UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:

INGENIERIA DE SISTEMAS

Trabajo de titulación previa a la obtención del título de:

INGENIEROS DE SISTEMAS

TEMA:

DESARROLLO DEL PROYECTO CIRCUITOS ALTERNATIVOS DE COMERCIALIZACIÓN – CIALCO MÓDULO DE ADMINISTRACIÓN DEL PORTAL Y BITÁCORA DE UTILIZACIÓN

AUTORES:

MIGUEL ANGEL MACHASILLA NICOLALDE

FRANCISCO JAVIER BONILLA LARCO

TUTOR:

WASHINGTON RAÚL PADILLA ARIAS

Quito, septiembre de 2016

CESIÓN DE DERECHOS DE AUTOR

Nosotros Francisco Javier Bonilla Larco, con documento de identificación Nº 1718388893 y Miguel Angel Machasilla Nicolalde, con documento de identificación 1716598634, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de grado intitulado: DESARROLLO DEL PROYECTO CIRCUITO ALTERNATIVO DE COMERCIALIZACIÓN – CIALCO MÓDULO DE ADMINISTRACIÓN DEL PORTAL Y BITÁCORA DE UTILIZACIÓN, mismo que ha sido desarrollado para optar por el título de: INGENIEROS EN SISTEMAS, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer los derechos cedidos anteriormente.

En la aplicación a lo determinado por la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada.

En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

Francisco Javier Bonilla Larco
CI: 171838889-3

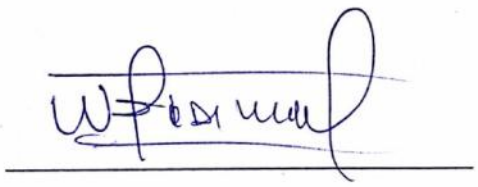
Miguel Ángel Machasilla Nicolalde
CI: 171659863-4

Quito, septiembre de 2016

DECLARACIÓN DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el proyecto, DESARROLLO DEL PROYECTO CIRCUITO ALTERNATIVO DE COMERCIALIZACIÓN – CIALCO MÓDULO DE ADMINISTRACIÓN DEL PORTAL Y BITÁCORA DE UTILIZACIÓN realizado por Francisco Javier Bonilla Larco y Miguel Ángel Machasilla Nicolalde, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, Septiembre del 2016



Washington Raúl Padilla Arias

C.I.: 1707492888

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE ANEXOS

Resumen

Un Circuito Alternativo de Comercialización (CIALCO), es una forma de comercio basado en la venta directa del agricultor hacia el consumidor, minimizando así la intervención de terceras personas que encarecen el producto. El Ministerio de Agricultura, Ganadería, Acuacultura y Pesca, MAGAP, hasta la fecha recauda información sobre CIALCO mediante formularios impresos, lo que genera pérdida de información valiosa para un mejor manejo de las bases de los agricultores ecuatorianos.

Dado este hecho, que es de importancia para mejorar la productividad ecuatoriana en materia de agricultura, se ha trabajado en el análisis, diseño, desarrollo e implementación del módulo de Administración y Bitácora de Utilización para el Sistema de Información Circuito Alternativo de Comercialización - CIALCO, el mismo que permite tener un control de acceso mediante la autentificación de usuarios, menús dinámicos, y privilegios de manipulación de datos CRUD (por sus siglas en inglés: CREATE, READ, UPDATE, DELETE), a la información almacenada en el sistema. Además, este proyecto registra cada una de las transacciones realizadas dentro del sistema CIALCO, proporcionando información detallada que permite obtener informes estadísticos dentro del sistema.

Para un mejor control de las transacciones realizadas dentro del sistema se crearon campos de auditoría los cuales contienen la información detallada de usuario de creación de registros, fecha de creación del registro, usuario de modificación del registro, fecha de modificación del registro, identificador zonal a la cual pertenece el usuario y estado del registro (Activo / Inactivo, eliminación lógica del registro).

Abstract

An Circuito Alternativo de Comercialización (CIALCO), is a form of trade, based on the direct sale of the farmer or producer to the consumer, minimizing the intervention of third parties that raise the price of the product. El Ministerio de Agricultura, Ganadería, Acuacultura y Pesca, MAGAP, till the date CIALCO collects information on paper forms, resulting in loss of valuable information for better management of the bases of Ecuadorian farmers.

Given this fact, that it is important to improve Ecuador's productivity in agriculture, has worked on the analysis, design, development, and implementation of Management module and Record Utilization for Sistema de Información Circuito Alternativo de Comercialización (CIALCO), the same that allows access control through user authentication, dynamic menus, and privileges CRUD handling data (for its acronym in English: CREATE, READ, UPDATE, DELETE), the information stored in the system. In addition, this project registers each transaction within the system CIALCO, providing detailed information that allows for statistical reports within the system.

For better control of the performed transactions within the system fields were created audit which contain the detailed user information record creation, creation date registry, user registry modification, modification date of registration, zonal identifier created which it belongs to the user and registration status (active / inactive, logic deregistration).

Introducción

Actualmente existe un creciente uso de internet, el cual se ha convertido en uno de los medios de difusión más populares a nivel mundial. A provechando este hecho, el Departamento de Coordinación de Redes Comerciales del Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP), se ha interesado en hacer partícipe a la comunidad de los diferentes Circuitos Alternativos de Comercialización (CIALCO) con el fin de facilitar y promover el acceso a mercados de agricultores campesinos, para la provisión sostenible de alimentos sanos y el desarrollo digno de las familias rurales. Debido a lo anteriormente expuesto, se ha puesto en marcha el desarrollo del Sistema de Información de Circuitos Alternativos (CIALCOs), aplicando tecnología Web.

El módulo de Administración y Bitácora de Utilización, se enfoca en la seguridad, auditoría y seguimiento continuo de las diferentes transacciones que realizan los usuarios internos y externos dentro del Sistema de Información CIALCO, mediante el control de accesos y recursos, evitando el uso indebido de la información almacenada en el sistema.

CAPÍTULO I

ESTADO DEL ARTE

1.1 Antecedentes

La Coordinación General de Redes Comerciales nace de la consolidación del Proyecto de Circuitos Alternativos de Comercialización (CIALCO), con la misión de articular la oferta de políticas del gobierno central a través del impulso de redes comerciales locales asociativas de la agricultura familiar campesina para la provisión sostenible de alimentos sanos y suficientes para la alimentación nacional y el desarrollo digno y sostenible de las familias rurales. (Ministerio de Agricultura, Ganadería, Acuacultura y Pesca, 2016).

El manejo de la información de los diferentes CIALCOS, se lo ha realizado de forma manual, mediante formularios físicos que se los llena en cada una de las localidades en las que estos se realizan, posteriormente escaneados y pasados a hojas de cálculo en EXCEL. Por este motivo se presentan varias problemáticas, en una primera instancia el llenado de los formularios no cuenta con un estándar, éste depende del criterio del usuario de campo dando paso a una mala calidad de la información como datos incompletos, ubicaciones y/o direcciones mal definidas, duplicidad de información, dificultando la generación de algún tipo de estadística que ayude en la toma de decisiones.

Así nace el convenio entre el Ministerio de Agricultura, Ganadería, acuacultura y Pesca (MAGAP) y la Facultad de Sistemas de la Universidad Politécnica Salesiana para la creación del Sistema de Información de Circuitos Alternativos de Comercialización - CIALCO, agilitando los tiempos en los diferentes procesos y facilitando la toma de decisiones del Ministerio para el mejoramiento continuo de los Circuitos Alternativos de Comercialización (CIALCO).

1.2 Justificación

Es de interés de la oficina de Redes Comerciales del Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP), organizar, planificar, desarrollar, implementar herramientas que apoyen la definición de políticas gubernamentales que estimulen el desarrollo de los Circuitos Alternativos de Comercialización (CIALCO) de productos agropecuarios fomentando la relación directa entre productor y consumidor eliminando la intermediación a través de: ferias, fincas, canasta, canasta comunal, compra pública y exportaciones de producción agro-familiar.

La oficina de Redes de Comercialización del Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP), cuenta con información del movimiento de los productos que se comercializan en cada CIALCO en una hoja de cálculo, que no cuenta con las facilidades para mantener un sistema estadístico, además la información registrada es de los años 2010 y 2014, lo que dificulta mantener una base de datos consolidada para generar un sistema de apoyo en la toma de decisiones.

Por estos motivos y viendo la necesidad del Ministerio para fortalecer el desarrollo digno de vida de las comunidades rurales, se crea el Sistema de Información de Circuitos Alternativos de Comercialización (CIALCO) con la finalidad de tener un registro adecuado y ordenado de la información que se obtiene de los diferentes CIALCOs, productores y organizaciones, que ayude a generar nuevas estrategias y políticas gubernamentales para fomentar el crecimiento continuo de la producción agro-familiar.

1.3 Objetivo General

Controlar los procesos de parametrización del sistema y determinar los niveles de seguridad y accesibilidad de los usuarios a cada uno de los módulos.

1.4 Objetivos Específicos

- Generar perfiles de usuario.
- Mejorar la seguridad y accesibilidad de acuerdo al perfil de usuario.
- Monitorear continuamente las actividades que realizó el usuario en el portal.
- Generar el consumo del servicio WEB proporcionado por el Departamento de Redes Comerciales del Ministerio de Agricultura, Ganadería, Acuacultura, pesca (MAGAP), que devuelve la información personal mediante el número de identificación.
- Generar datos estadísticos de usuarios que han utilizado el portal.

1.5 Marco metodológico

Dentro de la Ingeniería de Software, existen varias metodologías usadas para estructurar, planificar y controlar el proceso de desarrollo de un sistema informático. Entre ellas podríamos nombrar algunas como:

Desarrollo Rápido de aplicaciones o RAD (por sus siglas en ingles Rapid Application Development).

Scrum.

Proceso Racional Unificado o RUP (por sus siglas en inglés Rational Unified Process.

Programación Extrema o XP (eXtreme Programming).

Para el desarrollo del Módulo de Administración del Portal y Administración de Bitácora de Utilización se utilizó la metodología XP (eXtreme Programming).

1.5.1 Programación Extrema o XP (eXtreme Programming)

Es una metodología de desarrollo de la Ingeniería de Software formulada por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999). (WIKIPEDIA, Programación extrema, 2015). Es una metodología ágil que tienen como principal objetivo aumentar la productividad a la hora de desarrollar un sistema. Tiene la capacidad de adaptarse rápidamente a los cambios de requerimientos en cualquier fase de desarrollo del proyecto.

Características:

Desarrollo iterativo e incremental.

Programación en parejas, dos personas trabajando en un mismo ordenador teniendo como objetivo una mejor calidad de código fuente, mayor productividad.

Frecuente integración con el cliente.

Corrección de errores antes de añadir nuevas funcionalidades.

Hacer entregas frecuentes.

Propiedad de código compartido, es decir, que todo el personal este en la capacidad de corregirlo.

Simplicidad en el código.

1.5.1.1 Fases de la metodología XP

1.5.1.1.1 Planificación

En esta fase el cliente describe concretamente las funcionalidades y/o requerimientos que el sistema debe realizar, posteriormente y en conjunto con los programadores se establecen los tiempos y prioridades de desarrollo.

Historias de usuario: Consiste en la descripción de los pasos o actividades que realiza un usuario para llevar a cabo un determinado proceso. Estas historias de usuarios se las utiliza en la fase de desarrollo y también en las de pruebas.

Plan de publicación: Se establecen los tiempos y prioridades de desarrollo para la implementación de cada una de las historias de usuario.

Iteraciones: son presentaciones que se realizan según el plan de publicación, con el objetivo de revisar y probar las historias de usuarios que se van desarrollando.

Velocidad de proyecto: hace referencia al tiempo de desarrollo del proyecto, se lo puede estimar basándonos en el número de historias de usuario que se desarrollan en una iteración.

Programación en parejas: la metodología XP aconseja que la programación se la realice en parejas, es decir, dos programadores en un mismo ordenador con la finalidad de incrementar la productividad y la calidad del software desarrollado.

1.5.1.1.2 Diseño

Diseños simples: la metodología XP recomienda hacer diseños simples y sencillos, que sean entendibles y fáciles de implementar.

Glosario de términos: usar un glosario de términos y una correcta especificación de nombre de métodos y clases ayudará a comprender el diseño y facilitara su posterior implementación.

Refactorizar: mejorar y modificar la estructura y codificación de código sin alterar su funcionalidad.

Tarjetas C.R.C. (Clase – Responsabilidad – Colaboración): son tarjetas que representan a los objetos (Clases) que intervienen en el proyecto. Estas tarjetas se

dividen en tres secciones, el nombre de la clase, sus responsabilidades (Atributos y métodos) y sus colaboradores (Clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades). Ver Figura 1.

Tarjeta C.R.C

Figura 1. Secciones que conforman una tarjeta CRC
Fuente: (Modelo CRC, 2004)

1.5.1.1.3 Codificación

Es la fase en donde se codifica todo el diseño, mediante estándares de programación que mantienen el código consistente y facilitan su compresión. Entre sus principales elementos que menciona XP referentes a la codificación están:

El cliente siempre debe estar presente en cada una de las fases de programación.

Unidades de prueba o test.

Toda la producción de código debe ser hecha en pareja.

No trabajar horas extras, un programador agotado no genera un código de calidad.

1.5.1.1.4 Pruebas

Es uno de los pilares fundamentales de la metodología XP, es la fase donde se pone a prueba todo lo desarrollado e implementado en el proyecto.

Unidades de test o pruebas

Implementación**:** el código será implantado en la versión a publicar.

Protección contra fallos: dar solución inmediata en caso de tener fallos.

Pruebas de aceptación: Es la evaluación del cliente.

Fases de la Metodología XP

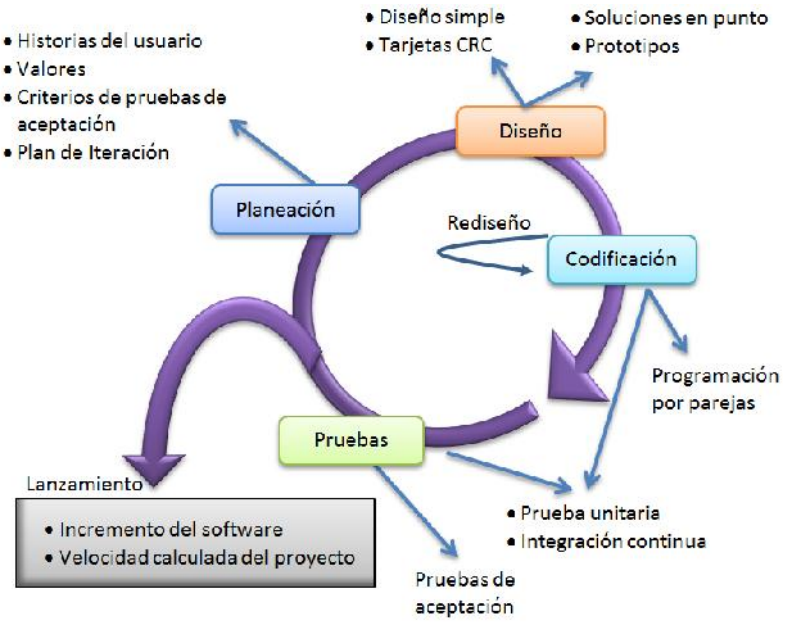


Figura 2. Fases de la Metodología XP
Fuente: (ISSUU, 2010)

CAPÍTULO II

MARCO REFERENCIAL Y TEÓRICO

2.1 Marco referencial o institucional

El objetivo del Departamento de Redes Comerciales de Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP) es mejorar y ordenar la información de los diferentes CIALCOs, para facilitar la toma de decisiones que promuevan la venta de productos de los agricultores campesinos. Para ello se crea un Sistema de Información de Circuitos Alternativos de Comercialización CIALCO. Desarrollado con tecnología WEB y con una administración basada en la autentificación de usuario que nos permite tener un control de acceso a las diferentes opciones del sistema.

