = "mayo"; break;
			case 5: mes = "junio"; break;
			case 6: mes = "julio"; break;
			case 7: mes = "agosto"; break;
			case 8: mes = "septiembre"; break;
			case 9: mes = "octubre"; break;
			case 10: mes = "noviembre"; break;
			case 11: mes = "diciembre"; break;
		}
		//recuperar año
		int anio = 1900 + fecha.getYear();//calendario.get(Calendar.YEAR);
```

```
                //devolver string
                String fechaTexto = /*diaSemana + " " + */diaMes + " de " + mes + " de " + anio;
        }
        @SuppressWarnings("deprecation")
        public static String formatearHora(Date fecha)
        {
                SimpleDateFormat printFormat = new SimpleDateFormat("HH:mm");
                String horaTexto = printFormat.format(fecha);
                return horaTexto;
        }
}
```

4.4.4 Configuración de plataforma ECM

En esta sección se presentará todas las configuraciones y adaptaciones realizadas a la plataforma ECM con el fin de alinear las funcionalidades de este sistema para soportar el proyecto en curso.

4.4.4.1 Configuración del repositorio documental

A continuación se establecen los parámetros necesarios activar las funciones del repositorio documental, entre estos parámetros se configuran el almacenamiento, búsqueda, notificaciones, autenticación, entre otros.

- **Sistema de archivos**

La plataforma ECM almacena los documentos del repositorio documental dentro de un sistema de archivos al que tenga acceso servidor. Para especificar el directorio donde se va a almacenar el repositorio hay que editar el archivo alfresco-global.properties que se encuentra en la ruta /opt/banred/tomcat/shared/classes y establecer la siguiente configuración:

```
dir.root=/opt/banred/alf_data
dir.contentstore=${dir.root}/contentstore
dir.contentstore.deleted=${dir.root}/contentstore.deleted
dir.auditcontentstore=${dir.root}/audit.contentstore
# Ubicación de los índices de Lucene
dir.indexes=${dir.root}/lucene-indexes
dir.indexes.backup=${dir.root}/backup-lucene-indexes
```

- **Base de datos**

Como complemento al sistema de archivos descrito anteriormente, la plataforma ECM utiliza una base de datos relacional para almacenar los METADATOS, la estructura documental, los permisos del repositorio documental, entre otros. Para que la plataforma ECM pueda trabajar con una el motor de base de datos MySQL, es necesario modificar el archivo alfresco-global.properties que se encuentra en la ruta /opt/banred/tomcat/shared/classes y añadir la siguiente configuración:

```
db.driver=org.gjt.mm.mysql.Driver
db.username=root
db.password=**********
db.name=alfresco42c
db.url=jdbc:mysql://localhost/${db.name}
db.pool.initial=10
db.pool.max=275
db.pool.validate.query=SELECT 1 FROM DUAL
db.pool.idle=-1
```

- **Motor de transformación**

En la plataforma ECM se puede transformar un documento de texto de un formato a otro, por ejemplo, un archivo .doc, .xls, .ppt a un archivo .pdf. Para tener acceso a estas acciones de transformación en la plataforma ECM, se debe instalar la aplicación LibreOffice. Esto es opcional y se puede hacer en cualquier momento después de que la plataforma ECM esté instalada. Para habilitar el motor de transformación se necesita configurar el archivo alfresco-global.properties que se encuentra en la ruta /opt/banred/tomcat /shared/classes y añadir la siguiente configuración:

```
ooo.exe=/opt/alfresco-4.2.c/libreoffice/program/soffice.bin
ooo.enabled=true
ooo.port=8100
jodconverter.enabled=false
jodconverter.officeHome=/opt/alfresco-4.2.c/libreoffice
jodconverter.portNumbers=8100
```

- **Correo electrónico**

Para que la plataforma ECM pueda enviar notificaciones por correo electrónico hay que agregar los siguientes parámetros en el archivo alfresco-global.properties ubicado en la ruta /opt/banred/tomcat /shared/classes y añadir la siguiente configuración:

```
mail.host=smtp.gmail.com
mail.port=465
mail.protocol=smtps
mail.username=yujo.kun.diego@gmail.com
mail.password=*********
mail.smtps.starttls.enable=true
mail.smtps.auth=true
### Test message when Alfresco starts ###
mail.testmessage.send=true
mail.testmessage.to=zannafar@me.com
mail.testmessage.subject=Outbound mail from Alfresco, server started.
mail.testmessage.text=Outbound SMTP email subsystem is working. Installed on ${dir.root}
```

- **Motor de búsqueda**

Cuando un documento se almacena en el repositorio documental pasa por un proceso de indexación de sus METADATOS y su contenido con el fin de recuperar estos documentos mediante búsquedas.

Esta indexación se ejecuta durante un determinado intervalo de tiempo, para este proyecto se asignaron los parámetros del motor de búsqueda más adecuados para las características del servidor. Dichas configuraciones son hechas en el archivo alfresco-global.properties ubicado en la ruta /opt/banred/tomcat/shared/classes y añadir la siguiente configuración:

```
index.subsystem.name=lucene
index.recovery.mode=AUTO
lucene.maxAtomicTransformationTime=100
lucene.query.maxClauses=10000
lucene.indexer.batchSize=1000000
lucene.indexer.cacheEnabled=true
lucene.indexer.maxDocIdCacheSize=100000
lucene.indexer.maxDocumentCacheSize=100
lucene.indexer.maxIsCategoryCacheSize=-1
lucene.indexer.maxLinkAspectCacheSize=10000
```

lucene.indexer.maxParentCacheSize=100000
lucene.indexer.maxPathCacheSize=100000
lucene.indexer.maxTypeCacheSize=10000
lucene.indexer.maxFieldLength=10000

- **Configuración autenticación con Directorio Activo**

Con el objetivo de evitar que los usuarios posean múltiples contraseñas, dentro de la institución, la plataforma ECM puede ser configurada para que reutilice el mismo nombre de usuario y contraseña creado previamente para un funcionario mediante el uso de sistemas de autenticación como Directorio Activo, OpenLDAP o Kerberos. En el proyecto actual se utiliza un servidor de autenticación con Directorio Activo. Para realizar esta configuración hay que realizar los siguientes pasos.

- **Configuración de autenticación**

Configurar la conexión de autenticación al servidor de directorio activo, creando el archivo ldap-ad-authentication.properties dentro de la siguiente ruta /opt/banred/tomcat/shared/classes/ alfresco/extension/subsystems/Authentication/ldap-ad/ldap-banred. Añadir los siguientes parámetros:

ldap.authentication.active=true
ldap.authentication.allowGuestLogin=true
ldap.authentication.userNameFormat=%s@applications.banred-test.fin.ec
ldap.authentication.java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
ldap.authentication.java.naming.provider.url=ldap://servidorldap:389
ldap.authentication.java.naming.security.authentication=simple
ldap.authentication.escapeCommasInBind=false
ldap.authentication.escapeCommasInUid=false
ldap.authentication.defaultAdministratorUserNames=Administrador

- **Configuración de sincronización de usuarios**

La plataforma ECM necesita crear una réplica de los usuarios y grupos definidos en el servidor de directorio activo, con el fin de especificar los roles y permisos que van a tener estos usuarios sobre las carpetas y documentos del repositorio documental. Para esta sincronización se necesita especificar el árbol de dominio, los atributos de

los usuarios y los grupos que se desea recuperar desde el directorio activo, para esto se deben modificar el archivo ldap-ad-authentication.properties especificado en el paso anterior y colocar los siguientes parámetros:

```
ldap.synchronization.active=true
ldap.synchronization.java.naming.security.authentication=simple
ldap.synchronization.java.naming.security.principal=administrador@applications.banred-test.fin.ec
ldap.synchronization.java.naming.security.credentials=********
ldap.synchronization.queryBatchSize=1000
ldap.synchronization.attributeBatchSize=1000
ldap.synchronization.groupQuery=(&(objectclass\=group)(description\=Alfresco))
ldap.synchronization.groupDifferentialQuery=(&(objectclass\=group)(description\=Alfresco)(!(whenChanged<\={0})))
ldap.synchronization.personQuery=(&(objectclass\=user)(userAccountControl\:1.2.840.113556.1.4.803\:\=512))
ldap.synchronization.personDifferentialQuery=(&(objectclass\=user)(userAccountControl\:1.2.840.113556.1.4.803\:\=512)(!(w
henChanged<\={0})))
ldap.synchronization.groupSearchBase=cn\=Users,dc\=applications,dc\=banred-test,dc\=fin,dc\=ec
ldap.synchronization.userSearchBase=cn\=Users,dc\=applications,dc\=banred-test,dc\=fin,dc\=ec
ldap.synchronization.modifyTimestampAttributeName=whenChanged
ldap.synchronization.timestampFormat=yyyyMMddHHmmss'.0Z'
ldap.synchronization.userIdAttributeName=sAMAccountName
ldap.synchronization.userFirstNameAttributeName=givenName
ldap.synchronization.userLastNameAttributeName=sn
ldap.synchronization.userEmailAttributeName=mail
ldap.synchronization.userOrganizationalIdAttributeName=company
ldap.synchronization.defaultHomeFolderProvider=largeHomeFolderProvider
ldap.synchronization.groupIdAttributeName=cn
ldap.synchronization.groupDisplayNameAttributeName=name
ldap.synchronization.groupType=group
ldap.synchronization.personType=user
ldap.synchronization.groupMemberAttributeName=member
ldap.synchronization.enableProgressEstimation=true
ldap.synchronization.br.user.jobTitle=title
ldap.synchronization.br.user.department=department
ldap.synchronization.br.user.personalPhone=homePhone
ldap.synchronization.br.user.mobile=mobile
ldap.synchronization.br.user.roleCode=postalCode
ldap.synchronization.br.user.companyPhone=telephoneNumber
ldap.synchronization.br.user.companyFax=facsimileTelephoneNumber
```

- **Configuración de ejecución de tarea programada de sincronización**

Una vez definida la sincronización se necesita configurar a la plataforma ECM para que ejecute cada determinado intervalo de tiempo esta tarea, en este caso se ha definido que se ejecute esta tarea cada 10 minutos. Para estas configuraciones se necesita agregar los siguientes parámetros en el archivo alfresco-global.properties dentro de la ruta /opt/banred/tomcat/shared/classes/alfresco:

```
synchronization.synchronizeChangesOnly=true
synchronization.import.cron=0 0,10,20,30,40,50 * * * ?
synchronization.syncWhenMissingPeopleLogIn=true
synchronization.syncOnStartup=false
synchronization.autoCreatePeopleOnLogin=true
synchronization.loggingInterval=100
synchronization.workerThreads=2
synchronization.allowDeletions=true
```

- **Cadena de autenticación**

La plataforma ECM puede configurarse para utilizar varios servidores de autenticación, mediante la definición de una cadena de autenticación, para definir esta configuración se necesita modificar el archivo alfresco-global.properties ubicado en dentro de la ruta /opt/banred/tomcat/shared/classes/alfresco:

```
### Cadena autenticacion
authentication.chain=alfrescoNtlm1:alfrescoNtlm,ldap-banred:ldap-ad
```

La nomenclatura de la cadena de autenticación se la realiza de la siguiente manera: *nombre de autenticación seguida de dos puntos y finalmente se especifica el tipo de autenticación*. En la cadena definida anteriormente *alfrescoNtlm* hace referencia al tipo de autenticación propio de la plataforma ECM, además se añade a la cadena la autenticación de tipo *ldap-ad* con nombre *ldap-banred*. Si se usa más de un tipo de autenticación se lo separa mediante una coma. Como se puede ver en la configuración detallada anteriormente se está utilizando la autenticación definida por defecto en la plataforma ECM y la autenticación de Directorio Activo.

- **Auditoría**

Se ha configurado a la plataforma ECM para que realice una auditoría sobre los documentos que se encuentran dentro del repositorio documental mediante la creación de archivos XML de registro de transacciones. Para la auditoría se establece los siguientes archivos dentro de la ruta /opt/banred/tomcat/shared/classes/alfresco/ extension/audit:

- lecturaDocumentos.xml, el cual contiene la configuración de auditoría sobre la lectura de documentos.
- creacionDocumentos.xml, contiene la configuración para registrar la auditoría de creación de documentos en el repositorio documental.
- modificarMetadatosDocumentos.xml, dentro de este archivo se establece las configuraciones para registrar los cambios de METADATOS sobre los documentos del repositorio documental.
- modificarDocumentos.xml, este archivo contiene la configuración para registrar los cambios sobre los documentos del repositorio documental.
- eliminacionDocumentos.xml, permite registrar los documentos que son eliminados dentro del repositorio documental.

Dentro de la configuración dentro de estos archivos de auditoría se necesita especificar los extractores de información, el componente que se va a auditar y que valores de auditoría se requieren recuperar, a continuación se presenta el contenido más relevante dentro del archivo leerArchivo.xml:

```
<Audit
   xmlns="http://www.alfresco.org/repo/audit/model/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.alfresco.org/repo/audit/model/3.2 Alfresco-audit-3.2.xsd">

<DataExtractors>
<DataExtractor name="simpleValue" registeredName="auditModel.extractor.simpleValue"/>
</DataExtractors>

<PathMappings>
<PathMap source="/Alfresco-api/post/ContentService/getReader" target="/LeerArchivo"/>
</PathMappings>

<Application name="LeerArchivo" key="LeerArchivo">
<AuditPath key="args">
        <AuditPath key="nodeRef">
<RecordValue key="referenciaDocumento" dataExtractor="simpleValue"/>
</AuditPath>
        </AuditPath>
<AuditPath key="result">
<RecordValue key="value" dataExtractor="simpleValue"/>
</AuditPath>
</Application>
</Audit>
```

- **METADATOS personalizados sobre documentos del repositorio documental**

Para agregar METADATOS personalizados en el repositorio documental se utiliza un concepto llamado “modelos de contenido”, el cual es usado para representar los objetos del negocio, los cuales tienen propiedades y características que pueden heredar de un tipo base. La personalización de modelos dentro de la plataforma ECM está limitada únicamente por la imaginación y los requerimientos del negocio. Factura, receta médica o película serían ejemplos de contenidos personalizados.

Siendo el sistema Alfresco un sistema flexible para desarrollar aplicaciones de gestión de contenidos, los modelos de contenido proveídos por la plataforma ECM son genéricos, de hecho, para necesidades de gestión documental básicas probablemente es suficiente. Sin embargo Pero se puede aprovechar la potencia y funcionalidad de tener un modelo personalizado a las necesidades del negocio. Para la creación de un modelo de contenido se deben realizar los siguientes pasos.

- **Definir modelos de contenido de la institución**

Consiste en definir los tipos de contenido a utilizar en la implementación, de estos tipos se especificará qué propiedades o METADATOS y asociaciones conforman. En este proyecto se necesita puntualmente lo siguiente:

Tabla 14. Tipos de documentos de la institución

Documento	Tipo de dato	Nombre METADATO
Memorando	String	Número Memorando
Memorando	String	Lugar
Memorando	Date	Fecha
Memorando	String	Receptor
Memorando	String	Cargo Receptor
Memorando	String	Emisor
Memorando	String	Cargo Emisor
Memorando	String	Referencias/Tema
Informe Interno	String	Número Informe

Documento	Tipo de dato	Nombre METADATO
Informe Interno	String	Lugar
Informe Interno	Date	Fecha
Informe Interno	String	Receptor
Informe Interno	String	Cargo Receptor
Informe Interno	String	Emisor
Informe Interno	String	Cargo Emisor
Informe Interno	String	Referencias/Tema
Oficio	String	Número oficio
Oficio	String	Lugar
Oficio	Date	Fecha
Oficio	String	Título receptor
Oficio	String	Receptor
Oficio	String	Cargo receptor
Oficio	String	Institución receptor
Oficio	String	Emisor
Oficio	String	Cargo emisor
Acta reuniones	String	Número acta
Acta reuniones	String	Responsable reunión
Acta reuniones	String	Cargo responsable reunión
Acta reuniones	String	Tipo reunión
Acta reuniones	String	Tema reunión
Acta reuniones	String	Lugar
Acta reuniones	Date	Fecha convocatoria
Acta reuniones	Integer	Tiempo estimado
Acta reuniones	User	Participantes

Fuente. Pedro Caicedo y Diego Godoy

Como se puede observar, existen METADATOS similares entre los cuatro tipos de documentos, dentro de la plataforma ECM para modelar estos documentos es posible definir un “aspecto” el cual agrupa un conjunto de METADATOS con el fin de permitir el re uso entre los diferentes tipos de contenidos.

- **Extender el modelo de contenido de la plataforma ECM**

Una vez conocido qué tipos documentales se utilizarán en la institución, el siguiente paso es extender el modelo de contenido de la plataforma ECM con estos nuevos tipos de documentos. Un modelo de contenido se lo especifica dentro de un archivo XML. Este modelado de documentos en el proyecto actual está ubicado dentro de el siguiente directorio /opt/banred/tomcat/shared/classes/alfresco/extension/banred, dentro del archivo modeloDocumentos.xml. Dentro de este modelo se puede destacar la siguiente información:

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="banred:modeloContenidobanred" xmlns="http://www.alfresco.org/model/dictionary/1.0">

<description>Modelo banred</description>
<author>Diego Godoy - Pedro Caicedo</author>
<version>1.0</version>

<imports>
<!-- Importar definicion de modelo de diccionario de Alfresco -->
<import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d"/>
<!-- Importar definicion de modelo de contenido de Alfresco -->
<import uri="http://www.alfresco.org/model/content/1.0" prefix="cm"/>
<!-- Importar definicion de modelo de listas de Alfresco -->
<import uri="http://www.alfresco.org/model/datalist/1.0" prefix="dl"/>
</imports>

<namespaces>
<namespace uri="http://www.banred.fin.ec/modelo/documentos/1.0" prefix="banred"/>
</namespaces>

<types>
<type name="banred:documentoGenerico">
<parent>cm:content</parent>
</type>
<type name="banred:memorando">
<parent>banred:documentoGenerico</parent>
<mandatory-aspects>
<aspect>banred:datosIdentificacionDocumento</aspect>
<aspect>banred:datosUbicacion</aspect>
<aspect>banred:datosTiempo</aspect>
<aspect>banred:datosReceptor</aspect>
<aspect>banred:datosEmisor</aspect>
<aspect>banred:datosReferenciaDocumento</aspect>
</mandatory-aspects>
</type>
    …
```

```
    …
<type name="banred:acta">
<parent>banred:documentoGenerico</parent>
<mandatory-aspects>
<aspect>banred:datosIdentificacionDocumento</aspect>
<aspect>banred:datosResponsableReunion</aspect>
<aspect>banred:datosGeneralesReunion</aspect>
<aspect>banred:datosUbicacion</aspect>
<aspect>banred:datosTiempo</aspect>
<!--aspect>banred:datosParticipantesReunion</aspect-->
<aspect>banred:participantes</aspect>
<aspect>banred:datosDuracionReunion</aspect>
</mandatory-aspects>
</type>
</types>
<aspects>
<aspect name="banred:datosUbicacion">
<properties>
<property name="banred:lugar">
<title>Lugar</title>
<description>Lugar de emision o creacion del documento</description>
<type>d:text</type>
</property>
</properties>
</aspect>
<aspect name="banred:datosTiempo">
<properties>
<property name="banred:fecha">
<title>Fecha documento</title>
<description>Fecha y hora (opcional) de emision o creacion del documento</description>
<type>d:date</type>
</property>
</properties>
</aspect>
    …
</aspects>
</model>
```

- **Especificar etiquetas**

Se necesita crear un archivo de etiquetas dentro del directorio de configuración de extensión el cual almacene los títulos y las descripciones de los elementos que contenga el archivo XML creado en el paso anterior, para lo cual se creó un archivo llamado banred.properties dentro de la ruta /opt/banred/tomcat/shared/classes/alfresco/extension/banred. En el contenido de este archivo se puede presentar las siguientes etiquetas relevantes:

```
#Etiquetas tipos de documentos
banred_modeloContenidobanred.type.banred_memorando.title=Documento memorando
banred_modeloContenidobanred.type.banred_informe.title=Documento informe
banred_modeloContenidobanred.type.banred_oficio.title=Documento oficio
banred_modeloContenidobanred.type.banred_acta.title=Documento acta
#Etiquetas de metadatos
banred_modeloContenidobanred.property.banred_lugar.title=Lugar
banred_modeloContenidobanred.property.banred_fecha.title=Fecha del documento
banred_modeloContenidobanred.property.banred_nombreReceptor.title=Nombre del receptor
banred_modeloContenidobanred.property.banred_cargoReceptor.title=Cargo del receptor
banred_modeloContenidobanred.property.banred_tituloReceptor.title=T\u00edtulo del receptor
banred_modeloContenidobanred.property.banred_numeroDocumento.title=N\u00famero del documento
```

- **Crear archivo de contexto que importe archivos de configuración a la plataforma ECM**

Es necesario tener en cuenta que el archivo generado para definir el modelo de contenido de la institución debe ir acompañado de un fichero de contexto, el cual se encarga de publicar el modelo mientras está en ejecución. Para lo cual, se ha creado un archivo llamado banredModel-context.xml, ubicado en la ruta /opt/banred/tomcat/shared/classes/alfresco/extension. El contenido de este archivo es el siguiente:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN' 'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>
<bean id="modelo.banred.carga" parent="dictionaryModelBootstrap" depends-on="dictionaryBootstrap">
<property name="models">
<list>
<value>alfresco/extension/banred/modeloDocumentos.xml</value>
</list>
</property>
</bean>
</beans>
```

- **Modelo de procesos**

El la plataforma ECM integra un motor de BPM, el cual aplica flujos de trabajo profesionales y avanzados. Este motor puede utilizarse para realizar circuitos de validación complejos. Los flujos de trabajo permiten definir tareas en serie y en

paralelo. Además, se pueden establecer "alarmas" para gestionar la finalización de determinadas tareas. En este caso se creó un proceso que administre el ciclo de vida de los documentos dentro de la institución, desde la creación hasta la firma electrónica de los mismos.

El modelo de procesos es el conjunto de tipos de contenidos que a diferencia del "modelo de contenidos" estos están enfocados a representar los formularios del cliente que se desea presentar en el proceso, es decir este modelo de procesos tiene la estructura muy similar al modelo de documentos creado anteriormente. Este es un punto muy importante cuando se crea un proceso en el sistema, por lo general por cada proceso que se despliega en el sistema existe un modelo de procesos que contiene los formularios de este proceso. Para la generación de estos formularios se han definido 3 modelos de procesos que se detallarán en el siguiente punto. Estos modelos se encuentran el la ruta /opt/banred/tomcat/shared/classes/alfresco/extension/banred/formularios y son los siguientes:

- 01-procesoGestionDocumental/ gestionDocumentalFormularios.xml
- 02-procesoCrearDocumento/crearDocumentoFormularios.xml.
- 03-procesoFirmarDocumento/firmarDocumentoFormularios.xml

A continuación se presenta el código más relevante que presente la configuración de estos formularios:

```
<?xml version="1.0" encoding="UTF-8"?>

<model name="ges:modeloProcesogesocumento"  xmlns="http://www.alfresco.org/model/dictionary/1.0">

<!-- Importar modelos definidos en Alfresco -->
<imports>
<import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d" />
<import uri="http://www.alfresco.org/model/bpm/1.0" prefix="bpm" />
<import uri="http://www.banred.fin.ec/modelo/documentos/1.0" prefix="banred" />
</imports>

<!-- Identificador del documento, uri => Nombre Largo, prefix => Nombre corto -->
<namespaces>
<namespace uri="http://www.banred.fin.ec/procesos/gestiondocumental/tareas" prefix="ges" />
</namespaces>
```

```
<types>
<!-- Formulario de tareas de proceso firmar documento  -->
<type name="ges:iniciarProceso">
<parent>bpm:startTask</parent>
<properties>
                        <property name="ges:tipoDocumento">
                                <type>d:text</type>
                                <constraints>
                                        <constraint ref="ges:opcionesTipoDocumento" />
                                </constraints>
                        </property>
</properties>
</type>

<!-- Formulario de revision de documento  -->
<type name="ges:revisarDocumento">
<parent>bpm:workflowTask</parent>
<properties>
                        <property name="ges:documentoAprobado">
                                <type>d:text</type>
                                <constraints>
                                        <constraint ref="ges:opcionesRevisarDocumento" />
                                </constraints>
                        </property>
</properties>
<mandatory-aspects>
<aspect>banred:datosTramite</aspect>
</mandatory-aspects>
</type>

<!-- Formulario de seleccion de corregir documento -->
<type name="ges:corregirDocumentoPorUsuario">
<parent>bpm:workflowTask</parent>
<properties>
                        <property name="ges:corregirDocumentoPorUsuario">
                                <type>d:text</type>
                                <constraints>
                                        <constraint ref="ges:opcionesRevisarDocumento" />
                                </constraints>
                        </property>
</properties>
<mandatory-aspects>
<aspect>banred:datosTramite</aspect>
</mandatory-aspects>
</type>

<type name="ges:corregirDocumento">
<parent>bpm:workflowTask</parent>
<mandatory-aspects>
<aspect>banred:datosTramite</aspect>
```

</mandatory-aspects>
</type>
</types>
</model>

4.4.4.2 Despliegue de nuevos procesos

Para diseñar y desplegar procesos en el motor BPM Activiti es necesario crear archivos XML que implementen el estándar BPMN 2.0. Como breve resumen, este estándar engloba el diagrama del proceso que norma el la secuencia del trámite y la imagen o diagrama que representa proceso dentro de un único archivo.

- **Creación de archivos de definición de procesos**

Para el proceso de gestión documental implementado en la institución se han creado tres diferentes archivos de procesos, un proceso principal llamado "Gestión documental" y dos subprocesos llamados "Crear documento" y "Firma electrónica" que complementan al proceso principal. Estos procesos han sido generados dentro de la siguiente ruta /opt/banred/tomcat/shared/classes/alfresco/extension/banred/procesos/GestionDocumental. Esta carpeta contiene los siguientes archivos de proceso:

- 01-GestionDocumental.bpmn20.xml.
- 02-CrearDocumento.bpmn20.xml.
- 03-FirmaDigital.bpmn20.xml.

A continuación se presenta el código XML relevante del proceso principal *01-GestionDocumental.bpmn20.xml*. Como primer punto se presentará la llamada al formulario inicial:

```
<startEvent id="inicio" name="Inicio" activiti:formKey="ges:iniciarProceso"></startEvent>
```

Llamada a tareas del proceso, como se puede visualizar, dentro de estas tareas de usuario se hace una llamada al formulario que se encuentra dentro del modelo de procesos definidos anteriormente, las cuales se presentan a continuación:

```
<userTask id="revisarDocumentoGerente" name="Revisar documento" activiti:assignee="${usuarioGerente}" activiti:formKey="ges:revisarDocumento">
…
<userTask id="CorregirDocumentoResponsableProceso" name="Corregir documento" activiti:assignee="${usuarioResponsableProceso}" activiti:formKey="ges:corregirDocumento">
…
<userTask id="CorregirDocumentoCoordinador" name="Corregir documento" activiti:assignee="${usuarioCoordinador}" activiti:formKey="ges:corregirDocumento">
…
<userTask id="RevisarDocumentoResponsableProceso" name="Revisar documento" activiti:assignee="${usuarioResponsableProceso}" activiti:formKey="ges:revisarDocumento">
```

Llamada a subprocesos, se puede visualizar que en esta llamada se especifican las variables que se envían hacia el subproceso y las variables que se recuperan desde el subproceso como se muestra a continuación:

```
<callActivity id="callCrearDocumento" name="Crear documento" calledElement="processCrearDocumento">
<extensionElements>
<activiti:in source="bpm_package" target="bpm_package"></activiti:in>
<activiti:in source="companyhome" target="companyhome"></activiti:in>
<activiti:in source="initiator" target="initiator"></activiti:in>
<activiti:in source="logger" target="logger"></activiti:in>
<activiti:in source="search" target="search"></activiti:in>
<activiti:in source="session" target="session"></activiti:in>
<activiti:in source="modificainforme" target="modificainforme"></activiti:in>
<activiti:in source="bpm_workflowDescription" target="bpm_workflowDescription"></activiti:in>
<activiti:in source="bpm_workflowDueDate" target="bpm_workflowDueDate"></activiti:in>
<activiti:in source="bpm_workflowPriority" target="bpm_workflowPriority"></activiti:in>
<activiti:in source="bpm_sendEmailNotifications" target="bpm_sendEmailNotifications"></activiti:in>
<activiti:in source="ges_tipoDocumento" target="ges_tipoDocumento"></activiti:in>
<activiti:in source="workflowinstanceid" target="numeroTramite"></activiti:in>
<activiti:out source="bpm_package" target="bpm_package"></activiti:out>
<activiti:out source="banred_lugar" target="banred_lugar"></activiti:out>
<activiti:out source="banred_fecha" target="banred_fecha"></activiti:out>
<activiti:out source="banred_responsableReunion" target="banred_responsableReunion"></activiti:out>
<activiti:out source="banred_cargoResponsableReunion"
        target="banred_cargoResponsableReunion"></activiti:out>
<activiti:out source="banred_tipoReunion" target="banred_tipoReunion"></activiti:out>
<activiti:out source="banred_temaReunion" target="banred_temaReunion"></activiti:out>
<activiti:out source="banred_duracionReunion" target="banred_duracionReunion"></activiti:out>
<activiti:out source="banred_participantesDeReunion" target="banred_participantesDeReunion"></activiti:out>
</extensionElements>
</callActivity>
```

Para utilizar variables entre la tarea y el proceso en general, se puede especificar acciones a ejecutar dentro de los diferentes eventos de la tarea. A continuación se presenta el código relevante para realizar esta recuperación:

```
<userTask id="RevisarDocumentoResponsableProceso" name="Revisar documento"
        activiti:assignee="${usuarioResponsableProceso}" activiti:formKey="ges:revisarDocumento">
<extensionElements>
<activiti:taskListener event="complete" class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
<activiti:field name="script">
<activiti:string>execution.setVariable('ges_documentoAprobado',
                task.getVariable('ges_documentoAprobado'));</activiti:string>
</activiti:field>
</activiti:taskListener>
<activiti:taskListener event="create" class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
<activiti:field name="script">
<activiti:string>//Recuperar documento
                var documento = bpm_package.children[0];
                //Dar permiso al usuario
                documento.setPermission("Consumer",usuarioResponsableProceso);
                //Presentar el ID del tramite
                task.setVariable('banred:numeroTramite', workflowinstanceid.replace("activiti$",""));
                //Fechas de expiracion
                if (typeof bpm_workflowDueDate != 'undefined') task.dueDate = bpm_workflowDueDate;
                if (typeof bpm_workflowPriority != 'undefined') task.priority = bpm_workflowPriority;</activiti:string>
</activiti:field>
<activiti:field name="runAs">
<activiti:string>admin</activiti:string>
</activiti:field>
</activiti:taskListener>
</extensionElements>
</userTask>
```

Selección de diferentes rutas del proceso mediante condiciones basadas en información recolectada por el cliente, a continuación se presenta la recuperación de variables y evaluación del camino del proceso mediante el valor recuperado:

```
<exclusiveGateway id="gatewayCorregirDocumentoResponsableProceso" name="Corregir
documento?"></exclusiveGateway>
<sequenceFlow id="flow28" name="Si" sourceRef="gatewayCorregirDocumentoResponsableProceso"
targetRef="CorregirDocumentoResponsableProceso">
                <conditionExpression xsi:type="tFormalExpression">
                <![CDATA[${ges_documentoAprobado=='SI'}]]></conditionExpression>
</sequenceFlow>
<sequenceFlow id="flow29" name="No" sourceRef="gatewayCorregirDocumentoResponsableProceso"
targetRef="scriptNotificarNoConformidadResponsableProceso">
                <conditionExpression xsi:type="tFormalExpression">
                <![CDATA[${ges_documentoAprobado=='NO'}]]></conditionExpression> </sequenceFlow>
```

Notificaciones a correo electrónico mediante la ejecución de tareas automáticas del sistema:

```
<serviceTask id="scriptNoConformidad" name="Notificar no conformidad"
        activiti:class="org.alfresco.repo.workflow.activiti.script.AlfrescoScriptDelegate">
<extensionElements>
<activiti:field name="script">
<activiti:string>logger.log("Enviar mail");
        var documento = bpm_package.children[0];
        try {
            var subject = "[PRUEBA] Documento no aprobado por el gerente";
            var emailTemplatePath = "Data Dictionary/Email Templates/Notify Email Templates";
            var emailTemplate = companyhome.childByNamePath(emailTemplatePath +
        "/notify_user_email_es.html.ftl");
            var emailFrom = initiator.properties.email;
            var mail = actions.create("mail");
            mail.parameters.to = "zannafar@me.com"; //"test." + usuarioResponsableProceso +
        "@magmasoft.com.ec";
            mail.parameters.subject = subject;
            mail.parameters.from = emailFrom;
            mail.parameters.template = emailTemplate;
            // placeholder body text; the template will have the message
            mail.parameters.text = "Tarea requiere accion";
            mail.execute(documento);
        }catch(ex)
        {}
</activiti:string>
</activiti:field>
</extensionElements>
</serviceTask>
```

- **Archivo de contexto para carga de archivos de configuración de procesos**

Es necesario tener en cuenta que los archivos XML generados con las definiciones de procesos y los modelos de los procesos que contienen los formularios de usuario de los procesos, necesitan ser publicados mediante un archivo de contexto de igual manera que se cargaron los modelos de contenido en la plataforma ECM. Para realizar esta publicación lo cual, se ha creado un archivo llamado banred-procesos-context.xml, ubicado en la ruta /opt/banred/tomcat/shared/classes/alfresco/extension. El contenido de este archivo es el siguiente:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN' 'http://www.springframework.org/dtd/spring-beans.dtd'>
<beans>
<bean id="cargaProcesoGestionDocumental" parent="workflowDeployer">
```

```
<property name="workflowDefinitions">
<!-- Ruta referente al archivo(s) bpmn20.xml -->
<list>
                        <props>
                                <prop key="engineId">activiti</prop>
                                <prop key="location">alfresco/extension/banred/procesos/ GestionDocumental/01-
                                        GestionDocumental.bpmn20.xml</prop>
                                <prop key="mimetype">text/xml</prop>
                                <prop key="redeploy">false</prop>
                        </props>
                        <props>
                                <prop key="engineId">activiti</prop>
                                <prop key="location">alfresco/extension/banred/procesos/GestionDocumental/02-
                                        CrearDocumento.bpmn20.xml</prop>
                                <prop key="mimetype">text/xml</prop>
                                <prop key="redeploy">false</prop>
                        </props>
                        <props>
                                <prop key="engineId">activiti</prop>
                                <prop key="location">alfresco/extension/banred/procesos/GestionDocumental/03-
                                        FirmaDigital.bpmn20.xml</prop>
                                <prop key="mimetype">text/xml</prop>
                                <prop key="redeploy">false</prop>
                        </props>
</list>
</property>
<!-- DEFINIR EL MODELO DE TAREAS (FORMULARIOS)-->
<property name="models">
<list>
<value>alfresco/extension/banred/formularios/01-procesoGestionDocumental/gestionDocumentalFormularios.xml
</value>
<value>alfresco/extension/banred/formularios/02-procesoCrearDocumento/crearDocumentoFormularios.xml</value>
<value>alfresco/extension/banred/formularios/03-procesoFirmarDocumento/firmarDocumentoFormularios.xml
        </value>
</list>
</property>
<property name="labels">
<list>
<value>alfresco/extension/banred/banred</value>
</list>
</property>
</bean>
</beans>
```

4.4.4.3 Interoperabilidad con otros sistemas mediante exposición de servicios web

La plataforma ECM dispone de diferentes APIs para el desarrollo de extensiones, componentes e interacción con otras aplicaciones. En este caso para la interoperabilidad deseada se hace el uso de servicios web los cuales proporcionan una API de servicios para documentos del repositorio documental accesibles vía REST, es decir mediante un URL de acceso a la plataforma ECM, se expone la funcionalidad para la gestión de documentos que presenta el repositorio documental.

