UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:

INGENIERÍA DE SISTEMAS

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

TEMA:
ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN EN ANDROID PARA DISPOSITIVOS MÓVILES, DE MENSAJERÍA, CONSULTAS Y NOTIFICACIONES PARA LA FEDERACIÓN DE ESTUDIANTES DE LA CARRERA DE SISTEMAS DE LA UNIVERSIDAD POLITÉCNICA SALESIANA CAMPUS SUR

AUTORES:
ANDREA PAULINA MARTÍNEZ INCA
MICHAEL VICENTE FLORES SANMARTÍN

DIRECTOR:
ALONSO RENÉ ARÉVALO CAMPOS

Quito, junio de 2015

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE TITULACIÓN

Nosotros, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

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

Quito, junio de 2015.

_____	_____
Andrea Paulina Martínez Inca	Michael Vicente Flores Sanmartín
C.C. 0604018671	C.C. 1722795414

DEDICATORIA

Dedico todo este trabajo sobre todo a Dios y a mi ángel que siempre me acompañado en este esfuerzo y sé que siempre lo harás, te lo dedico a ti Naincho. También a mi familia que es mi pilar fundamental para siempre seguir adelante en especial se lo dedico a mi Tuti y mi mami por todos sus esfuerzos que día a día hacen por verme feliz y a todas mis hermanas y mi hermano por su fiel apoyo y confianza en mí, y sin duda alguna también dedicada a ti Michael ya que a pesar de todas las dificultades ahora podemos compartir juntos esta meta culminada.

Andrea Paulina Martínez Inca

Yo dedico el resultado del presente trabajo a Dios, a mi madre Teresa Sanmartín, a mis dos hermanas Gabriela Flores y Pilar Flores por todo el apoyo que me brindaron tanto en las jornadas de desvelo, como en todo momento, nunca existirán las palabras que describan mi gratitud por ustedes.

A las personas que me apoyaron y me brindaron su comprensión y paciencia en los momentos difíciles esto es por ustedes.

Michael Vicente Flores Sanmartín

AGRADECIMIENTOS

A la Universidad Politécnica Salesiana por aportar con todo lo que necesitábamos en el desarrollo de la tesis, también un agradecimiento muy especial nuestro tutor el Ingeniero Rene Arévalo quien con toda su ayuda y conocimientos nos ayudó a concluir este trabajo, y sin duda alguna a todos los profesores que intervinieron en este proceso estudiantil durante todos estos años de esfuerzo.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1 3

ANÁLISIS DE CONTEXTO 3

1.1 Justificación 3

1.2 Formulación del problema 3

1.3 Importancia del desarrollo de la aplicación Android denominada “UPSY” 4

1.4 Objetivos 5

1.4.1 Objetivo general 5

1.4.2 Objetivos específicos 5

1.5 Alcance 5

1.6 Metodología utilizada para el desarrollo de UPSY 8

1.6.1 Valores de XP 9

1.6.2 Practicas XP 11

CAPÍTULO 2 13

MARCO TEÓRICO 13

2.1 Google Android 13

2.1.1 Historia 13

2.2 Base de datos SQLITE 16

2.3 SDK Manager 17

2.4 Programación extrema (XP) 17

2.4.1 El Proceso XP 18

2.5 Lenguaje Unificado de Modelado (UML) 22

2.5.1 Diagramas UML 22

2.5.1.1 Diagrama de clases 22

2.5.1.2 Diagrama de objetos 23

2.5.1.3 Diagrama de casos de uso 24

2.5.1.4 Diagrama de estados 24
2.5.1.5 Diagrama de secuencia 25
2.5.1.6 Diagrama de despliegue 25
2.6 Servicios Web 26
CAPÍTULO 3 30
ANÁLISIS DEL SISTEMA Y DISEÑO 30
3.1 Especificación de requisitos del software 30
3.1.1 Definición de roles de usuario 30
3.1.2 Levantamiento de historias de usuario 30
3.1.3 Tarjetas CRC (Clase Responsabilidades Colaboradores) 35
3.2 Diseño del Sistema 37
3.2.1 Diseño de la base de datos 37
3.2.1.1 Diseño conceptual de las base de datos 38
3.2.1.2 Diseño físico de las bases de datos 41
3.2.1.3 Diccionario de la base de datos del servidor 44
3.2.1.4 Diccionario de la base de datos de la aplicación Android “UPSY” 48
3.2.2 Diagrama de clases del proyecto 50
3.2.2.1 Diagrama de clases de aplicación Android 50
3.2.2.2 Diagrama de clases de los servicios web 52
3.2.2.3 Diagrama de clases del portal web 53
3.2.3 Diccionario de clases 54
3.2.3.1 Diccionario de clases aplicación Android 54
3.2.3.2 Diccionario de clases servicios web 71
3.2.4 Diagrama de casos de uso 73
3.2.4.1 Diagramas de caso de uso del servidor web 74
3.2.4.2 Diagramas de caso de uso del cliente Android 76
3.2.5 Diagrama de secuencia 82