2.1.1 Sistema de información

Un Sistema de Información (SI) es un conjunto de elementos orientados al tratamiento y administración de datos e información. Dichos elementos forman parte de alguna de las siguientes categorías: (WIKIPEDIA, Sistema de información, 2016).

Persona.

Actividades o técnicas de trabajo.

Datos.

Ciborra (2002) define a los Sistemas de Información como el estudio que trata la inserción y el uso de la tecnología en las organizaciones, instituciones, y la sociedad en general. (Fundación Wikimedia, Inc, 2016).

Si bien es cierto que un Sistema de Información puede mejorar y ordenar el manejo de los datos, también pueden existir factores externos o internos que puedan hacer un uso indebido de los mismos.

Con estos antecedentes, es fundamental implementar dentro de un Sistema de Información un módulo de Administración y Bitácora de su uso que permita el control integral de las transacciones que se pueden realizar con los datos que contiene el sistema, permitiendo una buena calidad de información, veraz y oportuna.

2.1.2 Seguridad informática

La seguridad informática establece normas que minimicen los riesgos de la información, entre ella podemos tener; restricciones de accesos, perfiles de usuarios, protocolos, etc. Que permitan un buen nivel de seguridad y de igual forma maximizar el desempeño de las organizaciones en general.

La seguridad informática es orientada a la protección de activos informáticos tales como:

Los usuarios: son las personas que utilizan la estructura tecnológica y gestiona la información.

La información: siendo el principal activo, reside en la infraestructura computacional y es utilizada por los usuarios.

2.1.3 Amenazas

Es todo elemento o acción capaz de atentar contra la seguridad de la información. Entre los cuales nombraremos:

Personas

Personal de la organización.

Ex – empleados.

Hackers.

Intrusos remunerados.

Amenazas por el efecto

Robo de la información.

Destrucción de la información.

Suplantación de identidades.

Publicación de datos personales o confidenciales.

Cambios de información.

Marco teórico

Se describe los conceptos y características de las herramientas que se utilizaron para el desarrollo del Módulo de Administración y Bitácora de Utilización del Sistema de Información CIALCO

2.1.4 Herramientas

2.1.4.1 Motor de base de datos PostgreSQL

Es un sistema de gestión de base de datos objeto – relación de código abierto que utiliza un modelo cliente/servidor, que garantiza la estabilidad del sistema.

Características:

Soporta varios tipos de datos.

Es fácil de administrar.

Es multiplataforma.

Alta concurrencia.

Soporta grandes cantidades de datos.

Es estable.

2.1.4.2 Java

Es un lenguaje de programación multiplataforma orientado a objetos que permite crear aplicaciones tipo cliente, cliente – servidor, y aplicaciones WEB.

Características:

Sistema orientado a objetos.

Es portable.

Robusto.

Dinámico

Maneja código abierto.

2.1.4.3 Java Enterprise Edition (JEE)

Es una plataforma de programación que permite el desarrollo de aplicaciones empresariales, JEE maneja la arquitectura de N capas. Vea Figura 3.

Capa cliente: puede estar constituida por aplicaciones de escritorio o navegadores web, es la capa a la que accede directamente.

Capa WEB: se encuentra en el servidor WEB y contiene la lógica de presentación que se utiliza para recibir las peticiones de la capa de cliente y generar las respuestas apropiadas a las mismas.

Capa de negocio: es donde se procesan todas las funcionalidades del sistema.

Capa de datos: son los servicios que proporcionan los datos persistentes utilizados por la lógica de negocio.

La plataforma JEE permite combinar varias tecnologías para la creación de aplicación o sistemas seguros y eficientes, entre ellas podemos citar, Frameworks que facilitan el desarrollo como JSF (Java Server Face), EJB (Enterprise Java Beans), HTML, Servidores Web, APIs de persistencia para la conexión a base de datos, Servlets y la implementación de la arquitectura MVC (Modelo Vista Controlador) que permite diferenciar y separar las tres capas anteriormente definidas.

2.1.4.4 Hibernate

Hibernate es una herramienta de mapeo objeto/relacional que facilita la iteración entre el lenguaje de programación en este caso JAVA y una base de datos relacional. El término de mapeo hace referencia a la técnica para la representación de datos desde un modelo de objeto a un modelo de datos relacional con un esquema SQL.

Hibernate no solo se ocupa del mapeo desde las clases Java a las tablas de la base de datos, sino que también facilita los procesos de consultas y recuperación de datos.

Características:

Maneja su propio lenguaje de consultas llamado HQL (Hibernate Query Language).

Facilita la programación.

Agilita los tiempos de desarrollo.

Es multiplataforma.

2.1.4.5 Framework

Son infraestructuras que encapsulan conceptos, prácticas y criterios que facilitan el desarrollo de software.

Características:

Compatibilidad de lenguajes.

Permite la reutilización de código.

Fácil de utilizar.

2.1.4.6 Java Persistence API (JPA)

Es un modelo de persistencia que permite el mapeo de una base de datos en Java (sistema orientado a objetos).

Características:

No requiere de configuraciones avanzadas.

Es versátil ya que permite el remplazo de librerías sin realizar cambios de código.

Maneja su propio lenguaje de consulta.

2.1.4.7 Arquitectura JPA

Para explicar la arquitectura de funcionamiento del Api JPA explicaremos algunos conceptos importantes, para ello no apoyaremos en el siguiente diagrama.

Arquitectura JPA

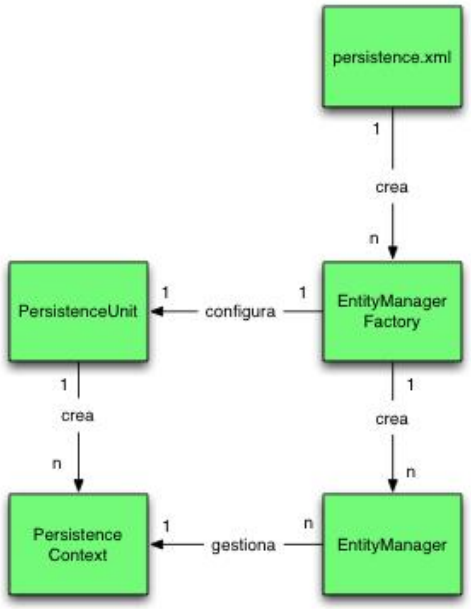


Figura 3. Arquitectura JPA
Fuente: (Arquitectura Java, 2016)

Persistence.xml: es el fichero de configuración que permite la conexión a la base de datos y donde se definen las entidades que se van a gestionar.

EntityManagerFactory: permite la creación y gestión de las múltiples instancias EntityManager. Esto se lo conoce como Persistence Unit.

EntityManager: gestiona la persistencia de los objetos, funciona como una instancia de consulta conocido como PersistenceContext.

2.2.1.7.1 Relaciones múltiples de la entidad

Existen 4 tipos de relaciones de entidades que se pueden utilizar en JPA.

Uno a uno: cada entidad se relaciona con una sola instancia de otra entidad. Para ello se utiliza la anotación de java @OneToOne.

Uno a muchos: una entidad puede tener relación con varias instancias de otra entidad. Su anotación en Java es @OneToMAny.

Muchos a uno: múltiples instancias de una entidad pueden relacionarse con una instancia de otra entidad. Su anotación en Java es @MAnyToOne.

Muchos a muchos: varias instancias de una entidad pueden relacionarse con varias instancias de otra entidad. Su anotación en Java es @ManyToMany.

2.1.4.8 Java Server Face (JSF)

Java Server Face (JSF) es un framework de Java que utiliza un patrón de diseño Modelo Vista Controlador (MVC), permite que el desarrollo de entornos WEB de una manera dinámica, permitiendo que él programador genere una división física y lógica del código.

Características:

Permite la reutilización de código.

Reduce tiempos de desarrollo.

Es compatible con otras tecnologías como HTML y java Script.

Forma parte JEE.

2.1.4.9 Primefaces

Es una librería de componentes para JSF, entre los cuales se pueden nombrar; editor de HTML, autocompletar, cartas, graficas, paneles, etc. Soporta Ajax, es compatible con otras librerías de contenido como JBOSS.

2.1.4.10 Servidor JBOSS

Es un servidor de aplicaciones basado en JEE, puede ser utilizado en cualquier sistema operativo y ofrece servicios de persistencia de objeto/relación Hibernate, aumenta el rendimiento de aplicaciones JAVA, maneja una licencia de código

abierto (GPL/LGPL), orientado a arquitecturas de servicios, Middleware para cualquier objeto de Java.

2.1.4.11 Spring

Spring es un framework de Java de código abierto. Ayuda a crear aplicaciones de alta calidad, más rápidamente. Spring proporciona un modelo de programación coherente y consistente.

El Spring Framework proporciona un modelo de programación y configuración completa para las aplicaciones empresariales modernas basadas en Java. Un elemento clave de Spring es el apoyo infraestructural a nivel de aplicación: Spring se centra en la "fontanería" de las aplicaciones empresariales para que los equipos pueden centrarse en la lógica de negocios a nivel de aplicación, sin ataduras innecesarias a los entornos de implementación específicos. (springla, 2015).

2.1.4.12 UML (Lenguaje Unificado de Modelado)

Es un estándar adoptado a nivel internacional para la creación de esquemas, diagramas y documentación relativa a los desarrollos de software. Incluye aspectos conceptuales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos.

Diagrama de caso de uso: es la representación de como un actor opera o interactúa con el sistema. Figura 4.

Diagrama de Caso de Uso

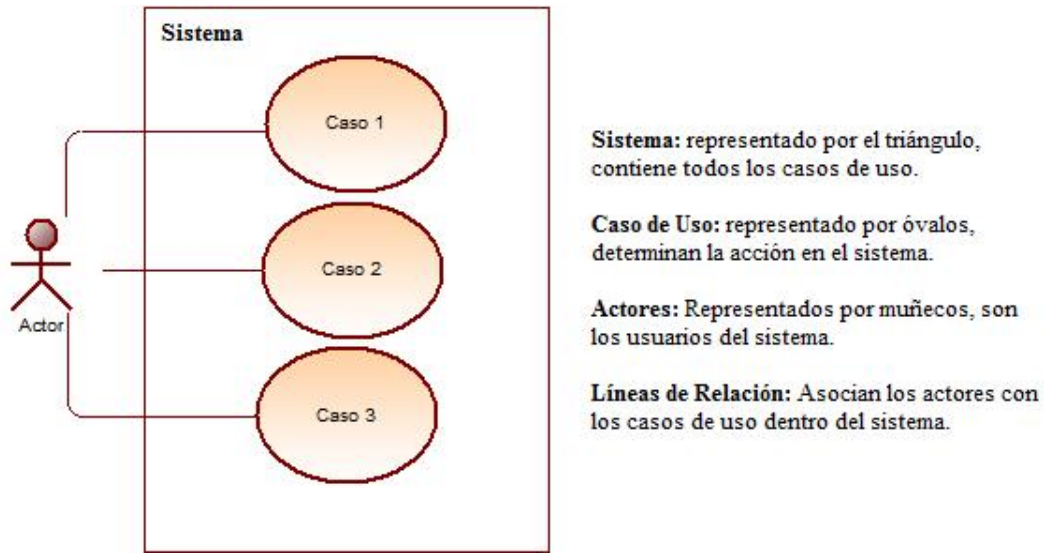


Figura 4. Arquitectura de Caso de Uso
Fuente: Francisco Bonilla y Miguel Machasilla

Diagrama de flujo: representa el flujo de trabajo paso a paso de los procesos que debe realizar el sistema software.

Diagrama de Flujo

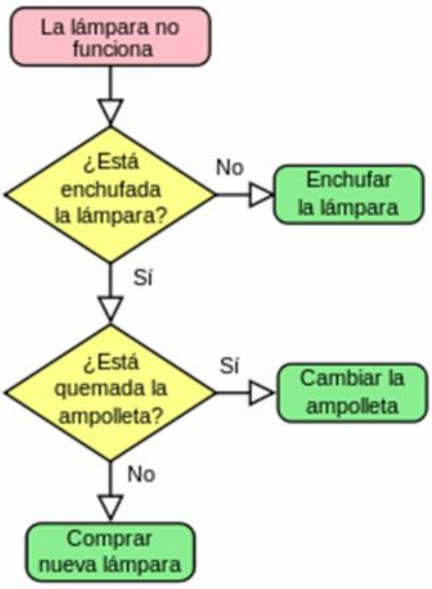


Figura 5. Diagrama de flujo sencillo con los pasos a seguir si una lámpara no funciona.
Fuente: (Wikimedia Commons, 2015)

Diagrama de Clases: describe la estructura de un sistema mediante clases que contendrán atributos, métodos y sus relaciones con otras clases.

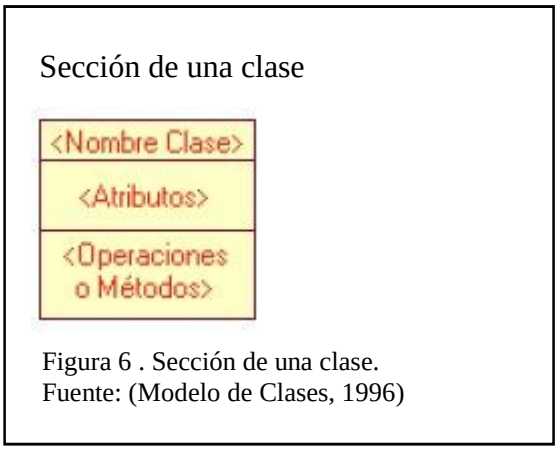


Figura 6 . Sección de una clase.
Fuente: (Modelo de Clases, 1996)

Para la creación del sistema, se ha utilizado como IDE de desarrollo al Spring Tool Suite, como lenguaje de programación Java, Hibernate y JPA para la persistencia datos, MAVEN para el manejo de librerías y ficheros, JSF con PrimeFaces que permiten implementar una arquitectura MVC.

CAPÍTULO III

ANÁLISIS Y DISEÑO

En el presente capitulo se detalla el análisis y el diseño del Módulo de Administración y Bitácora de Utilización para el Sistema de Información CIALCO.

3.1 Análisis

3.1.1 Análisis del problema

Viendo la necesidad del Departamento de Redes Comerciales del Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP Quito – Ecuador) de mejorar la calidad de la información almacenada en el Sistema de Información CIALCO, se realizó un análisis para la implementación del Módulo de Administración y Bitácora de su uso, que controla las distintas opciones y acciones que se pueden realizar dentro del sistema, para contar con datos veraces y oportunos.

Para ello nos basamos en la metodología XP en su fase de diseño. Mediante la recopilación de historias de usuarios se pudo determinar las necesidades o requerimientos que el Módulo de Administración y Bitácora de Utilización debe contener para contar con un funcionamiento óptimo del sistema que garantice así un mejor manejo de la información de los CIALCOS.

Seguidamente se establecieron prioridades a los requerimientos y se otorgaron tiempos de desarrollo, definiendo así el cronograma para las revisiones de avances, entregas, correcciones de errores, publicación de versiones.

3.1.2 Historias de usuarios

Tabla 1. Historia de usuario Nº 1

Historia de usuario			
Número:	1	**Usuario:**	Julio Cabezas
Nombre Historia:	Administraciones de Sistemas		
Prioridad en el sistema:	Alta	**Riesgo de desarrollo:**	Alta
Puntos Estimados:	4	**Iteración asignada:**	4
Programador responsable:	Francisco Bonilla		
Descripción: El módulo de Administración y Bitácora de utilización debería ser manejado mediante definición de perfiles, uno que maneje la parametrización, que pueda realizar la creación de usuarios, según el perfil se despliegan las opciones del sistema mediante la creación de menús dinámicos, de igual forma determinar los privilegios de manipulación de los datos CRUD, por otra parte un perfil con el cual se permita el ingreso de la información de los diferentes CIALCOs.			
Observaciones: De la descripción podemos enumerar los siguientes requerimientos: **R1.** Creación de perfiles de usuarios, los que se parametrizaran por defecto son Administrador y Técnico. **R2.** El administrador tendrá acceso a CRUD de parametrizaciones del sistema, CRUD de información de usuarios, CRUD de menús, CRUD de perfiles, asignación de perfiles al usuario, asignación del menú al perfil, el menú contendrá las opciones y pantallas. **R3.** El técnico tendrá acceso al CRUD de los CIALCOs. **R4.** El ingreso a la aplicación debe ser mediante autentificación de usuario y contraseña.			