- **Verificar usuario dentro del portal**

Se creó un servicio REST que verifica si existe el usuario solicitado en el URL dentro de la persistencia de usuarios del portal, además este servicio devuelve el cargo del usuario dentro del portal (administrador o funcionario). Los archivos necesarios para la exposición de este servicio se presentan a continuación:

El URL de acceso al servicio web REST para esta verificación se encuentra en la siguiente ruta:
/opt/banred/tomcat/shared/classes/alfresco/extension/templates/webscripts/ec/fin/banred/portal/usuarios.get.desc.xml

```
<webscript>
<shortname>Usuarios portal BANRED</shortname>
<description></description>
<url>/ec/fin/banred/portal/usuarios?nombreusuario={userName}</url>
<format default="json">any</format>
<family>BANRED</family>
</webscript>
```

El código de verificación de los datos del usuario que ejecuta este servicio web es el siguiente:

```
public class UsuariosPortalBanred extends AbstractWebScript
{
    …
        public void execute(WebScriptRequest req, WebScriptResponse res) throws IOException
        {
```

```
                    // NOTA: Este web script debe ser ejecutado en un ambiente HTTP Servlet
                    /*if (!(req instanceof WebScriptServletRequest)) {
                    throw new WebScriptException(
                        "La recuperacion de contenido debe ser ejecutada en un ambiente HTTP Servlet");
                    }
                    HttpServletRequest httpReq = ((WebScriptServletRequest)req).getHttpServletRequest();
                    // recuperar parametros URL
                    String expediente = httpReq.getParameter("numeroExpediente");*/
                    String nombreUsuario = req.getParameter("nombreusuario");
                    ConexionBD datos = new ConexionBD();
                    ArrayList<Usuario> usuarios = null;
                    try {
                            usuarios = datos.consultarDatos(nombreUsuario);
                    } catch (SQLException e) {
                            System.out.println("Error al ejecutar la conexion con la base de datos");
                            e.printStackTrace();
                    } catch (ClassNotFoundException e) {
                            e.printStackTrace();
                    }
                    String respuesta ="";
                    if(usuarios != null)
                    {
                            if(usuarios.size()>0)
                            {
                                    respuesta = …
                        }
            }
            …
}
```

Para finalizar, se necesita definir un archivo de contexto que cargue la clase Java en el repositorio documental. El archivo generado para realizar esta carga es banred-webscripts-context.xml, el cual se encuentra en la siguiente ruta: /opt/banred/tomcat/shared/classes /alfresco/extension. La configuración de este archivo se presenta a continuación:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN' 'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>
<!-- Register the Java-backed web scripts -->
<bean id="webscript.ec.fin.banred.portal.usuarios.get"
class="ec.fin.banred.alfresco.webscripts.UsuariosPortalBanred"
parent="webscript">
</bean>
</beans>
```

- **Firma electrónica**

Se creó un servicio web REST que permita la presentación del Applet que está embebido dentro del servidor el cual contiene el aplicativo de firma electrónica. Este servicio contiene tres componentes ubicados dentro de la ruta: /opt/banred/tomcat/shared/classes/alfresco/extension/ templates/webscripts/firma, los cuales se presenta a continuación:

Este servicio web se ejecuta mediante la llamada al siguiente URL relativo /banred/firma/custom/documento?nodeRef={nodeRef} dentro de la plataforma ECM. Se presenta a continuación el código necesario para presentar el Applet del aplicativo de firma dentro de una página HTML básica, este código se creó dentro del archivo firma.get.html.ftl en la ruta descrita anteriormente.

```
<#assign
urlDocumento="api/node/content/${documento.nodeRef.storeRef.protocol}/${documento.nodeRef.storeRef.identifier}/${documento.nodeRef.id}
/${documento.name?url}">
<html>
<head>
        <title>Applet de Firma.</title>
</head>
<body>
        <div id="areaAppletFirma"></div>
        <script type="text/javascript">//<![CDATA[
        var SERVER_URI = window.location.protocol + "//" + window.location.host;
        var SERVER_NAME = window.location.host.substring(0,window.location.host.indexOf(":"));
        var SERVER_PORT = window.location.host.substring(window.location.host.indexOf(":") + 1);
        var SHARE_PROXY_URI = SERVER_URI + "/share/proxy/Alfresco/";
        document.getElementById("areaAppletFirma").innerHTML = "<APPLET id=\"appletFirmaElectronica\"" +
"codebase=\""+ SERVER_URI +"/Alfresco/firma/Librerias/\" " +
"code=\"com.firmaelectrónica.interfaz.AppletFirma\"" +
"archive=\"firmaDigital.jar,bcprov-jdk16-143.jar,commons-codec-1.6.jar,commons-logging-1.1.1.jar,fluent-hc-4.2.1.jar,httpclient-
4.2.1.jar,httpclient-cache-4.2.1.jar,httpcore-4.2.1.jar,httpmime-4.2.1.jar,iText-2.1.0.jar,sunpkcs11.jar\"" +
                                                                                "width=700 height=480>" +
        "<param name =\"servidor\" value = \""+ SERVER_NAME +"\" />" +
        "<param name =\"puerto\" value = \""+ SERVER_PORT +"\" />" +
        "<param name=\"ticket\" value=\"${session.ticket}\">" +
        "<param name =\"nodeRef\" value = \"${documento.nodeRef}\" />" +
        "<param name=\"pathsinfirmar\" value=\""+ SERVER_URI +"/Alfresco/s/${urlDocumento}\">" +
                                                                                "</APPLET>";
        //]]></script>
</body>
</html>
```

4.4.4.4 Implementación de componente de creación de documentos .docx.

Dentro de la plataforma ECM, mediante el uso del componente Java para generación de documentos en formato .docx descrito en el punto 4.4.3, se agrega la funcionalidad al repositorio documental de generar documentos con formato .docx y reflejar los METADATOS especificados en el proceso documental por parte de los usuarios dentro del contenido de estos documentos.

Para implementar esta funcionalidad se realizaron los siguientes pasos:

Como primer punto se genera un archivo llamado banredModificarDocumentoAccion.jar, el cual es un contenedor de aplicaciones Java y se lo desplegó dentro de la plataforma ECM en la ruta /opt/banred/tomcat/webapps/alfresco/WEB-INF/lib el cual contiene todos los elementos del componente de creación de documentos.

Como segundo punto, de manera similar al de los modelos de contenido, se necesita crear una archivo de contexto que cargue esta funcionalidad al entorno de ejecución de la plataforma ECM, para lo cual se creó un archivo llamado banred-actions-context.xml ubicado dentro de la ruta /opt/banred/tomcat/shared/classes/alfresco/extension, el cual contiene el siguiente código relevante:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN' 'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>
<bean id="modificaDocumentosMemorandoScript" parent="baseJavaScriptExtension"
                class="ec.fin.banred.alfresco.javascript.ModificaDocumentoGenericoAccion">
<property name="extensionName">
<value>modificamemorando</value>
</property>
<property name="contentService" ref="contentService" />
<property name="searchService" ref="searchService" />
<property name="nodeService" ref="NodeService" />
<property name="metadatos">
        <value>${banred.documentos.memorando.metadatos}</value>
</property>
<property name="parametros">
```

```
        <value>${banred.documentos.memorando.parametros}</value>
</property>
</bean>
  …
</beans>
```

De manera complementaria a esta funcionalidad, se especifica en el archivo alfresco-global.properties la relación entre los METADATOS que se recuperan desde el repositorio documental y las variables dentro del contenido .docx, por ejemplo, el valor del METADATO con nombre *banred: numeroDocumento* se escribirá en la variable *numero_documento* dentro del contenido del archivo .docx. A continuación se presenta la configuración para asociar los METADATOS del documento de tipo memorando y las variables presentes en el archivo .docx que serán reemplazadas, de manera similar se configuran estas asociaciones con los otros tipos de documentos.

```
banred.documentos.memorando.metadatos=banred:numeroDocumento, banred:nombreEmisor, banred:cargoEmisor,
banred:nombreReceptor, banred:cargoReceptor, banred:lugar, banred:fecha, banred:temaReferencia
banred.documentos.memorando.parametros=numero_documento,remitente,cargo_remitente,destinatario,
                    cargo_destinatario,lugar,fecha,referencias
```

4.4.4.5 Implementación de aplicativo de firma electrónica dentro del sistema de gestión documental.

Como se describió en el punto 4.4.4.3.2, se creó un servicio web de tipo REST que presenta una página HTML la cual hace una referencia al Applet del aplicativo de firma, con el fin de presentar esta página HTML dentro del contexto de la interfaz de usuario de la plataforma ECM se necesita crear un conjunto de archivos de presentación los cuales de detalla a continuación.

- **Creación de interfaz para presentación de aplicativo de firma.**

Para crear una nueva página dentro de la interfaz de la plataforma ECM se necesita crear un archivo XML, el cual contiene la información de la nueva página. Se ha creado un archivo llamado firmaElectronica.xml ubicado en la ruta /opt/banred/tomcat/shared/classes/alfresco/web-extension/site-data/pages, el cual contiene el nombre de la página y el tipo de autenticación que solicita:

```
<?xml version='1.0' encoding='UTF-8'?>
<page>
<title>Firma electrónica</title>
<description>Pagina para firma electrónica</description>
<template-instance>firmaElectronica</template-instance>
<authentication>user</authentication>
</page>
```

A continuación, se especificó en un archivo XML complementario los elementos que componen la nueva página de la interfaz, dentro de esta que hasta ahora está en blanco se agrega un nuevo componente que será el que contenedor del Applet de firma. Este archivo de se llama firmaElectronica.xml ubicado en la ruta /opt/banred/tomcat/shared/classes/alfresco/web-extension/site-data/ template-instances/, el cual se destaca la configuración del contenedor del Applet:

```
<?xml version='1.0' encoding='UTF-8'?>
<template-instance>
<template-type>org/firma/firma-electrónica-pagina</template-type>
<properties>
<hasTreeview>true</hasTreeview>
</properties>
<components>
  …
<component>
<region-id>applet-firma</region-id>
<sub-components>
<sub-component id="default">
<evaluations>
<evaluation>
<url>/components/firma/applet-firma</url>
<properties>
<nodeRef>{nodeRef}</nodeRef>
</properties>
</evaluation>
</evaluations>
</sub-component>
</sub-components>
</component>
</components>
</template-instance>
```

Seguidamente se define, el contenido del componente “applet-firma” de la nueva página descrita en el punto anterior, en este componente básicamente se crea un *iframe,* elemento HTML, el cual hace una llamada al servicio web REST que

recupera el aplicativo de firma, dicha configuración se la hace en un archivo nombrado applet-firma.get.html.ftl ubicado en la ruta /opt/banred/tomcat/shared/classes/alfresco/web-extension/site-webscripts/org/alfresco/components/firma, el cual contiene la siguiente configuración:

```
<@standalone>
<@markup id="html">
<@uniqueIdDiv>
<iframe src="/share/proxy/Alfresco/banred/firma/custom/documento?nodeRef=${nodeRef}" width="800"
height="500" seamless></iframe>
</@>
</@>
</@standalone>
```

- **Agregar acción personalizada de firma electrónica**

Una vez creada la interfaz para presentar el aplicativo de firma electrónica, se procedió a agregar dentro de los formularios de tareas de los procesos una acción llamada “Firmar documento” sobre los documentos con formato .pdf, para esta configuración se modificó el archivo packageitems.ftl ubicado en la ruta /opt/banred/tomcat/shared/classes/alfresco/web-extension/site-webscripts/ org/alfresco/components/form/controls/workflow/packageitems.ftl

```
…
<#if form.data['prop_bpm_packageItemActionGroup']?? &&form.data['prop_bpm_packageItemActionGroup']?is_string &&
form.data['prop_bpm_packageItemActionGroup']?length &gt; 0>
<#local packageItemActionGroup = form.data['prop_bpm_packageItemActionGroup']>
<#local viewMoreAction = { "name": "view_more_actions",
"label": "form.control.object-picker.workflow.view_more_actions",
"link": documentLinkResolver }>
<#local firmaAction = { "name": "firma_actions", "label": "banred.etiquetas.workflow.firmadocumento",
"link": firmaLinkResolver }>
<#if packageItemActionGroup == "read_package_item_actions" ||
packageItemActionGroup == "edit_package_item_actions">
<#local actions = actions + [viewMoreAction]>
<#if field.control.params.tareaFirma??>
<#local actions = actions + [firmaAction]>
</#if>
…
```

4.4.4.6 Formularios del aplicativo

Los formularios de la plataforma ECM desarrollados cara a los procesos implementados se detallan a continuación.

- **Formularios de presentación y edición de METADATOS**

Los formularios dentro de la plataforma ECM presentan una interfaz para que el usuario pueda visualizar y/o editar los METADATOS que contiene un documento. Para configurar los formularios es necesario especificar que METADATOS se quiere presentar por cada uno de los diferentes tipos de documentos que se tiene creado dentro del repositorio documental. Para especificar estos formularios se ha creado el archivo banred-formularios-config.xml ubicado en la ruta /opt/banred/tomcat/shared/classes/alfresco/web-extension. Debido a la longitud de estas configuraciones, se va a presentar únicamente el formulario establecido para la edición de METADATOS de los documentos de tipo "oficio" a continuación:

```
<config evaluator="node-type" condition="banred:oficio">
<forms>
<form>
        <field-visibility>
<show id="cm:name" />
<show id="cm:creator" for-mode="view" />
<show id="cm:created" for-mode="view" />
<show id="cm:modifier" for-mode="view" />
<show id="cm:modified" for-mode="view" />
<show id="banred:numeroDocumento" />
<show id="banred:lugar" />
<show id="banred:fecha" />
<show id="banred:tituloReceptor" />
<show id="banred:nombreReceptor" />
<show id="banred:cargoReceptor" />
<show id="banred:institucionReceptor" />
<show id="banred:nombreEmisor" />
<show id="banred:cargoEmisor" />
</field-visibility>
<appearance>
<set id="datosDocumento" appearance="fieldset" label-id="banred.set.datosOficio" />
<field id="banred:numeroDocumento" set="datosDocumento" label-id="banred.campo.numeroDocumento" />
<field id="banred:lugar" set="datosDocumento" label-id="banred.campo.lugar" />
<field id="banred:fecha" set="datosDocumento" label-id="banred.campo.fecha" />
<field id="banred:tituloReceptor" set="datosDocumento" label-id="banred.campo.tituloReceptor" />
```

```
<field id="banred:nombreReceptor" set="datosDocumento" label-id="banred.campo.nombreReceptor" />
<field id="banred:cargoReceptor" set="datosDocumento" label-id="banred.campo.cargoReceptor" />
<field id="banred:institucionReceptor" set="datosDocumento" label-id="banred.campo.institucionReceptor" />
<field id="banred:nombreEmisor" set="datosDocumento" label-id="banred.campo.nombreEmisor" >
<control template="/org/alfresco/components/form/controls/userinfo.ftl" />
</field>
<field id="banred:cargoEmisor" set="datosDocumento" label-id="banred.campo.cargoEmisor" >
<control template="/org/alfresco/components/form/controls/userinfoCargo.ftl" />
</field>
</appearance>
</form>
</forms>
</config>
```

- **Formularios de tareas de usuarios dentro de los procesos cargados**

De manera similar a los formularios de para los diferentes tipos de documentos especificados en el paso anterior, para definir los formularios de las tareas de usuarios de un flujo se necesita especificar los METADATOS que se van a presentar en la interface, cabe recalcar que estos METADATOS son los campos que se desea que ingrese el usuario como alimentación de información al proceso, para realizar esta configuración se necesita definir el nombre de la tarea que se va a configurar, debido a la longitud de esta configuración, se presentará solamente la configuración necesaria para el formulario de inicio del proceso de “Gestión documental” y el formulario de “Ingreso de METADATOS para convocatoria”.

```
<!-- Formulario inicial -->
<config evaluator="string-compare" condition="activiti$processGestionDocumental" >
<forms>
<form>
<!-- 1. Que campos mostrar y el orden-->
<field-visibility>
<show id="bpm:workflowDescription" />
<show id="ges:tipoDocumento" />
<show id="bpm:workflowDueDate" />
<show id="bpm:workflowPriority" />
</field-visibility>
<!-- 2. Apariencia de los campos y agrupamiento-->
<appearance>
                    …
</appearance>
</form>
</forms>
</config>
```

```
<config evaluator="task-type" condition="banredwfc:llenarMetadatosFormularioWebConvocatoria">
<forms>
<form>
<field-visibility>
<!-- campos propios de Alfresco -->
<show id="message" />
<show id="banred:numeroTramite" />
<show id="bpm:priority" />
<show id="bpm:dueDate" />
<show id="bpm:status" />
<show id="transitions" />
<!-- campos de banred -->
<show id="banred:numeroDocumento" />
<show id="banred:lugar" />
<show id="banred:fecha" />
<show id="banred:responsableReunion" />
<show id="banred:cargoResponsableReunion" />
<show id="banred:tipoReunion" />
<show id="banred:temaReunion" />
<show id="banred:participantesDeReunion" />
<show id="banred:duracionReunion" />
</field-visibility>
<!-- 2. Apariencia de los campos y agrupamiento-->
<appearance>
                    …
</appearance>
</form>
</forms>
</config>
```

4.5 Diccionario de datos

A continuación se detalla el diccionario de la base de datos *potalbanred.*

Tabla 15. Resumen de tablas de base de datos portalbanred

Nombre	Documentación
USUARIO	
HISTORIAL	

Elaborado por: Pedro Caicedo y Diego Godoy

4.5.1 Resumen de Columnas Tabla USUARIO

Tabla 16. Resumen de columnas tabla USUARIO

Nombre	DataType	Limitaciones	Anulable	Documentación
ID	integer(10)	PK	No	
ESTADO	varchar(4)		No	
NOMBRE	varchar(128)		No	
CEDULA	varchar(16)		No	
TIPO	integer(10)		No	
Ldap	varchar(128)		No	

Elaborado por: Pedro Caicedo y Diego Godoy

4.5.1.1 Relación entre tablas

Tabla 17. Detalles relación RelaciónBD

RELACIONBD	
To	 HISTORIAL
Autor	Pedro Caicedo
Create Date Time	10/07/2013 10:46:39 AM
Last Modified	10/07/2013 10:47:16 AM
Identifying	False
To Multiplicity	0..*
From Multiplicity	1
Sync To Association	Yes
Datos Modelo	Physical

Elaborado por: Pedro Caicedo y Diego Godoy

4.5.1.2 Resumen de columnnas tabla HISTORIAL

Tabla 18. Resumen de columnas tabla HISTORIAL

Nombre	DataType	Limitaciones	Anulable	Documentación
ID	integer(10)	PK	No	
ESTADO	varchar(4)		No	
USUARIO	integer(10)	FK (USUARIO.ID)	No	
FECHA	timestamp		No	
ACCION	varchar(16)		No	

Elaborado por: Pedro Caicedo y Diego Godoy

4.6 Resultado de pruebas

A continuación se detallan los resultados de las pruebas de funcionalidad y optimización de código definidos en el plan de pruebas (ver anexos).

4.6.1 Prueba de funcionalidad

Culminado el proceso de construcción y parametrización de la solución propuesta, se realizaron las pruebas de funcionalidad (caja negra) junto al personal de BANRED.

Los criterios de estado especificados para los resultados de esta prueba son: *cerrado* y *pendiente*. Donde, *cerrado* indica que el resultado de ejecutar los pasos de una actividad ha sido correcto, *y pendiente* requiere de un reconocimiento más a fondo (revisión de código).

Seguidamente se citan los casos de prueba de funcionalidad definidos:

PRU - 001: Administración de Usuarios
PRU - 002: Auditoria BDD
PRU - 003: Auditoria ECM
PRU - 004: Parametrización BDD

PRU - 005: Parametrización ECM

PRU - 006: Parametrización LDAP

PRU - 007: Tareas Pendientes

PRU - 008: Estado Trámite

PRU - 009: Involucrados

PRU - 010: Hoja Ruta

Los casos de prueba más relevantes son mostrados a continuación, y los restantes ver en el anexo CASOS DE PRUEBA.

4.6.1.1 Caso de prueba administración de usuarios

Tabla 19. Caso de prueba administración de usuarios

PRU-001 Administración de Usuarios		
Propósito:	Verificar la correcta funcionalidad en cuanto a la Administración de Usuarios.	
Prerrequisitos:	Ingresar al sistema, usuario administrador	
Datos de Prueba:	Datos para Administración de Usuarios.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Seleccionar la opción Administrar usuarios 3. Ingresar filtros requeridos 4. Clic en el botón buscar 5. Seleccionar el usuario deseado 6. Clic en el link EDITAR 7. Asignar el perfil deseado 8. Clic en el botón guardar	
RESULTADO		
ID	**Descripción:**	**Estado**
1.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy

4.6.1.2 Caso de prueba auditoría base de datos

Tabla 20. Caso de prueba auditoría base de datos

PRU-002 Auditoria base de datos		
Propósito:	Verificar la correcta funcionalidad en cuanto a Auditoria BDD.	
Prerrequisitos:	Ingresar al sistema, usuario administrador	
Datos de Prueba:	Datos para buscar Auditoria BDD.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Clic en la opción Auditoria BDD 3. Ingresar filtros requeridos 4. Clic en el botón buscar	
RESULTADO		
ID	**Descripción:**	**Estado**
2.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy

4.6.1.3 Caso de prueba auditoría ECM

Tabla 21. Caso de prueba auditoría ECM

PRU-003 Auditoria ECM		
Propósito:	Verificar la correcta funcionalidad en cuanto a Auditoria ECM.	
Prerrequisitos:	Ingresar al sistema, usuario administrador	
Datos de Prueba:	Datos para buscar Auditoria ECM.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Clic en la opción Auditoria ECM 3. Ingresar filtros requeridos 4. Clic en el botón buscar	
RESULTADO		
ID	**Descripción:**	**Estado**
3.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy

4.6.1.4 Caso de prueba parametrización de base de datos

Tabla 22. Caso de prueba parametrización de base de datos

PRU-004 Parametrización de base de datos		
Propósito:	Verificar la correcta funcionalidad en cuanto a Parametrización BDD.	
Prerrequisitos:	Ingresar al sistema, usuario administrador	
Datos de Prueba:	Datos para administrar Parametrización BDD.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Clic en la opción Parametrización 3. Clic en la opción Parametrización BDD 4. Modificar los valores necesarios 5. Clic en el botón guardar 6. Reiniciar el servidor para reflejar los cambios	
RESULTADO		
ID	**Descripción:**	**Estado**
4.	**PASOS PARA REPRODUCIR:**	Cerrado

Fuente: Pedro Caicedo y Diego Godoy

4.6.1.5 Caso de prueba parametrización de ECM

Tabla 23. Caso de prueba parametrización de ECM

PRU-005 Parametrización ECM		
Propósito:	Verificar la correcta funcionalidad en cuanto a Parametrización ECM.	
Prerrequisitos:	Ingresar al sistema, usuario administrador	
Datos de Prueba:	Datos para administrar Parametrización ECM.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Clic en la opción Parametrización 3. Clic en la opción Parametrización ECM 4. Modificar los valores necesarios 5. Clic en el botón guardar 6. Reiniciar el servidor para reflejar los cambios	
RESULTADO		
ID	**Descripción:**	**Estado**
5.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy

4.6.1.6 Caso de prueba parametrización LDAP

Tabla 24. Caso de prueba parametrización LDAP

PRU-006 Parametrización LDAP		
Propósito:	Verificar la correcta funcionalidad en cuanto a Parametrización ldap.	
Prerrequisitos:	Ingresar al sistema, usuario administrador	
Datos de Prueba:	Datos para administrar Parametrización LDAP.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Clic en la opción Parametrización 3. Clic en la opción Parametrización ldap 4. Modificar los valores necesarios 5. Clic en el botón guardar 6. Reiniciar el servidor para reflejar los cambios	
RESULTADO		
ID	**Descripción:**	**Estado**
6.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy

4.6.2 Prueba de optimización de código

Las métricas de incidencias definidas por la herramienta CodePro Analytix para la optimización de código del proyecto, arrojan un conjunto de 40 categorías, sin embargo para fines prácticos del diagnóstico se consideraron únicamente las siguientes:

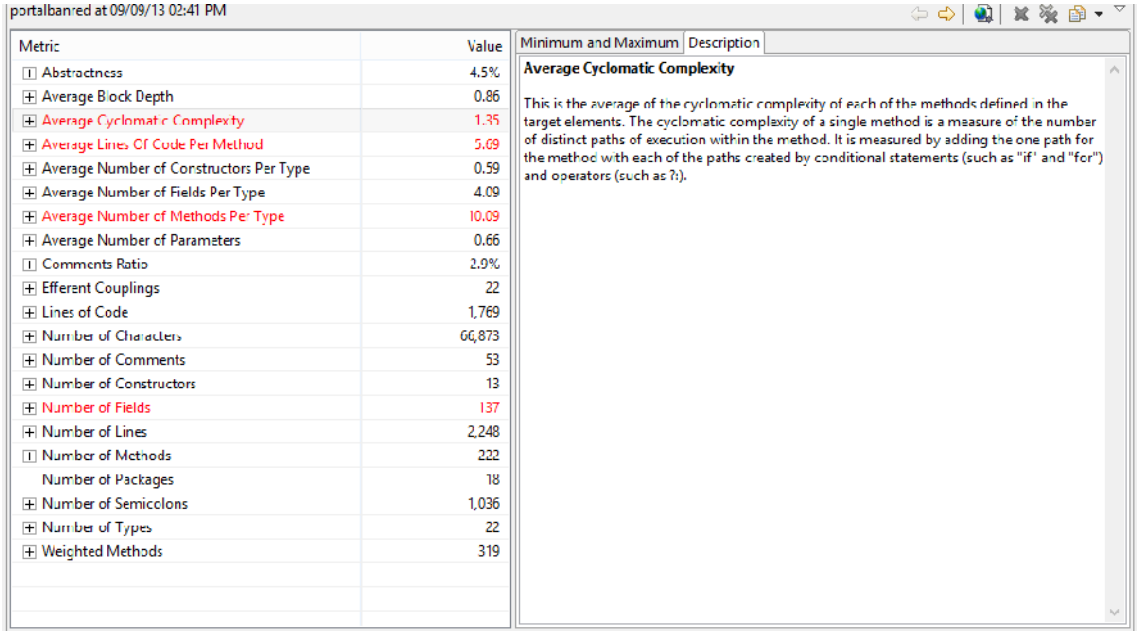
- Desempeño
- Complejidad de programa
- Código muerto o no alcanzable
- Estilo de código
- Convenciones de nomenclatura
- Manejo de excepciones
- Comentarios de código

De acuerdo a las explicaciones proporcionadas en el punto 3.3 MODELO DE DOMINIO, las pruebas de optimización de código se realizaron únicamente sobre el software construido, es decir, el portal web, el aplicativo de firma electrónica y el componente de creación de documentos con formato .docx.

4.6.2.1 Portal web

En la figura No 76 se muestra una imagen de la vista tabular de métricas obtenida por la herramienta CodePro Analytix, para los elementos del proyecto web previo a la optimización de código

Figura 76. Resumen de incidencias de resultados de código de portal web previo a la depuración



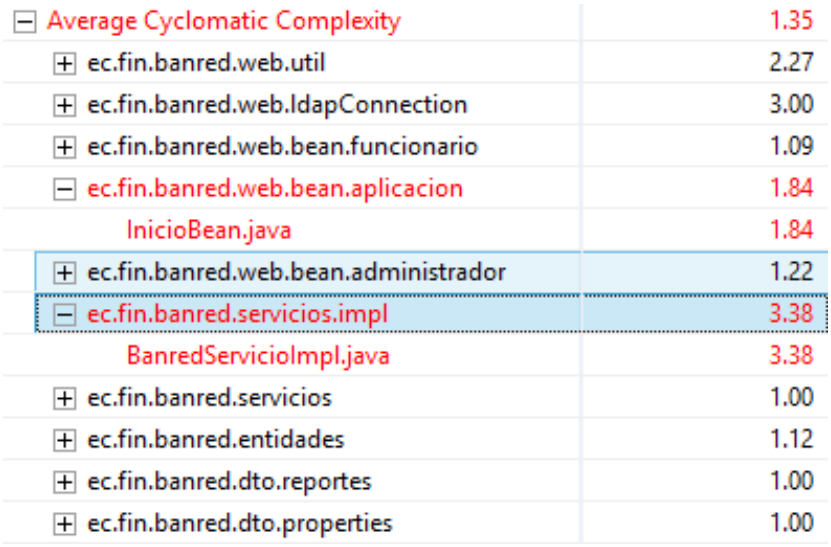
Fuente: CodePro Analytix.

Como muestra la figura anterior, la depuración de código se debe centrar en las incidencias de average cyclomatic complexity o media complejidad ciclomática, averange lines of code per method o media de líneas de código por método, average number of methods per type o media de número de metodos por tipo y number of fields onúmero de campos, cuyo significado y detalle se muestra en seguida.

- **Media Complejidad Ciclomática (Average Cyclomatic Complexity)**

Esta métrica hace referencia al número de caminos distintos de ejecución dentro de los métodos de las clases. En la figura mostrada a continuación, se indica el reporte de los componentes del aplicativo web que presentan incidencias del tipo media complejidad ciclomática:

Figura 77. Reporte de elementos del portal web con incidencias de media complejidad ciclomática



Fuente: CodePro Analytix.

Una vez analizadas el código fuente de los elementos con incidencias, se determinó que existían métodos únicos que debían ser descompuestos para mejorar el rendimiento del software.

Como referencia a la depuración de código realizada (fue similar para las clases restantes) se detalla la optimización de código realizada sobre método *inicializar()* de la clase *InicioBean* del *paquete ec.fin.banred.web.bean.aplicacion.*

La optimización realizada consistió en descomponer al método *inicializar()* en los métodos *inicializar(), perfilUser()* y *loginUser().* Ver anexo OPTIMIZACIÓN DE CODIGO.