3.2.5.1 Diagramas de secuencia servidor web 82
3.2.5.2 Diagramas de secuencia aplicación Android 84
3.2.6 Diagrama de componentes 87
3.2.7 Diagrama de despliegue 88
CAPÍTULO 4 89
4.1 Introducción etapa de pruebas 89
4.2 Tipos de pruebas realizadas 89
4.2.1 Hardware para pruebas 89
4.2.2 Pruebas de funcionalidad 91
4.2.3 Pruebas de tiempo de respuesta 97
4.2.3.1 Herramientas utilizadas 97
4.2.3.2 Pruebas de estrés 99
4.2.4 Pruebas de compatibilidad 103
CONCLUSIONES 105
RECOMENDACIONES 106
LISTAS DE REFERENCIAS 107

ÍNDICE DE TABLAS

Tabla 1. *Prácticas y reglas XP* 12
Tabla 2. *Versiones de Android* 15
Tabla 3. *Roles y requisitos de Usuario del Sistema* 30
Tabla 4. *Historia de Usuario Gestión de Registro de usuario* 31
Tabla 5. *Historia de Usuario Gestión de Notificaciones* 31
Tabla 6. *Historia de usuario gestión de chat* 32
Tabla 7. *Historia de usuario sincronización con el servidor* 33
Tabla 8. *Historia de usuario acceso al portal web* 33
Tabla 9. *Historia de usuario gestión de registro y envío de notificaciones* 34
Tabla 10. *Tarjeta CRC clase usuario* 35
Tabla 11. *Tarjeta CRC clase contacto* 35
Tabla 12. *Tarjeta CRC clase chat* 36
Tabla 13. *Tarjeta CRC clase notificación* 36
Tabla 14. *Tarjeta CRC clase registro y envío de notificaciones* 37
Tabla 15. *Tabla notificación de la base de datos PostgresSql* 44
Tabla 16. *Tabla tipo_notificacion de la base de datos PostgresSql* 44
Tabla 17. *Tabla Usuario_Notificación de la base de datos PostgresSql* 45
Tabla 18. *Tabla Usuario de la base de datos PostgresSql* 45
Tabla 19. *Tabla Envia_Recibe de la base de datos PostgresSql* 46
Tabla 20. *Tabla chat de la base de datos PostgresSql* 46
Tabla 21. *Tabla tipo_usuario de la base de datos PostgresSql* 46
Tabla 22. *Tabla Carrera de la base de datos PostgresSql* 47
Tabla 23. *Tabla Género de la base de datos PostgresSql* 47
Tabla 24. *Tabla Contacto de la base de datos PostgresSql* 47
Tabla 25. *Tabla Carrera de la base de datos SQLite* 48
Tabla 26. *Tabla Sexo de la base de datos SQLite* 48
Tabla 27. *Tabla Contacto de la base de datos SQLite* 48
Tabla 28. *Tabla Chat de la base de datos SQLite* 49
Tabla 29. *Tabla Perfil_Usuario de la base de datos SQLite* 49
Tabla 30. *Tabla Notificación de la base de datos SQLite* 50
Tabla 31. *Tabla Tipo_Notificación de la base de datos SQLite* 50
Tabla 32. *Clase CambiaPass* 54

Tabla 33. *Clase Carátula* 55
Tabla 34. *Clase Confirmapass* 55
Tabla 35. *Clase Guarda* 56
Tabla 36. *Clase Recupera* 57
Tabla 37. *Clase Registro* 58
Tabla 38. *Clase BuscarContacto* 59
Tabla 39. *Clase Cargadatos* 60
Tabla 40. *Clase Chat* 61
Tabla 41. *Clase Configuración* 62
Tabla 42. *Clase Contacto* 62
Tabla 43. *Clase DetalleContacto* 63
Tabla 44. *Clase DetalleNotificacion* 63
Tabla 45. *Clase ExpandableListAdapter* 64
Tabla 46. *Clase Listabusqueda* 65
Tabla 47. *Clase Notificacion* 66
Tabla 48. *Clase Perfil* 67
Tabla 49. *Clase SQLite* 68
Tabla 50. *Clase SQLiteHelper* 69
Tabla 51. *Clase TestServicio* 70
Tabla 52. *Clase WebService* 71
Tabla 53. *Clase WebServiceupsy* 72
Tabla 54. *Clase conectar* 72
Tabla 55. *Clase sentenciasSql* 73
Tabla 56.*Características equipo de pruebas Android* 90
Tabla 57. *Características equipo de pruebas del servidor* 90
Tabla 58. *Pruebas acceso al sistema 1* 92
Tabla 59. *Pruebas acceso al sistema 2* 93
Tabla 60. *Pruebas acceso al sistema 3* 94
Tabla 61. *Pruebas de interfaz de notificaciones* 94
Tabla 62. *Pruebas de interfaz de contactos 1* 95
Tabla 63. *Pruebas de interfaz de contactos 2* 95
Tabla 64. *Pruebas de interfaz de contactos 3* 96
Tabla 65. *Pruebas de interfaz de conversación* 96
Tabla 66. *Pruebas de compatibilidad* 103