Nota: Se describe las historias de usuario y se definen los requerimientos
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 2. Historia de usuario Nº 2

Historia de usuario			
Número:	2	**Usuario:**	Julio Cabezas
Nombre Historia:	Complemento al módulo de Administración y Bitácora		
Prioridad en el sistema:	Alta	**Riesgo de desarrollo:**	Alta
Puntos Estimados:	5	**Iteración asignada:**	2
Programador responsable:	Francisco Bonilla		
Descripción: Se requiere aumentar las Zonas para subdividir las provincias. Se necesita aumentar al usuario la zona en la cual trabaja, para que solo pueda visualizar la información de los CIALCOS que pertenecen a su Zona laboral, hay tener un usuario el cual puede ver la información de todas las zonas.			
Observaciones: De la descripción podemos enumerar los siguientes requerimientos: **R1.** Aumentar una tabla que contenga las Zonas, aumentar la relación de la Zona			

con las provincias.

R2. Aumentar la zona en la cual trabaja el usuario.
R3. Control para mostrar solo la información de los CIALCOs, de acuerdo a la zona asignada al usuario.
R4. Crear una zona general, el usuario que se le asigne la zona general, podrá ver y manipular la información de todos los CIALCOs, indistintamente de la zona a la que pertenezca la información.
R5. Registro continuo de la Bitácora de utilización con la siguiente información: acción realizada (Inicio de sesión, creación, consulta, edición, eliminación de registros), descripción, nombre y apellido de usuario, correo Electrónico, Dirección de IP, Fecha y hora en la cual se realizó la acción.

Nota: Se describe la segunda historia de usuario para complementar el módulo de administración y bitácora
Fuente: Francisco Bonilla y Miguel Machasilla

3.1.3 Plan de Trabajo / Iteraciones

Tabla 3. Plan de Trabajo e Iteraciones de la Historia de Usuario Nº1

Plan de Trabajo			
No. Historia de Usuario	**#Requerimiento**	**Prioridad**	**Fecha de Entrega**
1	R1	2	07/06/2016
1	R2	1	07/06/2016
1	R3	4	07/06/2016
1	R4	3	07/06/2016
Iteraciones			
Fecha	**Descripción**		
06/04/2016	Levantamiento de historia de usuario		
22/04/2016	Revisión de avances del requerimiento R2		
06/05/2016	Revisión final del R2, avances del R4		
20/05/2016	Revisión final del R4, avances del R3		
03/06/2016	Revisión final de R3		
04/06/2016 al 05/06/2016	Correcciones y modificaciones pequeñas		
07/06/2016	Entrega final de requerimientos		

Nota: se define las prioridades, fecha de entrega y la iteración de revisiones periódicas.
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 4. Plan de trabajo e Iteración para la Historia de Usuario No. 2

Plan de Trabajo			
No. Historia de Usuario	**#Requerimiento**	**Prioridad**	**Fecha de Entrega**
2	R1	1	21/06/2016
2	R2	2	21/06/2016
2	R3	3	21/06/2016
2	R4	4	21/06/2016
2	R5	5	21/06/2016
Iteraciones			
Fecha	**Descripción**		
07/06/2016	Levantamiento de historia de usuario		
14/06/2016	Revisión de avances de todos los requerimientos		
21/05/2016	Entrega final de requerimientos		

Nota: Se define las prioridades y fechas de entrega de los requerimientos.
Fuente: Francisco Bonilla y Miguel Machasilla

Diseño

Como ayuda para desarrollar el diseño del sistema de administración se utilizó el Lenguaje Unificado de Modelado (UML) para diagramar casos de uso, de flujo y diseñar la base de datos.

3.1.4 Diagramas de casos de uso

3.1.4.1 Diagrama de caso de uso Administrador

El siguiente diagrama muestra las acciones que desenvuelve el actor Administrador. Seguido describiremos los diferentes casos de usos que intervienen en el mismo.

Diagrama de Administrador

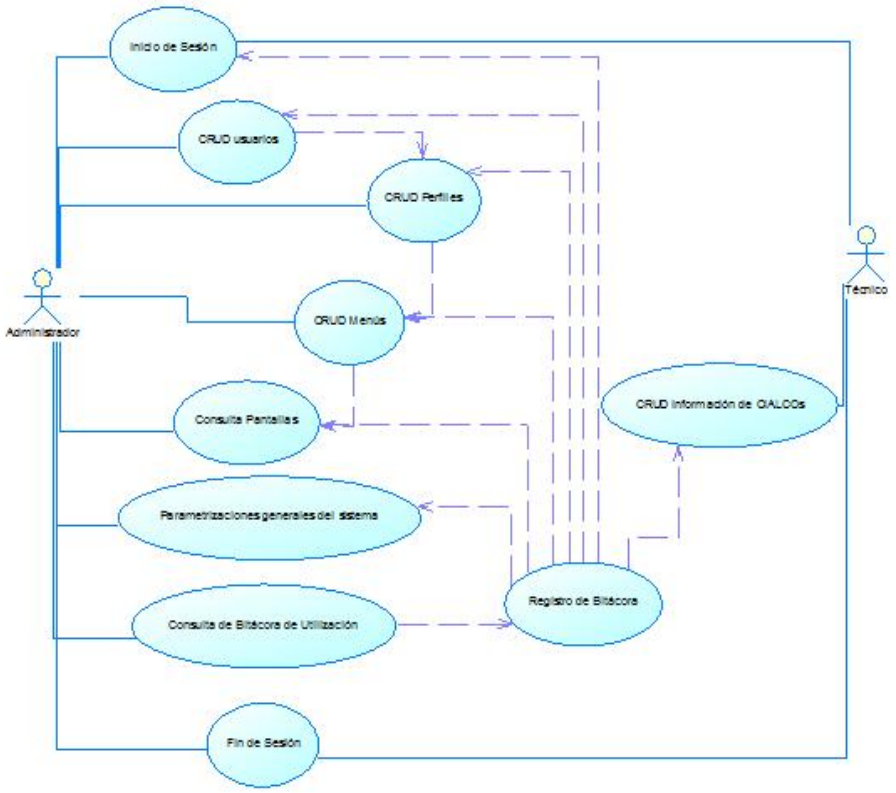


Figura 7. Acciones de un usuario con perfil de Administrador
Elaborado por: Francisco Bonilla y Miguel Machasilla

Tabla 5. Descripción del caso de uso del administrador

Descripción del caso de uso			
Nombre:	Diagrama caso de uso Administrador	**No. Figura**	7
Caso de Uso	**Descripción**		
Inicio de Sesión	Hace referencia a la acción de autentificación mediante el ingreso de usuario y contraseña		
CRUD usuarios	Se refiere a la creación, consulta, actualización y eliminación de un usuario, la acción de creación y modificación dependen de la existencia de perfiles para su vinculación		
CRUD Perfiles	Creación, consulta, actualización y eliminación de un perfil, las acciones de creación y modificación dependen de la asignación de menús.		
CRUD Menús	Creación, consulta, actualización y eliminación de un menú, las acciones de creación y modificación dependen de la asignación de pantallas.		
Consulta Pantallas	En las pantallas el usuario solo puede consultarlas ya que estas están directamente relacionadas a la parte de desarrollo.		
Parametrizaciones	El administrador tiene la capacidad de establecer las		

generales del sistema	parametrizaciones del sistema para que sean utilizadas por el usuario Técnico, entre ellas podemos nombrar, la manipulación de información de catálogos, provincias, cantones, parroquias, zonas, productos, etc.
Consulta de Bitácora	Tendrá el acceso a consultar la bitácora en la cual se detallan las acciones que cada uno de los usuarios hace dentro del sistema. Depende del registro de bitácora.
Registro de Bitácora	Depende de todas las acciones anteriores y de las acciones que realiza el actor Técnico
CRUD Información de CIALCOs	Creación, consulta, modificación, eliminación de registros de los CIALCOs.
Fin de Sesión	Salida del sistema

Nota: Toda acción que se realiza en el sistema graba un registro de bitácora con la información del usuario (Nombre, Apellidos, correo, dirección de IP, Fecha y hora en la que se realiza la acción, descripción de la acción, y de ser el caso registra el id del registro que es modificado.
Fuente: Francisco Bonilla y Miguel Machasilla

3.1.4.2 Diagrama de caso de uso de Técnico

El siguiente diagrama muestra las acciones que desenvuelve el actor Administrador.

Seguido describiremos los diferentes casos de usos que intervienen en el mismo.

Diagrama de caso de uso del actor Técnico

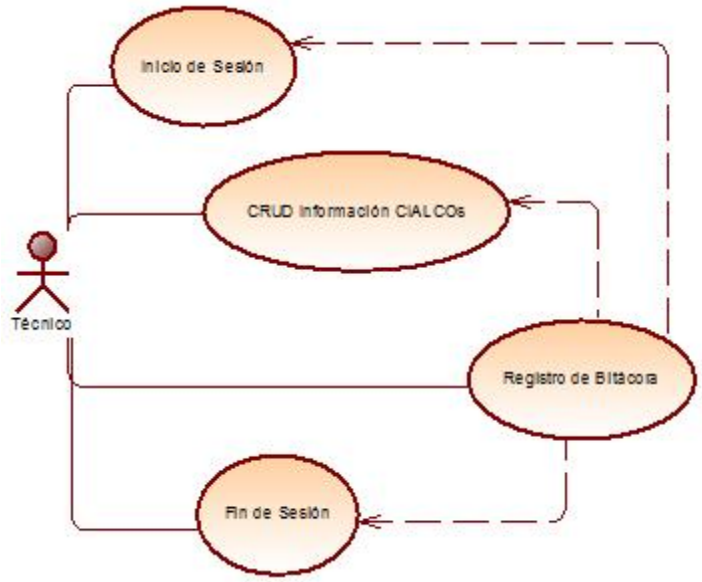


Figura 8. Descripción de acción de usuario con perfil Técnico
Elaborado por: Francisco Bonilla y Miguel Machasilla

Tabla 6. Descripción del caso de uso del técnico

Descripción del caso de uso			
Nombre:	Diagrama de caso de uso del actor Técnico	**No. Figura**	8
Caso de Uso	**Descripción**		
Inicio de Sesión	Hace referencia a la acción de autentificación mediante el ingreso de usuario y contraseña		
CRUD Información de CIALCOs	Creación, consulta, modificación, eliminación de registros de los CIALCOs.		
Registro de Bitácora	Depende de todas las acciones anteriores		
Fin de Sesión	Salida del sistema		

Nota: Toda acción que se realiza en el sistema graba un registro de bitácora con la información del usuario (Nombre, Apellidos, correo, dirección de IP, Fecha y hora en la que se realiza la acción, descripción de la acción, y de ser el caso registra el id del registro que es modificado.
Fuente: Francisco Bonilla y Miguel Machasilla

3.1.5 Diagrama de flujo

En el siguiente diagrama se observa el flujo de los perfiles que fueron definidos en la fase de análisis.

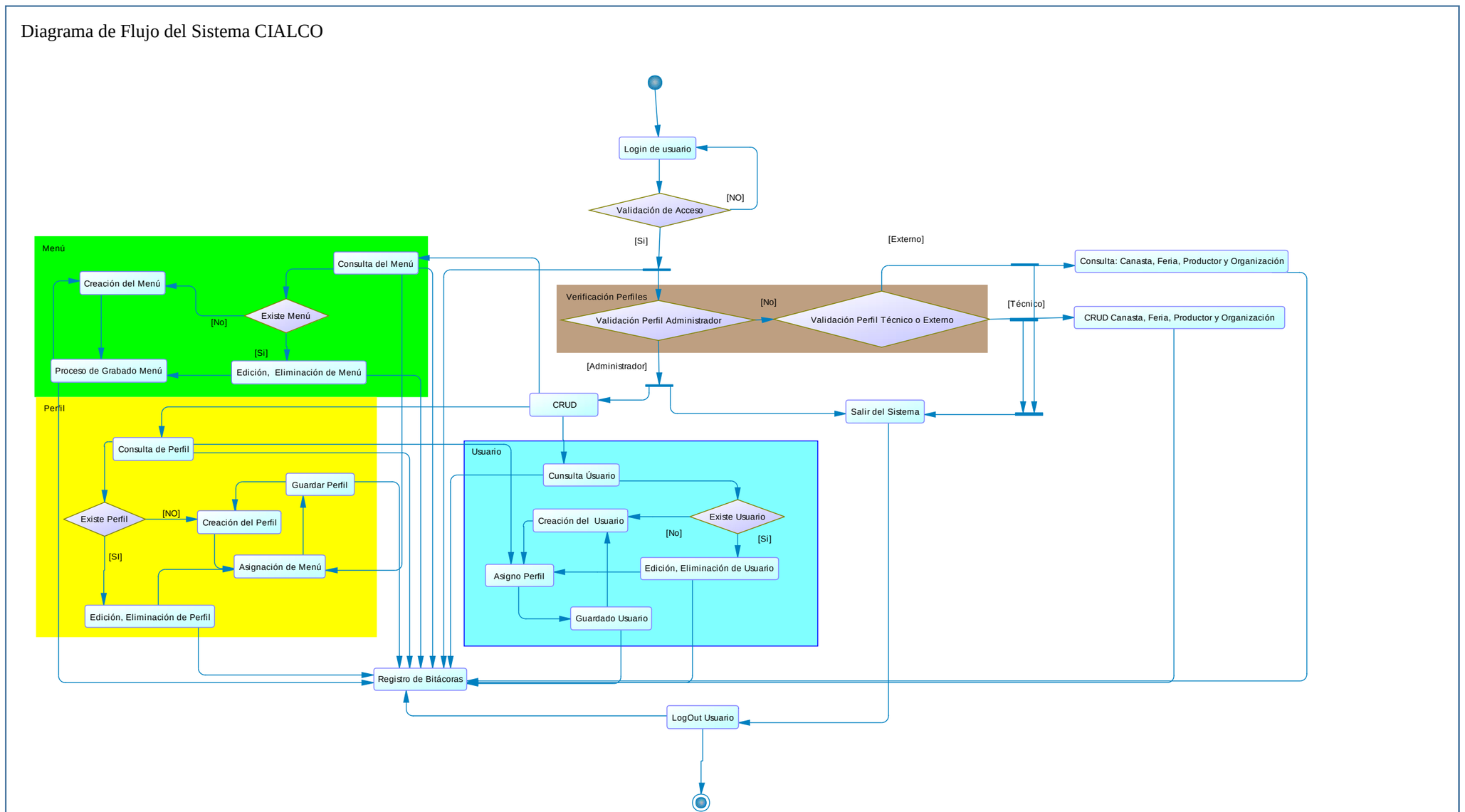


Figura 9. Diagrama de flujo del Sistema CIALCO
Elaborado por: Francisco Bonilla y Miguel Machasilla

Para dar inicio al sistema se debe ingresar las credenciales en el Login de usuario, si las validaciones no son correctas vuelve a solicitar las credenciales caso contrario verifica con que perfil se está accediendo.

Si el sistema verifica que las credenciales ingresadas son de perfil administrador le permite realizar acciones de crear, consultar, actualizar, y eliminar (CRUD) registros de usuarios, perfiles, pantallas, menús, catálogos y zonas.

Caso contrario las credenciales ingresadas son de técnico o de usuario externo.

Si el sistema verifica que las credenciales ingresadas son de perfil técnico le permite realizar acciones de crear, consultar, actualizar, y eliminar (CRUD), en los CIALCO de Canasta, Feria, Productor y Organización.