Posterior al proceso de depuración indicado, se ejecuto nuevamente el análisis de la herramienta de software y las incidencias fueron eliminadas, tal como lo muestra la figura siguiente:

Figura 78. Reporte de evaluación de elementos del portal web de tipo complejidad ciclomática posterior a depuración de código

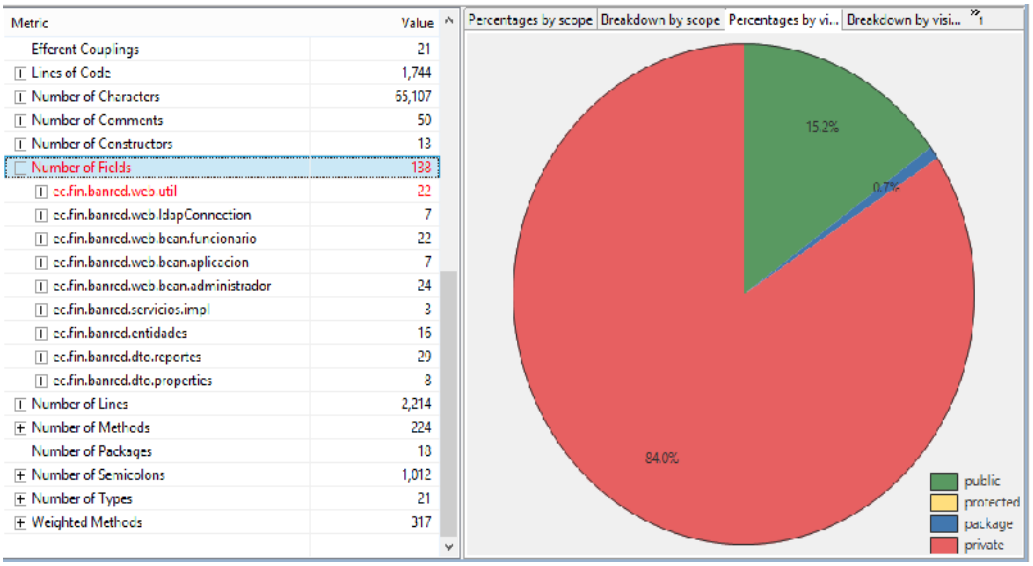
Average Cyclomatic Complexity	1.35
ec.fin.banred.web.util	2.27
ec.fin.banred.web.ldapConnection	3.00
ec.fin.banred.web.bean.funcionario	1.09
ec.fin.banred.web.bean.aplicacion	1.80
ec.fin.banred.web.bean.administrador	1.22
ec.fin.banred.servicios.impl	3.21
ec.fin.banred.servicios	1.00
ec.fin.banred.entidades	1.12
ec.fin.banred.dto.reportes	1.00
ec.fin.banred.dto.properties	1.00

Fuente: CodePro Analytix.

- **Número de campos (number of fields)**

Esta medida establece un desglose basado en el ámbito del campo (estático o de instancia) y su visibilidad (público, paquete, protegida o privada). En la figura siguiente se muestra el reporte de la métrica resultante de número de campos sobre los componentes del proyecto.

Figura 79. Reporte de elementos del portal web con incidencias de número de campos

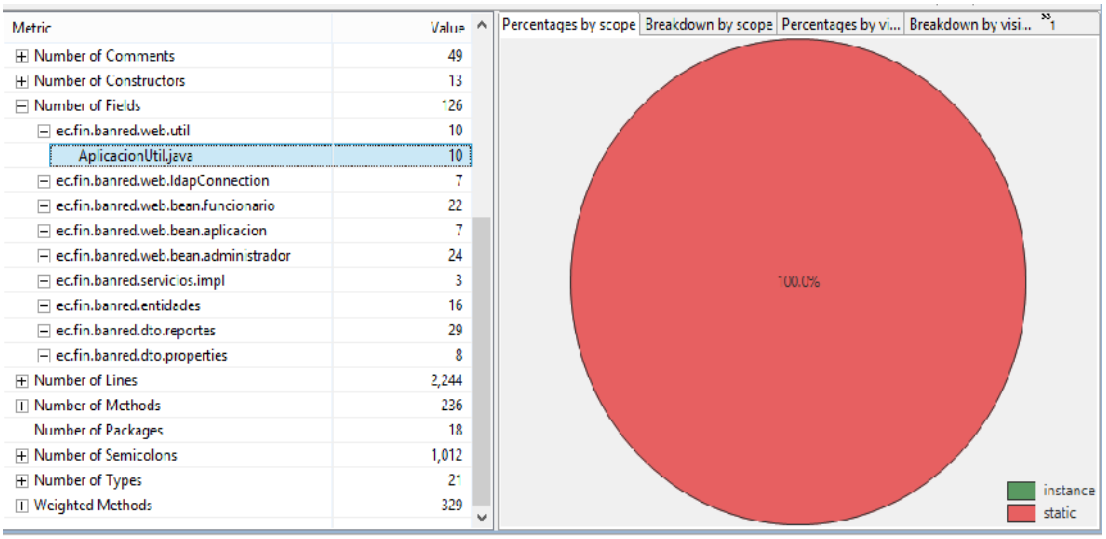


Fuente: CodePro Analytix

Analizado el código fuente de la clase *AplicacionUtil* que se encuentra dentro del paquete *ec.fin.banred.web.util*, reportado con incidencia, se determinó que se debía crear varios métodos *get()* que retornen los valores de las variables estáticas definidas en la clase. El detalle de esta depuración de código se la puede observar en el anexo OPTIMIZACIÓN DE CÓDIGO.

Una vez optimizado el código se volvió a ejecutar el análisis de la herramienta y las novedades fueron eliminadas, tal como lo muestra figura siguiente:

Figura 80. Reporte de elementos del portal web con incidencias de número de campos posterior a la optimización de código

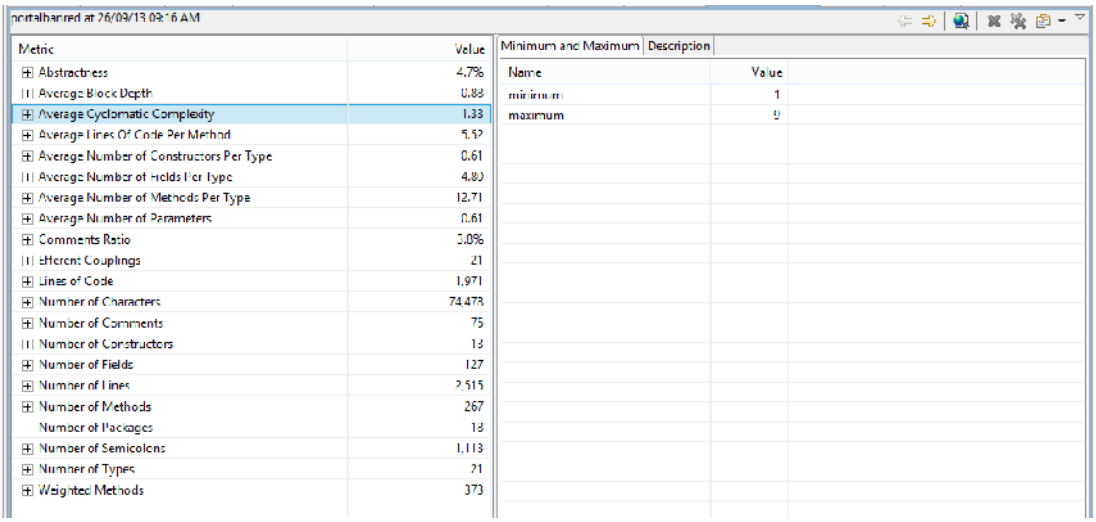


Fuente: CodePro Analytix.

Respecto a las méticas de *media de número de metodos por tipo y número de campos*, éstas se solucionaron al aplicar la métrica de complejidad cicomática y el número de atributos por clase indicada.

Posterior a la mitigación de todas las incidencias presentas por el reporte de métricas (ver figura 81), se ejecutó nuevamente un análisis general sobre todos los componentes del aplicativo web con la herramienta CodePro Analytix, y se estableció que el código del portal web esta corregido y óptimo.

Figura 81. Resumen de incidencias de resultados de código de portal web posterior a la depuración

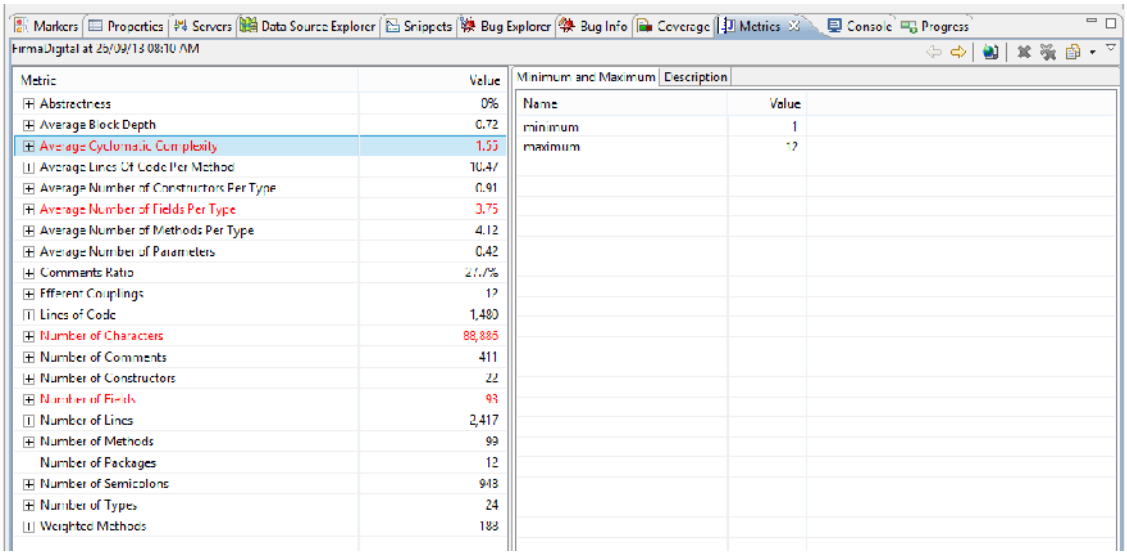


Fuente: CodePro Analytix.

4.6.2.2 Aplicativo de firma electrónica

Similar al proceso de validación de código del portal web, en la Figura 82 se muestra una imagen de la vista tabular de métricas obtenida por la herramienta CodePro Analytix, para los elementos del aplicativo de firma electrónica previo a la optimización de código.

Figura 82. Resumen de incidencias de resultados de código de aplicativo de firma electrónica previo a la depuración



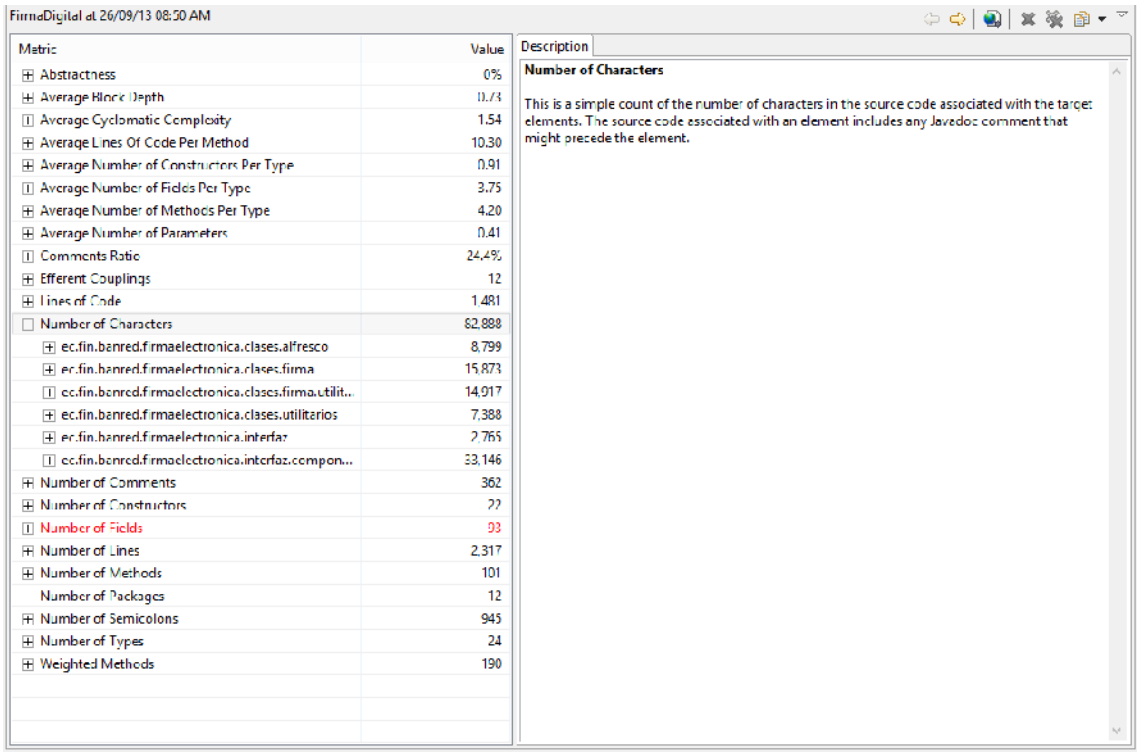
Fuente: CodePro Analytix.

Como muestra la figura No 103, la depuración de código se debe centrar en las incidencias de average cyclomatic complexity o media complejidad ciclomática, average number of fields per type o media de número de campos por tipo, number of characters o número de caracteres y number of fields o número de campos, cuyo significado se muestra a continuación.

- Media complejidad ciclomática: Esta métrica hace referencia al número de caminos distintos de ejecución dentro de los métodos de las clases.
- Media de número de campos por tipo es: Este es el promedio del número de campos definidos en la clase.
- Número de caracteres: Este es un simple recuento del número de caracteres en el código fuente de las clases, interfases, etc. El código fuente asociado con un elemento incluye cualquier comentario Javadoc que podrían preceder el elemento de la clase.
- Número de campos:Esta medida establece un desglose basado en el ámbito del campo (estático o de instancia) y su visibilidad (público, paquete, protegida o privada).

Posterior a la mitigación de todas las incidencias presentas por el reporte previo, se ejecutó nuevamente el análisis general de los componentes del aplicativo de firma electrónica con la herramienta CodePro Analytix, determinando que el código está corregido y óptimo, a excepción de la métrica *número de campos* debido a quecada campo (field) esta representando un componente en la interfaz del Applet, como por ejemplo, un botón, un input, etc y no se puede modificar o eliminar. Ver resultados en figura siguiente:

Figura 83. Resumen de incidencias de resultados de código de aplicativo de firma electrónica posterior a la depuración



Fuente: CodePro Analytix.

CONCLUSIONES

- La automatización del nuevo proceso de negocio de gestión documental, maneja de forma efectiva el ciclo de vida de los documentos administrativos de BANRED, disminuyendo significativamente la carga operativa de su personal en la ejecución de las tareas de: creación de trámites (definidos en este proyecto), versionamiento, búsqueda, despacho, revisión de involucrados, almacenamiento y socialización de documentos.

- Con la implementación de la solución propuesta, se disminuirá de forma sustancial la impresión de papel relacionada a los flujos de trabajo automatizados y sus documentos anexos, puesto que su gestión no requiere de medios físicos.

- El ECM ALFRESCO, es la única herramienta de código abierto del mercado que cumple los estándares de rendimiento y escalabilidad empresarial de ECM's definidos y valorados por Gartner Inc. y The Forrester Wave, dos de las principales compañías de asesoramiento e investigación en tecnologías de la información del mundo.

- En caso que BANRED opte por otro ECM, el modelo de arquitectura definido en la solución propuesta, así como el Applet de firma electrónica, podrán implementarse en el nuevo software.

- El *Proceso Unificado* es una metodología iterativa e incremental, muy completa y bien documentada, que proporcionó al desarrollo de la solución propuesta, tanto en la configuración del EMC como en la construcción de software, los diagramas y modelos necesarios, para probar, integrar y ejecutar el producto de forma efectiva.

- La integración de la herramienta Directorio Activo con el portal web y el ECM, administra los inicios de sesiones de usuario, la gestión de áreas administrativas de BANRED, y demás información que es consumida por la herramienta ECM y el portal web para gestionar los flujos de trabajo documentales y los roles de accesos, eliminando la duplicidad de información y disminuyendo la carga

operativa de los administradores de red y de servicios del Proceso de Tecnología de BANRED, en caso de implementar la solución propuesta en un escenario de producción.

- PDF es el formato estándar definido para estampar la firma electrónica en los documentos empresariales, puesto que el programa Adobe Reader realiza una validación de datos con la entidad de certificación, Banco Central del Ecuador, mediante el envío de datos de la firma digital del usuario (vía Internet). En caso que las credenciales estén o no vigentes, la entidad de certificación informa de esta novedad al programa Adobe Reader y ésta estampa dicho resultado en el documento PDF.

- El proceso de firma electrónica así como la edición de documentos en línea, requieren de una conexión permanente a Internet, caso contrario se invalidarán estas funcionalidades.

RECOMENDACIONES

- Implementar una gestión del cambio organizacional enfocada a ordenar y controlar el nuevo esquema de trabajo, a fin de disminuir la resistencia de su personal frente al nuevo proceso documental.

- Controlar la impresión de documentos, asignando cuotas de impresión a los usuarios de la red y generando reportes mensuales sobre dichas impresiones, mediante el uso de herramientas de administración de recursos de red, como por ejemplo: Configuration Manager de Microsoft. Esta actividad deberá estar sujeta a las políticas de gestión del cambio.

- Optar por el ECM ALFRESCO como el nuevo gestor documental de BANRED,

- Usar los modelos y diagramas de la metodología UP complementado con la notación BPMN para los modelos de requerimientos, casos de uso, despliegue y finalmente documentar los diagramas de procesos requeridos.

- Generar una política de gestión de directorio de usuarios con la herramienta Directorio Activo, donde se norme la creación, baja y modificación de cuentas de usuario y se especifiquen los tipos de datos y los valores que acepta cada campo de software. A continuación se siguieren las siguientes pautas:}

    - Nombre de usuario con primer nombre y primer apellido en mayúsculas.
    - Manejo de contraseña segura con uso de caracteres, números y letras.
    - El campo destinado al registro de la cédula solo debe poseer valores numéricos para el caso de ciudadanía ecuatoriana y valores alfanuméricos para cédulas extranjeras.
    - En caso que personal de BANRED cese en sus funciones, se sugiere deshabilitar su usuario de DA y no eliminarlo, puesto que los usuarios poseen registros de transacciones realizadas en los diversos sistemas empresariales.
    - Documentar las asociaciones de los campos del DA y las herramientas de software que consumen dicha información.

- Por otra parte, se sugiere el uso de DA 2007 o superior puesto que el campo *nombre de usuario* utilizado en los inicios de sesión no es Case Sensitive y no se ha normado el uso de mayúsculas o minúsculas.
- Adicionalmente se deberá establecer de forma clara el ámbito de responsabilidades de los usuarios *administrador de red*, *administrador ECM* y *administrador de portal*, puesto que cada uno gestiona diferentes servicios y recursos dentro de la Institución.

- Usar el programa Adobe Reader y el formato PDF para el estampado de firma digital. De la investigación realizada, solo este software realiza la validación de firma electrónica ante la entidad de certificación, de forma gratuita.
- De manera contrapuesta, se siguiere que el motor de transformación de documentos de tipo docx a pdf y al editor de texto requerido para implementar los procesos, sea de paga puesto que las herramientas LibreOffice y OpenOffice, utilizadas en este proyecto, presentan problemas con los formatos para tabas, gráficas y otros elementos dentro del documento de texto.

- Otorgar permisos de navegación a Internet a todos los usuarios de la LAN de BANRED para obtener todos los beneficios que una herramienta ECM ofrece y por supuesto para proporcionar validez jurídica a los documentos a través del estampado de firma electrónica en los mismos. BANRED posee una política para navegación y uso del Internet.

- Generar políticas para normar el uso de la herramienta de gestión documental, para lo cual se sugiere lo siguiente:
  - El usuario administrador del ECM, no puede generar flujos de trabajo documental, para tal fin se utilizará el usuario existente en DA.
  - En caso de presentarse problemas con la conexión a Internet, o la prohibición del uso de esta, se creará un directorio dentro de *Mis Documentos* de la máquina cliente con el nombre DOCUMENTOS ECM, donde se almacenarán todos los documentos que se descargue el usuario desde la plataforma ECM.

LISTA DE REFERENCIAS

Adams, C. y Lloyd, S. (2002). *Concepts, Standards, and Deployment Considerations*. Boston, United States: Pearson Education

Carlisle, A. y Lloyd, S. (2003). *Understanding PKI second edition.* Boston, MA, Estados Unidos: Pearson Education.

Cano, J. (2003). *Consideraciones sobre el Estado del arte del Peritaje Informático en Latinoamérica.* Recuperado el 20 de noviembre de 2012 de http://derechoytics.uniandes.edu.co/

Comisión de las Naciones Unidas para el derecho mercantil internacional (2003). *CNUDMI Anuario Volumen XXXIV*. Recuperado el 10 de diciembre de 2012 de http://www.uncitral.org/pdf/spanish/yearbooks/yb-2003-s/2003_vol_B_Ebook_S.pdf

Contreras, I. (2009). *La firma electrónica y la función notarial en Jalisco: Homologación Federal y Estatal*. Recuperado el 15 de noviembre de 2012 de http://www.revistanotarios.com/files/La-Firma-electronica.pdf

DICAMPUS (2012). *Modelo de dominio*. Recuperado el 1 de noviembre de 2012 de http://www.wikos.es/UserFiles/1/File/03-Enterprise%20Architect%20.pdf

ECURED (2013). *Ingeniería de Software*. Recuperado el 8 de enero de 2013 de http://webcache.googleusercontent.com/search?q=cache:http://www.ecured.cu/index.php/Ingeniería_de_software

Empresa BANRED. (2012). *Misión y Visión.* Recuperado de http://www.portal.banred.fin.ec/index.php?option=com_content&task=view&id=21&Itemid=71

Gilbert, M., Shegda, K., Chin, K., Tay, G., Koehler-Kruener, H. (2012). *Magic Quadrant for Enterprise Content Management*. Recuperado el 15 de mayo de 2013 de http://www.gartner.com/technology/reprints.do?id=1-CKSZ07&ct=121021&st=sg

Hernandez, E. *El lenguaje Unificado de Modelado (UML)*. Recuperado el 30 de agosto de 2012 de http://www.disca.upv.es/enheror/pdf/ActaUML.PDF

Instituto de Ingeniería Eléctrica. (2009). *Normas de salud ocupacional para las empresas privadas.* Recuperado el 20 de mayo de 2013 de: http://iie.fing.edu.uy/ense/asign/desasoft

Jacobson, I., Booch, G., y Rumbaugh, J. (2000). *El proceso unificado de la ingeniería de software.* Madrid, España: Pearson Education.

LEFISPedia. *Criptografía*. Recuperado el 10 de noviembre de 2012 de http://www.lefis.org/wiki/doku.php?id=es:criptografia

Ley de Comercio Electrónico, Firmas y Mensajes De Datos (2002). *Contenido y estatutos de la ley.* Recuperado el 20 de agosto de 2012 de http://comercioexterior.com.ec/qs/sites/default/files/Ley%20de%20Comercio%20Electronico.pdf

MKM Publicaciones. (2012). *La oficina sin papeles es una utopía*. Recuperado el 7 de julio de 2012 de http://www.mkm-pi.com/byte-ti/la-oficina-sin-papeles-es-una-utopia/

Object Management Group (2012). Business Process Model and Notation.Recuperado el 31 de agosto de 2012 de http://www.bpmn.org/

Peñaranda, H. (2011). *La firma electrónica digial en Venezuela*. Recuperado el 10 de noviembre de 2012 de http://pendientedemigracion.ucm.es/info/nomadas/29/hectorpenaranda.pdf

Pressman, R. (1993): Ingeniería de Software, Un enfoque práctico. New York City, NY, Estados Unidos: Mc Graw Hill.

Revista de Cultura, diario Clarín de Argentina (2008). Arqueología de la firma. Recuperado el 31 de agosto de 2012 de http://edant.revistaenie.clarin.com/notas/2008/05/24/01678682.html

Revista Escuela de Ingeniería de Antioquia (2008). *Un patrón de interacción entre diagramas de actividades UML y sistemas WORKFLOW*. Medellin, Colombia

Ruiz, F. (2010). *Apuntes históricos sobre la firma y su trascendencia como signo identificatorio*. Recuperado el 10 de noviembre de 2012 de http://grafo-logico.blogspot.com/2010/02/apuntes-historicos-sobre-la-firma-y-su.html

Serrano, J. (2012). Apuntes históricos sobre la firma como signo identificativo. Recuperado el 20 de diciembre de 2012 de http://peritocaligrafoalmeria.blogspot.com/2012/04/apuntes-historicos-sobre-la-firma-como.html

Schneier, B. y Banisar, D. (1997). The Electronic Privacy Papers. Hoboken, NJ, Estados Unidos: John Wiley & Sons

Schmuller J. (2000). *Aprendiendo UML en 24 horas*. México: Prentice Hall.

slideshare. *El proceso Unificado.* Recuperado el 30 de agosto de 2012 de http://www.slideshare.net/Sofylutqm/el-proceso-unificado-3943047#btnNext

Stallings, W. (2010). *Cryptography and Network Security: Principles and Practice*. Estados Unidos: Prentice Hall.

Sumano, M., Fernández, J., y Cortés, M. (2011). *Ingeniería de software II manual de buenas prácticas*. México: Recuperado el 15 de diciembre de 2012 de http://www.uv.mx/personal/asumano/files/2010/07/MP_IS2_2011.pdf

UNAM. *Modelos de procesos del software.* Recuperado el 20 de agosto de 2012 de http://www.paginaspersonales.unam.mx/files/1065/Modelos_de_procesos_del_software.pdf

Universidad Ecotec (2008). *Modelos de ingeniería de software*. Recuperado el 20 de agosto de 2012 de https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCgQFjAA&url=http%3A%2F%2Fdocs.universidadecotec.edu.ec%2Ftareas%2F2012E%2FCOM347%2Falum%2F2008292548_833_2012E_COM347_Modelos_o_Paradigmas_de_la_Ingenier_a_de_Software-Deber2.docx&ei=Tk-GUvSzG4bNkAenp4CADA&usg=AFQjCNHFEb2Fxqtu4NPEjUzTPwwYEUlxbg&sig2=_qaEUyROSt0VNGhBflAuxg&bvm=bv.56643336,d.eW0

GLOSARIO

Directorio Activo: Es el término que usa Microsoft para referirse a su implementación de servicio de directorio en una red distribuida de computadores.

AJAX: JavaScript asíncrono y XML, Asynchronous JavaScript And XML por sus siglas en inglés, es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).

Aplicativo: Componente de una aplicación Java que se ejecuta en el contexto de otro programa, comúnmente en un navegador web.

Applet: Componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo en un navegador web en el equipo cliente.

Archivística: Es el estudio teórico y práctico de los principios, procedimientos y problemas concernientes al almacenamiento de documentos, buscando que dicha documentación se mantenga en el tiempo, pueda ser consultada y clasificada.

Archivo .properties: Los ficheros .properties son archivos que nos permiten almacenar variables de configuración de una aplicación Java.

Archivo .log: Permiten el registro oficial de eventos durante un rango de tiempo en particular en una aplicación.

Archivo de contexto: Son los archivos encargados de cargar configuraciones dentro de aplicaciones Java Empresariales.

BPM: Se llama Gestión de procesos de negocio (Business Process Management o BPM en inglés) a la metodología empresarial cuyo objetivo es mejorar la eficiencia a través de la gestión de los procesos de negocio, que se deben modelar, organizar, documentar y optimizar de forma continua. Como su nombre sugiere, BPM se enfoca en la administración de los procesos dentro de una organización.

CCS: Las hojas de estilo en cascada CCS (Cascading Style Sheets, por sus siglas en inglés) hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas.

Clase de objetos: Una clase de objetos define los atributos y operaciones de los objetos. Los objetos se crean en tiempo de ejecución mediante la instanciación de la definición de la clase.

Ciclo de vida del software: Utilizado a menudo como otro nombre para el proceso del software. Originalmente acuñado para referirse al modelo en cascada del proceso del software.

Confiabilidad: La confiabilidad de un sistema es una propiedad total que tienen en cuenta la seguridad del sistema, la fiabilidad, la disponibilidad, la protección y otros atributos. Refleja el grado en el cual los usuarios pueden confiar en el sistema.

Construcción: Proceso de compilar los componentes o unidades que forman un sistema y enlazarlos con otros componentes para crear un programa ejecutable. La construcción del sistema está normalmente automatizada de modo que se minimiza la recompílación. Esta automatización puede ser incorporada a un sistema de procesamiento de lenguajes (como en Java) o puede implicar herramientas CASE para apoyar la construcción del sistema.

Desarrollo iterativo: Enfoque para el desarrollo de software en el que se entrelazan los procesos de especificación, diseño, programación y pruebas.

Directorio: Contenedor virtual en el que se almacenan una agrupación de archivos de datos y otros subdirectorios.

DRA: es un proceso de desarrollo de software que comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades CASE.

EJB: Enterprise java beans, modelo de componentes basado en Java.

Escenario: Descripción de una forma típica en la que se utiliza un sistema o un usuario lleva a cabo alguna actividad.

ECM: Enterprise Content Management, sistema que permite la organización y el almacenamiento de documentos de una empresa. El término abarca estrategias, métodos y herramientas que se utilizan en todo el ciclo de vida del contenido.

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

Herramienta CASE: Herramienta software, como un editor del diseño o un depurador de programas, utilizada para apoyar una actividad en el proceso de desarrollo del software.

Hibernate: Es una herramienta de Mapeo objeto-relacional que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

ID (identifier): Nombre que identifica a un elemento único. Un ID puede ser una palabra, un número, una letra, un símbolo o cualquier combinación de estos.

Ingeniería de sistemas: Proceso que trata de la especificación de un sistema, la integración de sus componentes y las pruebas de que el sistema cumple sus requerimientos. La ingeniería de sistemas no sólo trata el sistema software, sino el sistema socio-técnico entero: software, hardware y procesos operativos.

Interfaz: Especificación de los atributos y operaciones asociados con un componente software. La interfaz es utilizada como el medio de tener acceso a la funcionalidad del componente.

Interfaz de Programación de Aplicaciones (API): Interfaz, generalmente especificada como un conjunto de operaciones, definida por un programa de aplicación que permite acceder a la funcionalidad del programa. Esto significa que no sólo se puede acceder a esta funcionalidad a través de la interfaz de usuario, sino que otros programas pueden utilizarla directamente.

Java: Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel.

JSF: JavaServer Faces (JSF) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas.

LDAP: Protocolo Ligero de Acceso a Directorios (en español) que hacen referencia a un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

Lenguaje de Modelado Unificado (UML): Lenguaje gráfico utilizado en el desarrollo orientado a objetos que incluye varios tipos de modelos del sistema que proporcionan distintas vistas de un sistema. UML se ha convertido en un estándar de facto para el modelado orientado a objetos.

Lenguaje Estructurado de Consultas (SQL): Lenguaje estándar utilizado para la programación de bases de datos relaciónales.

LOGS: Registros de auditoría de un sistema informático.

Lógica de negocio: En análisis y diseño orientado a objetos, este término describa la parte de un sistema que se encarga de las tareas relacionadas con los procesos de un negocio, tales como ventas, control de inventario, contabilidad, etc. Son rutinas que realizan entradas de datos, consultas a los datos, generación de informes y más específicamente todo el procesamiento que se realiza detrás de la aplicación visible para el usuario (Backend).

Metadata o Metadato: (Meta+datos) es un término que se refiere a datos sobre los propios datos.

Modelo de procesos: Representación abstracta de un proceso. Los modelos de procesos pueden ser representados desde varias perspectivas y mostrar las actividades implicadas en un proceso, los objetos utilizados en el proceso, las restricciones que se aplican al proceso y los roles de las personas involucradas en el proceso.

Office: Es una suite de oficina que abarca e interrelaciona aplicaciones de escritorio, servidores y servicios para los sistemas operativos Microsoft Windows y Mac OS X.

Open Source: Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

PDF: Portable Document Format (acrónimo del inglés) formato de documento portátiles un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems.

Persistencia de objetos: Se entiende por persistencia (en programación) como la acción de preservar la información de un objeto de forma permanente (guardar), pero a su vez también se refiere a poder recuperar la información del mismo (leer) desde los datos en una tabla, de un archivo plano, etc., para que pueda ser nuevamente utilizada,

PKCS#11: Interfaz de dispositivo criptográfico ("Cryptographic Token Interface" o cryptoki). Define un formato de fichero usado comúnmente para almacenar claves privadas con su certificado de clave pública protegido mediante clave simétrica.

Proceso del software: Conjunto relacionado de actividades y procesos implicados en el desarrollo y evolución de un sistema software.

Reglas de negocio: Las Reglas del Negocio o Conjunto de Reglas de Negocio describe las políticas, normas, operaciones, definiciones y restricciones presentes en

una organización y que son de vital importancia para alcanzar los objetivos misionales.

Rest: Es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide web.

Richfaces: Richfaces es una biblioteca de código abierto basada en Java que permite crear aplicaciones web con Ajax.

Servicio Web: Componente software independiente al que se puede acceder a través de Internet utilizando protocolos estándares. SOAP (Simple Object Access Protocol) se utiliza para el intercambio de información en servicios web. WSDL (web Service Description Language) se utiliza para definir las interfaces de los servicios web.

Sistema crítico: Sistema informático cuyo fallo de funcionamiento puede causar importantes pérdidas económicas, humanas o medioambientales.

Software: El software se refiere a los programas y datos almacenados en un ordenador. En otras palabras, son las instrucciones responsables de que el hardware (la máquina) realice su tarea.

String: Sucesión de caracteres (letras, números u otros signos o símbolos).

Token: Es un dispositivo electrónico que se le da a un usuario autorizado de un servicio computarizado para facilitar el proceso de autenticación, llamado también token de seguridad o token criptográfico.

Tomcat: También llamado Apache Tomcat, funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Web: Sistema de documentos (o páginas web) interconectados por enlaces de hipertexto, disponibles en Internet.

Workflow: Flujo de trabajo en español, es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.