ÍNDICE DE FIGURAS

Figura 1. Valores de la metodología XP 9
Figura 2. Línea de desarrollo Android 14
Figura 3. Ejemplo creación de base de datos SQLite 16
Figura 4. Pantalla de Android SDK Manager. 17
Figura 5. Muestra las tareas dentro del proceso XP. 18
Figura 6. Descripción de las etapas del proceso XP 21
Figura 7. Representación de un diagrama de clase 23
Figura 8. Descripción de un diagrama de objetos 23
Figura 9. Diagrama de casos de uso 24
Figura 10. Representación de un diagrama de estados 24
Figura 11. Descripción de un diagrama de secuencia 25
Figura 12. Ejemplo de un diagrama de despliegue 26
Figura 13. Ejemplo de uso de servicios web 27
Figura 14. Arquitectura básica de protocolos de servicios web 29
Figura 15. Diseño conceptual de la base de datos SQLite 39
Figura 16. Diseño conceptual de la base de datos en PostgresSql 40
Figura 17. Diseño físico de la base de datos SQLite 42
Figura 18. Diseño físico de la base de datos en PostgresSql 43
Figura 19. Diagrama de clases aplicación Android "UPSY" 51
Figura 20. Diagrama de clases de los servicios web 52
Figura 21. Diagrama de clases del portal web 53
Figura 22. La figura muestra el diagrama de caso de uso de acceso al portal web ... 74
Figura 23. La figura muestra el caso de uso de la gestión de 75
Figura 24. La figura muestra el acceso a la aplicación Android 76
Figura 25. Caso de uso consultar notificaciones 78
Figura 26. Muestra el caso de uso del chat dentro de la aplicación Android 79
Figura 27. Muestra el caso de uso en el que el usuario realiza una conversación 81
Figura 28. Diagrama de secuencia acceso al portal web 83
Figura 29. Diagrama de secuencia de la gestión de notificaciones 83
Figura 30. Diagrama de secuencia del ingreso a la aplicación Android 84
Figura 31. Diagrama de secuencia consulta de notificaciones 85
Figura 32. Diagrama de secuencia del chat 86

Figura 33. Diagrama de secuencia de conversaciones 87
Figura 34. Diagrama de componentes 88
Figura 35. Muestra el diagrama de despliegue de toda la aplicación Android desarrollada. 88
Figura 36. Aplicación Jmeter 98
Figura 37. Aplicación SoapUI 98
Figura 38. Gráfico en Jmeter de pruebas de rendimiento 101
Figura 39. Muestra los resultados de los tiempos de respuesta del WebService ... 102

RESUMEN

La presente tesis, es el resultado de la investigación sobre la estructura y metodología de desarrollo XP (Extreme Programing), aplicada sobre el Sistema Operativo Android y sus correspondientes APIs (Interfaz de programación de aplicaciones), permitiéndonos desarrollar una aplicación Android para dispositivos móviles que tenga conexión de red inalámbrica y que se encuentren dentro de las instalaciones de la Universidad Politécnica Salesiana (UPS) Campus Sur, con funciones de: mensajería, consultas y notificaciones. Permitiendo la comunicación entre los usuarios registrados a atreves de un módulo de chat, la recepción de notificaciones emitidas por el usuario administrador mediante un portal web, mismo que será responsable de la publicación y emisión de las notificaciones de eventos dentro de la UPS, facilitando el acceso a la información a los estudiantes.

ABSTRAC

This thesis is the result of research on the structure and development methodology XP (Extreme Programing), applied on the Android operating system and their corresponding APIs (application programming interface), allowing us to develop an Android application for mobile devices has wireless network connection and are within the premises of the Salesiana Polytechnic University (UPS) South Campus, with functions: messaging, consultations and notifications. Allowing communication between registered to dare a chat module users receive notifications issued by the administrative user through a web portal, it will be responsible for publishing and broadcast event notifications within the UPS, facilitating access to information to students.

INTRODUCCIÓN

Las aplicaciones para dispositivos móviles como: Smartphone (teléfonos inteligentes), tablets y otros, han obtenido una mayor demanda gracias a la aparición de dispositivos con mayores características tecnológicas, que nos brindan la posibilidad de desarrollar aplicaciones cada vez más grandes, cuyas ventajas facilitan el acceso a información, a la comunicación y demás facilidades para el usuario.

Considerando el antecedente anterior se desarrolló una aplicación Android denominada UPSY para teléfonos inteligentes, que está compuesta por tres componentes: un portal web para uso del usuario administrador, una aplicación Android que se instala dentro del teléfono inteligente y una colección de servicios web que permiten la interacción de los componentes de la aplicación Android.

UPSY contiene una interfaz de usuario con acceso a módulos de chat y notificaciones. Dentro del módulo de chat se encuentra la gestión de contactos para el envío de solicitudes, añadir nuevos contactos, recibir solicitudes y de esta manera poder iniciar una sesión de chat con los contactos confirmados. Por otro lado el módulo de notificaciones permite al usuario recibir notificaciones que son enviadas y actualizadas por el usuario administrador desde el portal web de notificaciones, dichas notificaciones publicadas se encuentra divididas en: información general, eventos estudiantiles, gestión FEUPS y noticias institucionales, las cuales se sincronizan con la aplicación Android por medio de los Servicios web, lo que posibilita al usuario poder revisar las notificaciones publicadas por el usuario administrador en tiempo real.

Para brindar las funcionalidades menciona la aplicación Android trabaja en conjunto con un servidor paralelo que contiene: una base de datos Postgres, una aplicación web para el administrador y un conjunto de Servicios web (Servicios web). Demostrando algunas de las posibilidades de Android como son: bases de datos embebidas (SQLITE), manejo de tareas asíncronas, métodos de notificación, consumo de recursos HTTP (Protocolo de Transferencia de Hipertexto) por medio de Apache Commons HTTP Client embebidos y demás características que están a disposición del desarrollador.

El capítulo uno, está compuesto de la justificación del desarrollo de la aplicación Android, los objetivos, formulación del problema, la metodología de desarrollo que se utilizó y el alcance de la aplicación Android junto con una descripción de los módulos que integran a UPSY.