Si el sistema verifica que las credenciales ingresadas son de perfil externo le permite consultar registros dentro de los CIALCOs.

El usuario logueado en el sistema puede realizar un LogOut en cualquier momemento y de esta forma salir del sistema.

Si el sistema detecta inactividad durante 10 minutos independientemente del usuario utilizado inmediatamente solicitara el ingreso de las credenciales, adicional se puede salir del sistema en cualquier momento.

3.1.6 Tarjetas C.R.C.

En las siguientes tablas y mediante las Tarjetas C.R.C de la metodología XP definimos las clases con sus respectivos nombres, responsabilidades (Atributos y Métodos) y colaboradores (Clases con las que trabaja en conjunto).

Tabla 7. Tarjeta C.R.C de Usuario

Usuario	
Responsabilidades	**Colaboradores**
Usuario, Clave primaria – Tipo Dato Intiger(8)	Persona
Nombre usuario, Tipo dato Character(15)	Zona
Contraseña, Tipo dato Character(100)	
Registro de auditoría (Usuario de Creación, Fecha Creación, Usuario de modificación, fecha de modificación, estado.	

Nota: Contiene los datos del usuario que permiten la autentificación y acceso al sistema.
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 8. Tarjeta C.R.C Usuario – Perfil

Usuario – Perfil	
Responsabilidades	**Colaboradores**
Usuario Perfil, Clave primaria – Tipo Dato Integer(8)	Perfil
Crear permiso, tipo dato Boolean	Usuario
Editar permiso, tipo dato Boolean	
Consultar permiso, tipo dato Boolean	
Eliminar permiso, tipo dato Boolean	
Registro de auditoría (Usuario de Creación, Fecha Creación, Usuario de modificación, fecha de modificación, estado.	

Nota: Maneja la relación entre la Clase Perfil VS Usuario, también se definen los privilegio de acceso CRUD a la información
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 9. Tarjeta C.R.C Perfil

Perfil	
Responsabilidades	**Colaboradores**
Perfil, Clave primaria – tipo dato Integer(8)	Menú
Nombre Perfil, tipo dato VarChar(100)	
Descripción, tipo dato VarChar(100)	
Registro de auditoría (Usuario de Creación, Fecha Creación, Usuario de modificación, fecha de modificación, estado.	

Nota: Maneja la información de los perfiles, al cual se le asigna el menú.
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 10. Tarjeta C.R.C Menú

Menú	
Responsabilidades	**Colaboradores**
Menú, clave primaria – tipo dato Integer(8)	Pantalla
Nombre, tipo dato VarChar(100)	Menú Padre
Icono, tipo dato VarChar(20)	
Registro de auditoría (Usuario de Creación, Fecha Creación, Usuario de modificación, fecha de modificación, estado.	

Nota: Maneja todas las opciones del menú que se muestran de acuerdo al perfil. Dependiendo de la jerarquía en la cual se desea crear el menú esta puede tener una referencia a sí misma.
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 11. Tarjeta C.R.C Pantalla

Pantalla	
Responsabilidades	**Colaboradores**
Pantalla, clave primaria – tipo dato Long	
Url, tipo dato Varchar(100)	
Nombre Pantalla, tipo dat Varchar (200)	
Registro de auditoría (Usuario de Creación, Fecha Creación, Usuario de modificación, fecha de modificación, estado.	

Nota: Esta clase contiene las URLs de las pantallas de consultas del sistema, Debe ser usadas solo en caso de nuevos desarrollos.
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 12. Tarjeta C.R.C Bitácora

Bitácora	
Responsabilidades	**Colaboradores**
Bitácora, clave primaria – tipo dato Integer(8)	Usuario
Acción, tipo dato Varchar(250)	
Detalle acción, tipo dato Varchar(250)	
Correo, tipo dato Varchar(350)	
Dirección de IP, tipo dato VarChar (20)	
Registro de auditoría (Usuario de Creación, Fecha Creación, Usuario de modificación, fecha de modificación, estado.	

Nota: Clase que contiene los datos de todas las transacciones que los usuarios realizan dentro del sistema.
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 13. Tarjeta C.R.C Persona

Persona	
Responsabilidades	**Colaboradores**
Persona, clave primaria – tipo dato Integer(8)	
Nombre, tipo dato Varchar(150)	
Apellido, tipo dato Varchar(150)	
Nombre completo tipo dato (350)	
Cédula Personal, dato tipo Varchar(13)	
Edad, tipo dato Integer (2)	
Mail, tipo dato Varchar(250)	
Genero, tipo dato Character(1)	
Teléfono fijo, tipo dato Character(20)	
Teléfono celular, tipo dato Character(20)	
Registro de auditoría (Usuario de Creación, Fecha Creación, Usuario de modificación, fecha de modificación, estado.	

Nota: maneja los datos personales de los usuarios.
Fuente: Francisco Bonilla y Miguel Machasilla

3.1.7 Diseño Lógico de la Base de Datos

De acuerdo con el análisis de las Tarjetas C.R.C elaboramos el diseño del diagrama lógico de la base de datos del Módulo de Administración y Bitácora de Utilización. Como se muestra en la Figura 10.

Diagrama Lógico de la Base del Módulo de Administración y Bitácora de Utilización

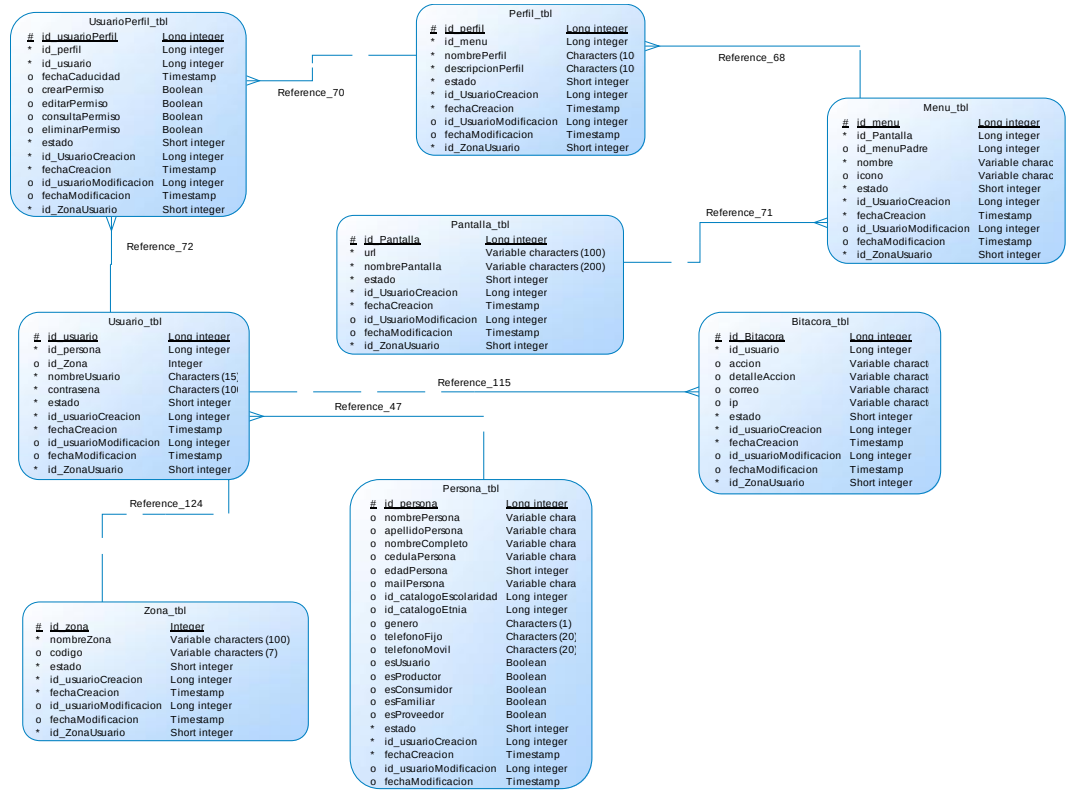


Figura 10. Diagrama Lógico de la base del Módulo de Administración y Bitácora de Utilización. Elaborado por: Francisco Bonilla, Miguel Machasilla.

3.1.8 Diseño Físico de la Base de Datos

En la siguiente figura se muestra el diseño físico de la base de datos en la cual observamos las relaciones entre las diferentes tablas.

Diagrama físico de la base de datos del Módulo de Administración y Bitácora

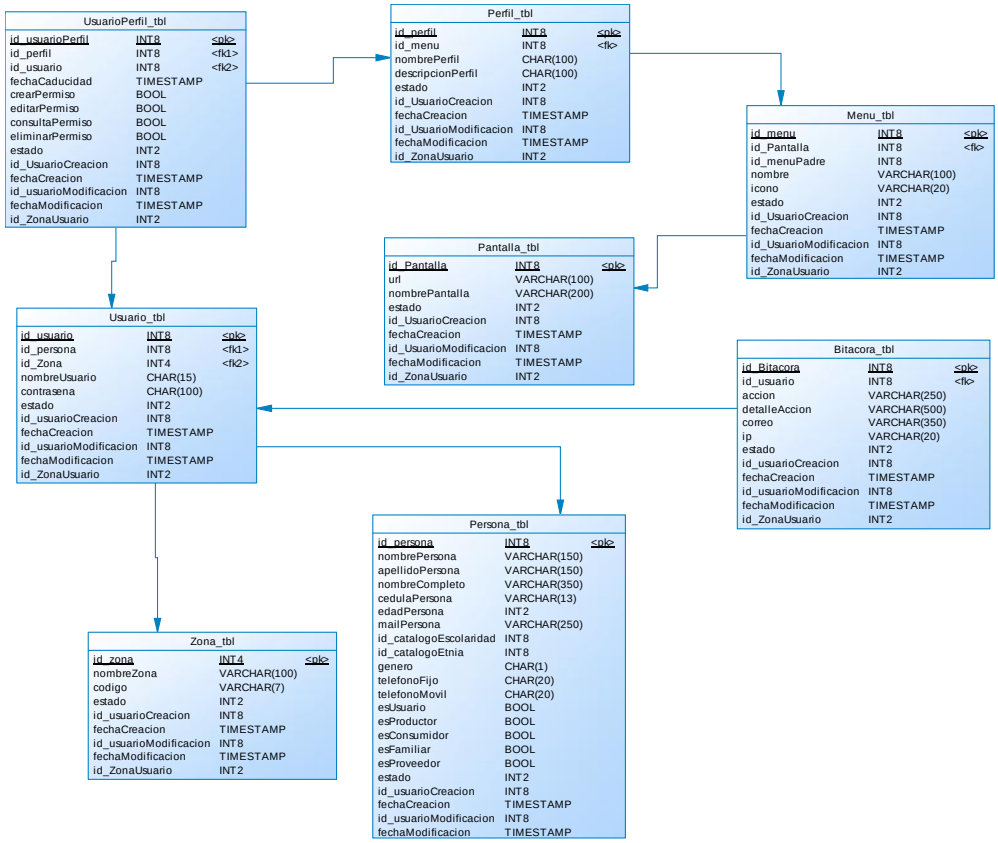


Figura 11. Diagrama físico de la base de datos del Módulo de Administración y Bitácora de Utilización.
Elaborado por: Francisco Bonilla, Miguel Machasilla

CAPÍTULO IV

CONSTRUCCIÓN Y PRUEBAS

En el presente capítulo se detalla la construcción del Módulo de Administración y Bitácora de Utilización, el cual se ha basado en las tecnologías mencionadas en el marco teórico, seguidamente se ha realizado pruebas, las que verifican el funcionamiento del módulo de acuerdo a los requerimientos detallados en las historias de usuario.

4.1 Construcción

4.1.1 Construcción de la Base de Datos

Para la creación de la base de datos utilizamos la herramienta PowerDesigner, que permite generar el script de creación de la Base de Datos desde el modelo físico diseñado en el capítulo dos. Este Script contiene las sentencias SQL para la creación de las tablas, índices, claves primarias y foráneas, atributos, etc. Que serán utilizados posteriormente por la aplicación.

En la tabla se presenta una parte del contenido del Script de creación de la base de datos y la forma de crear una base de datos en PostgreSQL.

Tabla 14. Script de Creación de la Base de Datos

Script de Creación de la Base de Datos
/*==*/ /* DBMS name: PostgreSQL 8 */ /* Created on: 07/07/2016 18:11:43 */ /*==*/

```
/*==============================================*/
/* Tale: PERFIL_TBL                             */
/*==============================================*/
create table PERFIL_TBL (
   ID_PERFIL           INT8              not null,
   ID_MENU             INT8              not null,
   NOMBREPERFIL        CHAR(100)         not null,
   DESCRIPCIONPERFIL   CHAR(100)         not null,
   ESTADO              INT2              not null,
   ID_USUARIOCREACION  INT8              not null,
   FECHACREACION       TIMESTAMP         not null,
   ID_USUARIOMODIFICACION INT8              null,
   FECHAMODIFICACION   TIMESTAMP         null,
   ID_ZONAUSUARIO      INT2              not null,
   constraint PK_PERFIL_TBL primary key (ID_PERFIL)
);


/*==============================================*/
/* Index: INDEX_9                               */
/*==============================================*/
create unique index INDEX_9 on PERFIL_TBL (
ID_PERFIL
);
```

```
/*==============================================*/
/* Table: PERSONA_TBL                           */
/*==============================================*/
create table PERSONA_TBL (
   ID_PERSONA          INT8              not null,
   NOMBREPERSONA       VARCHAR(150)      null,
   APELLIDOPERSONA     VARCHAR(150)      null,
   NOMBRECOMPLETO      VARCHAR(350)      null,
   CEDULAPERSONA       VARCHAR(13)       null,
   EDADPERSONA         INT2              null,
   MAILPERSONA         VARCHAR(250)      null,
   ID_CATALOGOESCOLARIDAD INT8           null,
   ID_CATALOGOETNIA    INT8              null,
   GENERO              CHAR(1)           null,
   TELEFONOFIJO        CHAR(20)          null,
   TELEFONOMOVIL       CHAR(20)          null,
   ESUSUARIO           BOOL              null,
   ESPRODUCTOR         BOOL              null,
   ESCONSUMIDOR        BOOL              null,
   ESFAMILIAR          BOOL              null,
   ESPROVEEDOR         BOOL              null,
   ESTADO              INT2              not null,
   ID_USUARIOCREACION  INT8              not null,
   FECHACREACION       TIMESTAMP         not null,
```

```
    ID_USUARIOMODIFICACION INT8              null,
    FECHAMODIFICACION    TIMESTAMP           null,
    ID_ZONAUSUARIO       INT2                not null,
    constraint PK_PERSONA_TBL primary key (ID_PERSONA)
);

/*==============================================*/
/* Index: CEDULAPERSONA                         */
/*==============================================*/
create unique index CEDULAPERSONA on PERSONA_TBL (
CEDULAPERSONA
);
```

Nota: Generación de sentencias SQL para la creación de la base de datos.

Fuente: Francisco Bonilla y Miguel Machasilla.

Creación de la base de datos en PosgresSQL

Ingresamos la consola de Administración de PosgreSQL pgAdmin. Damos clic en la en PostgreSQL 9.4 (Localhost:5432) que se encuentra en la sección de búsqueda de objetos como se muestra en la figura.

Consola de Administración de PostgreSQL

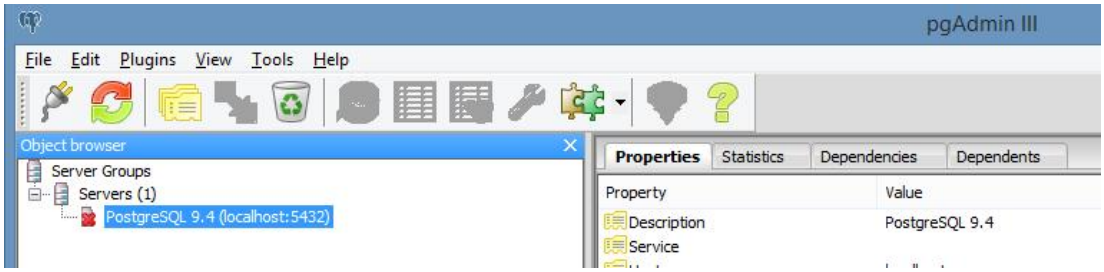


Figura 12. Ingreso al entorno de administración de PostgreSQL.
Elaborado por: Francisco Bonilla y Miguel Machasilla.