XML: Lenguaje de Marcado Extensible. XML es un lenguaje de marcado de texto que soporta el intercambio de datos estructurados. Cada campo de datos se delimita por etiquetas que proporcionan información sobre ese campo. XML se utiliza ampliamente en la actualidad y se ha convertido en la base de los protocolos para los servicios web.

ANEXOS

ÍNDICE

1.1 Modelo de requisitos 1

1.1.1 Requisitos funcionales: 1

1.1.1.1 Plataforma ECM 2

Repositorio Documental 2

Tareas de Usuario 5

Flujos de trabajo 6

Plantillas 7

Gestión de Usuarios y Perfiles 8

Gestión de sesiones del sistema 9

Colaboración 10

Configuración de plataforma ECM 11

Auditoría 12

1.1.1.2 Portal web 13

Gestión de sesiones del sistema 13

Gestión de Usuarios y Perfiles 14

Parametrización del portal 14

Reportes 15

Auditoría 16

1.1.2 Requisitos no funcionales: 17

1.1.2.1 Rendimiento 17

1.1.2.2 Seguridad 17

1.1.2.3 Fiabilidad 18

1.1.2.4 Disponibilidad 19

1.1.2.5 Mantenibilidad 20

1.1.2.6 Usabilidad 20

1.2 Detalle de interfaz de usuario 22
1.2.1 Portal web 22
1.2.1.1 Interfaz de usuario funcionario 22
1.2.1.2 Interfaz de usuario *administrador* 26
1.2.2 Plataforma ECM 32
1.2.3 Aplicativo de firma electrónica 43
1.3 Plan de entrevista para dimensionamiento del proceso gestión documental de BANRED 45
1.4 Modelo de domino plataforma ECM ALFRESCO 47
1.5 Plan de pruebas 48
Funcionalidad 48
Optimización de código 48
Configuración del ambiente de pruebas 49
Cronograma de trabajo 49
1.6 Casos de prueba 50
1.6.1 Pruebas de funcionalidad 50
1.6.1.1 Caso de prueba tareas pendientes 50
1.6.1.2 Caso de prueba estado trámite 50
1.6.1.3 Caso de prueba involucrados 51
1.6.1.4 Caso de prueba Hoja de Ruta 52
1.6.2 Optimización de código 53
1.6.2.1 Media Complejidad Ciclomática (Average Cyclomatic Complexity)…. ..... ……53
1.6.2.2 Número de campos (number of fields) 55
1.7 Lista de referencias 58
1.8 Manual de usuario “Simulación de interoperabilidad de la plataforma ECM con portal web" 59

ÍNDICE DE FIGURAS

Figura 1. Arquitectura Alfresco 19
Figura 2. Mis tareas pendientes 22
Figura 3. Estado de trámites 23
Figura 4. Involucrados del trámite 24
Figura 5. Hoja de ruta 25
Figura 6. Administración de usuarios 26
Figura 7. Parámetros de conexión con plataforma ECM 27
Figura 8. Parámetros de conexión con Directorio Activo 28
Figura 9. Parámetros de conexión con base de datos 29
Figura 10. Auditoría de plataforma ECM 30
Figura 11. Auditoría del portal web 31
Figura 12. Iniciar un trámite 32
Figura 13. Formulario llenar metadatos en formulario web 33
Figura 14. Formulario ingresar información al cuerpo del documento 34
Figura 15. Formulario detalles del documento 35
Figura 16. Formulario edición de documento con Google Docs 37
Figura 17. Formulario revisar documento 38
Figura 18. Formulario corregir documento 39
Figura 19. Formulario firmar documento electrónicamente 40
Figura 20. Formulario detalles del trámite 42
Figura 21. Interfaz firma electrónica 43
Figura 22. Parámetros de firma electrónica 44

ÍNDICE DE TABLAS

Tabla 1. Requisitos funcionales repositorio documental 2
Tabla 2. Requisitos funcionales tareas de usuario 5
Tabla 3. Requisitos funcionales flujos de trabajo 6
Tabla 4. Requisitos funcionales plantillas de documentos 7
Tabla 5. Requisitos funcionales gestión de usuarios y perfiles 8
Tabla 6. Requisitos funcionales gestión de sesiones del sistema 9
Tabla 7. Requisitos funcionales colaboración 10
Tabla 8. Requisitos funcionales plataforma ECM 11
Tabla 9. Requisitos funcionales auditoría 12
Tabla 10. Requisitos funcionales sesiones portal web 13
Tabla 11. Requisitos funcionales usuarios y perfiles portal web 14
Tabla 12. Requisitos funcionales parametrización portal web 14
Tabla 13. Requisitos funcionales reportes 15
Tabla 14. Requisitos funcionales auditoría portal web 16
Tabla 15. Recomendaciones de rendimiento 17
Tabla 16. Tipos de algoritmos de firma digital 17
Tabla 17. Acuerdo de Nivel de Servicio (SLA) 19
Tabla 18. Caso de prueba tareas pendientes 50
Tabla 19. Caso de prueba estado trámite 50
Tabla 20. Caso de prueba involucrados 51
Tabla 21. Caso de prueba Hoja de Ruta 52

1.1 Modelo de requisitos

El modelo de requisitos estructura los requerimientos de usuario final, los cuales se dividen en requisitos funcionales y no funcionales.

Los primeros definen la expectativa del usuario frente al sistema, mientras que los requisitos no funcionales describen los atributos que debe tener el software una vez construido, y estos pueden ser; de portabilidad, eficiencia, confiabilidad, robustez, rendimiento, etc.

1.1.1 Requisitos funcionales:

Los requisitos funcionales han sido registrados en una tabla, cuyos campos se detallan a continuación:

- Código: Identificador del requisito funcional. El formato definido es: ERS-001 (número secuencial).
- Detalle: Texto que describe la funcionalidad requerida en el software.
- Entrada: Datos, información o salida de otros procesos, los cuales producirán la funcionalidad requerida.
- Salida: Resultado o garantía de éxito en el cumplimiento del requerimiento funcional.
- Actor: Usuario o proceso del sistema. Los roles[1]de usuario definidos en el proyecto fueron:
- Observación: Información adicional complementaria.

Seguidamente se describen los requerimientos planteados por los usuarios de BANRED, durante las reuniones de levantamiento de requerimientos:

[1]Clasificación mediante la cual se definen distintos privilegios de operación para los usuarios del sistema.

1.1.1.1 Plataforma ECM

- **Repositorio Documental**

Tabla 1. Requisitos funcionales repositorio documental

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-001	El sistema permitirá la creación de documentos	Opción seleccionada	Documento creado	Todos los usuarios	La ofimática de ALFRESCO es Google Docs y la establecida en el proyecto es Microsoft Office 2010
ERS-002	El sistema será capaz de asignar la "cuenta de usuario" autenticada al sistema operativo a uno de los METADATOS del documento creado en el repositorio documental.	Cuenta de Directorio Activo, documento	Variable asignada a METADATA de documento	Proceso de sistema	La cuenta de usuario está asociada a la sesión actual de sistema operativo
ERS-003	El sistema permitirá la edición de documentos	Opción seleccionada, documento a modificar	Documento modificado	Todos los usuarios	
ERS-004	El sistema permitirá consultar documentos en la plataforma ECM por el contenido dentro del documento y por su METADATA	Opción seleccionada,			
ERS-005	El sistema permitirá eliminar documentos	Opción seleccionada, documento	Documento eliminado	Todos los usuarios	
ERS-006	El sistema permitirá iniciar flujos de trabajo	Opción seleccionada	Flujo de trabajo iniciado	Todos los usuarios	
ERS-007	El sistema permitirá crear directorios (carpetas)	Opción seleccionada	Nuevo directorio	Todos los usuarios	
ERS-008	El sistema permitirá eliminar directorios	Opción seleccionada,	Directorio eliminado	Todos los usuarios	

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
		directorio			
ERS-009	El sistema permitirá modificar directorios.	Opción seleccionada, directorio	Directorio modificado	Todos los usuarios	
ERS-010	El sistema permitirá categorizar los documentos del repositorio como favoritos	Opción seleccionada, documento	Documento categorizado como favorito	Todos los usuarios	
ERS-011	El sistema proporcionará un explorador de directorios y archivos	Petición de usuario, criterio de ordenamiento de archivos y directorios	Organización de archivos y directorios en la plataforma ECM	Todos los usuarios	El explorador de directorios y archivos mostrará los siguientes campos: descripción, fecha de modificación, usuario modificador, tamaño del archivo
ERS-012	El sistema permitirá descargar documentos del repositorio documental al equipo del usuario	Opción seleccionada, archivo	Documento descargado a equipo cliente	Todos los usuarios	
ERS-013	El sistema permitirá editar propiedades de los documentos	Opción seleccionada, documento	Propiedad de documentos actualizada	Todos los usuarios	Las propiedades del documento están definidas como METADATA
ERS-014	El sistema permitirá la lectura de documentos de otros usuarios, a través de la asignación de permisos en el repositorio documental	Opción seleccionada, documento, usuario permitido	Permiso de lectura sobre documento asignado a un nuevo usuario	Todos los usuarios	
ERS-015	El sistema permitirá subir documentos del equipo local al repositorio documental de herramienta ECM	Opción seleccionada, documento	Almacenamiento de archivo en repositorio documental	Todos los usuarios	
ERS-016	El sistema mostrará la ruta actual en la que se encuentra el usuario, en la barra de direcciones del ECM	Ubicación dentro del sistema	Mostrar ruta actual de usuario	Todos los usuarios	
ERS-017	El sistema permitirá la creación de documentos	Opción seleccionada	Documento creado	Todos los usuarios	La ofimática de ALFRESCO es Google Docs y la establecida en el proyecto es Microsoft Office 2007
ERS-018	El sistema será capaz de asignar la “cuenta de usuario” autenticada al sistema	Cuenta de Directorio	Variable asignada a METADATA	Proceso de sistema	La cuenta de usuario está asociada a la sesión actual de sistema

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
	operativo a uno de los METADATOS del documento creado en el repositorio documental.	Activo, documento	de documento		operativo
ERS-019	El sistema permitirá la edición de documentos	Opción seleccionada, documento a modificar	Documento modificado	Todos los usuarios	
ERS-020	El sistema permitirá consultar documentos en la plataforma ECM por el contenido dentro del documento y por su METADATA	Opción seleccionada,			
ERS-021	El sistema permitirá eliminar documentos	Opción seleccionada, documento	Documento eliminado	Todos los usuarios	
ERS-022	El sistema permitirá iniciar flujos de trabajo	Opción seleccionada	Flujo de trabajo iniciado	Todos los usuarios	
ERS-023	El sistema permitirá crear directorios (carpetas)	Opción seleccionada	Nuevo directorio	Todos los usuarios	
ERS-024	El sistema permitirá eliminar directorios	Opción seleccionada, directorio	Directorio eliminado	Todos los usuarios	
ERS-025	El sistema permitirá modificar directorios.	Opción seleccionada, directorio	Directorio modificado	Todos los usuarios	
ERS-026	El sistema permitirá categorizar los documentos del repositorio como favoritos	Opción seleccionada, documento	Documento categorizado como favorito	Todos los usuarios	
ERS-027	El sistema proporcionará un explorador de directorios y archivos	Petición de usuario, criterio de ordenamiento de archivos y directorios	Organización de archivos y directorios en la plataforma ECM	Todos los usuarios	El explorador de directorios y archivos mostrará los siguientes campos: descripción, fecha de modificación, usuario modificador, tamaño del archivo
ERS-028	El sistema permitirá descargar documentos del repositorio documental al equipo del usuario	Opción seleccionada, archivo	Documento descargado a equipo cliente	Todos los usuarios	

Elaborado por: Pedro Caicedo y Diego Godoy

- **Tareas de Usuario**

Tabla 2. Requisitos funcionales tareas de usuario

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-029	El sistema permitirá crear y delegar tareas de usuario para iniciar los flujo de trabajo de gestión documental	Opción seleccionada, tarea, usuario designado, documentos anexos, Tiempo estimado de ejecución, prioridad de ejecución, estado tarea	Delegación de nueva tarea y Notificación mediante mensaje de correo electrónico	Todos los usuarios	El estado de las tareas son: activas, pendientes y completadas Las tareas tendrán las siguientes prioridades: alta, media y/o baja
ERS-030	El sistema permitirá modificar tareas de usuario.	Opción seleccionada, tarea	Tarea modificada	Todos los usuarios	
ERS-031	El sistema permitirá eliminar tareas de usuario.	Opción seleccionada, tarea	Tarea eliminada	Usuario administrador	
ERS-032	El sistema permitirá realizar búsquedas de tareas por diversos criterios	Opción seleccionada, criterio de búsqueda	Lista tareas	Todos los usuarios	Los criterios de búsqueda son: fecha de vencimiento y prioridad

Elaborado por: Pedro Caicedo y Diego Godoy

- **Flujos de trabajo**

Tabla 3. Requisitos funcionales flujos de trabajo

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-033	El sistema debe permitir la automatización de flujos de trabajo	Opción seleccionada, Requerimiento de cliente, modelo flujo de trabajo, usuarios involucrados	Flujo de trabajo automatizado	Usuario administrador	
ERS-034	El sistema permitirá la asignación de usuarios a los flujos de trabajo	Opción seleccionada, usuarios del sistema, flujo de trabajo	Usuarios asignados a flujos de trabajo	Usuario administrador	
ERS-035	El sistema será capaz de asociar los usuarios vinculados a los flujos de trabajo de gestión documental a través del análisis de la cuenta de usuario autenticada	Proceso de sistema, usuario autenticado, flujo de trabajo	Asignación de usuarios a flujo de trabajo	Proceso de sistema	Cada usuario tiene asociado una Unidad Organizativa y un grado jerárquico en Directorio Activo
ERS-036	El sistema debe permitir la edición de flujos de trabajo existentes	Opción seleccionada, Requerimiento de cliente, cambio en modelo de flujo de trabajo, flujo de trabajo, usuarios involucrados	Flujo de trabajo actualizado	Usuario administrador	
ERS-037	El sistema permitirá establecer las cuentas de correo electrónico asociados a los flujos de trabajo	Flujos de trabajo, cuenta de correo electrónico	Cuenta de correo electrónico asociada al flujo de trabajo	administrador	
ERS-038	El sistema debe permitir la eliminación de flujos de trabajo existentes	Opción seleccionada,	Flujo de trabajo eliminado	Usuario administrador	

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
		flujo de trabajo			
ERS-039	El sistema será capaz mostrar una lista de los flujos de trabajo activos, pendientes y/o completados asignados a los usuarios	Opción seleccionada, usuario de sistema	Lista de los flujos de trabajo asociados al usuario	Todos los usuarios	
ERS-040	El sistema mostrará un filtro de búsqueda para los flujos de trabajo por fecha de vencimiento	Fecha vencimiento, flujo de trabajo, usuario sistema	Lista de flujos de trabajo encontrados	Todos los usuarios	Los valores del filtro son: hoy, mañana, próximos siete días, con retraso y sin fecha
ERS-041	El sistema proporcionará un filtro de búsqueda para los flujo de trabajo por prioridad y fecha de inicio	Tipo prioridad, flujo de trabajo, usuario de sistema	Lista flujos de trabajo encontrados	Todos los usuarios	Los estados de la prioridad son: alta, baja y media Los fechas estarán dadas por: Los últimos 7 , 14 y 28 días
ERS-042	El sistema permitirá consultar los usuarios involucrados en los flujos de trabajo	Opción seleccionada, usuarios, flujo de trabajo	Usuarios del flujo	Todos los usuarios	

Elaborado por: Pedro Caicedo y Diego Godoy

- **Plantillas**

Tabla 4. Requisitos funcionales plantillas de documentos

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-043	El sistema permitirá la creación de plantillas de documentos o clases documentales	Opción seleccionada, plantilla, METADATOS, diseño plantilla	Nueva plantilla	Usuario administrador	Las plantillas definidas serán: memorandos, oficios, informes, convocatorias y actas de trabajo
ERS-044	El sistema permitirá la eliminación de plantillas de documentos	Opción seleccionada, plantilla	Plantilla eliminada	Usuario administrador	No es requisito que esta administración sea gráfica o por medio de formularios. Esta administración puede ser a través de archivos o tablas de

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
					configuración (en formato XML)
ERS-045	El sistema permitirá la modificación de plantillas de documentos	Opción seleccionada, plantilla, METADATOS, diseño plantilla	Plantilla modificada	Usuario administrador	No es requisito que esta administración sea gráfica o por medio de formularios. Esta administración puede ser a través de archivos o tablas de configuración (en formato XML)
ERS-046	Es deseable que el sistema permita la edición de METADATA de las plantillas definidas	Opción seleccionada, plantilla, METADATA	Plantilla actualizada	Usuario administrador	No es requisito que esta administración sea gráfica o por medio de formularios. Esta administración puede ser a través de archivos o tablas de configuración (en formato XML)
ERS-047	Es deseable que el sistema permita la eliminación de metada de las plantillas definidas.	Opción seleccionada, plantilla, METADATA a eliminar	Plantilla actualizada	Usuario administrador	No es requisito que esta administración sea gráfica o por medio de formularios. Esta administración puede ser a través de archivos o tablas de configuración (en formato XML)

Elaborado por: Pedro Caicedo y Diego Godoy

.

- **Gestión de Usuarios y Perfiles**

Tabla 5. Requisitos funcionales gestión de usuarios y perfiles

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-048	El sistema permitirá actualizar la información de su directorio de usuarios con la información de usuarios del Directorio de sistema operativo (Directorio Activo)	Tarea programada de ECM, información de usuarios desde Directorio Activo, permiso de consulta,	Actualización de registros de usuarios en plataforma ECM	Proceso del sistema	La tarea programada se ejecutará una vez por día

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-049	El sistema permitirá asignar perfiles a los usuarios del ECM	Rol de usuario, usuario	Asignación de perfil de usuario	administrador	Los perfiles de usuario serán: administrador, gerente, líder de proceso y coordinador
ERS-050	El sistema permitirá modificar los perfiles de usuario del ECM	Rol de usuario, usuario	Perfil de usuario modificado	administrador	Los perfiles de usuario serán: administrador, gerente, líder de proceso y coordinador

Elaborado por: Pedro Caicedo y Diego Godoy

- **Gestión de sesiones del sistema**

Tabla 6. Requisitos funcionales gestión de sesiones del sistema

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-051	El sistema proporcionará un medio de autenticación de usuarios e ingreso al mismo	Mecanismo de autenticación de usuarios, credenciales de usuario	Ingreso al sistema	Todos los usuarios	
ERS-052	El sistema deberá utilizar las credenciales de usuario del sistema operativo (Directorio Activo) para ingresar al ECM	Usuario y contraseña de Directorio Activo	interfaz de usuario iniciada correspondiente al perfil en cuestión	Proceso del sistema	
ERS-053	El sistema mostrará un mensaje de error al usuario en caso que las credenciales de autenticación ingresadas sean inválidas o incorrectas.	Credenciales de autenticación inválidas o incorrectas	Mensaje de error	Proceso de sistema	
ERS-054	El sistema permitirá cerrar sesiones de usuario	Petición de salida del sistema	Sesión cerrada	Todos los usuarios	

Elaborado por: Pedro Caicedo y Diego Godoy

- **Colaboración**

Tabla 7. Requisitos funcionales colaboración

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-055	El sistema permitirá crear sitios de colaboración	Opción seleccionada, nombre sitio de colaboración, lista de usuarios miembros	Nuevo sitio de colaboración	Usuario administrador	
ERS-056	El sistema permitirá eliminar sitios de colaboración	Opción seleccionada, sitio de colaboración	Sitio de colaboración eliminado	Usuario administrador	
ERS-057	El sistema permitirá editar sitios de colaboración	Opción seleccionada, sitio de colaboración	Sitio de colaboración actualizado	Usuario administrador	
ERS-058	El sistema permitirá crear cuentas miembro de sitios de colaboración	Opción seleccionada, nombre y datos generales de usuario	Nueva cuenta de usuario miembro de sitio de colaboración	Todos los usuarios	
ERS-059	El sistema permitirá abandonar los sitios de colaboración de los cuales los usuarios son miembros	Opción seleccionada, sitio de colaboración, cuenta de usuario	Desvinculación de usuario a sitio de colaboración	Todos los usuarios	
ERS-060	El sistema permitirá realizar búsqueda de sitios colaborativos públicos				

Elaborado por: Pedro Caicedo y Diego Godoy

- **Configuración de plataforma ECM**

Tabla 8. Requisitos funcionales plataforma ECM

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-061	El sistema permitirá cambiar la configuración de conexiones con aplicaciones complementarias	Ruta de instalación de la aplicación, puerto de conexión, estado de habilitación de aplicación	Conexión con sistema complementario	administrador	Los aplicativos complementarios son: • OpenOffice.org 3. • ImageMagick. • SWFTools.
ERS-062	El sistema permitirá cambiar los parámetros de conexión con el programa de firma electrónica	IP del servidor ECM, puerto de comunicación, URL del API remoto	Habilitación de programa de firma electrónica		
ERS-063	El sistema permitirá cambiar los parámetros de configuración del protocolo ldap	IP del servidor, puerto, protocolo, dominio, usuario de consultas ldap	Parámetros de configuración ldap actualizados	administrador	Con esta configuración se establecerá la autenticación de usuarios con Directorio Activo de Windows
ERS-064	El sistema permitirá modificar la cadena de conexión a la base de datos de la plataforma ECM	Nombre de la base de datos, credenciales de usuario de conexión de base de datos, puerto, dirección IP de motor de base de datos	Parámetros de conexión de base de datos actualizados	administrador	
ERS-065	El sistema permitirá la configuración de los parámetros del motor de búsqueda de documentos	Ubicación de índices de búsqueda, tiempo de indexación de documentos, límites de búsqueda,	Motor de búsqueda habilitado	Usuario administrador	El motor de búsqueda es Lucene

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
		establecimiento de memoria cache,			
ERS-066	El sistema permitirá la configuración de acceso ftp	IP del servidor, puerto de comunicación , carpeta compartida	Protocolo FTP habilitado	Usuario administrador	
ERS-067	El sistema permitirá configurar los parámetros de acceso al repositorio documental por medio de carpetas compartidas	Carpeta que se quiere mostrar, nombre unidad de red, roles de usuario, protocolo SAMBA, puerto de comunicación	Acceso al repositorio por medio de carpetas compartidas habilitado	Usuario administrador	

Elaborado por: Pedro Caicedo y Diego Godoy

- **Auditoría**

Tabla 9. Requisitos funcionales auditoría

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-070	El sistema será capaz de llevar el registro del historial de cambios sobre los documentos en la plataforma ECM	Acción del usuario	Registro en logs	sistema	Se considerará creación, eliminación y actualización de documentos

Elaborado por: Pedro Caicedo y Diego Godoy

1.1.1.2 Portal web

- **Gestión de sesiones del sistema**

Tabla 10. Requisitos funcionales sesiones portal web

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN.
ERS-071	El sistema proporcionará un medio de autenticación de usuarios e ingreso al mismo	Mecanismo de autenticación de usuarios, credenciales de usuario	Ingreso al sistema	Todos los usuarios del portal	
ERS-072	El sistema deberá utilizar las credenciales de usuario del sistema operativo (Directorio Activo) para ingresar al portal	Usuario y contraseña de Directorio Activo	interfaz de usuario iniciada correspondiente al perfil en cuestión	Proceso del sistema	
ERS-059	El sistema mostrará un mensaje de error al usuario en caso que las credenciales de autenticación ingresadas sean inválidas o incorrectas.	Credenciales de autenticación inválidas o incorrectas	Mensaje de error	Proceso de sistema	
ERS-073	El sistema permitirá cerrar sesiones del portal web	Petición de salida del sistema	Sesión cerrada	Todos los usuarios del portal	

Elaborado por: Pedro Caicedo y Diego Godoy

- **Gestión de Usuarios y Perfiles**

Tabla 11. Requisitos funcionales usuarios y perfiles portal web

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-074	El sistema permitirá editar el perfil de usuario del portal	Nombre usuario o cédula	Actualización de perfil de usuario	administrador	Los perfiles de usuario serán: administrador y funcionario. El usuario asignado por defecto será tipo funcionario
ERS-075	El sistema será capaz de asignar por defecto el perfil tipo funcionario a un usuario	Usuario	Rol de usuario asignado	Proceso de sistema	

Elaborado por: Pedro Caicedo y Diego Godoy

- **Parametrización del portal**

Tabla 12. Requisitos funcionales parametrización portal web

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-076	El sistema permitirá modificar la cadena de conexión a la base de datos del portal	Nombre de la base de datos, credenciales de usuario de conexión de base de datos, puerto, dirección IP de motor de base de datos	Parámetros de Conexión entre el portal y su base de datos actualizados	administrador	
ERS-077	El sistema permitirá modificar los parámetros de configuración para la integración con la plataforma ECM	IP del servidor o nombre de equipo, puerto, usuario conexión y contraseña,	Parámetros de conexión entre portal y ECM actualizados	administrador	API remoto permite controlar las acciones del sistema ALFRESCO desde un aplicativo externo.

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
		URL de API remoto			
ERS-078	El sistema permitirá modificar los parámetros de configuración de protocolo ldap	IP del servidor o nombre de servidor, puerto, protocolo, dominio, usuario de consultas ldap y contraseña	Parámetros de configuración ldap actualizados	administrador	

Elaborado por: Pedro Caicedo y Diego Godoy

- **Reportes**

Tabla 13. Requisitos funcionales reportes

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-079	El sistema permitirá mostrar un reporte referente al estado de las tareas de usuario de la plataforma ECM	Nombre usuario, fecha inicio, fecha fin, estado de tarea	Reporte solicitado	Todos los usuarios	Los estados de las tareas son: pendiente, en curso y completado
ERS-080	El sistema será capaz de mostrar un reporte desde la plataforma ECM acerca del estado del trámite de un documento y el usuario que en ese momento posee dicho trámite	Número del documento, usuario remitente y/o usuario destinatario; fecha de creación del documento y/o fecha de creación del archivo, asunto	Reporte solicitado	Todos los usuarios	Los estados del documento son: creación, revisión, aprobación y firma
ERS-081	El sistema será capaz de mostrar un listado	Número de	Reporte solicitado	Todos los	

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
	de los involucrados en el trámite	trámite		usuarios	
ERS-082	El sistema será capaz de mostrar un detalle del historial de los trámites en la plataforma ECM	Número de trámite	Reporte solicitado	Todos los usuarios	El historial del trámite se denominará “Hoja de Ruta”

Elaborado por: Pedro Caicedo y Diego Godoy

- **Auditoría**

Tabla 14. Requisitos funcionales auditoría portal web

COD.	DETALLE	ENTRADA	SALIDA	ACTOR	OBSERVACIÓN
ERS-070	El sistema será capaz de llevar el registro del historial de la gestión de usuarios del portal web	Acción del usuario	Registro en logs	Usuarios del portal	Se registrarán los cambios sobre los registros de los usuarios siguientes: número de cédula, nombre, tipo y nacionalidad.

Elaborado por: Pedro Caicedo y Diego Godoy

1.1.2 Requisitos no funcionales:

1.1.2.1 Rendimiento

El servidor de pruebas podrá soportar la concurrencia de hasta 50 usuarios, sin embargo el paso a producción deberá considerar los lineamientos mostrados a continuación:

Tabla 15. Recomendaciones de rendimiento

Número de usuarios	Recomendaciones de memoria / características de CPU por servidor
Para 50 usuarios concurrentes hasta 500 usuarios ocasionales	1 GB JVM RAM 2x servidor CPU (o 1 x Dual-core)
Para 100 usuarios concurrentes hasta 1000 usuarios ocasionales	1 GB JVM RAM 4x servidor CPU (o 2 x Dual-core)
Para 200 usuarios concurrentes hasta 2000 usuarios ocasionales	2 GB JVM RAM 8x servidor CPU (o 4 x Dual-core)

Fuente: (Alfresco, 2012)

Para el paso a producción, se deberá considerar un test de rendimiento simulando el uso concurrente de la herramienta, a través de pruebas de carga para análisis de rendimiento y verificación de funcionalidades de las aplicaciones web, donde se simulará peticiones al servidor como si accediera un navegador o muchos a la vez, sin que exista en realidad ninguno ejecutándose. (Alfresco, 2012)

1.1.2.2 Seguridad

A continuación se describen los algoritmos de firma digital y de Hash empleados en el proyecto:

Tabla 16. Tipos de algoritmos de firma digital

Nombre	Tipo	Longitud mínima de clave para asegurar seguridad
RSA	asimétrico	1024-bit
SHA	Función Hash	256-bit

Fuente: (Alfresco, 2012)

La información en la nube se cifra de forma segura utilizando una encriptación AES de 256 bits, asegurando que cualquier posible violación de la seguridad física de su centro de datos, no ponga en peligro el contenido. (Alfresco, Archivos compartidos una verdadera experiencia, 2012)

La plataforma ECM utiliza la herramienta Open Source log4j para gestionar logs, para que los programadores y administradores puedan controlar lo que ocurre en la aplicación en diferente nivel de detalle. (BLYX, 2012)

1.1.2.3 Fiabilidad

El motor de búsqueda presenta las siguientes características de fiabilidad:

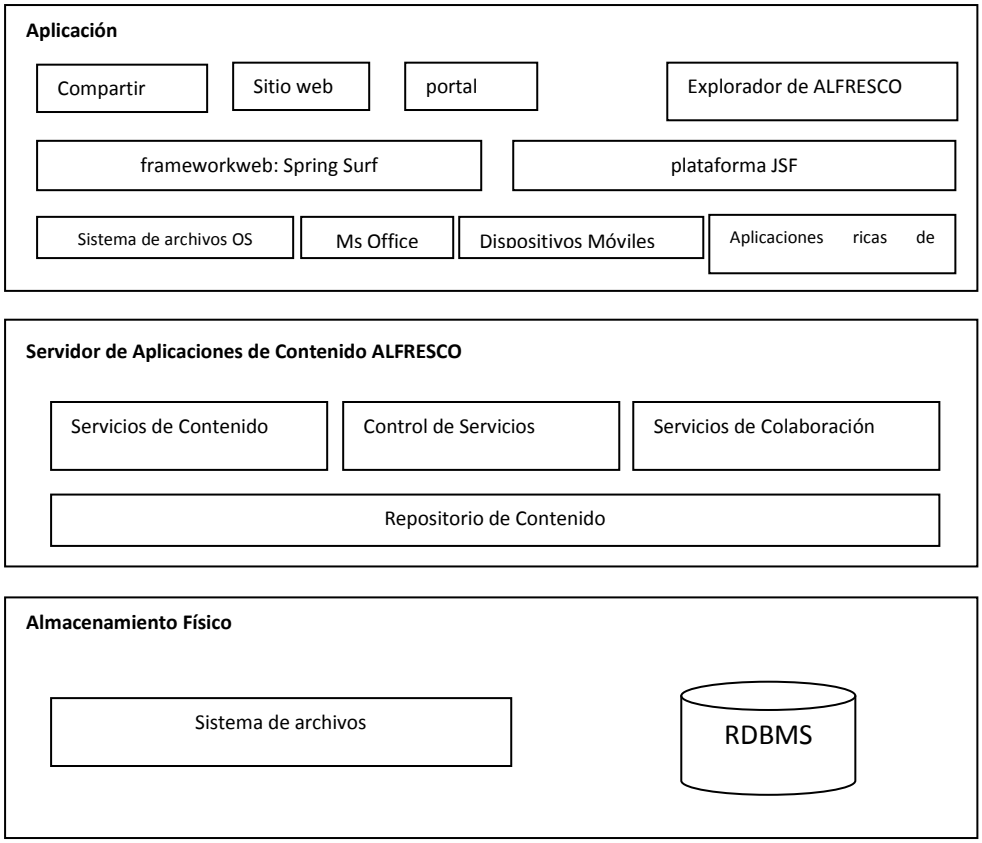
- Es el principal involucrado en todas las consultas y transacciones
- Es sensible a diferentes lenguajes de indización

El proceso de firma digital sustenta su fiabilidad en procesos criptográficos, los cuales pueden ser usados para:

- Confidencialidad (Cifrado de Datos)
- No-repudio (Firma Electrónica)
- Integridad de Datos (Firma Electrónica)
- Autenticación para usuarios, aplicaciones, y servicios (Firma Electrónica)

A continuación se muestra la arquitectura de la plataforma ECM:

Figura 1. Arquitectura Alfresco



Fuente: (Alfresco, Arquitectura Alfresco, 2012)

1.1.2.4 Disponibilidad

Debido a que la versión de ECM establecida en este proyecto es Open Source, no se puede garantizar una alta disponibilidad del servicio, puesto que no existe garantía del fabricante sobre los defectos del software que puedan generarse.

A continuación se detalla el Acuerdo de Nivel de Servicio (SLA) de la versión pagada del ECM ALFRESCO:

Tabla 17. Acuerdo de Nivel de Servicio (SLA)

Prioridad	Premier & Premier Advantage	Platino	Oro	Partner SLA
1	Sistema en producción sin servicio	resolución en 2 horas (24*7)	resolución en 4 horas	resolución en 16 horas

2	Sistema de desarrollo/producción severamente impactado	resolución en 8 horas	resolución en 16 horas	resolución en 16 horas
3	Preguntas/Como hacerlo/acrecentamiento	resolución en 8 horas	resolución en 16 horas	resolución en 16 horas

Fuente: (Alfresco, Soporte de suscripción de servicios, 2012)

1.1.2.5 Mantenibilidad

Se recomienda respaldar la data de la base de datos de la siguiente manera:

- Respaldos completos de forma quincenal.
- Respaldos diferenciales diarios.
- Respaldos transaccionales al menos cada 3 horas.