El capítulo dos, describe el desarrollo del marco teórico sobre: el sistema operativo Android y sus antecedentes, la metodología XP (Extreme Programming) que se utiliza para el diseño de la aplicación Android, el funcionamiento de los Servicios web, el lenguaje UML y los diagramas que serán utilizados en el proceso del desarrollo.

El capítulo tres, contiene todo el análisis de la aplicación Android detallando los requerimientos de software necesarios para el funcionamiento de la aplicación, se detallan los perfiles de usuario, levantamiento de tarjetas CRC (Clase, responsabilidades y colaboradores), se desarrolla el diseño del modelamiento de la base de datos del servidor web y de la base de datos de la aplicación Android, los diagramas que se usaron para detallar el funcionamiento de la aplicación.

El capítulo cuatro, se detalla la etapa de pruebas que se elaboran mediante un plan de pruebas, para verificar el comportamiento y la funcionalidad de la aplicación Android, finalmente se describe la fase de implementación determinado los requisitos mínimos para el funcionamiento tanto del servidor como de los teléfonos inteligentes donde se alojara la aplicación Android.

Concluimos señalando que el presente proyecto ha cumplido con el objetivo principal permitiendo el envío y recepción de mensajes entre estudiantes, el envío de notificación por el usuario administrador y la recepción de las mismas en la aplicación Android.

CAPÍTULO 1

ANÁLISIS DE CONTEXTO

1.1 Justificación

El propósito de esta aplicación denominada “UPSY” está orientado a brindar una nueva forma de difusión y comunicación de temas relacionados a la gestión de la Federación de estudiantes de la Universidad Politécnica Salesiana (FEUPS), noticias, eventos y demás actividades de apoyo a los estudiantes de la carrera de sistemas de la Universidad Politécnica Salesiana (UPS), dicha aplicación se desarrolló en Android aprovechando el actual auge de acceso a teléfonos inteligentes de los estudiantes de dicha carrera.

De esta manera UPSY facilita la comunicación de los estudiantes y la FEUPS, mejorando el acceso a la difusión de eventos, notificaciones institucionales y demás temas de interés para los estudiantes. A través de esta herramienta, se notificará sobre: reuniones, noticias, comentarios, sugerencias y demás eventos suscitados dentro de la UPS o que sean de importancia para la atención de los estudiantes, así también se permitirá la difusión de información e intercomunicación fácil y accesible de los estudiantes o miembros del Consejo de Carrera, por medio de un módulo de chat.

1.2 Formulación del problema

La Federación de Estudiantes de la Universidad Politécnica Salesiana (FEUPS), es un organismo autónomo de representación estudiantil y de participación en los diferentes niveles de gestión y animación de la Universidad, tanto a nivel nacional como dentro de las Sedes y Campus existentes. Esta se acoge a la Ley Orgánica de Educación Superior, Reglamento de Régimen Académico, a la Normativa de la Universidad Politécnica Salesiana (UPS) y al Estatuto de la Federación de Estudiantes. (Universidad Politécnica Salesiana, 2011)

La FEUPS se encuentra conformada por: El Presidente, Vicepresidente, Secretario y Tesorero, quienes conforman el organismo máximo de representación estudiantil de la UPS de las diferentes carreras. La FEUPS dentro de su funciones como se menciona en el Estatuto de la Federación de Estudiantes es la encargada de analizar

el desarrollo académico y administrativo de la carrera y proponer mejoras si se considera necesario, para lo cual La Asamblea de Carrera se reunirá ordinariamente una vez al mes.

Los representantes estudiantiles son los voceros oficiales de su carrera, ante las demás instancias de la Federación de Estudiantes y autoridades universitarias. Por tal motivo, debe prevalecer la colaboración de los representantes de curso, es decir, mantenerse siempre informados de las gestiones que realiza la FEUPS y de la misma manera receptar las opiniones de sus representados.

En la fase diagnóstica de este proyecto, se evidenció una deficiencia en los procesos de comunicación entre la FEUPS y los estudiantes, ya que estos últimos, no se encuentran totalmente informados sobre: la Gestión de la FEUPS, eventos, cursos u otras actividades que son de importancia para los estudiantes esto se produce por varios motivos, como por ejemplo: falta de medios de comunicación adecuados, falta de tiempo de los estudiantes, des interés de los estudiantes u otros factores que influyen en este proceso.

Este inconveniente dificulta el rol de intermediario que cumple la FEUPS para transmitir las opiniones de los estudiantes a las autoridades de la UPS.

1.3 Importancia del desarrollo de la aplicación Android denominada “UPSY”

Las aplicaciones para dispositivos móviles como: smartphone (teléfonos inteligentes), tablets y otros, han obtenido una mayor demanda gracias a la aparición de dispositivos con mayores características tecnológicas como: giroscopios, sensores de luz, sensores de altitud, memoria RAM, CPU, pantallas con mayor tamaño, geo localización, etc. Estas características permiten un abanico de posibilidades para dichos dispositivos que tienen como base un Sistema Operativo Android de tipo abierto, esto significa que toda la información del funcionamiento se encuentra publicado para el público en general, esta es una gran ventaja al momento del desarrollo ya que se encuentran disponibles muchas fuentes de consulta que facilitan el trabajo, pero aunque existen estas ventajas hay que considerar que el desarrollo de las aplicaciones Android es un mundo relativamente nuevo en nuestro país donde no se encuentra muchos desarrolladores cualificados sobre el manejo y desarrollo de las mismas.