Se desplegara el menú con las opciones que contiene el PostgreSQL, dar clic derecho sobre la opción Databases y escoger la opción New Database.

Entorno de Administración de PostgreSQL

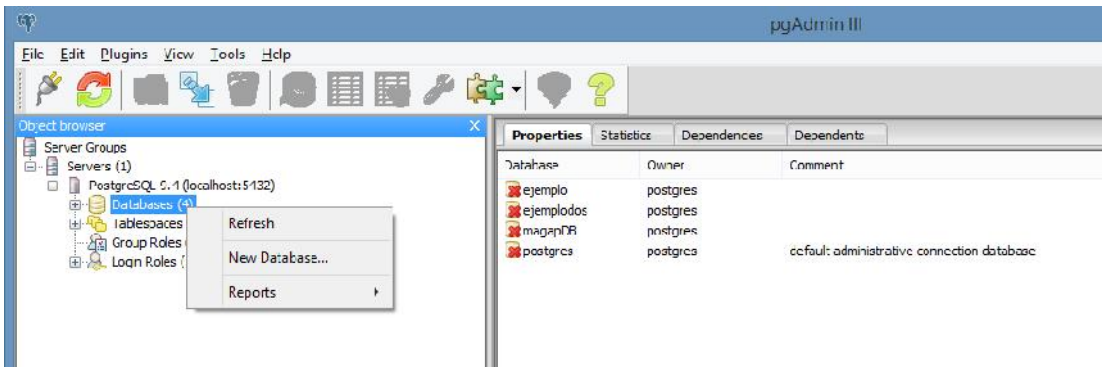


Figura 13. Opción de Creación de Base de Datos
Elaborado por: Francisco Bonilla y Miguel Machasilla

Se despliega la pantalla en la cual es necesario ingresar el nombre de la base de datos en este caso la llamaremos “magapDB”, y luego damos clic en OK.

Entorno de la Administración PostgreSQL

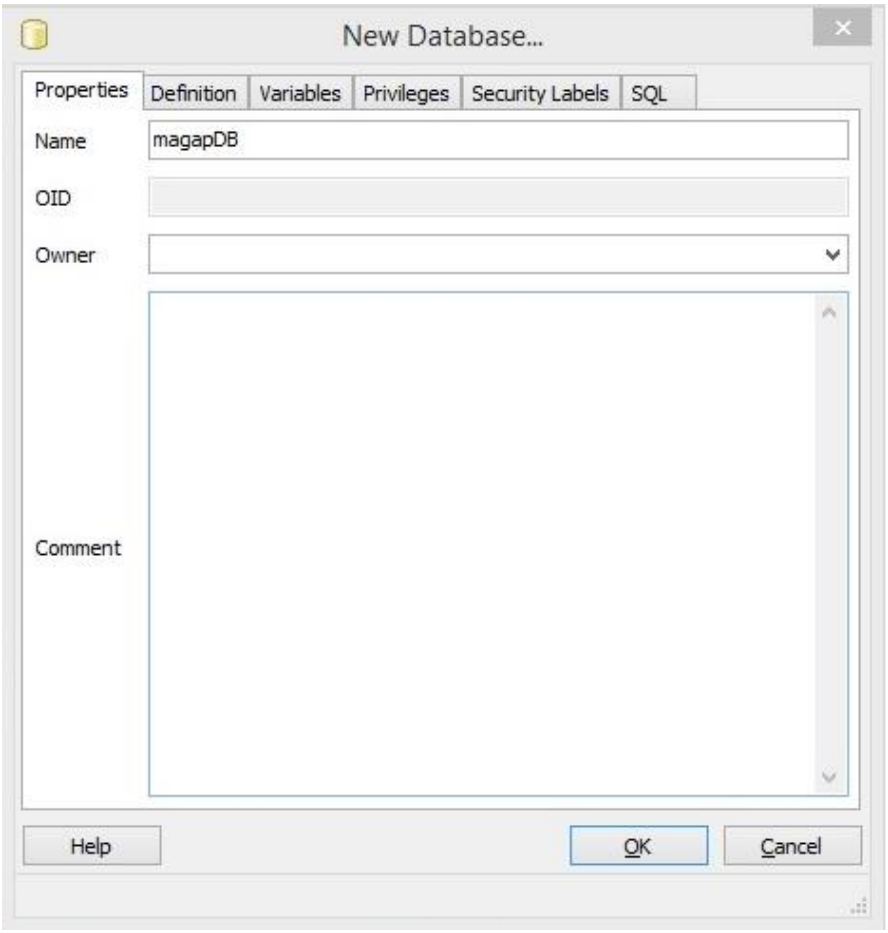


Figura 14. Ingreso del nombre de la Base de Datos
Elaborado por: Francisco Bonilla y Miguel Machasilla

Seleccione en el buscador de objetos la base creada y haga clic en la opción SQL de la barra de iconos para ingresar a la pantalla de Query del PostgreSQL. Copiar el código SQL del Script de creación de la base de datos y ejecutar.

Entorno de Administración PostgreSQL

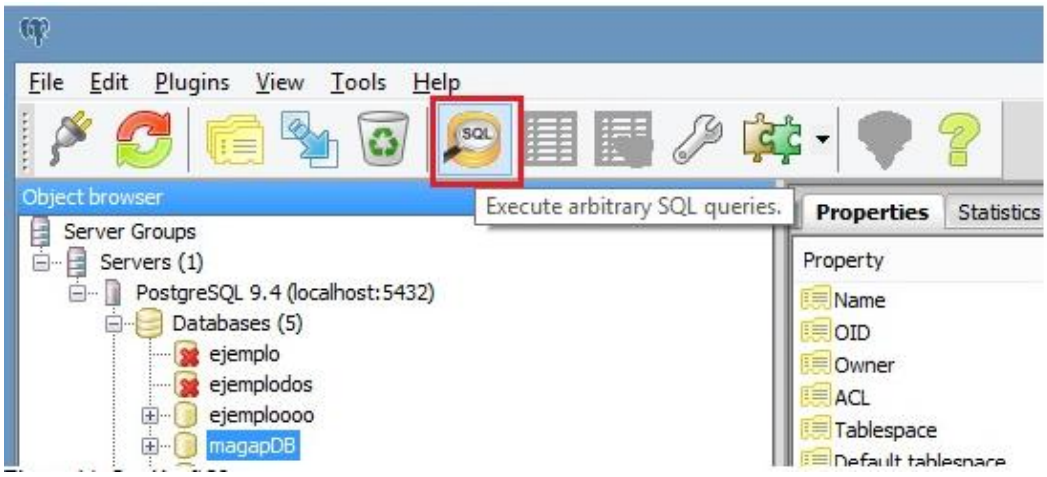


Figura 15. Opción SQL
Elaborado por: Francisco Bonilla y Miguel Machasilla

Entorno SQL Query de PstgreSQL

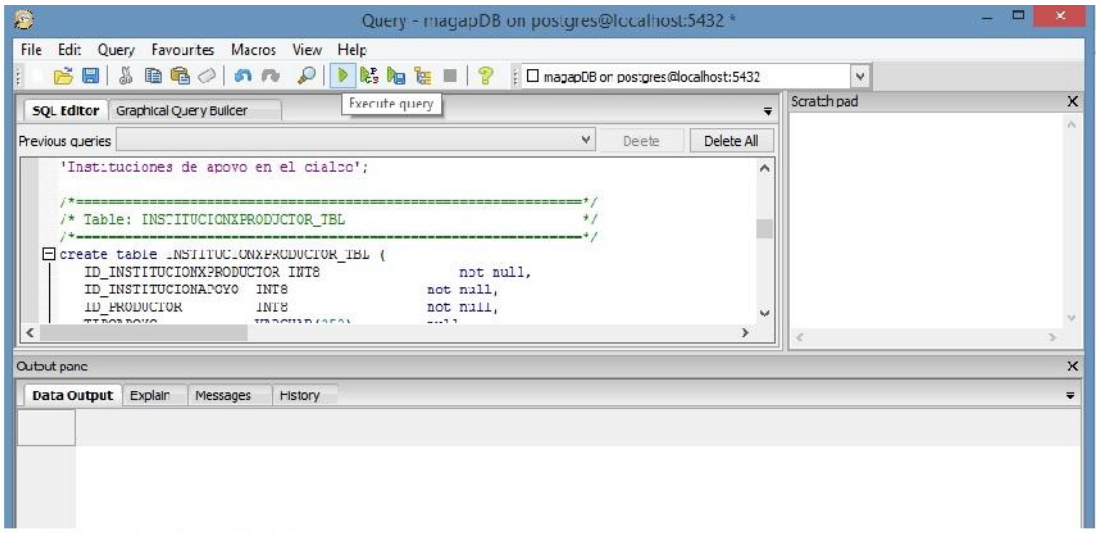


Figura 16. Ejecución de Scritp de Creación de la Base de Datos.
Elaborado por: Francisco Bonilla y Miguel Machasilla.

Luego de la creación de la base de datos, procedemos a ejecutar un segundo script que dará las parametrizaciones iniciales del sistema, para el funcionamiento. Entre estas parametrizaciones podemos mencionar:

Creación de los secuenciales de cada tabla.

Creación de los menús.

Creación de los perfiles iniciales Administrador y Técnico.

Asignación creación de dos usuarios, el uno para el manejo de la administración y el otro con el perfil de técnico para el ingreso general de la información de los CIALCOs.

La creación de Catálogos de información, esencial para el consumo de información de los CIALCOs.

Parametrización de las Zonas, provincias, cantones y parroquias.

4.1.2 Construcción de la interfaz del módulo de Administración y Bitácora

Para la creación de la interfaz, nos hemos basado en la tecnología adoptada anteriormente en el Sistema de Información CIALCO, para ello utilizaremos el IDE de desarrollo Spring Tool Suite, lenguaje Java en su Plataforma JEE, Hibernate y JPA para la persistencia con la base de datos, MAVEN para el manejo de librerías y ficheros y JSF con PrimeFaces que me permiten implementar una arquitectura MVC.

Ya que la interfaz es el medio de comunicación entre el usuario y el sistema debe ser fácil y no tener ningún tipo de complicaciones, es por ello que se ha propuesto al Departamento de Redes Comerciales (MAGAP) el diseño de una interfaz como se ve en siguiente figura.

Prototipo de Interfaz de usuario

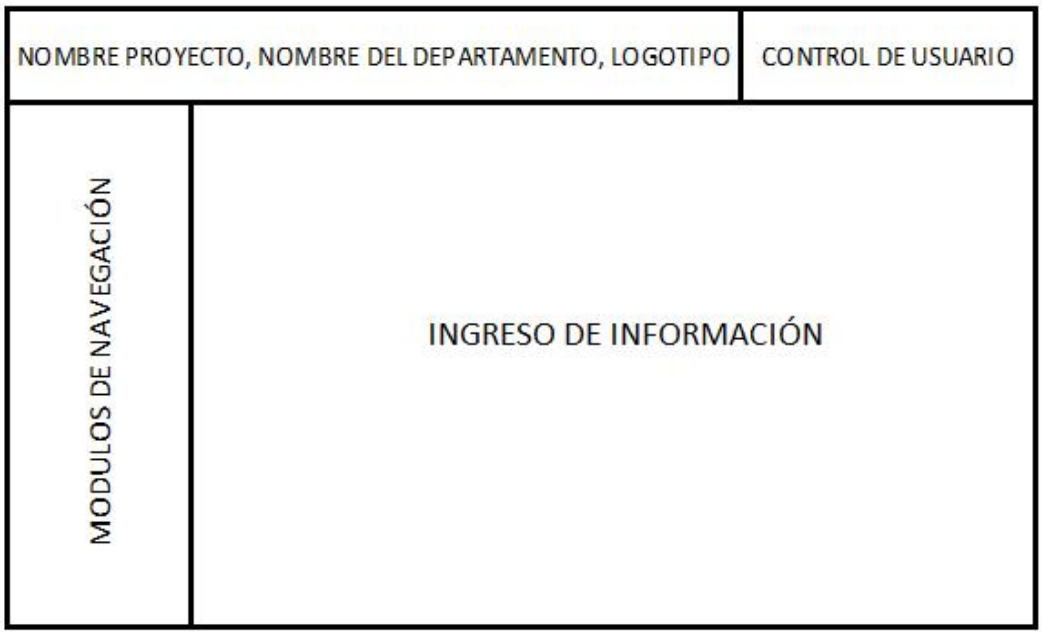


Figura 17. Prototipo de Interfaz de usuario
Elaborado por: Francisco Bonilla y Miguel Machasilla

Según estos referentes el proyecto cuenta con tres fases respetando la arquitectura de la plataforma JEE. La cual se describe a continuación.

IDE Spring Tool Suite

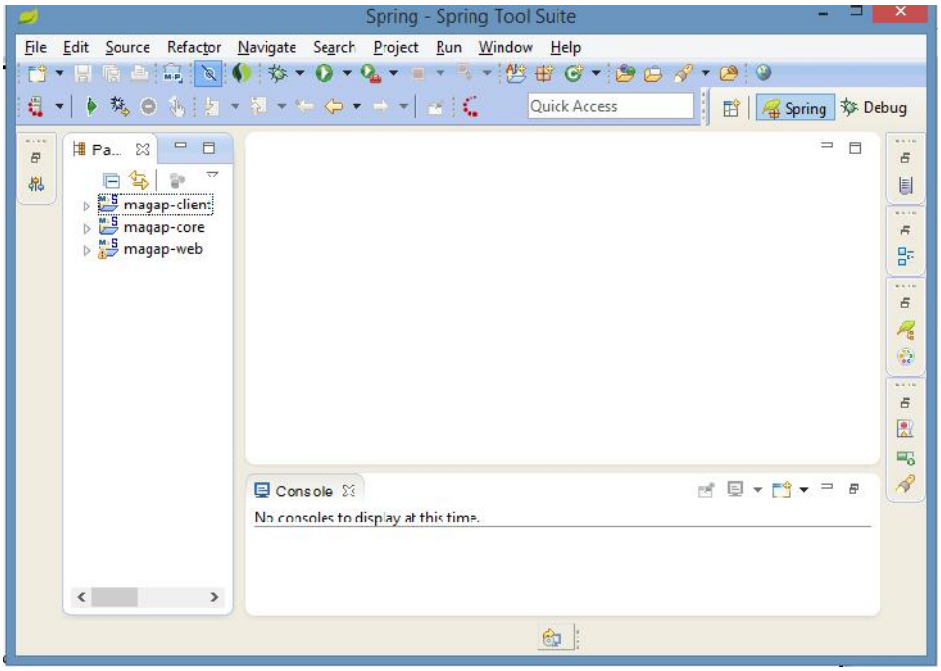


Figura 18. Definición de proyectos en JEE
Elaborado por: Francisco Bonilla y Miguel Machasilla

magap-client: contiene las Entidades o DTOs, la configuración para la conexión a la base de datos, hace referencia a la capa del servicio, y también contiene una parte esencial para el módulo de administración como es el proceso de encriptación y des encriptación de la contraseña del usuario en base 64 bits con ayuda del algoritmo de encriptación DES (Data Encryption Standard).

magap-core: contiene toda la lógica de negocio al igual que los conocidos DAO (Objeto de Acceso a Datos) que manejan la persistencia mediante Hibernate y JPA con la base de datos.

magap-web: es la parte visual del proyecto contiene las páginas web y los controladores o managedneans.