Las tareas de mantenimiento sobre el motor de base de datos deberán incluir las siguientes actividades:

- Defragmentación de tablas de las bases de datos
- Reconstrucción de índices
- Generación de estadísticas
- Limpieza de logs
- Generación de plan de respaldos, entre otras

Se sugiere realizar una copia del file system del servidor que contiene los documentos generados en la herramienta ECM de la siguiente manera

- Respaldos completos entre 15 y 30 días.
- Respaldos diferenciales diarios.

1.1.2.6 Usabilidad

- La aplicación debe seguir la Figura Corporativa de BANRED.
- Es deseable que se utilice la ofimática de Google Docs y Microsoft Office.

- Es deseable que las búsquedas alfabéticas ignoren la diferencia entre mayúsculas y minúsculas. Esto no deberá tenerse en consideración el caso que los datos a buscar explícitamente deban diferenciarlas.

1.2 Detalle de interfaz de usuario

1.2.1 Portal web

Seguidamente se muestran las interfaces de usuario del portal web organizado por sus roles de usuario *administrador y funcionario.*

1.2.1.1 Interfaz de usuario funcionario

- **Mis tareas pendientes**

Figura 2. Mis tareas pendientes

Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Indica el título de la tarea pendiente
2. Inditex descripción de la tarea pendiente
3. Indica la fecha de vencimiento de dicha tarea
4. Indica la prioridad la prioridad de la tarea
5. Indica el estado de la tarea
6. Permite ir a la tarea ejecutarla
7. Indica el usuario que se encuentra autenticado

- **Estado de trámites**

Figura 3. Estado de trámites

Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Campo para ingresar el número de trámite
2. Botón que busca estado de trámite
3. Panel que muestra el resultado en el que está en trámite
4. Indica el usuario que se encuentra autenticado

- **Involucrados del trámite**

Figura 4. Involucrados del trámite

Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Campo para ingresar el número de trámite
2. Botón que busca involucrados de un trámite
3. Muestra el tipo de tarea
4. Indica el nombre del involucrado
5. Muestra descripción del trámite
6. Indica la fecha de finalización
7. Indica el resultado del trámite
8. Indica un comentario del trámite
9. Indica el usuario que se encuentra autenticado

- **Hoja de ruta**

Figura 5. Hoja de ruta

Elaborado por: Pedro Caicedo y Diego Godoy

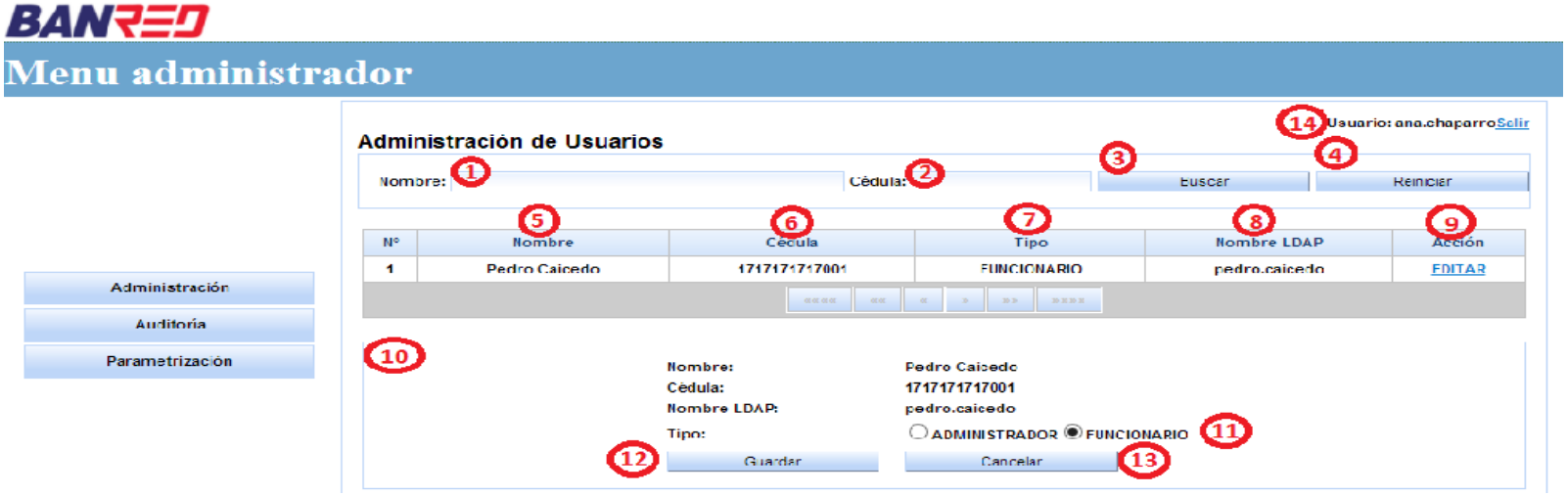
Descripción de componentes:

1. Campo para ingresar el número de trámite
2. Botón que busca información de un trámite
3. Indica el título del trámite
4. Indica el tipo de tarea
5. Indica quien realizó el trámite
6. Muestra la fecha de finalización
7. Indica el resultados del trámite
8. Indica comentarios del trámite
9. Permite ir al estado actual del trámite
10. Indica el usuario que se encuentra autenticado

1.2.1.2 Interfaz de usuario *administrador*

- **Administrador de usuarios**

Figura 6. Administración de usuarios



Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Campo para ingresar nombre
2. Campo para ingresar cédula
3. Botón para buscar información
4. Botón para reiniciar página
5. Indica nombre usuario
6. Indica cédula del usuario
7. Indica tipo de usuario
8. Indica nombre de usuario LDAP
9. Permite editar al usuario
10. Campo que muestra información del usuario
11. Campo que permite editar funcionario
12. Botón para guardar cambios realizados
13. Botón para cancelar cambios
14. Indica el usuario que se encuentra autenticado

- **Parámetros de conexión con plataforma ECM**

Figura 7. Parámetros de conexión con plataforma ECM

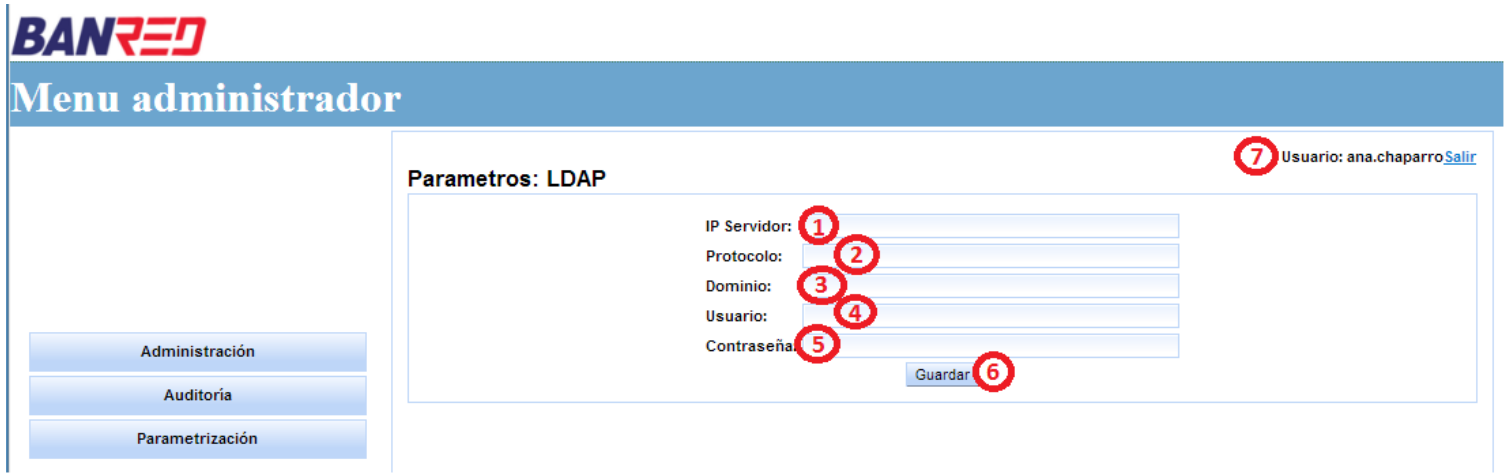
Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Campo para guardar la IP del servidor
2. Campo para almacenar puerto
3. Campo para almacenar nombre de usuario
4. Campo para almacenar contraseña
5. Campo para almacenar contexto
6. Botón para guardar cambios
7. Indica el usuario que se encuentra autenticado

- **Parámetros de conexión con Directorio Activo**

Figura 8. Parámetros de conexión con Directorio Activo



Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Campo para guardar la IP del servidor
2. Campo para almacenar puerto
3. Campo para almacenar nombre de usuario
4. Campo para almacenar contraseña
5. Campo para almacenar contexto
6. Botón para guardar cambios
7. Indica el usuario que se encuentra autenticado

- **Parámetros de conexión con base de datos**

Figura 9. Parámetros de conexión con base de datos

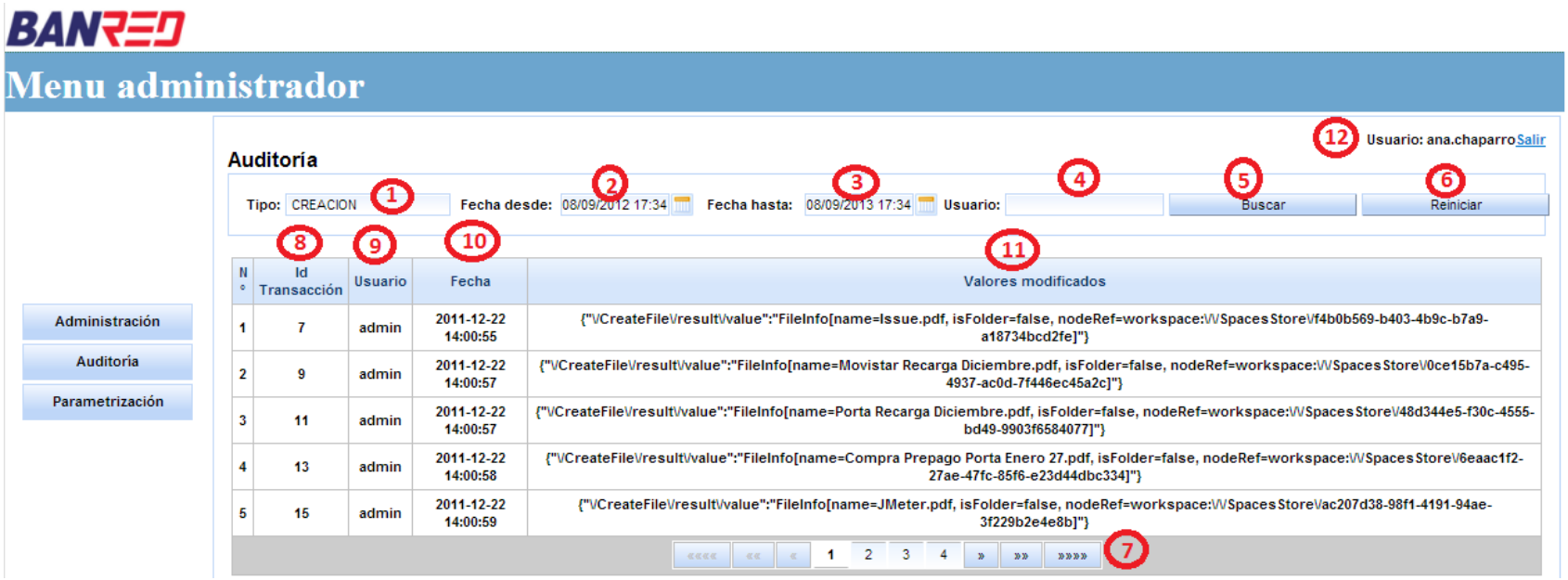
Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Campo para almacenar nombre de base de datos
2. Campo para almacenar nombre de usuario
3. Campo para almacenar contraseña
4. Campo para almacenar puerto
5. Campo para almacenar IP del servidor
6. Botón para guardar los parámetros establecidos
7. Indica el usuario que se encuentra autenticado

- **Auditoría de plataforma ECM**

Figura 10. Auditoría de plataforma ECM



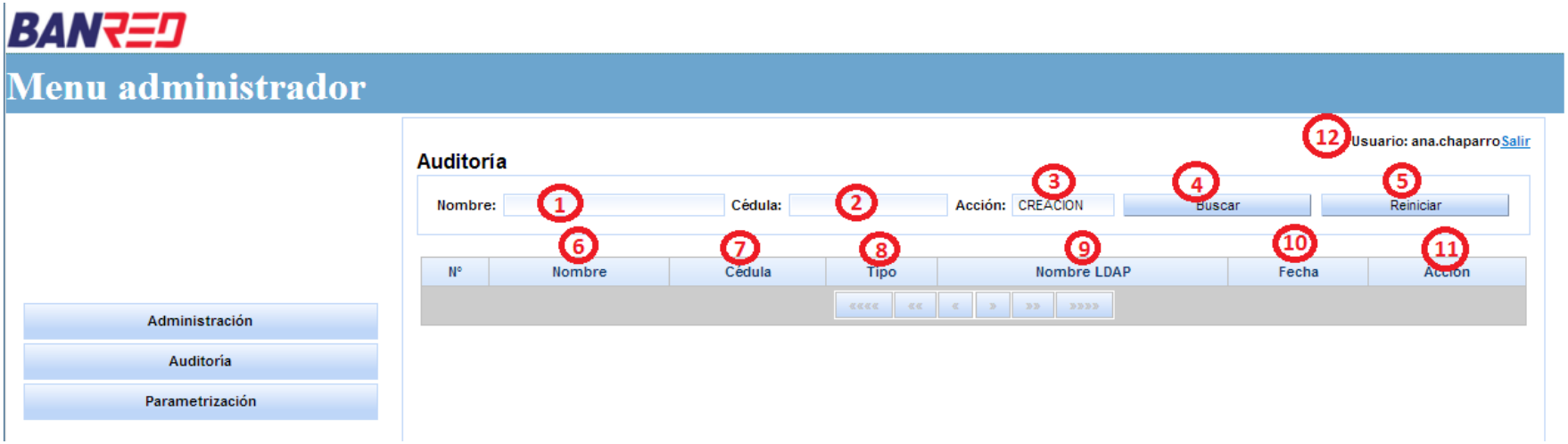
Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Campo para elegir el tipo de búsqueda
2. Campo para seleccionar fecha desde
3. Campo para seleccionar fecha hasta
4. Campo para escribir usuario
5. Botón para buscar información
6. Botón para reiniciar página
7. Muestra las páginas de información
8. Indican el id de la transacción
9. Indica usuario que realizó la acción
10. Indica que se realizó la acción
11. Indica los valores modificados
12. Indica el usuario que se encuentra autenticado

- **Auditoría del portal web**

Figura 11. Auditoría del portal web



Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

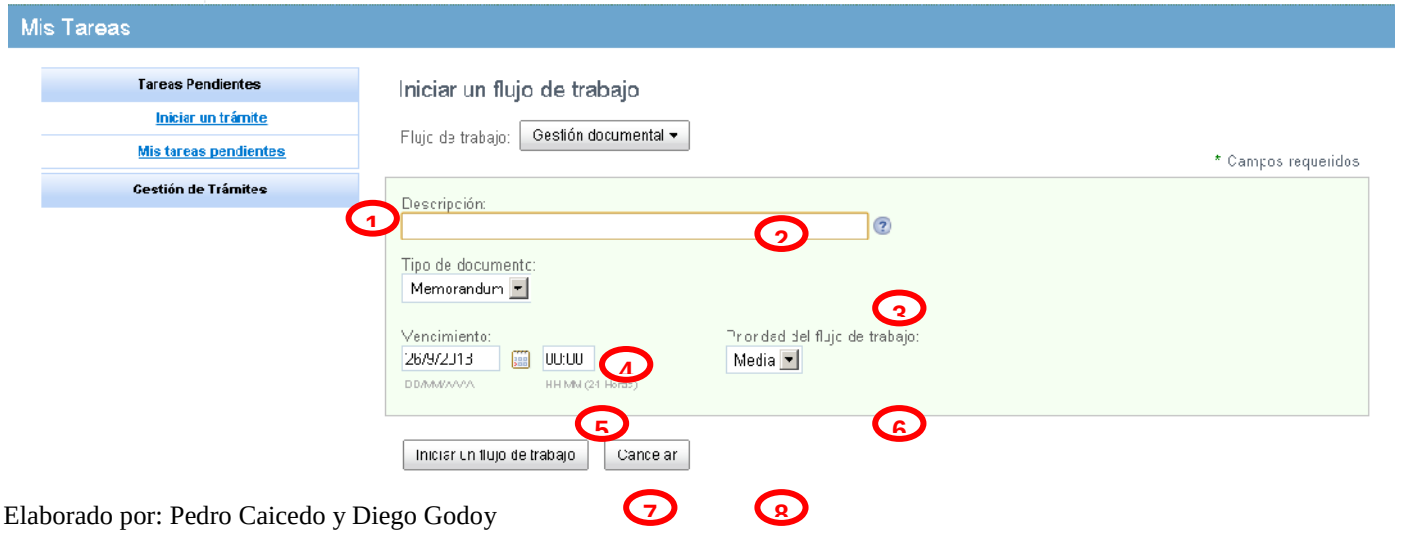
1. Campo para ingresar el nombre
2. Campo para ingresar la cedula
3. Campo para elegir el tipo
4. Botón para buscar información
5. Botón para reiniciar valores
6. Campo que indica el nombre
7. Campo que indica la cedula
8. Campo indica tipo usuario
9. Campo que indica nombre LDAP
10. Campo que indica fecha
11. Campo indica acción realizada
12. Indica el usuario que se encuentra autenticado

1.2.2 Plataforma ECM

Se utiliza en el portal web para ejecutar los trámites de la institución.

- **Iniciar un trámite**

Figura 12. Iniciar un trámite



Elaborado por: Pedro Caicedo y Diego Godoy

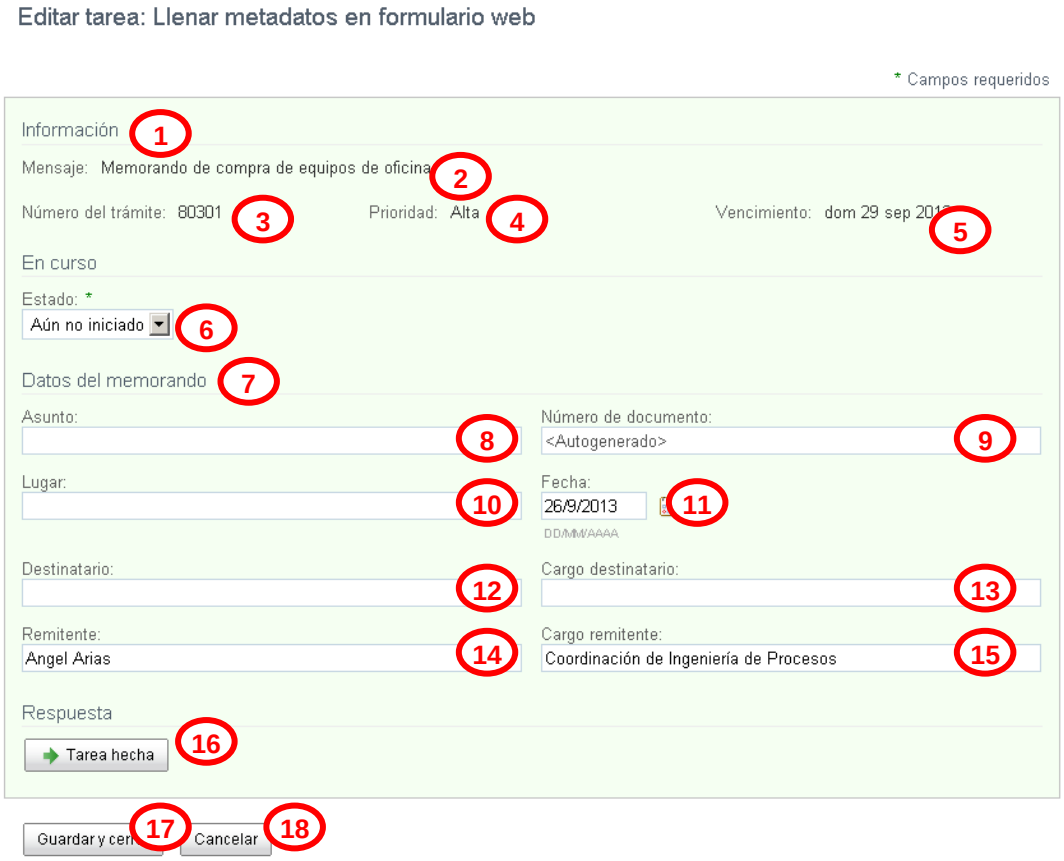
Descripción de componentes:

1. Menú de acceso
2. Listado de procesos implementados en la institución
3. Descripción del trámite a iniciar
4. Tipo de documento a crear
5. Fecha y hora de vencimiento
6. Prioridad del trámite a iniciar
7.
8. Botón para iniciar el trámite
9. Botón para cancelar el inicio del trámite

- **Formulario llenar metadatos en formulario web**

Esta tarea presenta la información necesaria para que un funcionario pueda crear un documento en el sistema. Los campos presentados en este formulario varían por cada tipo de documento (*acta*, *informe*, *oficio*, *memorando*). A continuación se presenta los campos para documentos de tipo *memorando*.

Figura 13. Formulario llenar metadatos en formulario web



Elaborado por: Pedro Caicedo y Diego Godoy

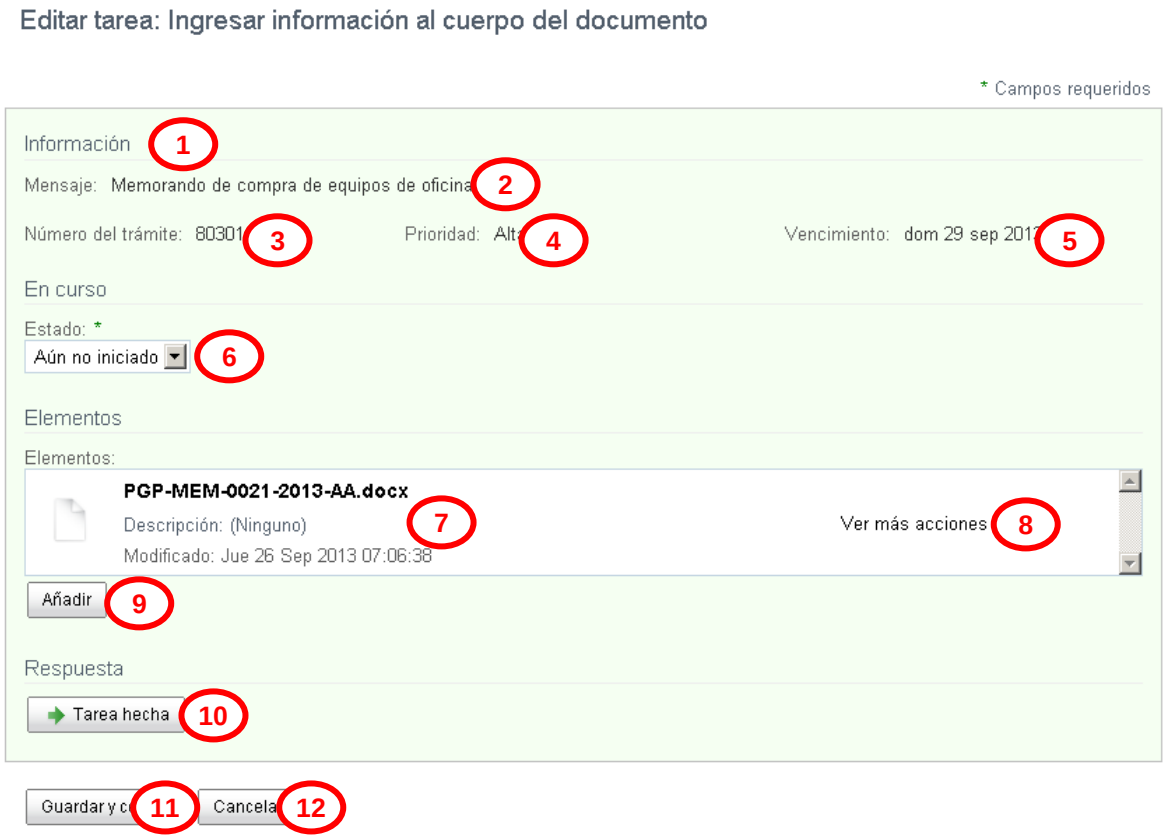
Descripción de componentes:

1. Información del trámite
2. Descripción del trámite
3. Número del trámite
4. Prioridad del trámite
5. Vencimiento del trámite
6. Estado de la tarea actual
7. Datos del documento
8. Asunto del documento
9. Número del documento
10. Lugar del documento
11. Fecha del documento
12. Nombre del destinatario
13. Cargo del destinatario
14. Nombre del remitente
15. Cargo del remitente
16. Botón para continuar tramite
17. Botón guardar y cerrar tarea
18. Botón cerrar tarea

- **Formulario ingresar información al cuerpo del documento**

Esta tarea presenta la información del trámite y el documento .docx creado con los datos especificados en el formulario anterior.

Figura 14. Formulario ingresar información al cuerpo del documento



Elaborado por: Pedro Caicedo y Diego Godoy

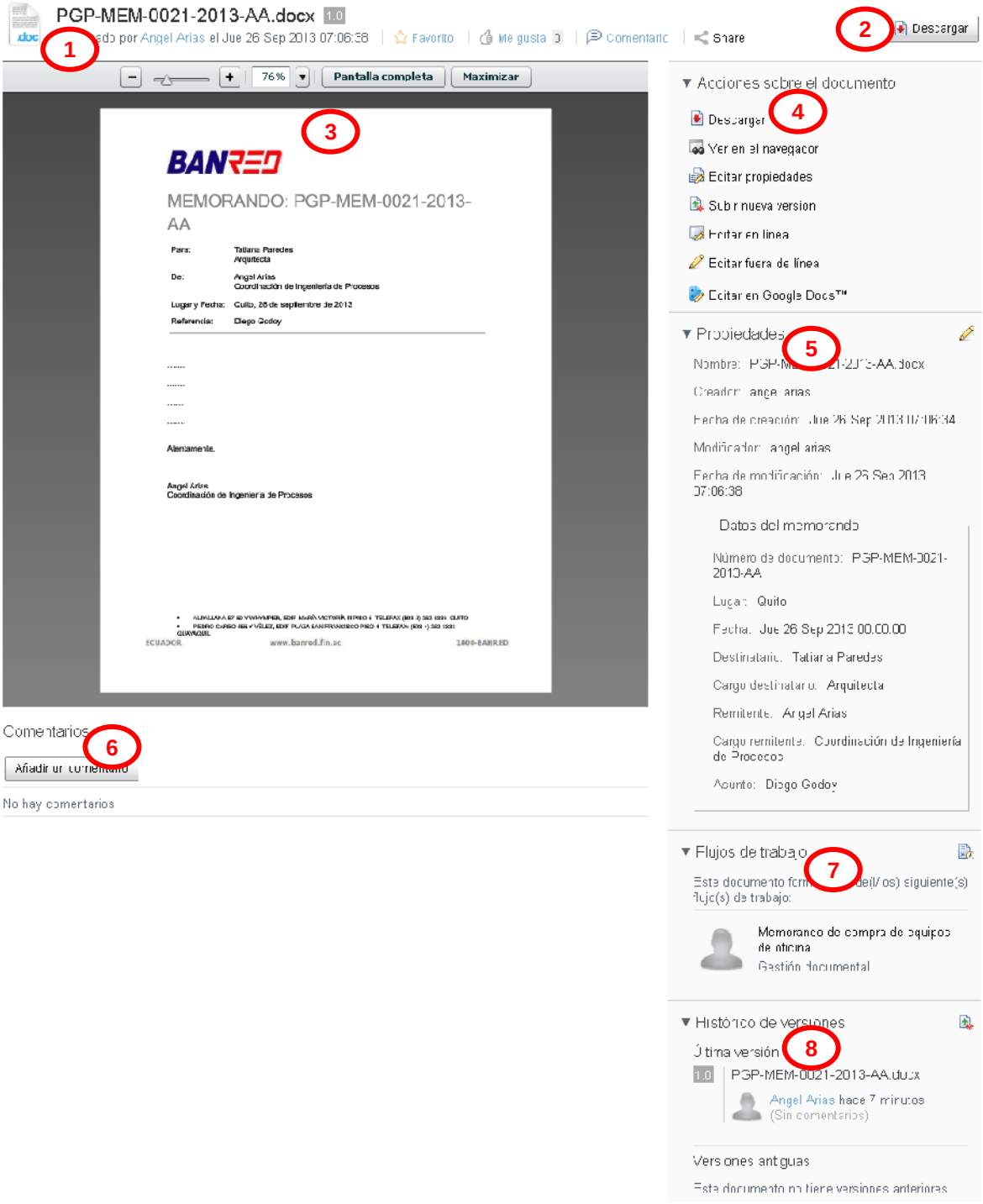
Descripción de componentes:

1. Información del trámite
2. Descripción del trámite
3. Número del trámite
4. Prioridad del trámite
5. Vencimiento del trámite
6. Estado de la tarea actual
7. Información documento generado
8. Acceso acciones sobre documento
9. Añadir adjuntos al trámite
10. Botón continuar trámite
11. Botón guardar y cerrar tarea
12. Botón cerrar tarea

- **Formulario detalles del documento**

Esta tarea presenta la información del documento .docx creado en el trámite.

Figura 15. Formulario detalles del documento



Elaborado por: Pedro Caicedo y Diego Godoy

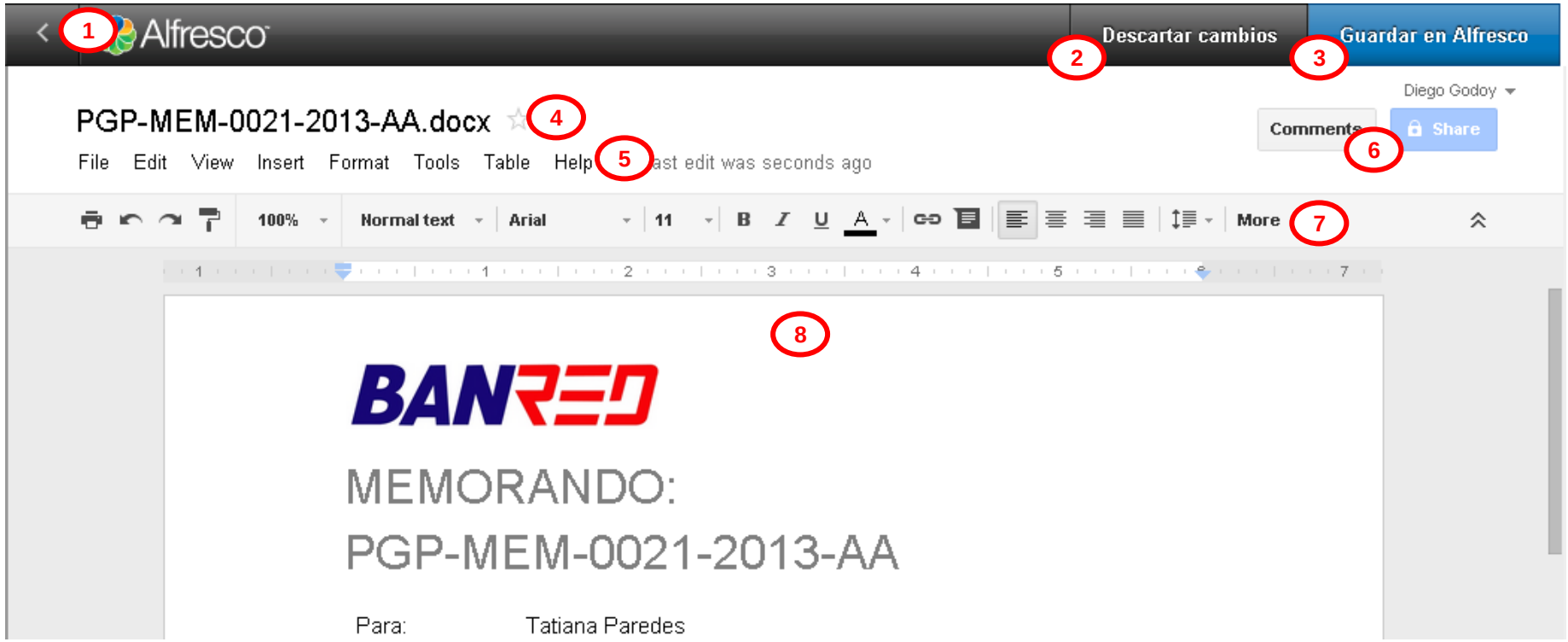
Descripción de componentes:

1. Nombre del documento
2. Botón descarga documento
3. Vista previa del documento
4. Acciones sobre el documento
5. METADATOS del documento
6. Comentarios sobre documento
7. Trámites donde pertenece el documento
8. Historial de versiones del documento

- **Formulario edición de documento con Google Docs**

Esta tarea edita la información del documento .docx creado en el trámite mediante Google Docs. Para habilitar esta funcionalidad es necesario contar con una cuenta de Google.