Otra venta de los dispositivos móviles que trabajan con el sistema operativo Android es su portabilidad, la que permite la transportación de la aplicación junto con el usuario, esto lo transforma al dispositivo de un carácter personal y de fácil acceso a la información contenida dentro del mismo.

Por tal motivo UPSY ofrece la posibilidad de comunicarse de forma gratuita con un listado de contactos elegidos y confirmados por el estudiante, paralelamente se muestran notificaciones institucionales de eventos y demás actividades que permiten que el estudiante se involucre con la FEUPS, lo que generara una mayor participación por parte de los estudiantes.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar e implementar una aplicación, en Android para dispositivos móviles, para manejo de: mensajería, consultas y notificaciones de la Federación de Estudiantes de la Carrera de Sistemas de la Universidad Politécnica Salesiana Campus Sur.

1.4.2 Objetivos específicos

- Presentar una aplicación que funcione en dispositivos móviles que trabajen con SO Android
- Implementar la aplicación para dispositivos móviles para uso de la federación de estudiantes y estudiantes de la facultad de sistemas.
- Desarrollar una aplicación de mensajería, chat, notificaciones, e información de la federación que trabaje en las instalaciones de la Universidad Politécnica Salesiana Campus Sur.

1.5 Alcance

En el presente proyecto está comprendido desde el análisis, diseño, desarrollo e implementación de una aplicación para dispositivos móviles que trabajen bajo el sistema operativo Android.

Para el desarrollo de esta propuesta tecnológica de software, se utilizó la metodología XP como marco base en todas las etapas del desarrollo, con la cual se pretendió

elaborar una aplicación Android que permita las características que se detallan a continuación.

De igual forma no se usó información proveniente de las bases de datos de la UPS. La base de datos con la que se trabajó, son de uso único y exclusivo de la herramienta. Se utilizarán dos tipos de bases de datos: una de tipo interna para dispositivos móviles y otra externa, denominada sistema gestor de base de datos que contendrá la información global. Estas dos bases de datos trabajarán de manera sincronizada para el buen funcionamiento de la aplicación.

Aplicación cliente Android

La aplicación Android se desarrollado en eclipse con el SDK Android y con base de datos SQLITE embebido en el sistema operativo Android para Smartphone.

- **Registro**

  Permite la creación de usuarios en los dispositivos móviles, solicitando datos de registro para la creación del perfil.

  Establece la comunicación con el servidor de base de datos para su registro y posterior obtención de una clave única dentro de la aplicación.

- **Seguridad**

  Validará la información de usuario y contraseña registrado en el servidor para su posterior ingreso a la aplicación, las contraseñas son encriptadas para precautelar la información del usuario, el método de encriptación que se uso es MD5.

- **Notificaciones**

  Es la pantalla principal donde se aprecian las notificaciones de todos los eventos, dentro de la Universidad Politécnica Salesiana Campus Sur según el cátalo de notificaciones, la cual contiene un listado de eventos separados por tipo según el catálogo de notificaciones anteriormente mencionado, con un contador de notificaciones pendientes; Las listas son de tipo desplegable para mostrando los eventos, siendo vínculos al detalla del evento seleccionado.

- **Contacto**

  Permite visualizar un listado de contactos que fueron vincularon exitosamente que usen el servicio de chat y notificación.

Seleccionando el contacto nos direcciona al chat con el contacto seleccionado donde se establece una comunicación de envío y recepción de mensajes de texto con el contacto seleccionado

- **Búsqueda de contacto**

  Nos permite buscar contactos dentro del servidor externo de base de datos que usen la aplicación de chat y notificaciones; esta búsqueda se realizara eligiendo entre varios criterios de búsqueda para encontrar con mayor facilidad al contacto, después de elegir los criterios de búsqueda se realizara la búsqueda que devolverá las coincidencias de búsqueda para que el usuario seleccione el contacto buscado para establecer la conversación, la misma que se entablara después de que el contacto encontrado reciba y acepte una solicitud de vinculo, caso contrario no se podrá realizar el emparejamiento de contacto.

- **Sincronización**

  Es el responsable de la comunicación entre el smartphone y el servidor de base de datos a través de servicios web. Se encarga de la búsqueda de actualizaciones de las notificaciones publicadas por parte del administrador, que son en formato de texto.

  Se encarga del envío y recepción de mensajes de texto entre el emisor y el receptor de mensajes.

- **Recuperación de contraseña**

  Permite recuperar la contraseña en el caso de pérdida u olvido de la misma, para ello se colocan los datos de identificación de usuario para verificar la información y resetear la contraseña.

Aplicaciones del Servidor

Consta con un servidor web Apache que aloja una aplicación para uso exclusivo del usuario administrador y una colección de servicios web, con una base de datos PostgresSql, sincronizada con la base de datos SQLite que se encuentra dentro del dispositivo móvil Android. El desarrollo estará realizado en Eclipse Juno junto con el SDK Android

- **Acceso al servidor**

Valida el acceso a la aplicación web, verificando los datos en la base de datos principal de la aplicación.

- **Publicación de notificaciones**

  Permite ingresar las notificaciones seleccionando el tipo al que corresponde; después de la revisión de la notificación se procederá a la difusión general por la Carrera seleccionada.