4.1.2.1 Construcción de los objetos Entidades o DTOs.

Para la creación de los DTOs primero debemos establecer la conexión con la base de datos lo cual lo hacemos en las configuraciones del archivo MagapDataSource.xml,

este archivo se encargara de contener el driver de conexión a la base de datos de PostgreSQL. A continuación se detalla el código de configuración que contiene el archivo mencionado.

Tabla 15. Driver de conexión a la base de datos de PostgreSQL.

Archivo MagapDataSource.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

	<!-- Data Source -->
	<bean id="dataSourcex"

	class="org.springframework.jdbc.datasource.DriverManagerDataSource">
		<property name="driverClassName">
			<value>org.postgresql.Driver</value>
		</property>
		<property name="url">
			<value>jdbc:postgresql://localhost:5432/magapDB</value>
		</property>
		<property name="username">
			<value>postgres</value>
		</property>
		<property name="password">
			<value>admin</value>
		</property>
	</bean>

	<bean id="magapTransactionManager"

	class="org.springframework.orm.hibernate3.HibernateTransactionManager"
>
		<property name="sessionFactory">
			<ref bean="sessionFactory" />
		</property>
	</bean>
</beans>
```

Nota: Configuración del Driver de conexión a la base de datos PostgreSQL.

Fuente: Francisco Bonilla y Miguel Machasilla

Citamos el código fuente de la entidad UsuarioDTO.java, alojado en el proyecto magap-client.

Tabla 16. Código entidad UsuarioDTO

Código entidad UsuarioDTO.java

```
package ec.gob.magap.dto;

import java.io.Serializable;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinColumns;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.PersistenceContext;
import javax.persistence.PersistenceContextType;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.hibernate.annotations.Proxy;

@Entity
@Table(name = "USUARIO_TBL")
@Proxy (lazy = false)
public class UsuarioDTO extends AuditoriaDTO implements Serializable {

	private static final long serialVersionUID = 2923730392886464654L;

	@Id
	@Column(name = "ID_USUARIO", nullable = false, insertable = true,
updatable = false, columnDefinition = "NUMBER")
	@SequenceGenerator(name = "ID_USUARIO_SEQ", sequenceName =
"ID_USUARIO_SEQ", initialValue = 1, allocationSize = 1)
	@GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"ID_USUARIO_SEQ")
```

```
	private Long idUsuario;

	@Column(name = "ID_PERSONA")
	private Long idPersona;
	@Column(name = "ID_ZONA")
	private Integer idZona;

	@Column(name = "NOMBREUSUARIO")
	private String nombreUsuario;

	@Column(name = "CONTRASENA")
	private String contrasena;

	@Transient
	private String ip;

	@PersistenceContext(type = PersistenceContextType.EXTENDED)
	@OneToMany(mappedBy="usuarioDTO", fetch = FetchType.LAZY,
cascade=CascadeType.PERSIST)
   private List<UsuarioPerfilDTO> usuarioPerfilDTOs;

	@PersistenceContext(type = PersistenceContextType.EXTENDED)
	@OneToMany(mappedBy="usuarioDTO", fetch = FetchType.LAZY,
cascade=CascadeType.PERSIST)
   private List<BitacoraDTO> bitacoraDTOs;

   @ManyToOne(optional = false, fetch = FetchType.LAZY)
	@JoinColumns({ @JoinColumn(name = "ID_PERSONA",
referencedColumnName = "ID_PERSONA", insertable = false, updatable = false)
})
	private PersonaDTO personaDTO;

	public Long getIdUsuario() {
		return idUsuario;
	}

	public void setIdUsuario(Long idUsuario) {
		this.idUsuario = idUsuario;
	}

	public Long getIdPersona() {
		return idPersona;
	}

	public void setIdPersona(Long idPersona) {
		this.idPersona = idPersona;
	}

	public Integer getIdZona() {
		return idZona;
```

```
	}

	public void setIdZona(Integer idZona) {
		this.idZona = idZona;
	}

	public String getNombreUsuario() {
		return nombreUsuario;
	}

	public void setNombreUsuario(String nombreUsuario) {
		this.nombreUsuario = nombreUsuario;
	}

	public String getContrasena() {
		return contrasena;
	}

	public void setContrasena(String contrasena) {
		this.contrasena = contrasena;
	}

	public String getIp() {
		return ip;
	}

	public void setIp(String ip) {
		this.ip = ip;
	}

	public List<UsuarioPerfilDTO> getUsuarioPerfilDTOs() {
		return usuarioPerfilDTOs;
	}

	public void setUsuarioPerfilDTOs(List<UsuarioPerfilDTO>
usuarioPerfilDTOs) {
		this.usuarioPerfilDTOs = usuarioPerfilDTOs;
	}

	public List<BitacoraDTO> getBitacoraDTOs() {
		return bitacoraDTOs;
	}

	public void setBitacoraDTOs(List<BitacoraDTO> bitacoraDTOs) {
		this.bitacoraDTOs = bitacoraDTOs;
	}

	public PersonaDTO getPersonaDTO() {
		return personaDTO;
	}
```

```
        public void setPersonaDTO(PersonaDTO personaDTO) {
                this.personaDTO = personaDTO;
        }
}
```

Nota: Ejemplo de una Emtidad

Fuente: Francisco Bonilla y Miguel Machasilla

A continuación se explica el código fuente de la entidad UsuarioDTO.java.

Las Entidades son Clases simples de Java con su definición de atributos y métodos, se la define como entidad con la anotación de JAVA @Entity, de igual manera y como se mencionó en el marco teórico se utilizan anotaciones que permiten las relaciones entre entidades en este caso @ManyToOne y @OneToMany. También se maneja estándares de programación de JAVA, podemos destacar; nombres de clases primera letra con mayúscula y el resto en minúscula, nombre de atributos y métodos primera letra en minúscula y si existen concatenaciones de palabras la primera es con mayúscula por ejemplo **idPersona** que en este caso es el nombre de uno de los atributos de la clase, se utiliza conceptos de herencia e implementación de interfaces, la anotación @Columna me permite identificar el campo de la tabla de la cual el atributo hace referencia.

4.1.2.2 Construcción de los objetos DAO.

Se alojan en el proyecto magap-core, estos actúan directamente a la base de datos mediante los criterios de Hibernate que se utilizan en la clase MagapGestor.java, ésta clase maneja toda la lógica de negocio del Sistema de Información CIALCO, mediante la implementación de interfaces Java y dando funcionalidades a cada uno de sus métodos dentro de ellas. El método **login** que se encuentra definido en la clase

MagapGestor.java, da la principal funcionalidad para el módulo de administración y bitácora ya que permite la autentificación de usuarios, utiliza el utilitario de des encriptación, inicia sesiones de usuarios siempre y cuando pasen el filtro de autentificación y permite que se carguen las diferentes opciones del sistema a las cuales tiene permisos un usuario.

A continuación se muestra el código del método **login** definido en la Clase MagapGestor.java.

Tabla 17. Código del método login definido en la Clase MagapGestor

Clase MagapGestor.java

```
@SuppressWarnings("unchecked")

        public UsuarioDTO login(String userName, String password)
                        throws MagapException {

                String passwE = Encrypt.getInstance().encryptBase64(password);
                DetachedCriteria criteriaUsuario = DetachedCriteria
                                .forClass(UsuarioDTO.class);
                criteriaUsuario.add(Restrictions.eq("nombreUsuario", userName));
                criteriaUsuario.add(Restrictions.eq("contrasena", passwE));
                criteriaUsuario.add(Restrictions.eq("estado",
Parameter.ESTADO_ACTIVO));
                criteriaUsuario.setFetchMode("personaDTO", FetchMode.JOIN);
                criteriaUsuario.setFetchMode("perfilDTO", FetchMode.JOIN);
                // criteriaUsuario.add(Restrictions.eq("perfilDTO.estado",
                // Parameter.ESTADO_ACTIVO));

                List<UsuarioDTO> usuarioDTOs = (List<UsuarioDTO>)
this.genericDAO
                                .findCriteria(criteriaUsuario);

                if (usuarioDTOs != null && !usuarioDTOs.isEmpty()) {
                        UsuarioDTO usuarioDTO = usuarioDTOs.iterator().next();

                        // Verifico si tienen perfiles
                        if (usuarioDTO.getUsuarioPerfilDTOs() != null) {
                                // Recorro los perfiles del usuario
                                for (Iterator<UsuarioPerfilDTO> it = usuarioDTO
                                                .getUsuarioPerfilDTOs().iterator();
```

```
it.hasNext();) {
                                        valor = it.next();
                                    }
                                    List<MenuDTO> menuDTOs =
findMenu(valor.getPerfilDTO()
                                                    .getIdMenu());
                                    valor.getPerfilDTO().setMenuDTOs(menuDTOs);
                                }
                                /*
                                 * if (usuarioDTO.getPerfilDTO() != null) {
List<MenuDTO> menuDTOs =
                                 * findMenu(usuarioDTO.getPerfilDTO() .getIdMenu());
                                 * usuarioDTO.getPerfilDTO().setMenuDTOs(menuDTOs);
}
                                 */
                                return usuarioDTO;
                            }
                            return null;
                        }
```

Nota: Código para el login de un usuario.

Fuente: Francisco Bonilla y Miguel Machasilla

4.1.2.3 Construcción de interfaz.

Para esta construcción nos basamos en el estándar de páginas web HTML 5 mediante la implementación del framework PrimeFace en conjunto con JSF. Los cuales son manejados a través de los ManagedBeans que se comunican con la capa de negocio y la de datos.

A continuación citamos el código fuente de una página web creada con PrimeFace y JSF llamada usuario.xhtml.

Tabla 18. Código fuente de una página web.

Código fuente de una página web.
<!DOCTYPE html> <ui:composition xmlns="*http://www.w3c.org/1999/xhtml*" xmlns:h="*http://java.sun.com/jsf/html*" xmlns:f="*http://java.sun.com/jsf/core*" xmlns:ui="*http://java.sun.com/jsf/facelets*" xmlns:p="*http://primefaces.org/ui*" xmlns:pe="*http://primefaces.org/ui/extensions*"

```
        xmlns:pt="http://xmlns.jcp.org/jsf/passthrough">

        <p:panel header="Usuario">
                <p:panel header="Datos personales" style="margin-bottom:20px">
                        <ui:param name="userTitle" value="Datos del suario" />
                        <h:panelGrid columns="4">
                                <h:outputText value="C&#233;dula: *" />
                                <p:inputText id="idced" required="true"
                                        requiredMessage="El campo C&#233;dula es
requerido"
                                        validatorMessage="Numero de c&#233;dula
invalida"

        value="#{usuarioController.personaDTO.cedulaPersona}"
                                        maxlength="10">
                                        <pe:keyFilter regEx="/[\d\-]/" />
                                        <f:validateLength minimum="10" />
                                        <f:validator validatorId="cedulaValidator" />
                                        <p:ajax event="blur" update="idced"
global="false" />
                                </p:inputText>

                                <h:outputText value="Nombres: *" />
                                <p:inputText required="true"
                                        requiredMessage="El campo Nombres es
requerido"

        value="#{usuarioController.personaDTO.nombrePersona}"
                                        maxlength="100">
                                        <pe:keyFilter regEx="/[a-z_\s]/i" />
                                </p:inputText>

                                <h:outputText value="Apellidos: *" />
                                <p:inputText required="true"
                                        requiredMessage="El campo Apellidos es
requerido"

        value="#{usuarioController.personaDTO.apellidoPersona}"
                                        maxlength="100">
                                        <pe:keyFilter regEx="/[a-z_\s]/i" />
                                </p:inputText>

                                <h:outputText value="Correo: *" />
                                <p:inputText id="pmail" required="true"
                                        requiredMessage="El campo Correo es
requerido"
                                        validatorMessage="El mail ingresado es
inv&#225;lido"

        value="#{usuarioController.personaDTO.mailPersona}">
```

```
                                        <pe:keyFilter regEx="/[a-z0-9_\.\-@]/i" />
                                        <f:validator validatorId="emailValidator" />
                                        <p:ajax event="blur" update="pmail"
global="false" />
                                </p:inputText>

                                <h:outputText value="Tel&#233;fono movil:" />
                                <p:inputMask
value="#{usuarioController.personaDTO.telefonoMovil}"
                                        mask="(99) 99-999-999" maxlength="10">
                                        <pe:keyFilter regEx="/[\d\-]/" />
                                </p:inputMask>

                                <h:outputText value="Tel&#233;fono fijo:" />
                                <p:inputMask
value="#{usuarioController.personaDTO.telefonoFijo}"
                                        maxlength="9" mask="(99) 999-9999">
                                        <pe:keyFilter regEx="/[\d\-]/" />
                                </p:inputMask>

                                <h:outputText value="G&#233;nero: *" />
                                <p:selectOneMenu
value="#{usuarioController.personaDTO.genero}"
                                        required="true" requiredMessage="El campo
G&#233;nero es requerido">
                                        <f:selectItem itemLabel="Seleccione..."
itemValue="" />
                                        <f:selectItem itemLabel="MASCULINO"
itemValue="M" />
                                        <f:selectItem itemLabel="FEMENINO"
itemValue="F" />
                                </p:selectOneMenu>

                                <h:outputText value="Etnia: *" />
                                <p:selectOneMenu

        value="#{usuarioController.personaDTO.idCatalogoEtnia}"
                                        required="true" requiredMessage="El campo
Etnia es requerido">
                                        <f:selectItem itemLabel="Seleccione..." />
                                        <f:selectItems
value="#{personController.etniaDTOs}" var="genero"

        itemLabel="#{genero.elementoCatalogo}"
                                                itemValue="#{genero.idCatalogo}" />
                                        <p:ajax event="change" partialSubmit="true"
process="@this"
                                                update="@this" global="false" />
                                </p:selectOneMenu>
```

```
                                <h:outputText value="Escolaridad: *" />
                                <p:selectOneMenu

        value="#{usuarioController.personaDTO.idCatalogoEscolaridad}"
                                        required="true" requiredMessage="El campo
Escolaridad es requerido">
                                        <f:selectItem itemLabel="Seleccione..." />
                                        <f:selectItems
value="#{personController.escolaridadDTOs}"
                                                var="genero"
itemLabel="#{genero.elementoCatalogo}"
                                                itemValue="#{genero.idCatalogo}" />
                                        <p:ajax event="change" partialSubmit="true"
process="@this"
                                                update="@this" global="false" />
                                </p:selectOneMenu>


                        </h:panelGrid>
                </p:panel>
                <p:panel header="Datos de usuario">
                        <h:panelGrid columns="4">
                                <h:outputText value="Usuario: *" />
                                <p:inputText required="true"
                                        requiredMessage="El campo Usuario es
requerido"

        value="#{usuarioController.usuarioDTO.nombreUsuario}"
                                        maxlength="100">
                                        <pe:keyFilter regEx="/[a-z_\s]/i" />
                                </p:inputText>

                                <h:outputText value="Contrase&#241;a: *" />
                                <p:password id="pass" required="true"
                                        requiredMessage="El campo
Contrase&#241;a es requerido"

        value="#{usuarioController.usuarioDTO.contrasena}" feedback="true" />

                                <p:outputLabel value="Perfil: *" />
                                <p:selectOneMenu id="page"

        value="#{usuarioController.usuarioPerfilDTO.idPerfil}"
                                        style="width:150px">
                                        <f:selectItem itemLabel="Seleccione..."
itemValue=""
                                                noSelectionOption="true" />
                                        <f:selectItems
value="#{indexPerfilController.perfilDTOs}"
                                                var="perfil"
```

```
itemLabel="#{perfil.nombrePerfil}"
                                        itemValue="#{perfil.idPerfil}" />
                                </p:selectOneMenu>
                                <h:outputText value="Zona: *" />
                                <p:selectOneMenu

        value="#{usuarioController.usuarioDTO.idZona}"
                                        required="true" requiredMessage="El campo
zona es requerido">
                                        <f:selectItem itemLabel="Seleccione..." />
                                        <f:selectItems
value="#{usuarioController.zonaDTOs}" var="zona"
                                                itemLabel="#{zona.nombreZona}"
                                                itemValue="#{zona.idZona}" />
                                        <p:ajax event="change" partialSubmit="true"
process="@this"
                                                update="@this" global="false" />
                                </p:selectOneMenu>
                        </h:panelGrid>
                </p:panel>