Figura 16. Formulario edición de documento con Google Docs



Elaborado por: Pedro Caicedo y Diego Godoy

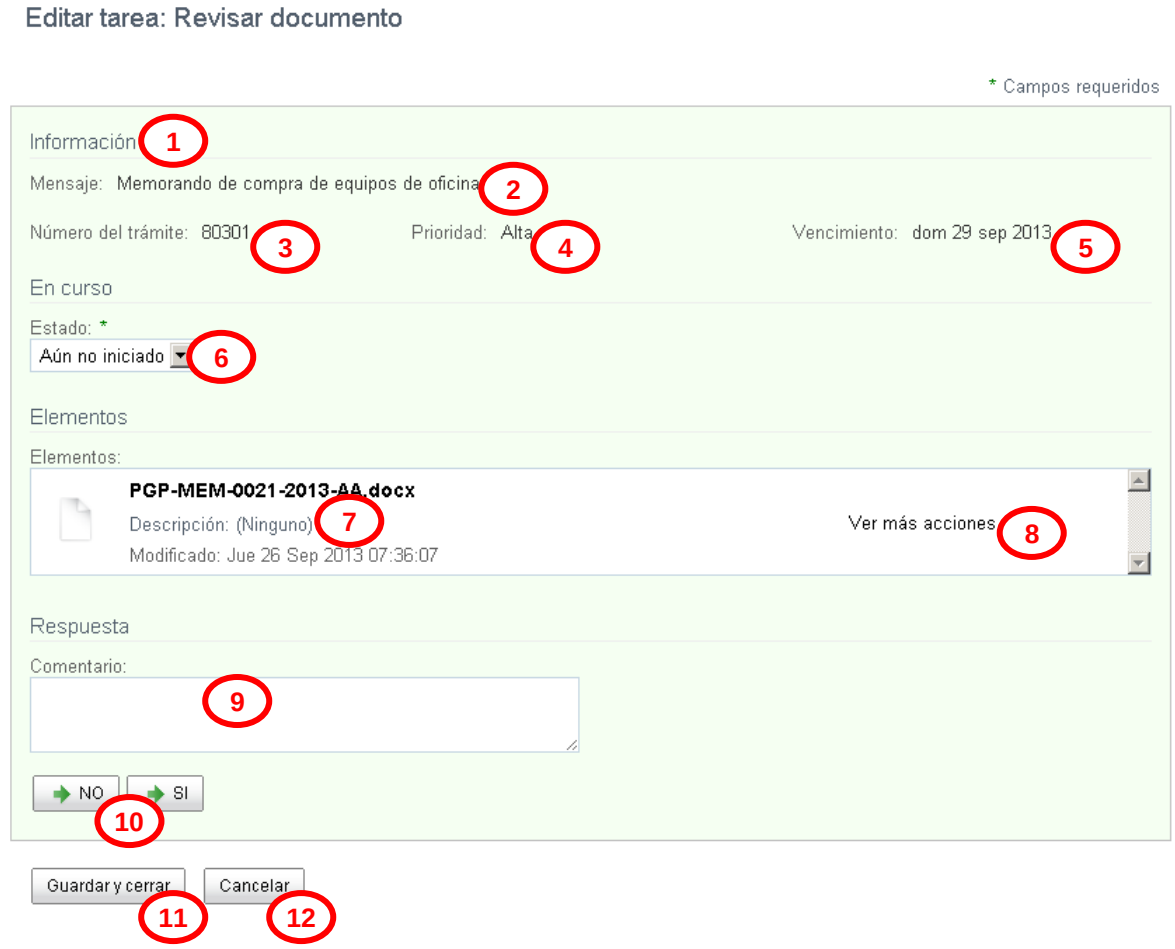
Descripción de componentes:

1. Regresar plataforma ECM
2. Descartar cambios hechos sobre documento
3. Botón para guardar cambios documento
4. Nombre del documento
5. Menú de opciones de Google Docs
6. Herramientas de colaboración Google Docs
7. Barra herramientas Google Docs
8. Área de trabajo de Google Docs

- **Formulario revisar documento**

Esta tarea realiza la revisión y aprobación del documento.

Figura 17. Formulario revisar documento



Elaborado por: Pedro Caicedo y Diego Godoy

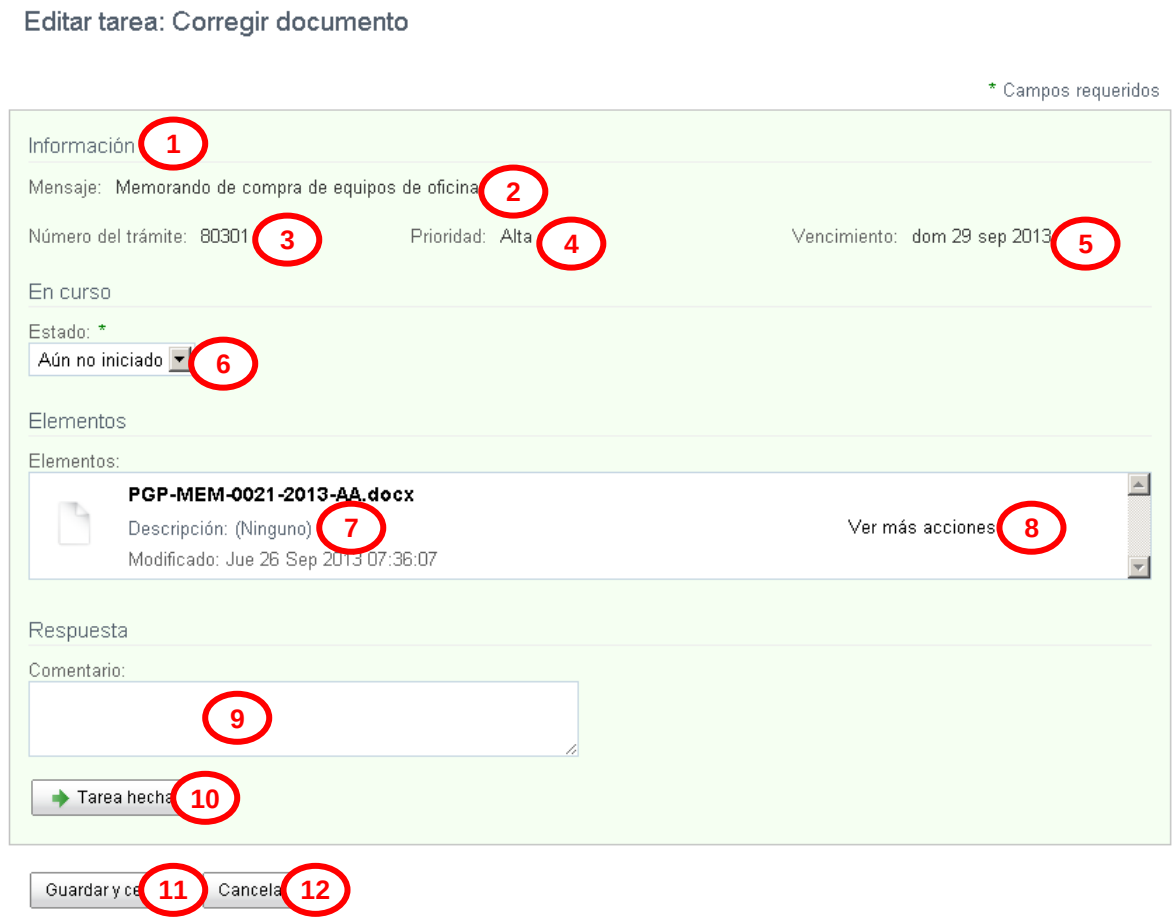
Descripción de componentes:

1. información del trámite
2. descripción del trámite
3. número del trámite
4. prioridad del trámite
5. vencimiento del trámite
6. estado de la tarea actual
7. información documento
8. ver contenido del documento
9. comentario de la revisión
10. aprobación del documento
11. botón guardar y cerrar tarea
12. botón cerrar tarea

- **Formulario corregir documento**

Esta tarea realiza la corrección necesaria al documento.

Figura 18. Formulario corregir documento



Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

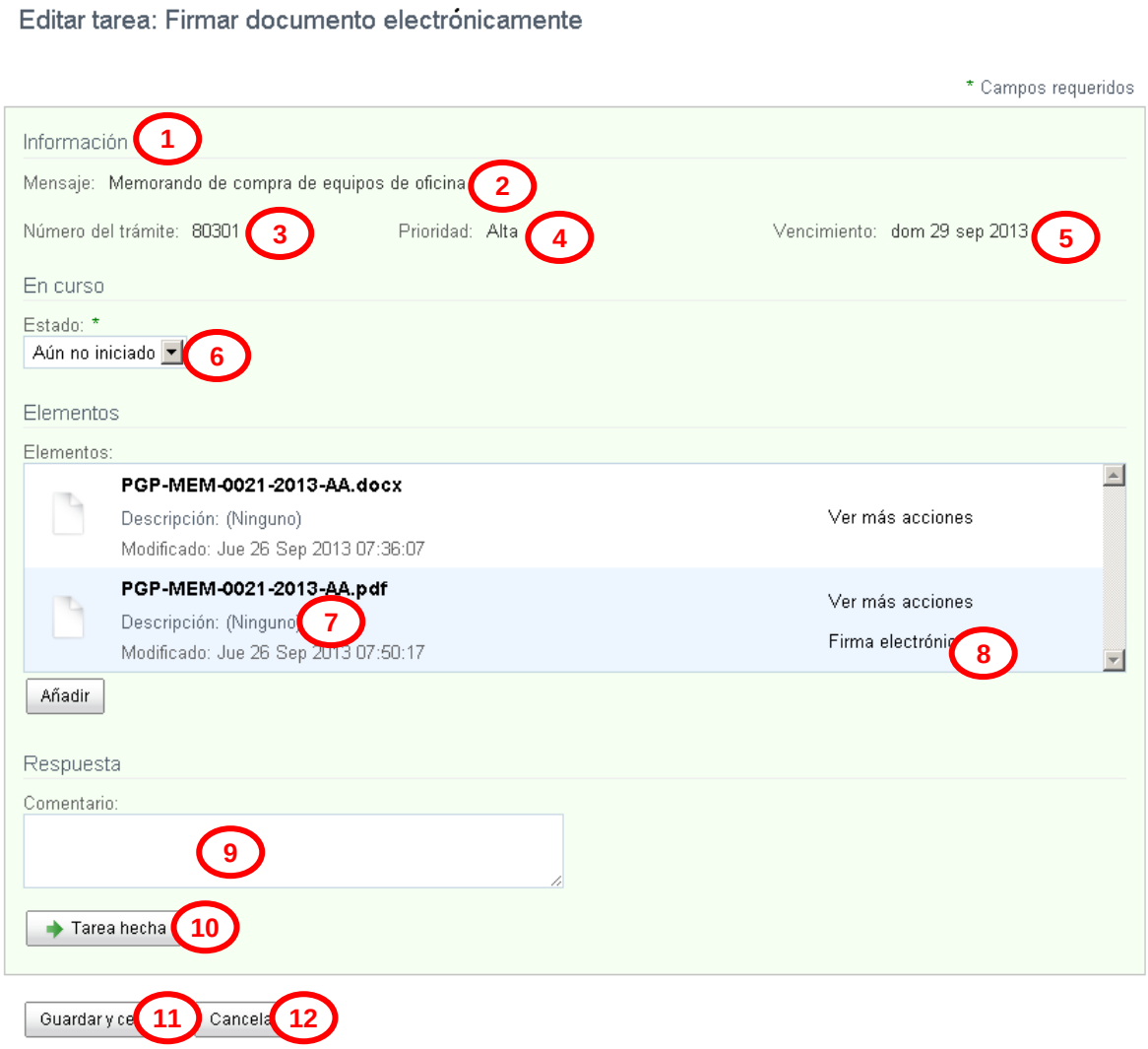
1. información del trámite
2. descripción del trámite
3. número del trámite
4. prioridad del trámite
5. vencimiento del trámite
6. estado de la tarea actual
7. información documento generado

8. ver contenido para editar documento
9. comentario de la corrección
10. botón continuar trámite
11. botón guardar y cerrar tarea
12. botón cerrar tarea

- **Formulario firmar documento electrónicamente**

Esta tarea registra la firma electrónica sobre el documento.

Figura 19. Formulario firmar documento electrónicamente



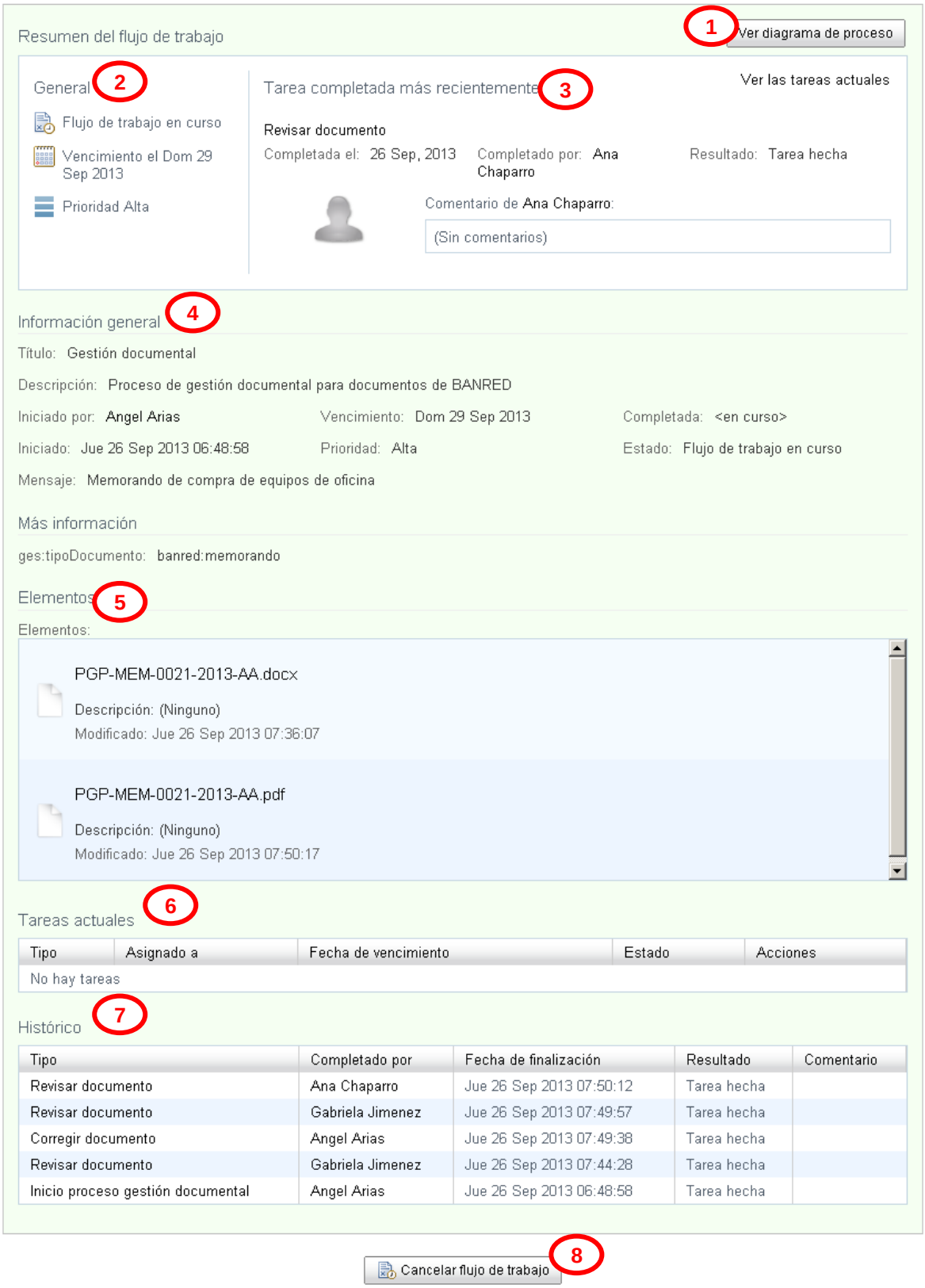
Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Información del trámite
2. Descripción del trámite
3. Número del trámite
4. Prioridad del trámite
5. Vencimiento del trámite
6. Estado de la tarea actual
7. Información documento con formato .pdf para firmar
8. Acción para llamar al aplicativo de firma electrónica
9. Comentario
10. Botón continuar trámite
11. Botón guardar y cerrar tarea
12. Botón cerrar tarea

- **Formulario detalles del trámite**

Figura 20. Formulario detalles del trámite



Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

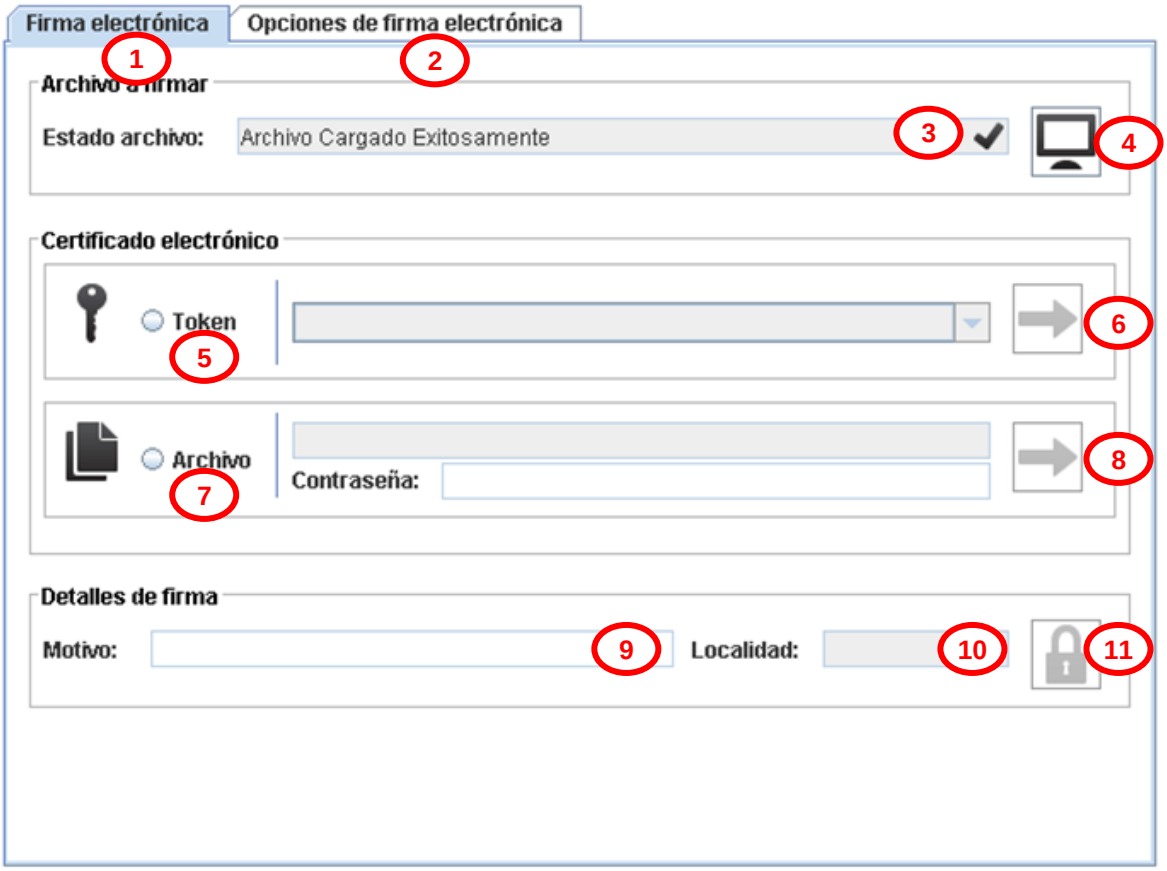
1. Ver diagrama del trámite
2. Detalles generales trámite
3. Última tarea realizada en el trámite
4. Detalles del trámite
5. Documentos del trámite
6. Tareas actuales del trámite
7. Histórico de actividades en el trámite
8. Cancelar el trámite

1.2.3 Aplicativo de firma electrónica

A continuación se describen los componentes del aplicativo de firma electrónica desarrollado en para este proyecto.

- **Interfaz de firma electrónica**

Figura 21. Interfaz firma electrónica



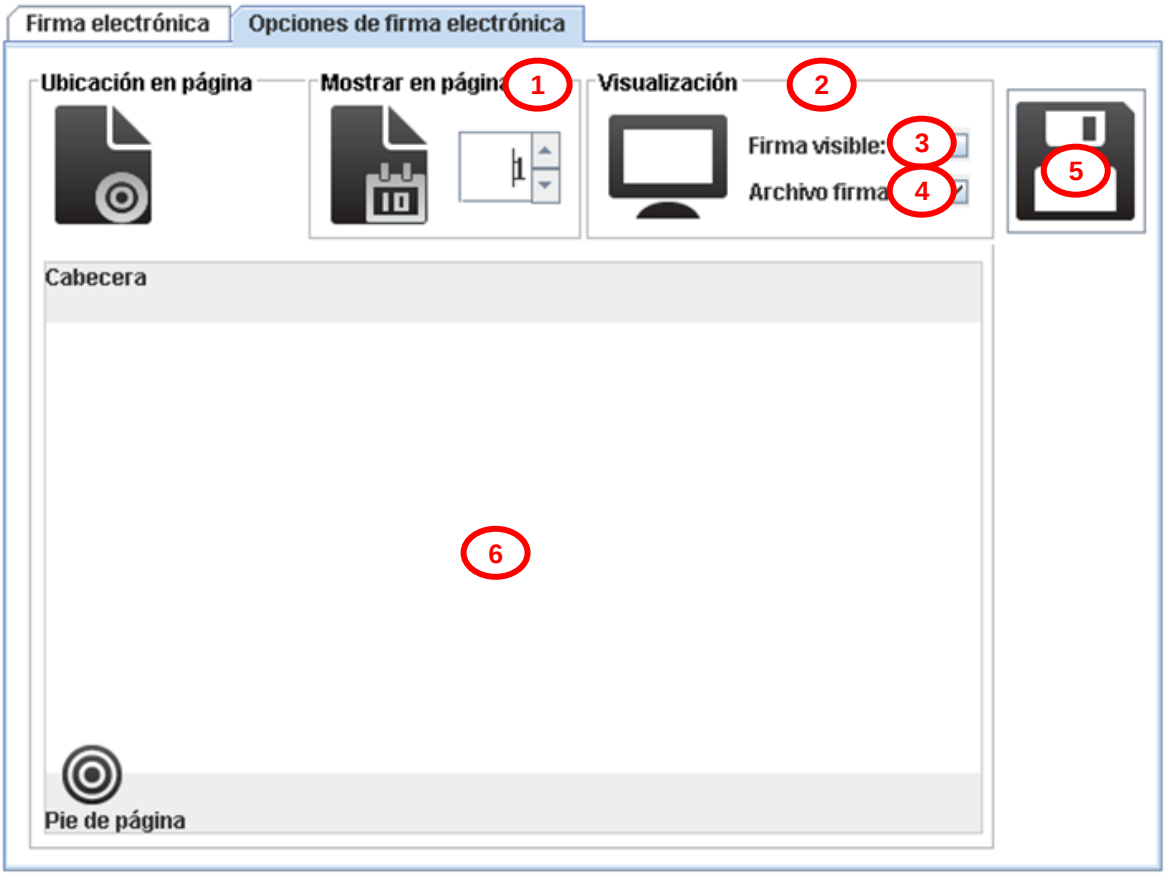
Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Pestaña firma electrónica
2. Pestaña opciones de firma
3. Estado archivo a firmar
4. Visualizar documento a firmar
5. Opción para firmar con token
6. Ingresar datos del token
7. Opción para firmar con archivo
8. Seleccionar certificado archivo
9. Motivo de firma
10. Localidad de firma
11. Botón para firmar documento

- **Parámetros de firma electrónica**

Figura 22. Parámetros de firma electrónica



Elaborado por: Pedro Caicedo y Diego Godoy

Descripción de componentes:

1. Mostrar firma en pagina
2. Opciones de visualización
3. Firma visible en documento
4. Ver archivo una vez firmado
5. Guardar opciones de firma
6. Ubicación de firma en página

1.3 Plan de entrevista para dimensionamiento del proceso gestión documental de BANRED

Con objeto de garantizar el éxito del proyecto, se solicita responder las siguientes preguntas:

1. Necesidades

1.1 ¿Cuáles son sus necesidades? (Indicar cada una en orden prioritario)
1.2 ¿Cuáles son sus objetivos con la implementación de un ECM?
1.3 ¿El diseño de la interfaz visual tiene una prioridad alta en su negocio?
1.4 Número usuarios:
1.5 ¿Cuál es el número de usuarios en que incrementa en el año?

2. Documentación

2.1 ¿Cuáles son los documentos administrativos que se gestionan?
2.2 ¿Cuáles son los atributos de cada tipo de documento?
2.3 ¿Qué formato de archivos usan?
2.4 ¿Los documentos se generan mediante plantillas de documentos?
2.5 ¿Se manejan versiones históricas de documentos?
2.6 ¿Cuenta con una política de organización y/o nombrado de documentos? (Entregar documentos):
2.7 ¿Puntualice la estructura jerárquica con la cual se almacenan los documentos?
2.8 ¿La estructura anterior, se encuentra almacenada de forma centralizada o distribuida? (Si es distribuida, indicar el esquema).
2.9 ¿Cómo se relaciona la estructura anterior con los permisos a usuarios? (Indicar los roles de acceso).
2.10 ¿Cómo un usuario común, accede a un documento?

3. Reportes

3.1 Detalle los reportes que se utilizan actualmente y los deseados.

3.2 ¿Se requieren reportes de auditoría?

4. Migración

4.1 ¿Cuál es su repositorio actual de documentos?

4.2 ¿En el nuevo flujo se solicitan versionamientos de documentos?

5. Procesos

5.1 ¿Los documentos tienen un flujo de aprobación?

5.2 ¿Cuál es el ciclo documental instaurado en su empresa?

6. Herramientas de software

6.1 ¿Poseen alguna herramienta para gestión documental? de ser positiva la respuesta,

6.1.1 ¿Cuál es esta?

6.1.2 ¿Existe integración con otras herramientas?

6.2 ¿Utilizan alguna herramienta para firmar electrónicamente los documentos digitales?, de ser positiva la respuesta,

6.2.1 ¿Cuál es esta?

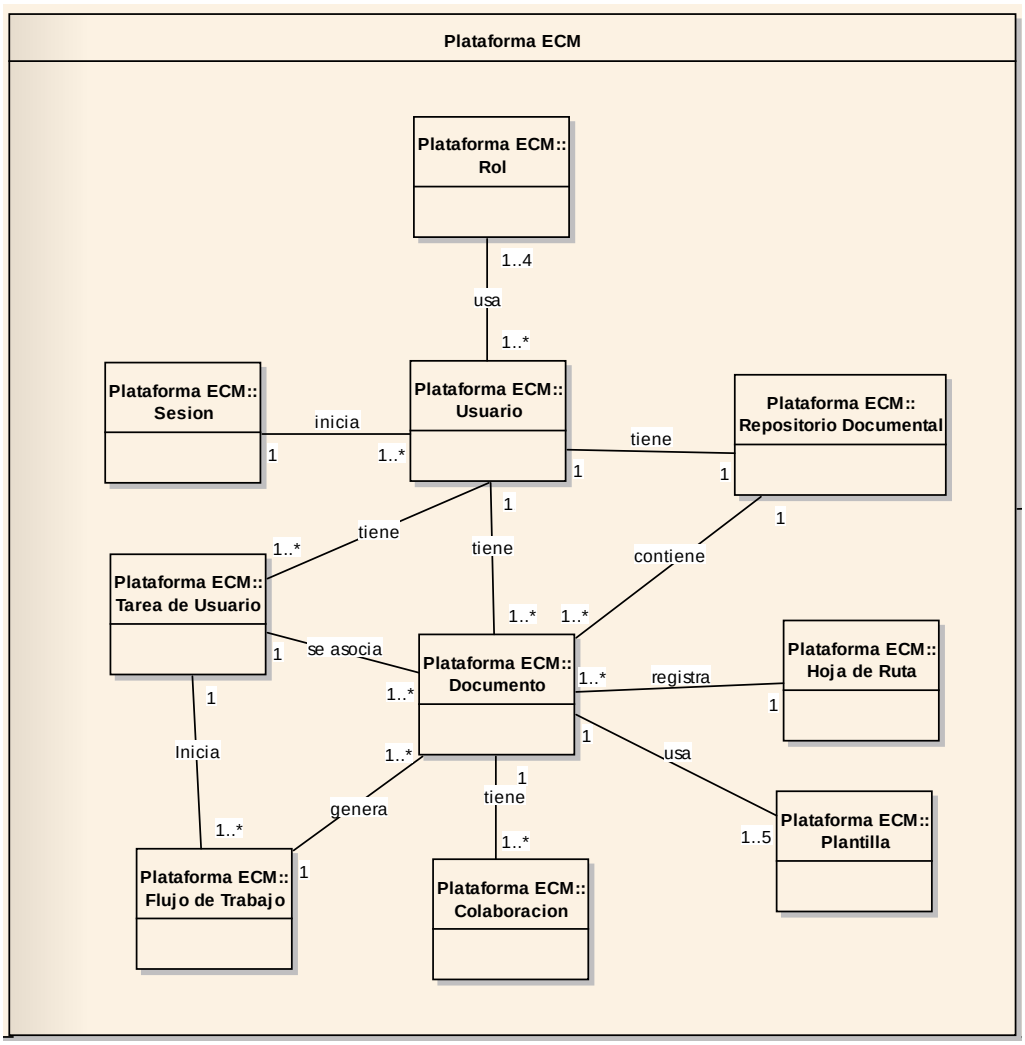
6.2.2 ¿Está integrada al ciclo de vida documental?

6.3 ¿Utilizan algún aplicativo para gestión de recursos de la red LAN? ?, de ser positiva la respuesta,

6.3.1 ¿Cuál es esta?

6.3.2 ¿Está integrada al ciclo de vida documental?

1.4 Modelo de domino plataforma ECM ALFRESCO



1.5 Plan de pruebas

El presente documento describe los métodos usados para verificar que la solución propuesta satisface la especificación requerida y las necesidades del cliente. Incluye los tipos de prueba, las herramientas de software, la configuración del ambiente de pruebas y el cronograma de trabajo.

TIPO DE PRUEBA	DESCRIPCIÓN TIPO DE PRUEBA	HERRAMIENTA ESPECÍFICA	DESCRIPCIÓN SOFTWARE
Funcionalidad	La prueba funcional es un proceso que permite encontrar discrepancias entre el software desarrollado y/o configurado y la especificación funcional. La prueba funcional es una actividad de caja negra y permite validar: • Los procesos y reglas de negocio establecidas • Que se cumplan los requerimientos funcionales definidos En esta prueba se validan los casos de uso que fueron aprobados por el cliente, y a partir de ellos se diseñan y ejecutan los casos de pruebas correspondientes.	No se utiliza software, debido a que este tipo de prueba verifica la implementación de los casos de uso en la solución propuesta de forma manual.	
Optimización de código	La prueba de optimización de código se enfoca hacia la la calidad del software, enfatizando las siguientes tareas. · Reducir el código que no se utiliza. · Reducir el código duplicado. · Y sobre todo, reducir las funcionalidades duplicadas, tanto a nivel de negocio como elementos de la arquitectura.	CodePro Analytix	El API CodePro Analytix de la compañía Instantiations para el IDE Eclipse, es una herramienta de gestión de la calidad del software. Ofrece un entorno para evaluación de código, métricas, análisis de dependencias, cobertura de código, generación de pruebas unitarias, etc.

Configuración del ambiente de pruebas

El ambiente de pruebas establecido se cita a continuación.

COMPONENTE	CONFIGURACIÓN	SOFTWARE INSTALADO Y CONFIGURADO	CANTIDAD
Servidor 1	**Procesador:** Core I3 2.27 GHz(4 núcleos) **Memoria:** 4GB DDR3 **Disco Duro:**100GB	• Ubuntu V. 12.04 Linux (64 bits) • MYSQL V. 5.1 • ALFRESCO 4.2 • Java JKD 1.7	1
Servidor 2	**Procesador:** Core I3 2.27 GHz (1 núcleo) **Memoria:** 1.5 GB **Disco Duro:** 20 GHz	• Windows Server 2003 (32 bits) • Directorio Activo 2003	1
Estaciones de Trabajo	**Procesador:** Intel Pentium V **Memoria RAM:** 1.5 GB **Disco Duro:** 80 GB **Red:** Acceso a la red local y a Internet	Sistema Operativo: Windows XP, Navegador web Chrome v. 29.0.1547.76 Office 2007.	1

Cronograma de trabajo

El cronograma de trabajo planificado para ejecución del plan de pruebas, se muestra a continuación.

TIPO DE PRUEBAS	TIEMPO DE EJECUCIÓN ESTIMADO (horas)
Funcionalidad	15
Optimización de código	20
Carga	15

1.6 Casos de prueba

1.6.1 Pruebas de funcionalidad

1.6.1.1 Caso de prueba tareas pendientes

Tabla 18. Caso de prueba tareas pendientes

PRU-007 Tareas Pendientes		
Propósito:	Verificar la correcta funcionalidad en cuanto a Tareas Pendientes.	
Prerrequisitos:	Ingresar al sistema, usuario funcionario.	
Datos de Prueba:	Datos para mostrar Tareas Pendientes.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Visualizar en la pantalla principal las tareas pendientes 3. Clic en el link Realizar Tarea	
RESULTADO		
ID	**Descripción:**	**Estado**
1.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy.