- **Reportes**

  Permite revisar un resumen de las actividades dentro de la aplicación como por ejemplo:

  - El uso de la aplicación.- se considerando la transmisión de datos entre usuarios sean Administradores o estudiantes, recepción y lectura de notificaciones, para determinar si es de utilidad la misma.
  - Detalle de publicaciones.- muestra un listado de las notificaciones publicadas en orden histórico
  - Usuarios activos.- es un detalle de los usuarios que en un periodo mínimo de tres meses atrás, han usado la aplicación en forma continua.

1.6 Metodología utilizada para el desarrollo de UPSY

La metodología utilizada en el diseño y desarrollo de la aplicación móvil para la FEUPS se denomina Extreme Programing (XP) o programación extrema que involucra el desarrollo ligero (o ágil) de aplicaciones de software, basada en una serie de valores y de buenas prácticas, que busca aumentar la productividad a la hora de desarrollar programas.

Este modelo de programación, se basa en una serie de metodologías de desarrollo de software, en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.

Una de las características principales de este método de programación, es que centra sus prioridades en las personas y no en los procesos. Los autores de XP han seleccionado las mejores prácticas para el desarrollo de aplicaciones, han profundizado en sus relaciones y en cómo se refuerzan las unas con las otras. El resultado de esta selección ha sido esta metodología única y compacta, por esto,

aunque no está basada en principios nuevos, constituye una nueva manera de ver el desarrollo de software.

El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos, aplicando el sentido común. (Beck, 2012)

1.6.1 Valores de XP

La metodología XP no es un conjunto de reglas estrictas a seguir, sino una forma de trabajo que busca una armonía entre los valores personales y los organizacionales, esta metodología tiene como punto de partida cinco valores fundamentales que se observan en la figura 1.

Cinco valores como está constituido XP

Figura 1. Valores de la metodología XP
Fuente: (Wells, 2013)

Descripción de los valores de la metodología XP

En la figura 1 expuesta se muestran los cinco valores, mismos que pasamos a explicar a continuación.

- **Comunicación:** todos son parte del equipo y nos comunicamos cara a cara todos los días. Trabajamos juntos en todo, desde los requerimientos hasta la programación. En equipo crearemos la mejor solución al problema. (Wells, 2013).

Esto nos indica que la comunicación que presenta esta metodología es de manera más informal entre los desarrolladores así como con el cliente dentro del desarrollo del proyecto.

- **Simplicidad**: la simplicidad implica que: "Desarrollaremos lo que sea solicitado y necesario, pero no más que eso. De esa forma, se maximiza el valor de la inversión realizada. Nos dirigiremos a nuestro objetivo a pasos simples y pequeños, mitigando las fallas a medida que ocurran. Crearemos algo de lo cual podamos sentirnos orgullos y que pueda mantenerse en el largo plazo a costos razonables" (Wells, 2013)
- **Coraje:** diremos la verdad en nuestros avances y estimados, no documentaremos excusas para el fracaso, pues planificamos para tener éxito. No tendremos miedo a nada pues sabemos que nadie trabaja solo. Nos adaptaremos a los cambios cuando sea que estos ocurran. (Wells, 2013).

  Este valor dentro de XP nos ayuda o tomar decisiones en beneficio del proyecto es decir los cambios que deber hacer dentro del código como la documentación sin importar el tiempo invertido lo que nos anima a eliminar lo que no sirve esto es lo que ayudará a implementar más fácilmente los cambios a futuro.
- **Respeto:** todos en el equipo dan y reciben el respeto que merecen como integrantes del equipo y los aportes de cada integrante son valorados por todos. Todos contribuyen, así sea simplemente con entusiasmo. Los desarrolladores respetan la experticia de los clientes y viceversa. La Gerencia respeta el derecho del equipo de asumir responsabilidad y tener autoridad sobre su trabajo. (Wells, 2013)
- **Retroalimentación:** nos tomaremos seriamente los compromisos con el usuario establecidos en todas las iteraciones, entregando software en funcionamiento en cada una. Mostraremos al usuario nuestro software frecuentemente y de forma temprana, escuchando cuidadosamente sus observaciones y realizando los cambios que sean necesarios. Adaptaremos nuestros procesos al proyecto y no al contrario. (Wells, 2013)

  En otras palabras habrá siempre una retroalimentación ya sea por parte del cliente, del sistema y del equipo mismo.

1.6.2 Practicas XP

La tabla 1. Muestra el conjunto de reglas y prácticas de la metodología XP que se pueden agrupar en: reglas y prácticas para la planificación, reglas y prácticas para el diseño, reglas y prácticas para el desarrollo y reglas y prácticas para las pruebas, que se utilizaron como marco de referencia para el proyecto