                <p:panel header="Permisos">
                        <p:selectManyCheckbox id="permisos"
                                value="#{usuarioController.selectedPermisos}">
                                <f:selectItem itemLabel="Crear" itemValue="crear"
/>
                                <f:selectItem itemLabel="Consultar"
itemValue="consultar" />
                                <f:selectItem itemLabel="Editar"
itemValue="editar" />
                                <f:selectItem itemLabel="Eliminar"
itemValue="eliminar" />
                        </p:selectManyCheckbox>
                </p:panel>

                <h:panelGroup layout="block" style="text-align: center">
                        <p:commandButton value="Guardar" styleClass="ui-
confirmdialog-no"
                                icon="ui-icon-disk" process="@all"
update="@form"

        actionListener="#{usuarioController.guardarForm}">
                                </p:commandButton>
                                <p:spacer width="5px"></p:spacer>
                                <p:commandButton value="Cancelar" type="button"
                                        styleClass="ui-confirmdialog-no" icon="ui-icon-
close" process="@all"
                                        update="@form">

                                </p:commandButton>
```

```
                </h:panelGroup>

        </p:panel>
</ui:composition>
```

Nota: Código ejemplo de una pagina WEB

Fuente: Francisco Bonilla y Miguel Machasilla

De este código obtenemos la siguiente visualización.

Visualización de Formulario usuario.xhtml

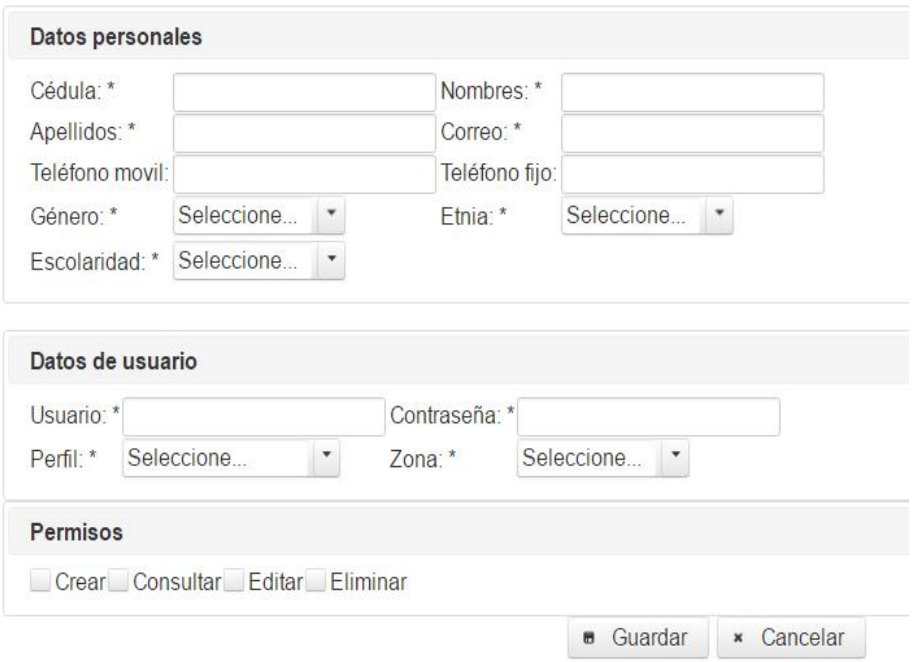


Figura 19. Pantalla de usuario.xhtml
Elaborado por: Francisco Bonilla y Miguel Machasilla

Pruebas

Mediante la utilización de las tablas de Casos de Pruebas de la metodología XP, se mencionan las siguientes pruebas.

Tabla 19. Caso de Prueba de la Historia de Usuario 1

Caso de Prueba de Aceptación			
Nº de Prueba:	1	**Nº de historia de usuario**	1
Nombre	Pruebas de funcionamiento de los requerimientos de la historia de usuario 1		
Descripción			
Se prueba el funcionamiento del módulo de administración y bitácora, es decir,			

autentificación de usuarios, CRUD de perfiles, CRUD de usuarios, CRUD Menús, CRUD parametrizaciones del sistema, accesos.	
Condiciones de Ejecución	
1. Se realizaran las pruebas en la máquina del desarrollador Francisco Bonilla.	
2. La presentación se lo realizará en las oficinas del departamento de Redes comerciales del MAGAP. En conjunto con el Ing. Julio Cabezas	
Pasos de ejecución	**Resultados de pruebas**
1. Verificar el ingreso al sistema mediante la autentificación de usuarios.	Verificado y Aprobado
2. Verificar la creación inicial de las parametrizaciones del sistema.	Verificado y Aprobado
3. Crear, editar, inhabilitar perfiles de usuarios.	Verificado y Aprobado
4. Creación de usuarios y pruebas de accesibilidad a la información de los CIALCOs.	Verificado y Aprobado
5. creación de menús para un nuevo perfil el cual lo utilizaremos en un usuario existente.	Verificado y Aprobado
6. Creación y edición de parametrizaciones como Elementos de los CIALCOs, para que puedan consumir en los formularios de CIALCOs.	Verificado y Aprobado
7. Comprobar las funcionalidades de los perfiles de administrador y técnico	Verificado y Aprobado
Evaluación de la prueba	Positiva

Nota: Se muestra las actividades realizadas y su resultado dando una calificación de positiva a los funcionamientos que se requieren en la historia de usuario Nº 1
Fuente: Francisco Bonilla y Miguel Machasilla

Tabla 20. Caso de Prueba de la Historia de Usuario Nº 2

Caso de Prueba de Aceptación			
Nº de Prueba:	2	**Nº de historia de usuario**	2
Nombre	Pruebas de funcionamiento de los requerimientos de la historia de usuario 2		
Descripción			
Se prueba el funcionamiento zonal del usuario, CRUD de zonas y vinculación de las provincias a las zonas, acceso a la información de acuerdo a la zona asignada al usuario.			
Condiciones de Ejecución			
1. Se realizaran las pruebas en la máquina del desarrollador Francisco Bonilla.			
2. La presentación se lo realizará en las oficinas del departamento de Redes comerciales del MAGAP. En conjunto con el Ing. Julio Cabezas			
Pasos de ejecución			**Resultados de pruebas**
1. Verificación de la creación de la tabla zonas para la asociación de las parroquias.			Verificado y Aprobado
2. Verificar el proceso de CRUD de zonas y la asociación de las provincias a las mismas.			Verificado y Aprobado
3. Aumentar el campo de zona al usuario para identificar la zona en la cual trabaja.			Verificado y Aprobado
4. Verificar que el usuario que tiene ya asignada una zona solo tenga acceso a la información generada en la misma zona			Verificado y Aprobado

5. Pruebas de la funcionalidad de acceso general de información por la zona general creada y asignada a un usuario.		Verificado y Aprobado
6. Verificar la información que se registra en la bitácora de utilización.		Verificado y Aprobado
Evaluación de la prueba	Positiva	

Nota: Se muestra las actividades realizadas y su resultado dando una calificación de positiva a los funcionamientos que se requieren en la historia de usuario Nº 2
Fuente: Francisco Bonilla y Miguel Machasilla

Conclusiones

- Utilizando la metodología XP se logró cumplir con el objetivo de creación de perfiles, dándole la administración de los mismos al usuario. Es decir, el usuario puede crear nuevos perfiles si así lo requiriese.
- Con la creación y administración de perfiles se mejoró la seguridad de la información y garantizó al acceso al sistema mediante la autentificación de usuarios.
- Con la creación de la Bitácora de Utilización se logró monitorear las diferentes acciones que realiza un usuario dentro del sistema.
- Con el almacenamiento de registros continuos de acciones que realizan los usuarios dentro del sistema, se logró generar estadísticas de usabilidad.
- Se logró consumir el servicio web para la consulta de información personal de los usuarios, productores, organizaciones, mediante su número de identificación Cédula o RUC.
- Con la creación de un sistema de administración de usuarios, se brinda al Departamento de Redes Comerciales del MAGAP tener control absoluto del sistema ya que permite crear y generar perfiles de usuarios con diferentes funcionalidades.

Recomendaciones

- Antes de empezar el desarrollo se debe tener definido el alcance con el cliente para que no se deba realizar cambios de gran escala en la aplicación.
- Indicar al cliente que debe realizar un respaldo de la base de datos periódicamente con la finalidad de asegurar la información ingresada al sistema.
- Capacitar al encargado de administrar el sistema dentro del Departamento de Redes Comerciales del MAGAP con la finalidad de asegurar la calidad de la aplicación entregada.

Uno de los problemas que se encontró al momento del desarrollo del sistema fue la falta de información como correos electrónicos, números de teléfonos celulares con lo que se limitó la generación de contraseñas para usuarios externos.

Trabajos Futuros

- Generar pruebas de ingreso masivos al sistema.
- Establecer una etapa de implementación y publicación en el portal web del MAGAP.
- Análisis de permisibilidad de creación de parametrizaciones del sistema realizados usuarios comunes.

Glosario de Términos	
CIALCO	Circuito alternativo de Comercialización
MAGAP	Ministerio de Agricultura, Ganadería, Acuacultura y Pesca
XP	Metodología eXtreme Programming
MVC	Modelo Vista Controlador
JSF	Java Server Face
JEE	Java Enterprise Edition
CRC	Clase, responsabilidad, colaboradores
API	Interfaz de Programación de aplicaciones
HTML	Lenguaje de marcas de hipertexto
EJB	Enterprise JavaBeans
DAO	Objeto de Acceso a Datos

LISTA DE REFERENCIA

Acción Ecológica. (Agosto de 2015). *Acción Ecológica*. Recuperado el 25 de Junio de 2016, de Acción Ecológica: http://www.accionecologica.org/soberania-alimentaria

Adictos al Trabajo. (2014). *Adictos al Trabajo*. Recuperado el 29 de Junio de 2016, de Adictos al Trabajo: https://www.adictosaltrabajo.com/tutoriales/introduccion-jsf-java/

Codejobs.biz. (30 de Septiembre de 2013). *Codejobs.biz*. Recuperado el 29 de Junio de 2016, de Codejobs.biz: https://www.codejobs.biz/es/blog/2013/09/30/que-es-jboss

Desarrollador Web. (02 de Enero de 2014). *Desarrollador Web*. Recuperado el 28 de Junio de 2016, de Desarrollador Web: http://www.desarrolloweb.com/articulos/que-es-mvc.html

Desarrollo Web. (02 de Septiembre de 2008). *Desarrollo Web Mapa Google*. Recuperado el 30 de Junio de 2016, de Desarrollo Web Mapa Google: http://www.desarrolloweb.com/articulos/incluir-mapa-google-web-5-minutos.html

Developers. (s.f.). *Developers Google Maps*. Recuperado el 30 de Junio de 2016, de Developers Google Maps: https://developers.google.com/maps/?hl=es

El club del Programador. (09 de Enero de 2012). *El club del Programador*. Recuperado el 28 de Junio de 2016, de El club del Programador:

http://www.elclubdelprogramador.com/2012/01/09/spring-introduccion-a-spring/

Fundación Wikimedia, Inc. (19 de 06 de 2016). *Sistema de información*. Obtenido de WIKIPEDIA La enciclopedia libre: https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n