1.6.1.2 Caso de prueba estado trámite

Tabla 19. Caso de prueba estado trámite

PRU-008 Estado Trámite	
Propósito:	Verificar la correcta funcionalidad en cuanto a Estado Tramite.
Prerrequisitos:	Ingresar al sistema, usuario funcionario.
Datos de Prueba:	Datos para buscar Estado Tramite.

PRU-008 Estado Trámite		
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Ir a la opción Gestión de Tramites 3. Seleccionar opción Estado Tramite 4. Ingresar el numero de trámite 5. Clic en el botón buscar	
RESULTADO		
ID	**Descripción:**	**Estado**
2.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy

1.6.1.3 Caso de prueba involucrados

Tabla 20. Caso de prueba involucrados

PRU-009 Involucrados		
Propósito:	Verificar la correcta funcionalidad en cuanto a Involucrados.	
Prerrequisitos:	Ingresar al sistema, usuario funcionario.	
Datos de Prueba:	Datos para buscar Involucrados.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Ir a la opción Gestión de Tramites 3. Seleccionar opción Involucrados 4. Ingresar el numero de trámite 5. Clic en el botón buscar	
RESULTADO		
ID	**Descripción:**	**Estado**
3.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy

1.6.1.4 Caso de prueba Hoja de Ruta

Tabla 21. Caso de prueba Hoja de Ruta

PRU-010 Hoja Ruta		
Propósito:	Verificar la correcta funcionalidad en cuanto a Hoja Ruta.	
Prerrequisitos:	Ingresar al sistema, usuario funcionario.	
Datos de Prueba:	Datos para buscar Hoja Ruta.	
Pasos:	**Flujo Principal** 1. Ingresar a la aplicación. 2. Ir a la opción Gestión de Tramites 3. Seleccionar opción Hoja Ruta 4. Ingresar el numero de trámite 5. Clic en el botón buscar	
RESULTADO		
ID	**Descripción:**	**Estado**
4.	**PASOS PARA REPRODUCIR:**	Cerrado

Elaborado por: Pedro Caicedo y Diego Godoy

1.6.2 Optimización de código

1.6.2.1 Media Complejidad Ciclomática (Average Cyclomatic Complexity)

- **Código inicial**

```
public void inicializar() {
		LOG.info("INICIO DEL BEAN");
		serverECM = AplicacionUtil.SERVER_ECM;
		server = AplicacionUtil.SERVER;
		try {
			usuario = (Usuario) AplicacionUtil.tomarAtributo("usuario");
			if (usuario == null) {
				if (AplicacionUtil.tomarParámetro("usuario") == null) {
					ExternalContext externalContext =
					FacesContext.getCurrentInstance().getExternalContext();

		externalContext.redirect(AplicacionUtil.SERVER_LOGIN);
				}else {
					//usuario == BUSCAR USUARIO
					usuario = new Usuario();
					//ELIMINAR LA LINEA DE ABAJO

				//usuario.setLdap(AplicacionUtil.tomarParámetro("usuario"));
					usuario.setLdap("ana.chaparro");

					usuario.setTipo(AplicacionUtil.tomarParámetro("tipo"));
				AplicacionUtil.setearAtributo("usuario", usuario);
				if(usuario.getTipo().equalsIgnoreCase("FUNCIONARIO")) {
						banderaMenu = false;
						mensaje = "Menu funcionario";
						redirect(usuario.getTipo());
					}

				if (usuario.getTipo().equalsIgnoreCase("ADMINISTRADOR")) {
						banderaMenu = true;
						mensaje = "Menu administrador";
						redirect(usuario.getTipo());
					}
				}
			}else {

				if (usuario.getTipo().equalsIgnoreCase("FUNCIONARIO")) {
					banderaMenu = false;
					mensaje = "Menu funcionario";
```

```
                    }

                    if (usuario.getTipo().equalsIgnoreCase("ADMINISTRADOR")) {
                        banderaMenu = true;
                        mensaje = "Menu administrador";
                    }
                }
            } catch (Exception e) {
        }
    }
```

- Código depurado

```
    Public void inicializar() {
        serverECM = AplicacionUtil.SERVER_ECM;
        server = AplicacionUtil.SERVER;
        try {
            usuario = (Usuario) AplicacionUtil.tomarAtributo("usuario");
            if (usuario == null) {
                loginUser();
            }else {
                perfilUser();
            }

        } catch (Exception e) {

        }
    }

    public void perfilUser() {
        if (usuario.getTipo().equalsIgnoreCase("FUNCIONARIO")) {
            banderaMenu = false;
            mensaje = "Menu funcionario";
        }

        if (usuario.getTipo().equalsIgnoreCase("ADMINISTRADOR")) {
            banderaMenu = true;
            mensaje = "Menu administrador";
        }
    }

    public void loginUser(){

        try {
            if (AplicacionUtil.tomarParámetro("usuario") == null) {
                ExternalContext externalContext = FacesContext.getCurrentInstance().getExternalContext();
                externalContext.redirect(AplicacionUtil.SERVER_LOGIN);
```

```
                }else {
                        usuario = new Usuario();
                        usuario.setLdap("ana.chaparro");

                        usuario.setTipo(AplicacionUtil.tomarParámetro("tipo"));
                        AplicacionUtil.setearAtributo("usuario", usuario);
                        if (usuario.getTipo().equalsIgnoreCase("FUNCIONARIO")) {
                                banderaMenu = false;
                                mensaje = "Menu funcionario";
                                redirect(usuario.getTipo());
                        }

                        if (usuario.getTipo().equalsIgnoreCase("ADMINISTRADOR")) {
                                banderaMenu = true;
                                mensaje = "Menu administrador";
                                redirect(usuario.getTipo());
                        }
                }
        } catch (Exception e) {
                LOG.info("Error en inicio Bean"+e.getLocalizedMessage());
        }
}
```

Como se observa, la optimización realizada fue descomponer al método *inicializar()* en los métodos *inicializar(), perfilUser()* y *loginUser().*

1.6.2.2 Número de campos (number of fields)

A continuación se muestra la depuración de código realizada sobre la clase *AplicacionUtil.*

- **Código inicial:**

```
Public class AplicacionUtil {

        public staticfinal String PATH_BDD="F://database.properties";
                public staticfinal String PATH_ECM="/opt/banred/ECM.properties";
                public staticfinal String PATH_ldap="/opt/banred/ldap.properties";

                public staticfinal String SERVER="http:////localhost:8080/portalbanred";
                public staticfinal String SERVER_LOGIN="http://192.168.0.4:8080/share/page/";
```

```
        public staticfinal String SERVER_LOGOUT=
"http://192.168.0.4:8080/share/page/dologout";

public staticfinal String SERVICIO_IMAGEN="http://192.168.0.4:8080/ALFRESCO/s/api/workflow-instances/activiti$$numeroTramite/diagram";
public staticfinal String SERVICIO_INVOLUCRADOS="http://192.168.0.4:8080/ALFRESCO/s/api/workflow-instances/activiti$$numeroTramite?includeTasks=true";
public staticfinal String SERVICIO_HOJA_RUTA="http://192.168.0.4:8080/ALFRESCO/s/api/workflow-instances/activiti$$numeroTramite?includeTasks=true";
public staticfinal String SERVICIO_TAREAS_PENDIENTES="http://192.168.0.4:8080/ALFRESCO/s/api/task-instances?authority=$usuario";
public staticfinal String SERVICIO_AUDITORIA="http://192.168.0.4:8080/ALFRESCO/s/api/audit/query/$tipo?verbose=true&fromTime=$inicio&toTime=$fin&limit=9999$usuario";
        public staticfinal String SERVER_ECM ="http://192.168.0.4:8080";
public staticfinal String MAS_INFORMACION_HOJA_RUTA="http://192.168.0.4:8080/share/page/workflow-details?workflowId=activiti$$numeroTramite&referrer=workflows&myWorkflowsLinkBack=true#current-tasks";
public staticfinal String REALIZAR_TAREA_TAREAS_PENDIENTES="http://192.168.0.4:8080/share/page/task-edit?taskId=$numeroTarea&amp;referrer=tasks";
        public staticfinal String USER_ADMIN="admin";
        public staticfinal String PASS_ADMIN="admin";
        public staticfinal String DOMINIO_ldap = "applications.banred-test.fin.ec";
        public staticfinal String USER_ldap = "administrador";
        public staticfinal String CLAVE_ldap = "banred2012";
        public staticfinal String SERVIDOR_ldap = "192.168.0.5";
public staticfinal String URL_CONEXION_ldap = "CN=Users,DC=applications,DC=banred-test,DC=fin,DC=ec";
```

Código Optimizado

```
Public class AplicacionUtil {

public staticfinal String MAS_INFORMACION_HOJA_RUTA="http://192.168.0.4:8080/share/page/workflow-details?workflowId=activiti$$numeroTramite&referrer=workflows&myWorkflowsLinkBack=true#current-tasks";
public staticfinal String REALIZAR_TAREA_TAREAS_PENDIENTES="http://192.168.0.4:8080/share/page/task-edit?taskId=$numeroTarea&amp;referrer=tasks";
public staticfinal String USER_ADMIN="admin";
public staticfinal String PASS_ADMIN="admin";
public staticfinal String DOMINIO_ldap = "applications.banred-test.fin.ec";
public staticfinal String USER_ldap = "administrador";
public staticfinal String CLAVE_ldap = "banred2012";
public staticfinal String SERVIDOR_ldap = "192.168.0.5";
public staticfinal String URL_CONEXION_ldap = "CN=Users,DC=applications,DC=banred-test,DC=fin,DC=ec";
public static String PATH_BDD(){
        return "F://database.properties";
}

public static String PATH_ECM(){
```

```
        return "/opt/banred/ECM.properties";
}

public static String PATH_ldap(){
        return "/opt/banred/ldap.properties";
}

public static String SERVER(){
        return "http:////localhost:8080/portalbanred";
}

public static String SERVER_LOGIN(){
        return "http://192.168.0.4:8080/share/page/";
}

public static String SERVER_LOGOUT(){
        return "http://192.168.0.4:8080/share/page/dologout";
}

public static String SERVICIO_IMAGEN(){
        return "http://192.168.0.4:8080/ALFRESCO/s/api/workflow-instances/activiti$$numeroTramite/diagram";
}

public static String SERVICIO_INVOLUCRADOS(){
        return "http://192.168.0.4:8080/ALFRESCO/s/api/workflow-
        instances/activiti$$numeroTramite?includeTasks=true";
}

public static String SERVICIO_HOJA_RUTA(){
        return "http://192.168.0.4:8080/ALFRESCO/s/api/workflow-
        instances/activiti$$numeroTramite?includeTasks=true";
}

public static String SERVICIO_TAREAS_PENDIENTES(){
        return "http://192.168.0.4:8080/ALFRESCO/s/api/task-instances?authority=$usuario";
}

public static String SERVICIO_AUDITORIA(){
        return
        "http://192.168.0.4:8080/ALFRESCO/s/api/audit/query/$tipo?verbose=true&fromTime=$inicio&toTime=$fin
        &limit=9999$usuario";
}

public static String SERVER_ECM(){
        return "http://192.168.0.4:8080";
        }
```

1.7 Lista de referencias

Alfresco. (17 de 08 de 2012). *Archivos compartidos una verdadera experiencia*. Obtenido de http://www.ALFRESCO.com/es/noticias/comunicados-de-prensa/ALFRESCO-da-el-salto-de-los-archivos-compartidos-una-verdadera

Alfresco. (20 de 08 de 2012). *Arquitectura Alfresco*. Obtenido de https://docs.google.com/file/d/0B3rUn70ppzM3OURucFhnbmlXdU0/edit

Alfresco. (21 de 08 de 2012). *Soporte de suscripción de servicios*. Obtenido de http://www.slideshare.net/ALFRESCO/ALFRESCO-support-subscription-services-explained

Alfresco. (8 de 10 de 2012). *Test rendimiento*. Obtenido de http://iaranda.wordpress.com/2011/02/02/test-de-rendimiento-en-ALFRESCO/

BLYX. (18 de 08 de 2012). *Consejos sobre logs Alfresco*. Obtenido de http://blyx.com/2011/06/02/consejos-sobre-los-logs-en-ALFRESCO/

Entreprise, A. (09 de 10 de 2012). *Comunicados de prensa*. Obtenido de http://www.ALFRESCO.com/es/noticias/comunicados-de-prensa/ALFRESCO-da-el-salto-de-los-archivos-compartidos-una-verdadera

1.8 Manual de usuario “Simulación de interoperabilidad de la plataforma ECM con portal web"

CONTENIDO

INTRODUCCIÓN ……1

INGRESO AL SISTEMA ……2

ESTRUCTURA Y FUNCIONALIDAD DEL SISTEMA ……..2

1.1 Menú de acceso ……3

1.2 Administración de usuarios ……....3

1.2.3 Auditoría…….……...6

1.2.4 Parametrización……...…..8

2.2. Tareas Pendientes ……...10

2.3. Gestión de Trámites ……...11

2.4. Ejecución de trámites ……...13

2.4.1 Iniciar trámite ……...13

2.4.2 Tarea llenar metadatos en formulario web….. ……...14

2.4.3 Tarea ingresar información en el cuerpo del documento ……15

2.4.4 Detalles del documento ……...16

2.4.5 Formulario editar documento con Google Docs ……18

2.4.6 Tarea revisar documento ……...18

2.4.7 Formulario corregir documento ……...19

2.4.8 Tarea firmar documento electrónicamente ……...20

Una vez firmado el documento se necesita dar un clic en el botón “Tarea hecha”.……21

2.4.9 Interfaz de firma electrónica ……...21

2.4.10 Parámetros de firma electrónica ……...22

INTRODUCCIÓN

El presente documento es el manual de usuario del portal BANRED, una herramienta que nos permite realizar la administración de usuarios en la sección de administrador, mientras que en la sección de funcionarios permite realizar tareas sobre trámites.

El objetivo principal es lograr la familiarización del usuario con la utilización de dicha herramienta.

INGRESO AL SISTEMA

La aplicación tiene dos tipos de usuarios con diferentes accesos y permisos los mismos que se detallaran a continuación:

Usuarios Funcionario: La información proporcionada es detallada, actualizada y dirigida a los usuarios que tienen que realizar acciones sobre trámites dentro de la organización.

Usuarios Administrativos: Son los administradores del modulo, tienen capacidad que de ver información de auditoría, así como también administrar información del sistema, y roles sobre los usuarios.

Para ingresar al **Portal BANRED**, se inicia abriendo un explorador de internet (browser) y se digita la siguiente dirección: http://direcciónIpdelServidor:8080/share.

A continuación, aparecerá la ventana de ingreso al Sistema, en donde, pueden ingresar los usuarios que se encuentren registrados.

En esta pantalla se deben realizar los siguientes pasos:

- Ingresar el nombre de usuario**.**
- Ingresar la contraseña.
- Hacer un clic en el botón “Iniciar sesión”.

ESTRUCTURA Y FUNCIONALIDAD DEL SISTEMA

De acuerdo a los requerimientos del usuario, la página principal del sistema se encuentra estructurada de la siguiente forma:

Opciones generales del sistema (parte superior derecha de la página)

- *Información de usuario registrado*
- *Salir*

Funcionalidades del sistema

Usuario administrador

Administración de usuarios del Portal

Auditoria del portal

Auditoria de plataforma ECM

Conexión con base de datos

Conexión con plataforma ECM

Conexión con Directorio Activo

Usuario funcionario

Tareas pendientes

Estado del trámite

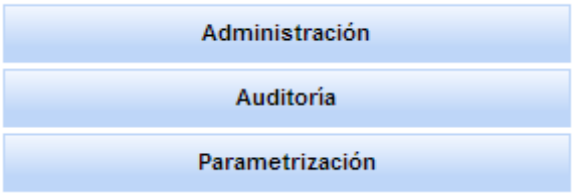
Ejecutar el trámite

FUNCIONALIDADES PARA USUARIO ADMINISTRADOR

La interfaz para usuario este usuario presentan todas las funcionalidades necesarias para administrar el portal web.

1.1 Menú de acceso

A continuación se presenta el menú de acceso que se presenta en el sistema para los usuarios administradores.



1.2 Administración de usuarios

En esta sección el administrador podrá tener acceso y ver toda la información acerca de todos los funcionarios que están dentro de la organización así también puede editar entre administrador y funcionario el perfil de un usuario.

1.2.1 Búsqueda de Usuarios

El sistema permite al usuario administrador tener acceso de la información de usuarios entre la cual puede buscar por los diferentes filtros que a continuación se presentan:

NOMBRE: el nombre debe ser escrito el usuario LDAP

CEDULA: la cédula debe ser el identificador del usuario

Para realizar una búsqueda de usuarios se necesita realizar lo siguiente:

Ingresar el nombre de usuario, cédula o caso contrario dejar en blanco estos filtros

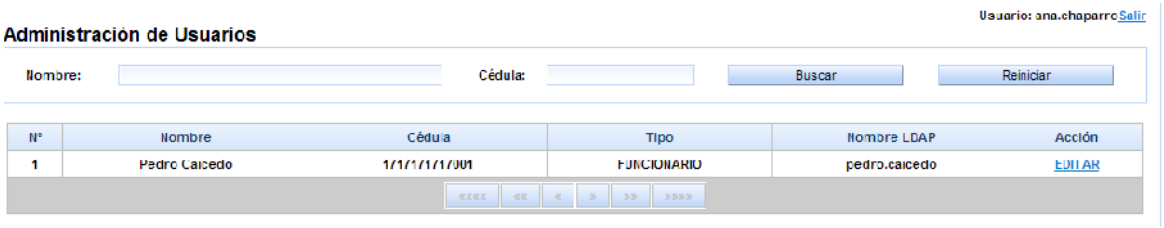
Presionar el botón buscar

Visualizar resultados

Al realizar la acción sobre el botón buscar se mostrará una grilla con la información de usuarios que encuentre en el Directorio Activo y / o en la Base de Datos.

La información debe estar registrada en el Active Directory y solo aquí se puede modificar sus valores.

A continuación se presenta un ejemplo de un resultado de búsqueda de usuarios en el sistema:



1.2.2 Edición de Usuarios

Para realizar la edición de un usuario es decir su perfil realizar las siguientes acciones:

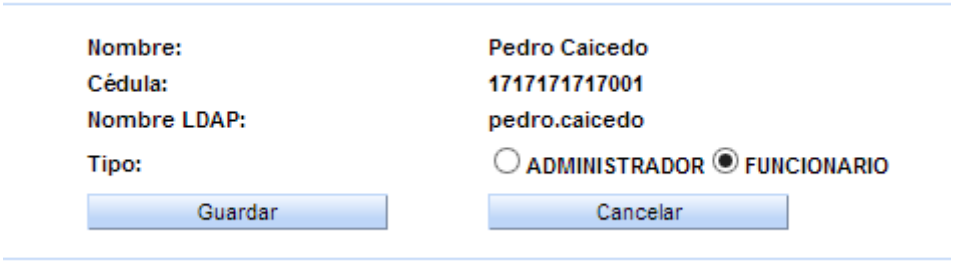
Ubicarse sobre el registro del usuario

Presionar clic en link editar como se presenta a continuación:

Una vez que se haya desplegado la información sobre el usuario seleccionado, el sistema permite:

Presionar clic sobre el perfil del portal que se desee asignar al usuario seleccionado.

Presionar en guardar, caso contrario cancelar para salir.



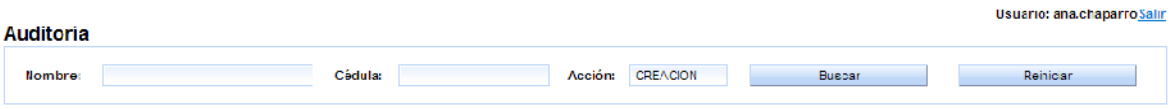
Los cambios que se registren en esta opción serán registrados en la base de datos y no en el Directorio Activo.

1.2.3 Auditoría

El menú auditoría presenta dos opciones dentro del sistema, la primera es para visualizar la auditoría interna del portal y la segunda es la consulta de auditoría a la plataforma ECM. A continuación se presenta la imagen de dicho menú:

1.2.3.1 Auditoria del portal

En esta opción se puede obtener información de auditoría realizada sobre registros basados en la base de datos, la información está basada en los registros que se genera en el menú anterior **Administración usuario.**



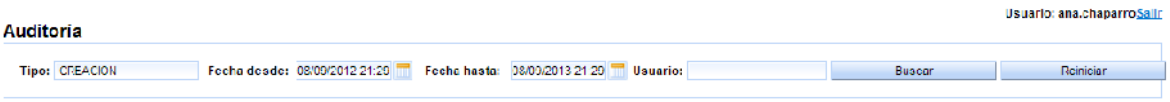
Para realizar una búsqueda de auditoría del portal se necesita seguir los siguientes pasos:

- Ingresar los campos disponibles como filtros.
- Presionar en buscar para mostrar información.
- Presionar reiniciar para limpiar información anterior.

La información que aquí se visualiza se encuentra en la base de datos, y esta se registra en la administración de usuarios.

1.2.3.2 Auditoria plataforma ECM

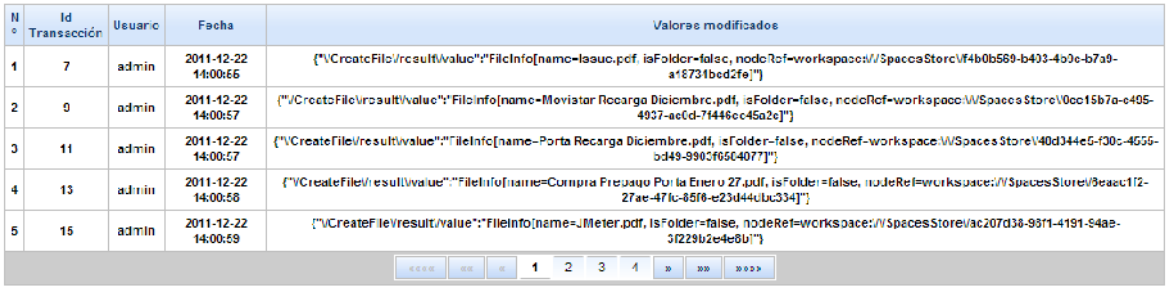
En esta opción se puede obtener información de auditoría realizada sobre la plataforma ECM.



Para realizar una búsqueda de información sobre la auditoría de la plataforma ECM se necesitan realizar los siguientes pasos:

- Ingresar los campos que tenemos como filtros.
- Presionar en buscar para mostrar información.
- Presionar reiniciar para limpiar información anterior.

A continuación se presentará la información recuperada de la plataforma ECM como se presenta en la siguiente imagen:



No	Id Transacción	Usuario	Fecha	Valores modificados
1	7	admin	2011-12-22 14:00:56	{"\/CreateFile\/result\/value":"FileInfo[name=Isauc.pdf, isFolder=false, nodeRef=workspace://SpacesStore/f4b0b569-b403-4b0e-b7a9-a18731bed2fe]"}
2	9	admin	2011-12-22 14:00:57	{"\/CreateFile\/result\/value":"FileInfo[name=Movistar Recarga Diciembre.pdf, isFolder=false, nodeRef=workspace://SpacesStore/0cc15b7e-e495-4937-ac0d-7f446cc45a2e]"}
3	11	admin	2011-12-22 14:00:57	{"\/CreateFile\/result\/value":"FileInfo[name=Porta Recarga Diciembre.pdf, isFolder=false, nodeRef=workspace://SpacesStore/40d344e5-f30c-4659-bd49-9903f6564077]"}
4	13	admin	2011-12-22 14:00:58	{"\/CreateFile\/result\/value":"FileInfo[name=Compra Prepago Porta Enero 27.pdf, isFolder=false, nodeRef=workspace://SpacesStore/3eaac1f2-27ae-47fc-85f6-e23d44dbc334]"}
5	15	admin	2011-12-22 14:00:59	{"\/CreateFile\/result\/value":"FileInfo[name=JMeter.pdf, isFolder=false, nodeRef=workspace://SpacesStore/ac207d38-9811-4191-84ae-3f229b2e4e6b]"}

« « « 1 2 3 4 » » »

La información que aquí se visualiza se encuentra en el ECM, y esta se registra desde el administrador de ALFRESCO.

1.2.4 Parametrización

El menú parametrización presenta las opciones dentro del portal que permiten hacer la conexión con aplicativos o sistemas externos, dichos sistemas comprenden Directorio Activo, base de datos y plataforma ECM:

1.2.4.1 Conexión con base de datos

Para realizar la actualización de conexión del portal con la base de datos en esta sección se necesita llenar todos los campos que se desea modificar y presionar el botón guardar.

Para que los cambios se refresquen en el sistema, el servidor debe ser reiniciado.

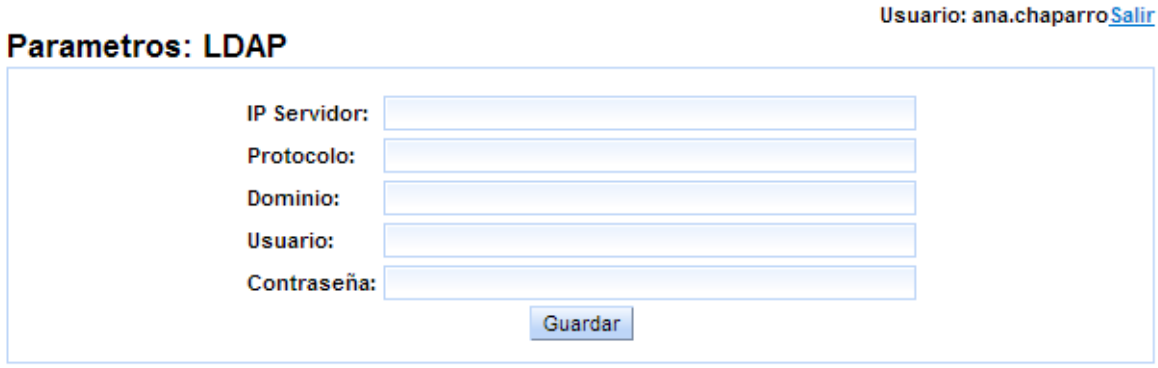
1.2.4.2 Conexión plataforma ECM

Para realizar la actualización de la conexión del portal con la plataforma ECM en esta sección se necesita llenar todos los campos que se desea cambiar y presionar el botón guardar.

Para que los cambios se refresquen en el sistema, el servidor debe ser reiniciado.

1.2.4.3 Conexión con Directorio Activo

Para realizar la actualización de la conexión del portal con el servicio de autenticación de Directorio Activo en esta sección se necesita llenar todos los campos que se desea cambiar y presionar el botón guardar.



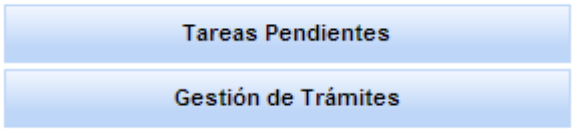
Para que los cambios se refresquen en el sistema, el servidor debe ser reiniciado.

FUNCIONALIDADES PARA USUARIO FUNCIONARIO

En esta sección el usuario funcionario podrá tener acceso a la información relacionada de los trámites de gestión documental.

2.1. Menú de opciones usuario funcionario

A continuación se presenta el menú de acceso que se presenta en el sistema para los usuarios funcionarios.



2.2. Tareas Pendientes

Esta opción informara al usuario registrado todas las tareas pendientes que tiene que realizar. Para realizar la tarea pendiente basta con dar un clic en la acción al lado derecho y esta re direccionara directamente hacia la tarea para realizarla.

No	Título	Descripción	Fecha Vencimiento	Prioridad	Estado	Acciones
1	Revisar documento	Flujo Angel	2013-08-10 10:22:49	Medium	On Hold	REALIZAR TAREA
2	Revisar documento	Prueba Andrres	2013-08-10 10:56:08	Medium	In Progress	REALIZAR TAREA
3	Ingresar informacion del cuerpo documento	Prueba hyo	2013-08-15 21:38:04	Medium	Not Yet Started	REALIZAR TAREA

Esta información es enviada desde la plataforma ECM, al igual la redirección que esta nos realiza hacia el formulario de cada tarea.

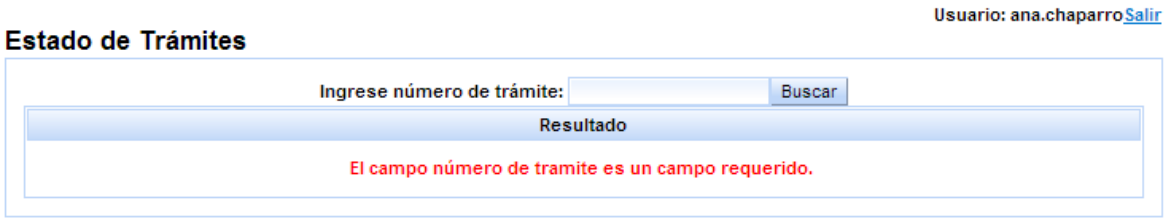
2.3. Gestión de Trámites

Dentro de esta sección el portal permite que los funcionarios puedan visualizar el estado de los trámites de la institución. Dentro de esta sección se presenta el siguiente submenú:

2.3.1.Estado del trámite

Para ubicar en qué estado en el que se encuentra un trámite específico basta con ingresar a esta sección y realizar los siguientes pasos:

- Ingresar el número de trámite ya que éste te requerido
- Presionar el botón buscar y visualizar la información



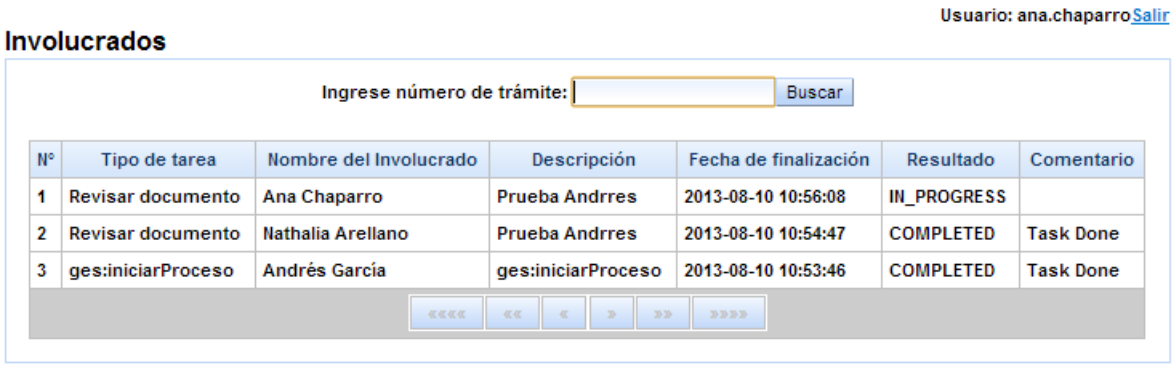
No dejar en blanco el campo número de trámite.

El área del resultado mostrara la información en el caso que exista.

2.3.2. Involucrados del trámite

Si se desea conocer la información de los usuarios que tienen que ver con cierto trámite se debe ingresar a la sección gestión de trámites involucrados y realizar los siguientes pasos:

- Ingresar el número de trámite en la sección indicada
- Presionar el botón buscar y visualizar la información



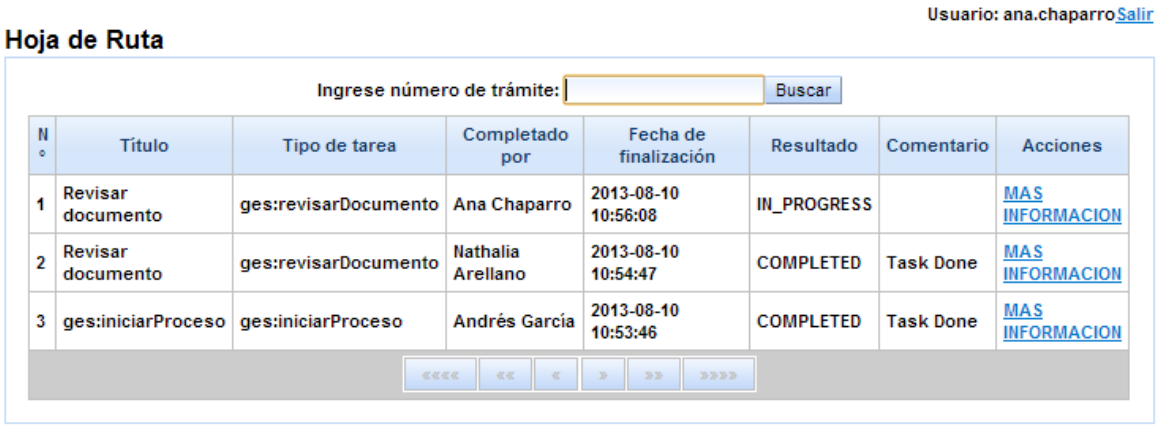
No dejar en blanco el campo número de trámite

El área del resultado mostrara la información en el caso que exista

2.3.3 Hoja de ruta

Si se desea conocer la información de hoja de ruta, se necesita ingresar a esta sección y realizar los siguientes pasos:

- Ingresar el número de trámite
- Dar clic en el botón buscar
- Visualizar la información deseada
- Para visualizar el detalle del trámite dar un clic en el link “MÁS INFORMACIÓN”.