Tabla 1. *Prácticas y reglas XP*

Planificación				Diseño				Desarrollo				Pruebas		
Historia de usuarios	Plan de entregas	Plan de interacciones	Reuniones diarias	Simplicidad	Soluciones	Recodificación	Metáforas	Uso de estándares	Programación dirigida por las pruebas	Programación en pares	Propiedad colectiva del código	Pruebas Unitarias	Detecciñon y correciñon de errores	Pruebas de aceptación
Substituye a documentación y casos de uso ya que son escritas por el propio cliente y en sus propias palabras	Se realizará en base a las estimaciones de tiempo entregadas por los desarrolladores	Se realiza una interacción al inicio de cada ciclo y son probadas de igual forma	Se realizan para mantener la comunicación con el equipo	Se propone siempre un diseño más simple que funcione	Programas para dar soluciones	Escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad	Concepto como una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo.	Promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación.	Se realiza pruebas constantes y no solo al final	Trabajan dos programadores, esto ayuda a obtener mejores diseños y minimizan errores	Propiedad colectiva y responsabilidades colectivas	Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados.	Se encuentran los errores y se los corrige inmediatamente y tener en cuenta para que no vuelva a ocurrir	El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Nota. Extreme Programing (XP) Fuente: (Joskowicz, 2008)
Elaborado por: Andrea Martínez y Michael Flores

CAPÍTULO 2

MARCO TEÓRICO

2.1 Google Android

Google Android es un sistema operativo basado en Linux usado especialmente en Smartphone y otros dispositivos móviles.

2.1.1 Historia

En Octubre de 2003 Android Inc. fue fundada por Andy Rubin, Rich Miner, Nick Sears y Chris White. Sus intenciones iniciales eran las de desarrollar un sistema operativo inteligente, que tuviese en cuenta la localización y los gustos de su propietario y actuase en consecuencia.

En el año 2005 Google se interesó por Android Inc. Y su trabajo. Decidió adquirir la compañía, para así poder tomar partido en el desarrollo y la toma de decisiones relacionadas con el sistema operativo que estaban desarrollando. Andy Rubin, Rich Miner y Chris White decidieron seguir con el proyecto, además de algunos trabajadores que tenía la compañía, y pasaron a trabajar para Google.

En el año 2007, La Open Handset Alliance, formada por compañías como Google, Sony, HTC, Samsung, T-Mobile, Sprint Nextel, Texas Instruments o Qualcomm anunciaron que estaban trabajando en un mismo objetivo: un sistema operativo abierto para una plataforma móvil. Ese mismo día, Android presentó su primer producto: un sistema operativo basado en Linux 2.6.

En el año 2008 salió al mercado el HTC Dream, el primer smartphone con el sistema operativo Android 1.0 Apple Pie. En apenas 6 meses, el HTC Dream vendió más de 1 millón de terminales en Estados Unidos, y unos 100.000 terminales más en Reino Unido.

En febrero de 2009, poco después del primer lanzamiento de Android al mercado, Google nos sorprendía con el lanzamiento de Android 1.1 Banana Bread. Esta versión de Android apenas introdujo cambios, y fue más bien una versión dedicada a arreglar fallos y bugs de Android 1.0 Apple Pie.

Eso sí, la única cosa que se añadió en Android 1.1 Banana Bread fue algo que potenció el sistema operativo: las actualizaciones automáticas.

Para dar salida a esta nueva herramienta, Google lanza en Abril de 2009 su siguiente versión: Android 1.5 Cupcake. A esta altura fue cuando comenzó a calar el hecho de que Google iba a utilizar nombres de postres para sus versiones de Android, además de que la salida sería en orden alfabético.

En el 2010 El primer teléfono en recibir la actualización fue el HTC Nexus One, dejando claro que Google iba a dar prioridad a sus terminales Nexus en las actualizaciones de software como se muestra en la figura 2. (Ochando, 2003)

Describe brevemente la línea y evolución de Android.

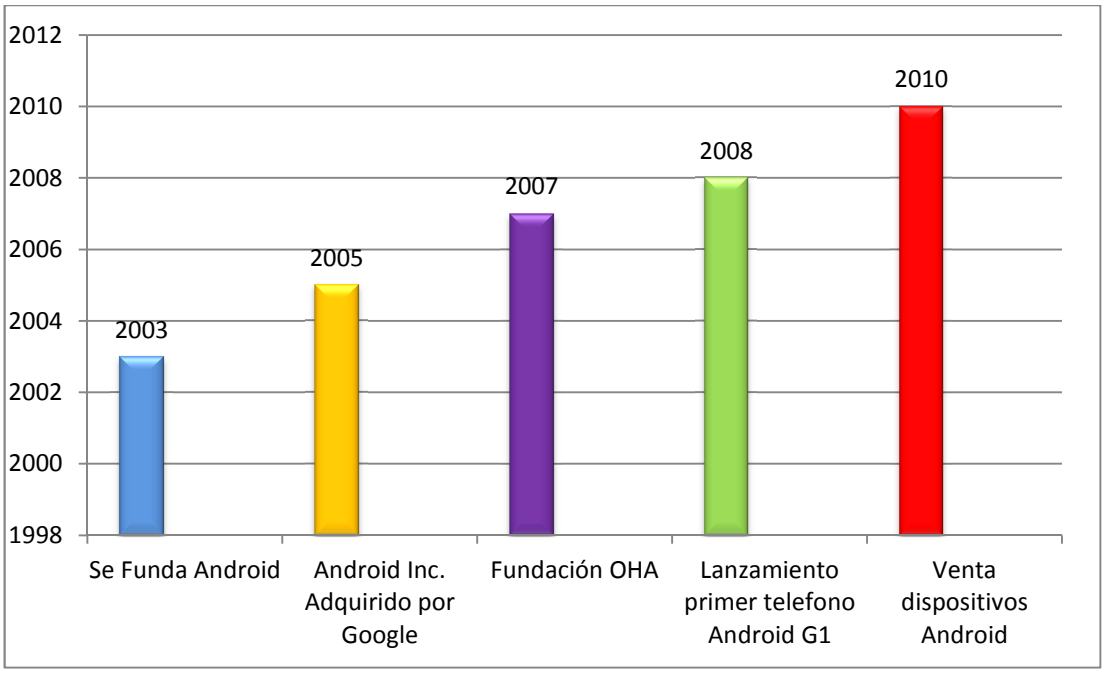

Figura 2. Línea de desarrollo Android
Fuente: (Pérez, 2011)
Elaborado por: Andrea Martínez y Michael Flores

El sistema operativo Android ha ido desarrollándose a la par de la tecnología, creciendo constantemente en funcionalidades y características conforme transcurre el tiempo, este crecimiento se resume en un conjunto versiones como se aprecia continuación en la tabla 2.