Ingenieria de Software. (2012). *Ingenieria de Software*. Recuperado el 25 de 06 de 2016, de Ingenieria de Software: http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html

Java Enterprise. (2006). *Java Enterprise*. Recuperado el 27 de Junio de 2016, de Java Enterprise: http://www.jtech.ua.es/j2ee/2006-2007/jee.html

Java. (s.f.). *Java*. Recuperado el 27 de Junio de 2016, de Java: https://www.java.com/es/download/faq/techinfo.xml

MAGAP CIALCOS. (1 de 1 de 2015). *MAGAP*. Obtenido de MAGAP: http://www.agricultura.gob.ec/coordinacion-de-redes-comerciales/

Ministerio de Agricultura, Ganadería, Acuacultura y Pesca. (08 de junio de 2016). *Coordinación de Redes Comerciales*. Obtenido de Programas y Servicios: http://www.agricultura.gob.ec/redes-comerciales/

Naughton, P. (1996). *Manual de Java.* España: McGRAW-HILL/INTERAMERICANA DE ESPAÑA S.A.

Pico.dev. (11 de Marzo de 2016). *Pico.dev*. Recuperado el 27 de Junio de 2016, de Pico.dev: https://picodotdev.github.io/blog-bitix/2016/03/introduccion-y-nuevas-caracteristicas-de-java-ee-7/

Procesos de Software. (10 de Diciembre de 2014). *Procesos de Software*. Obtenido de Procesos de Software: http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP

Salvador, S. A. (2012). *Ingeniería de Software.* Garceta.

Software, I. d. (2005). *Pressman Roger S.* McGraw Hill.

springla. (2015). *Spring Framework*. Obtenido de springla: https://springla.io/spring/spring-framework/

Springla. (2015). *Spring-Framework*. Recuperado el 27 de Junio de 2016, de Spring-Framework: https://springla.io/spring/spring-framework/

WIKIPEDIA. (22 de 02 de 2015). *Programación extrema*. Obtenido de WIKIPEDIA La enciclopedia libre: https://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema

WIKIPEDIA. (25 de 07 de 2016). *Sistema de información*. Obtenido de WIKIPEDIA La enciclopedia libre: https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n

Wikipedia. (08 de Abril de 2016). *Wikipedia Spring_Framework*. Recuperado el 27 de Junio de 2016, de Wikipedia Spring_Framework: https://es.wikipedia.org/wiki/Spring_Framework

ANEXOS

Anexo 1. Manual de Usuario

El usuario debe ingresar a un navegador web y escribir la dirección URL http://localhost:8080/magap-web-0.0.1-SNAPSHOT/index.jsf, al cargar la página se visualiza una pantalla que solicita la autenticación de usuarios. Figura 1.

Pantalla inicial

Figura 1. Pantalla inicial.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Para ingresar al sistema se debe colocar el usuario y la contraseña y presionar el botón “Entrar”. Si los datos ingresados por el usuario no son los correctos el sistema devuelve el siguiente mensaje: “Nombre de usuario o contraseña incorrectos”. Figura 2.

Autenticación

Figura 2. Autenticación.
Elaborado por: Francisco Bonilla y Miguel Machasilla

En caso de no ingresar algún dato de autentificación el sistema devuelve los siguientes mensajes “El campo de usuario es requerido” y “El campo de contraseña es requerido”. Figura 3.

Error De Autenticación

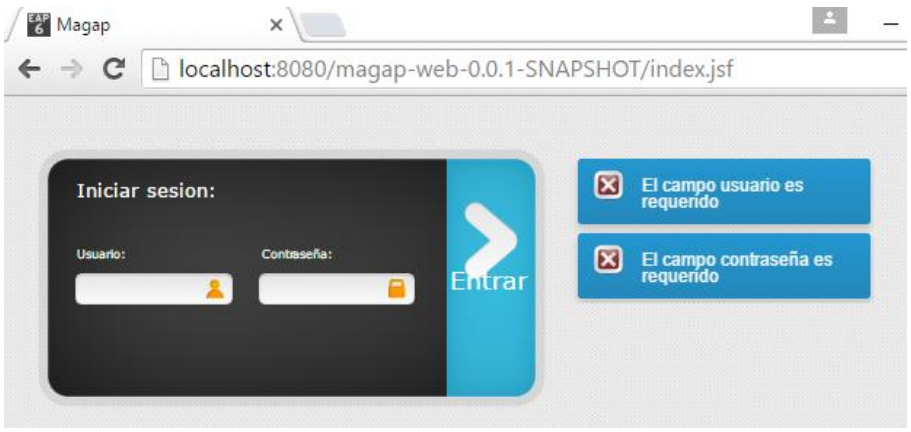


Figura 3. Error De Autenticación.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Si los datos ingresados de autentificación son correctos, el usuario ingresa al sistema visualizando las opciones que fueron asignadas a su perfil.

Funcionalidades del sistema

De acuerdo con el perfil de usuario (Administrador o Técnico), el sistema permite realizar las siguientes funcionalidades.

Usuarios con perfil administrador

El usuario administrador, tiene acceso a toda la paramtrización del sistema, usuarios, perfiles, menús, catálogos y bitácora de utilización que permiten el funcionamiento óptimo del sistema.

Una vez que se ingresa con el perfil administrador se presenta la pantalla principal del sistema, la cual se divide en: Cabecera, menú y cuerpo. Figura 4

Interfaz Gráfica

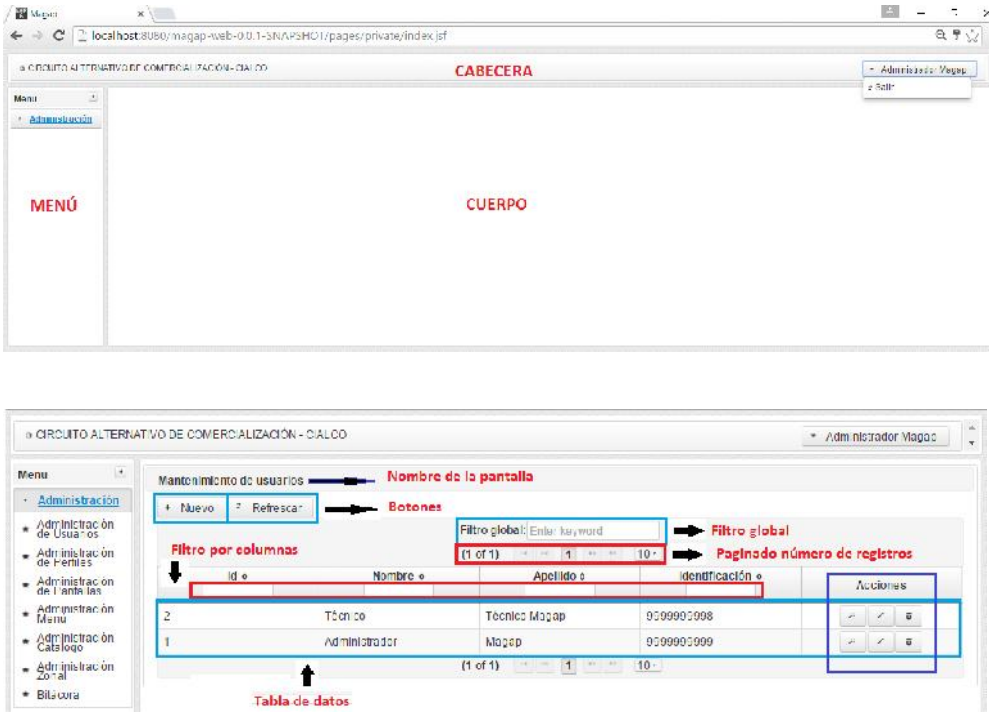


Figura 4. Interfaz Gráfica.
Elaborado por: Francisco Bonilla y Miguel Machasilla

En la **cabecera** se encuentra el nombre del sistema y el usuario que actualmente está utilizando sus diferentes opciones.

En el **Menú** se despliegan las opciones a las cuales el usuario tiene acceso.

En el **Cuerpo** se despliega el contenido de cada una de las opciones del menú.

Al dar clic en una de las opciones del menú se despliega la pantalla de consulta la cual maneja un estándar de presentación.

En la figura anterior se visualiza el nombre de la pantalla, los botones de creación de nuevo registro “Nuevo” y de refrescar “Refrescar”, la tabla de datos la cual contiene un filtro global, un paginado que controla el número de registros máximos que se quiere que aparezca por página, donde se puede filtrar por cada una de las columnas y por cada registro que se tiene en la tabla, con acciones de búsqueda, edición e inhabilitar.

Para crear un nuevo usuario se ingresa a la opción “Administración de Usuario”, se da clic en el botón “Nuevo” y se llena la siguiente información.

Se debe tener en cuenta que donde se encuentre asterisco * se debe llenar obligatoriamente la información.

En la segmento Datos del usuario en “Perfile” permite escoger si va ser Técnico o Administrador, “Zona” es el grupo de provincias al que va a pertenecer el usuario, como se aprecia en la siguiente tabla.

Tabla 1. Zonas

ZONAS	PROVINCIAS
ZONA GENERAL	Todas las provincias a nivel nacional
ZONA 1	Carchi, Esmeraldas, Imbabura, Sucumbíos.
ZONA 2	Napo, Pichincha, Orellana.
ZONA 3	Cotopaxi, Chimborazo, Pastaza, Tungurahua.
ZONA 4	Manabí, Santo Domingo de los Tsáchilas.
ZONA 5	Bolívar, Guayas, Los Ríos, Santa Elena.

ZONA 6	Azuay, Cañar, Morona Santiago.
ZONA 7	El Oro, Loja, Zamora Chinchipe.
ZONA GALAPAGOS	Galápagos

Nota: Se muestra las zonas de trabajo del técnico
Fuente: Francisco Bonilla y Miguel Machasilla

Los “Permisos” pueden ser crear, consultar, editar o eliminar de acuerdo a la función que realice el nuevo usuario dentro del sistema.

Para asignarle un perfil al nuevo usuario se ingresa a la opción “Administración de Perfiles”, se da clic en el botón “Nuevo” y se llena la siguiente información. Figura 5.

Administración de Perfiles

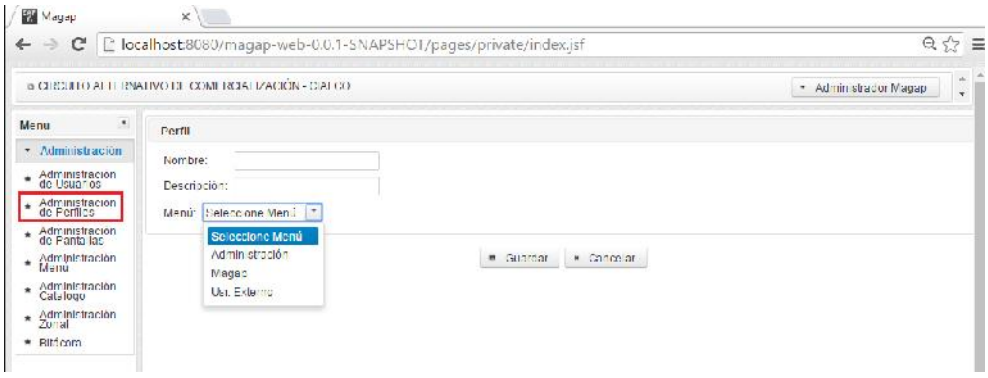


Figura 5. Administración de Perfiles.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Para asignar pantallas al nuevo usuario se ingresa a la opción “Administración de Pantallas”, se da clic en el botón “Nuevo” y se llena la siguiente información. Figura 6.

Administración de Pantallas

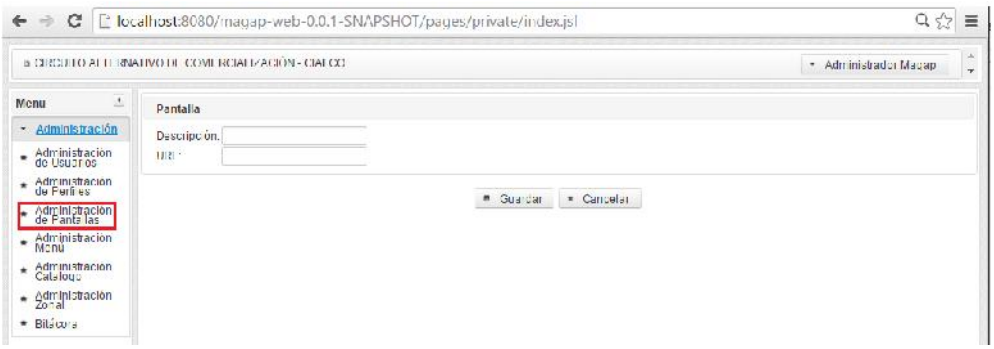


Figura 6. Administración de Pantallas.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Para asignar un nuevo menú al nuevo usuario se ingresa a la opción “Administración de Menú”, se da clic en el botón “Nuevo” y se llena la siguiente información. Figura7.

Administración de Menú

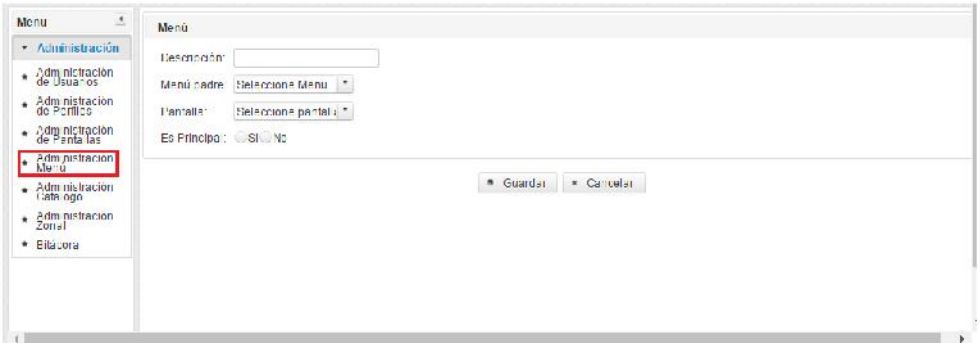


Figura 7. Administración de Menú.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Para asignar nuevos catálogos al usuario se ingresa a la opción “Administración de Catálogo”, se da clic en el botón “Nuevo” y se llena la siguiente información. Figura8.

Administración de Catálogo

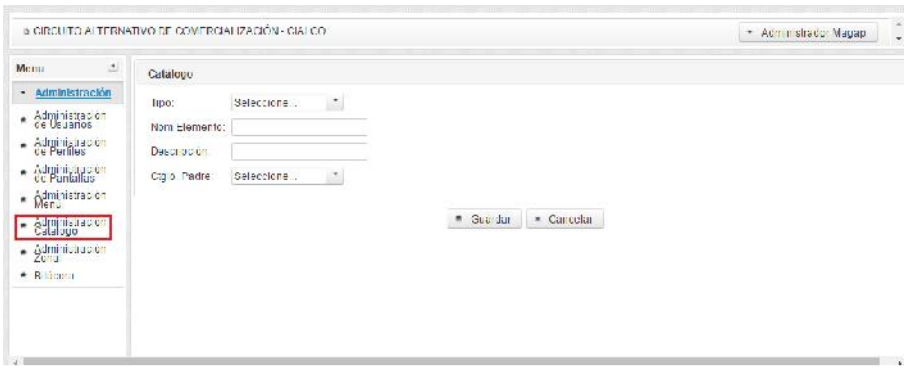


Figura 8. Administración de Catálogo.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Para asignar una nueva zona se ingresa a la opción “Administración Zonal”, se da clic en el botón “Nuevo” y se llena la siguiente información. Figura 9.

Administración Zonal

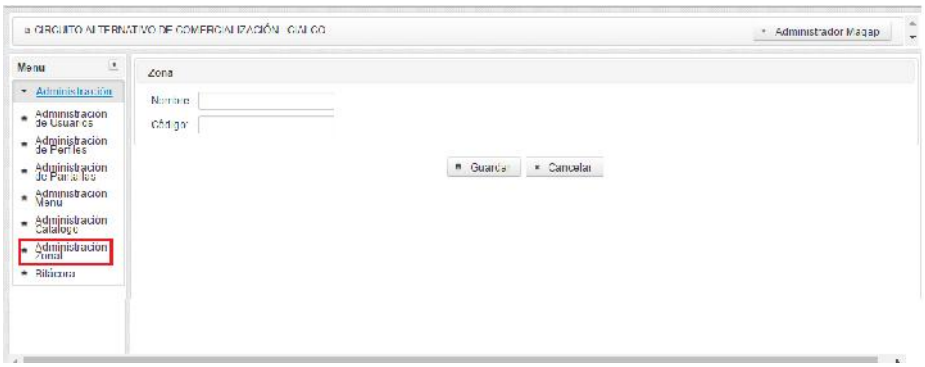


Figura 9. Administración Zonal.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Mediante la opción “Bitácora”, el administrador puede obtener los de utilización del sistema. Figura 10.

Bitácora

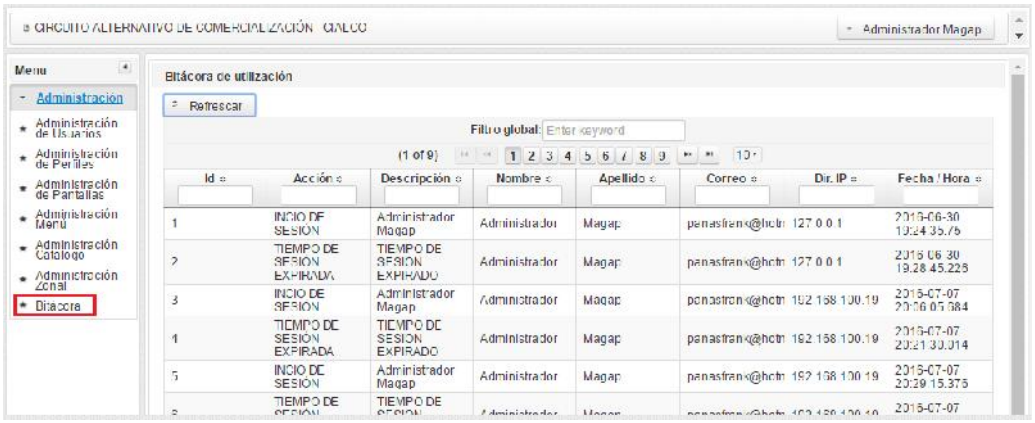


Figura 10. Bitácora.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Usuarios con perfil técnico

El usuario técnico una vez que ingresa al sistema con el perfil definido por el administrador tiene la capacidad de aplicar cambios al contenido, ya sea ingresar, modificar, consultar y eliminar información. Figura 11.

Pantalla usuario técnico

Figura 11. Pantalla usuario técnico.
Elaborado por: Francisco Bonilla y Miguel Machasilla

Anexo 2. Certificado de Aceptación

Ministerio
de **Agricultura, Ganadería,**
Acuacultura y Pesca

Av. Eloy Alfaro 30-350 y Amazonas
Telf. (593) 2 3960 100 / 3960 200
www.magap.gob.ec
Quito - Ecuador

Quito, 29 de Julio del 2016

CERTIFICADO

Por medio de la presente yo, ERIKA ALEXANDRA ZARATE BACA con número de cédula 1718974866, como Coordinadora General de Redes Comerciales – Subrogante - del MAGAP a cargo de la gestión de los CIRCUITOS ALTERNATIVOS DE COMERCIALIZACIÓN – CIALCO, certifico que el proyecto: DESARROLLO DEL PROYECTO CIRCUITO ALTERNATIVO DE COMERCIALIZACIÓN – CIALCO MÓDULO DE ADMINISTRACIÓN DEL PORTAL Y BITÁCORA DE UTILIZACIÓN, realizado por los estudiantes **Francisco Javier Bonilla Larco** con número de cédula 1718388893 y **Miguel Angel Machasilla Nicolalde** con número de cédula 1716598634, se encuentra finalizado, probado, aprobado y entregado.

Los estudiantes pueden hacer uso del presente, a su convenio.

Para constancia firma

Ministerio
de Agricultura, Ganadería,
Acuacultura y Pesca
Coordinación General de Redes Comerciales

ERIKA ZARATE BACA

COORDINADORA GENERAL DE REDES COMERCIALES – SUBROGANTE

MINISTERIO DE AGRICULTURA, GANADERÍA, ACUACULTURA Y PESCA

C.C 1718974866