No dejar en blanco el campo número de trámite

El área del resultado mostrara la información en el caso que exista

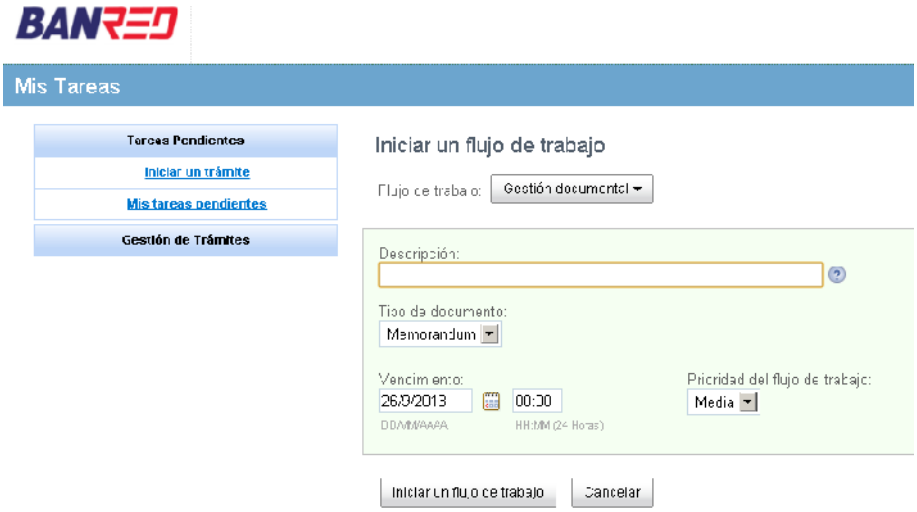
2.4. Ejecución de trámites

Dentro del portal se encuentra un conjunto de pantallas que permiten a los usuarios funcionarios realizar los trámites dentro de la institución.

2.4.1 Iniciar trámite

Dentro del portal, para iniciar un trámite se necesita ingresar al menú "Tareas pendientes" y seleccionar la opción "Iniciar un trámite". A continuación se presentará un formulario con los campos necesarios para iniciar un trámite en la institución, en el cual se necesita realizar los siguientes pasos:

- Seleccionar el trámite "Gestión documental" de la lista de trámites del sistema.
- Ingresar la descripción de qué se trata el trámite.
- Ingresar la fecha de caducidad del trámite.
- Ingresar la prioridad del trámite.
- Ingresar el tipo de documento que se generará dentro del trámite.



2.4.2 Tarea llenar metadatos en formulario web

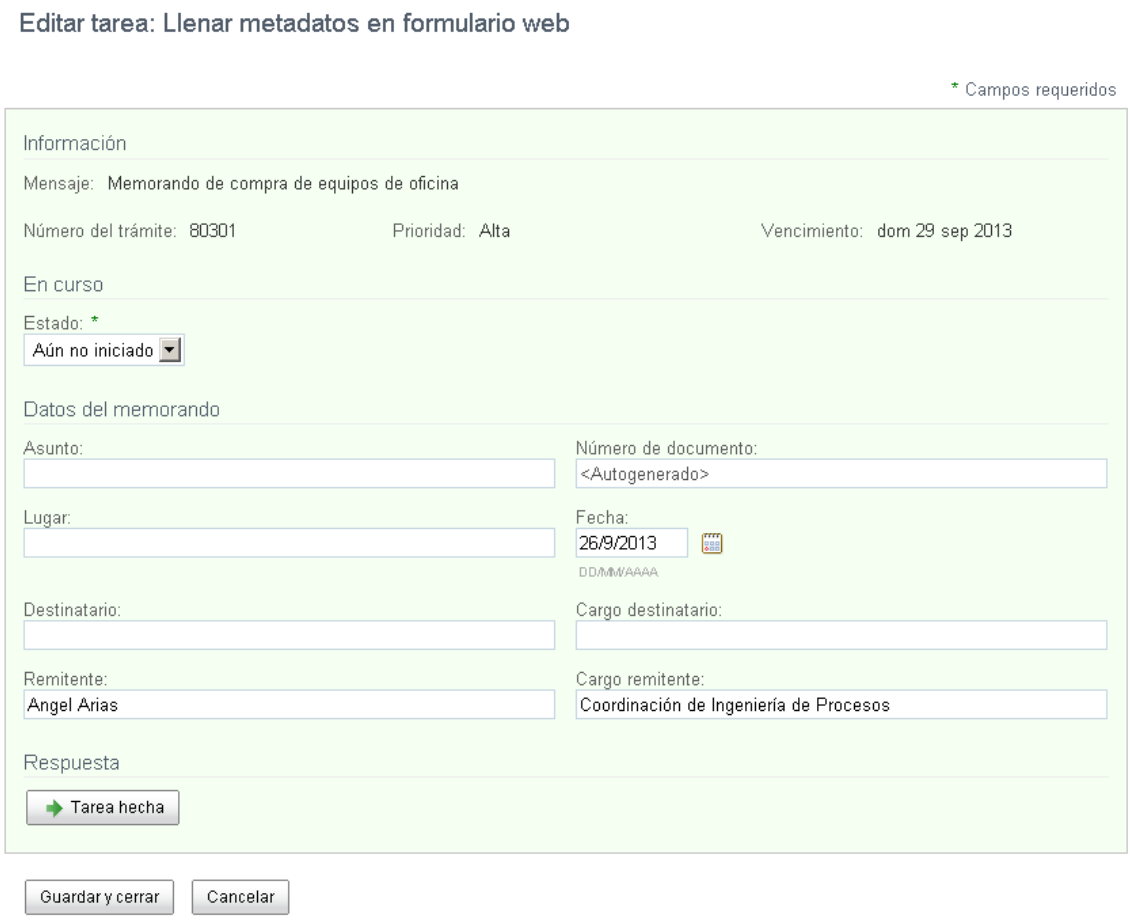
Dentro de este formulario se necesita ingresar la información que se desea reflejar sobre el documento que se va a crear en el trámite, es decir, este formulario genera un documento con la información recolectada.

Cabe recalcar que los campos presentados dependen del tipo de documento, sin embargo el proceso para crear un documento en el te trámite es igual para cada documento.

Para ingresar los METADATOS del documento a crear en este formulario se necesita realizar los siguientes pasos, para este ejemplo se ingresarán los datos de un documento de tipo memorando:

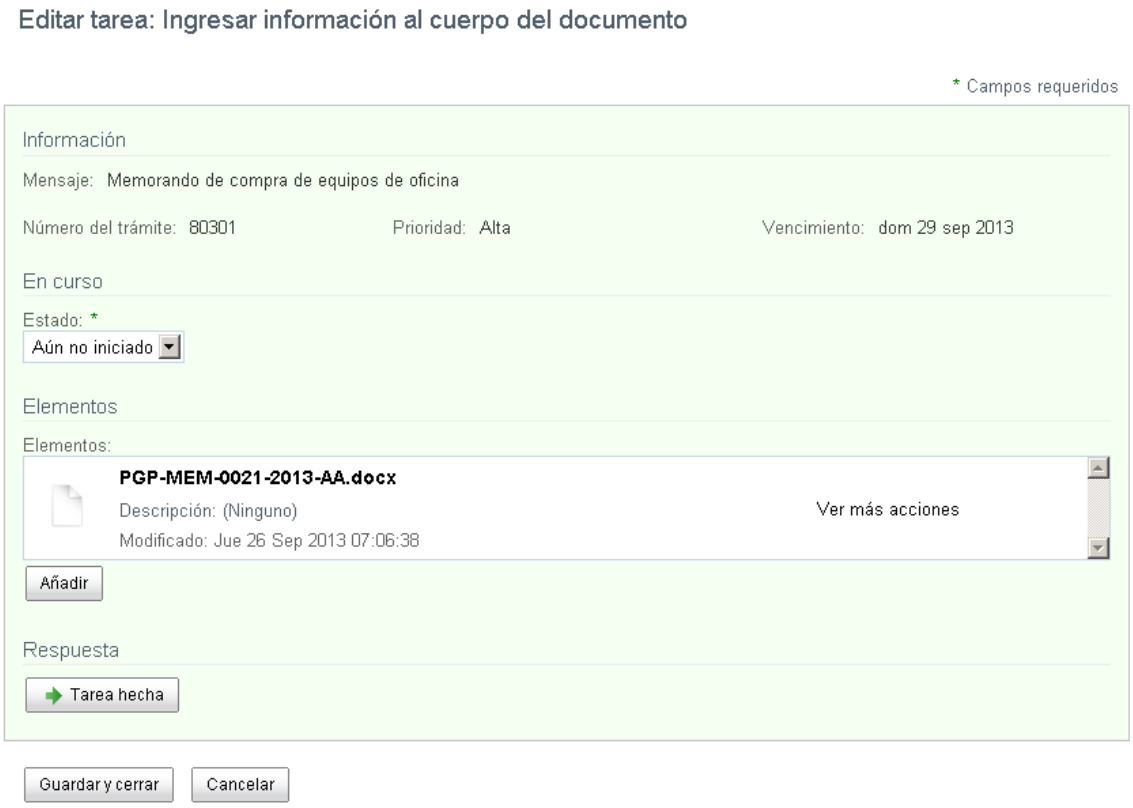
- Ingresar el asunto del documento.
- Ingresar el lugar de generación del documento.
- Ingresar la fecha del documento.
- Ingresar los datos del destinatario.
- Ingresar los datos del remitente.

Una vez ingresada esta información se debe dar clic en “Tarea hecha” para continuar con el trámite. Cabe recalcar que el número del documento se generará a partir de los datos del remitente.



2.4.3 Tarea ingresar información en el cuerpo del documento

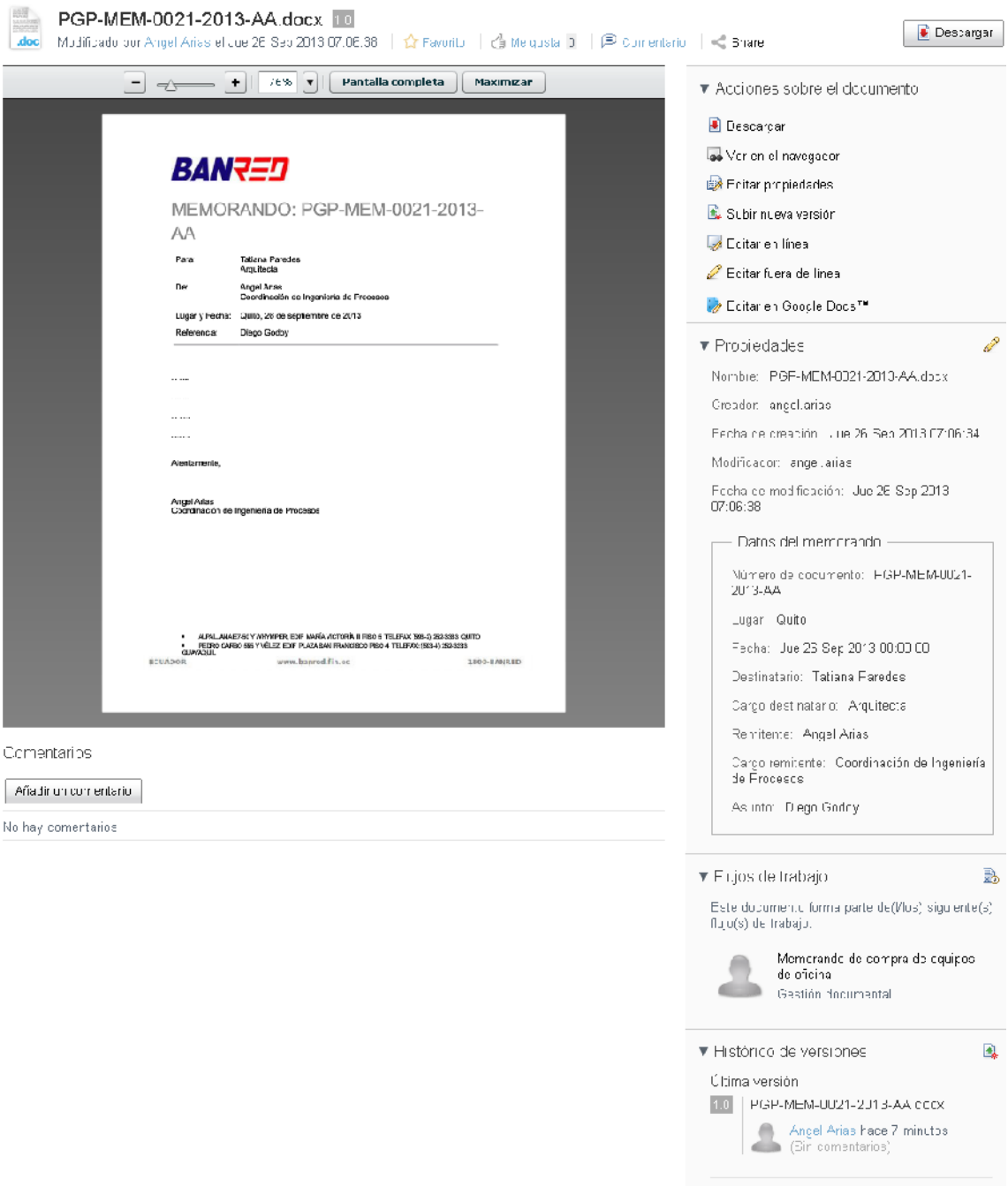
Una vez ingresada la información en la tarea anterior, el sistema procede a generar un documento dentro del trámite con dicha información. Dentro de esta tarea, el funcionario deberá ingresar el contenido necesario para complementar este documento. Como se presenta a continuación esta tarea contiene los datos del trámite, el documento generado y los controles para continuar con el trámite.



Para ingresar contenido dentro del documento se necesita seleccionar el link "Ver más acciones", el cual direccionará a la pantalla "Detalles del documento" que se explicará más adelante. Una vez realizado los cambios sobre el documento se necesita dar un clic en el botón "Tarea hecha" para continuar con el trámite.

2.4.4 Detalles del documento

Dentro de esta pantalla se presenta toda la información relacionada con el documento generado en el trámite. Esta pantalla contiene una vista previa del documento generado, las acciones disponibles del documento, la información de METADATOS, el histórico de versiones, entre otros. A continuación se presenta el contenido de esta pantalla:

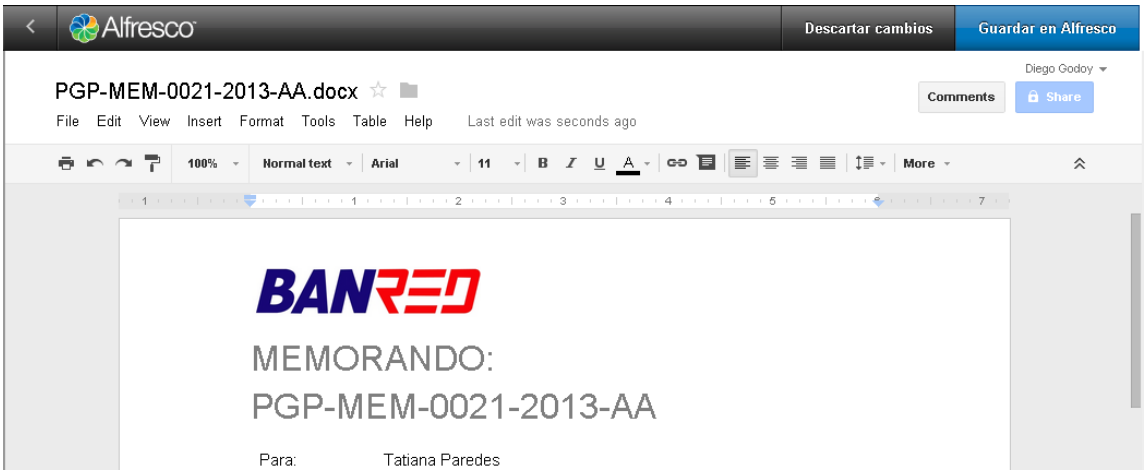


El principal objetivo de esta pantalla es permitir realizar la edición del contenido, para lo cual el usuario necesita dar un clic sobre el botón “Editar en Google Docs”, dicha acción permite presentar la interfaz de Google Docs para edición de documentos.

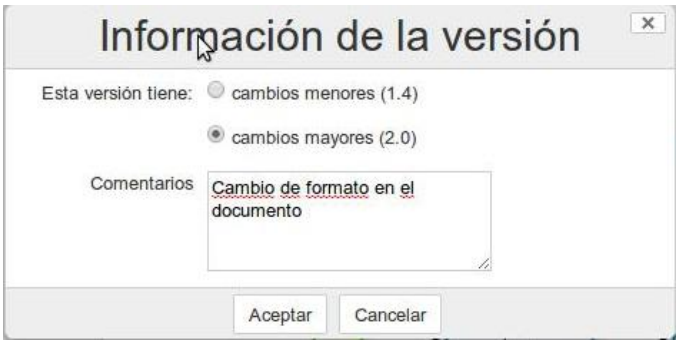
Para acceder a esta funcionalidad es necesario que el funcionario tenga una cuenta en Google.

2.4.5 Formulario editar documento con Google Docs

Al seleccionar la acción “Editar con Google Docs” en el punto anterior, se presentará el entorno de edición de documentos ofimáticos de Google dentro del portal, donde el usuario podrá realizar los cambios necesarios al contenido del documento.



Para guardar los cambios sobre las modificaciones realizadas al documento se necesita dar un clic en el botón “Guardar en ALFRESCO” e ingresar los detalles del cambio del documento en la siguiente pantalla.

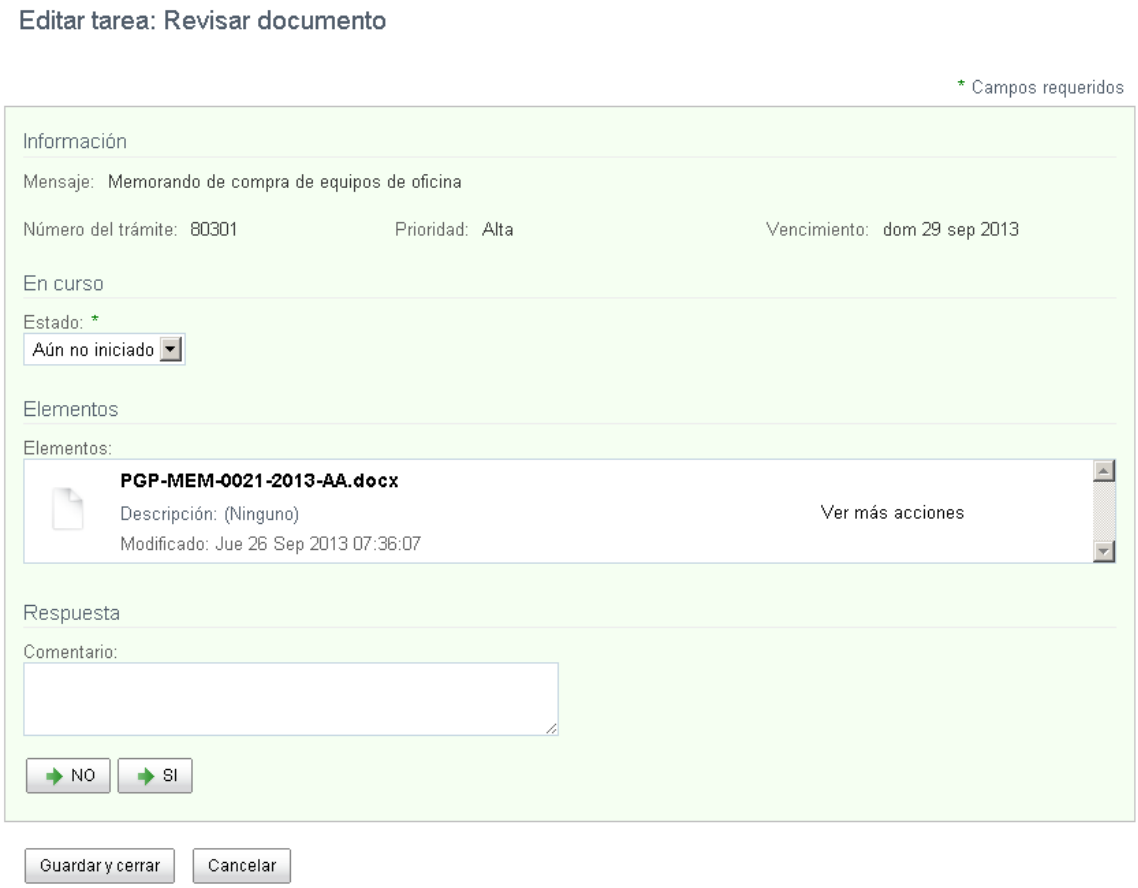


2.4.6 Tarea revisar documento

Dentro de esta tarea, el usuario asignado a revisar el documento podrá visualizar los datos generales del trámite, el documento adjunto al trámite y los controles para continuar con el trámite.

Para realizar la revisión del documento, se necesita seleccionar el link “Ver más acciones” sobre el área de documentos del trámite, donde el sistema direcciona a la

página de detalles del documento descrita anteriormente para que el revisor pueda ver la vista previa del documento.



Una vez revisado el documento se necesita regresar a la pantalla del trámite y seleccionar el botón “Si” para aceptar el contenido del documento o “No” para rechazar el contenido y el trámite se direccione a la tarea de corrección.

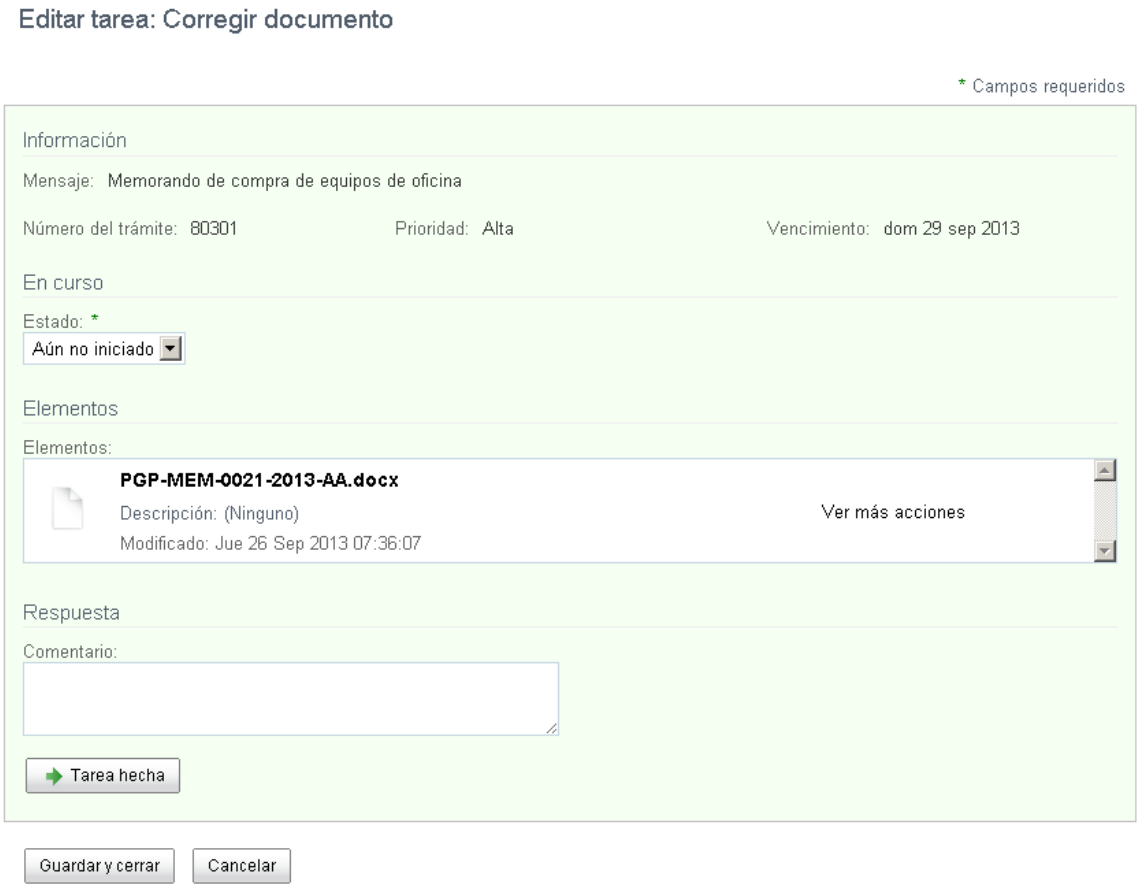
2.4.7 Formulario corregir documento

Si el documento no ha sido aprobado por el revisor, el sistema procederá a regresar el documento al usuario pertinente, en este caso el usuario necesita realizar los cambios solicitados por el revisor para poder continuar con el trámite.

Para realizar la modificación del documento se necesitan realizar los siguientes pasos en la pantalla actual:

- Seleccionar el link “Ver más acciones”.
- En detalles del documento seleccionar la acción “Editar con Google Docs”.
- Realizar los cambios solicitados.

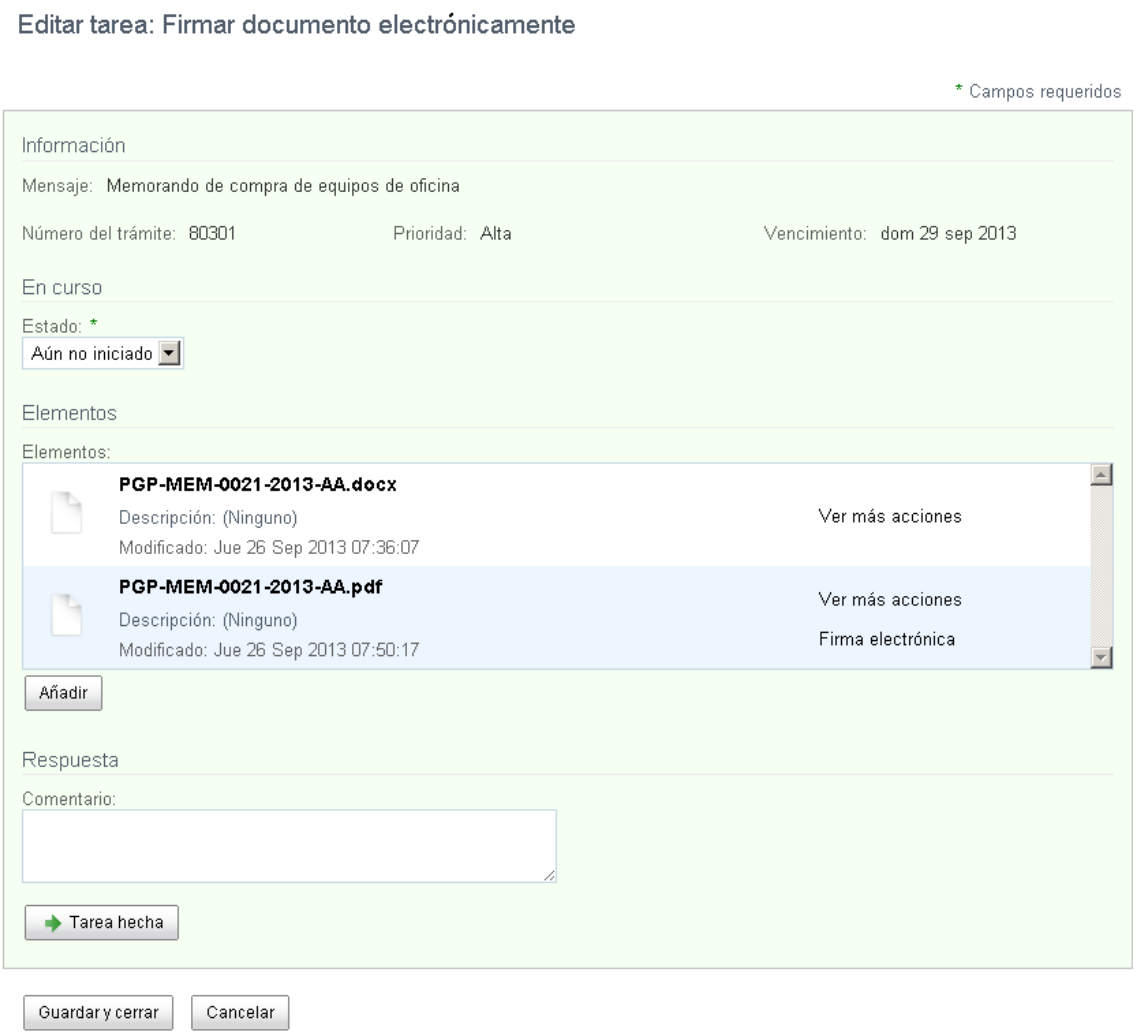
- Regresar a la pantalla del trámite y seleccionar el botón “Tarea hecha”.



2.4.8 Tarea firmar documento electrónicamente

Si los usuarios revisores aceptan el contenido del documento, el trámite procederá a continuar con la tarea de firma electrónica. Esta tarea presenta la información general del trámite, el documento del trámite que se aplicará la firma electrónica y los controles para continuar con el trámite.

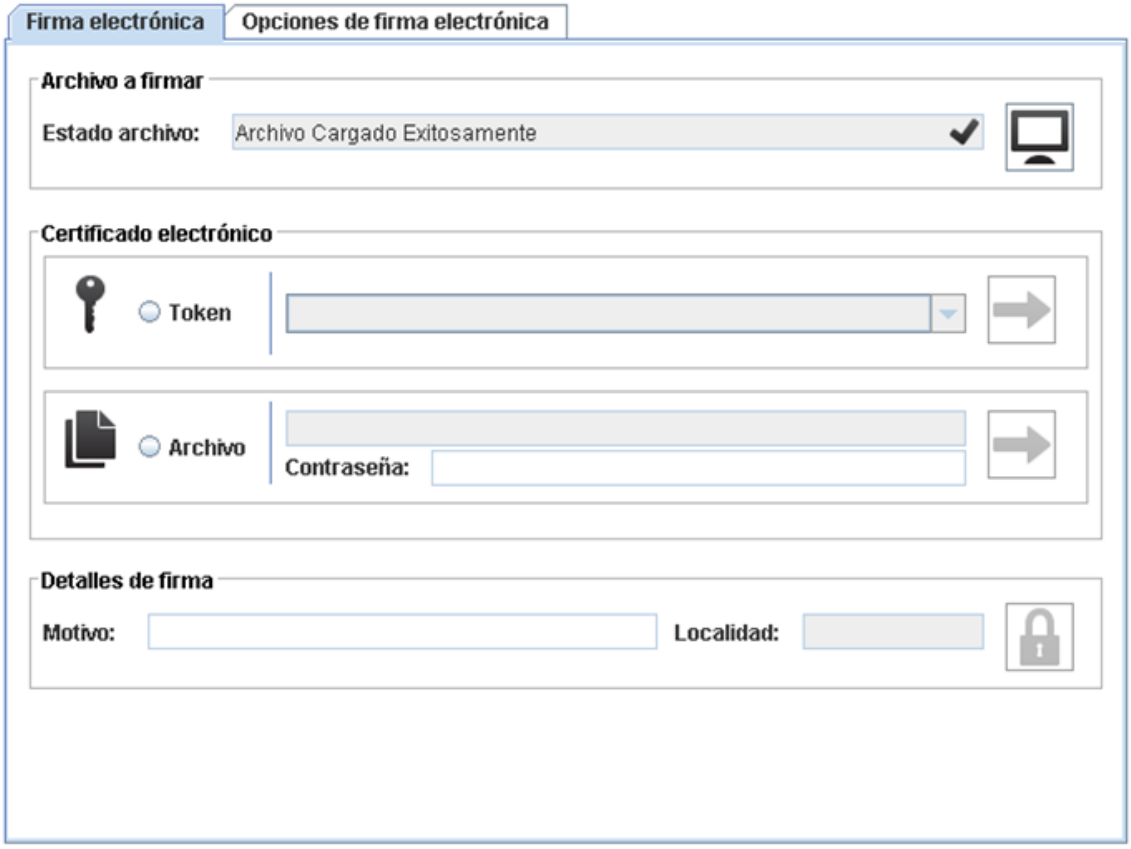
Para firmar el documento se necesita seleccionar la acción “Firma electrónica” sobre el documento del trámite, donde se presentará el aplicativo de firma electrónica detallado más adelante.



Una vez firmado el documento se necesita dar un clic en el botón “Tarea hecha”.

2.4.9 Interfaz de firma electrónica

Al seleccionar la opción “Firma electrónica” en el punto anterior, el sistema presentará el siguiente formulario:



Para firmar electrónicamente un documento se necesita realizar los siguientes pasos:

- Seleccionar la opción “Token”.
- Dar un clic en el botón “→”.
- Ingresar el nombre del Token.
- Ingresar la clave privada.
- Ingresar el motivo de la firma y la localidad.
- Seleccionar el botón “candado” para realizar la firma electrónica.

2.4.10 Parámetros de firma electrónica

Como funcionalidad complementaria a la firma electrónica, se puede especificar los parámetros de posición y visibilidad de la firma dentro del documento. Para modificar los parámetros por defecto de la firma electrónica se necesita realizar los siguientes pasos:

- Seleccionar la pestaña “Opciones de firma”.

- Seleccionar el número de página donde se visualizará la firma electrónica dentro de “Mostrar en página”.
- Seleccionar si la firma será visible en el documento dentro de “Visualización”.
- Seleccionar si se desea pre visualizar el archivo firmado dentro de “Visualización”.
- Seleccionar la posición de la firma electrónica dentro de “Ubicación en página”.
- Seleccionar el botón “guardar” para aplicar los cambios.