Tabla 2. *Versiones de Android*

Versiones android	
Nombre de versión	**Fecha de publicación y detalles**
1.0 (Apple Pie)	Lanzado el 23 de septiembre de 2008. Ventana de notificación desplegable
1.1 (Banana Bread)	Lanzado el 9 de febrero de 2009 Solución a errores de anterior versión.
1.5 (Cupcake) Basado en el kernel de Linux 2.6.27	Lanzado 30 de abril de 2009: La Posibilidad de grabar y reproducir videos.
1.6 (Donut) Basado en el kernel de Linux 2.6.29	Lanzado 15 de septiembre de 2009 Actualización de soporte para CDMA/EVDO, 802.1x, VPN y text-to-speech.
2.0 / 2.1 (Eclair) En Base el kernel de Linux 2.6.29	Lanzado 26 de octubre de 2009 Interfaz Nuevo de usuario en el navegador y soporte para HTML5.
2.2 (Froyo) Basado en el kernel de Linux 2.6.32	Lanzado 20 de mayo de 2010 Soporte para la instalación de aplicación en la memoria externa.
2.3 (Gingerbread) Basado en el kernel de Linux 2.6.35.7	Lanzado 6 de diciembre de 2010: Soporte NFC (Near Field Comunication).
3.0/3.1/3.2 (Honeycomb)	Brinda un mejor soporte para tablets.
4.0 (Ice CreamSandwich)	Versión que unifica el uso tanto en teléfonos, tablets, netbooks.
4.1/4.2/4.3 (JellyBean)	Dictado por voz mejorado con la novedad sin conexión a Internet.
4.4 (KitKat)	Permitió que dispositivos con 512MB de RAM
5.0(Lollipop)	Extensión de nuevas plataformas Google Wear, Google TV y Google Card

Nota. La tabla enlista las versiones existentes de Android. Fuente: (Android Curso UPV, 2011)
Elaborado por: Andrea Martínez y Michael Flores

2.2 Base de datos SQLITE

“Android tiene integrado en el propio sistema un API completa que nos permite manejar BBDD en SQLite. SQLite es un motor de base de datos que maneja archivos de poco tamaño, no necesita ejecutarse en un servidor, cumple el estándar SQL-92 y además, es código libre**.”** (Báez, 2006, pág. 55)

Para acceder a la información de la base de datos local se debe crear una clase que herede de SQLiteOpenHelper sobre la cual tendremos que adaptar/sobrescribir los métodos proporcionados para obtener la funcionalidad con la base de datos deseada. Básicamente en esta clase definimos los atributos de nuestra base de datos y el comportamiento ante creación y actualización de la misma. Los métodos que deberemos sobrescribir serán onCreate() y onUpgrade(), además de la constructora, donde podremos incluir todo lo que creamos necesario. En la figura 3 se detalla un ejemplo de la creación de una base de datos en SQLite con los métodos onCreate() y onUpgrade().

Creación de base de datos SQLite dentro de java

Código Android

```
public class DomicilioSQLiteHelper extends SQLiteOpenHelper {

    //Aqui creamos el String que creará la base de datos en el onCreate
    String creaBD= "CREATE TABLE Domicilio (calle TEXT, ciudad TEXT, CP
INTEGER, numero INTEGER)";

    public DomicilioSQLiteHelper(Context context, String nombre,
                        CursorFactory factory, int version) {
       super(context, nombre, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
       //Aqui se crea la BD, se llamaría solo cuando no exista
       db.execSQL(sqlCreate);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int a, int b) {
       /*Utilizamos como opción de actualización la de borrado de la tabla
anterior, para luego crearla vacia con la nueva versión*/
       db.execSQL("DROP TABLE IF EXISTS Domicilio"); //Borrado anterior
       db.execSQL(sqlCreate); //Creación nueva
    }
}
```

Figura 3. Ejemplo creación de base de datos SQLite
Fuente: (Báez, 2006, pág. 56)

2.3 SDK Manager

Es el Kit de Desarrollo de Software (SDK) o Android SDK el cual se encuentra disponible en varias versiones. Mediante el kit podemos realizar aplicaciones y ejecutar el emulador de la versión de Android.

Al aparecer una nueva versión de Android, Google se encarga de liberar el código fuente y publica el SDK con la nueva versión, lo cual ayuda a adaptar las aplicaciones a las nuevas versiones.

La instalación es muy fácil, se descarga de la página oficial y al ejecutar el SDK Manager se deberán seleccionar las casillas desmarcadas para instalar todas las versiones de Android así como todas las herramientas como se muestra en la figura 4.

Pantalla que muestra el SDK manager

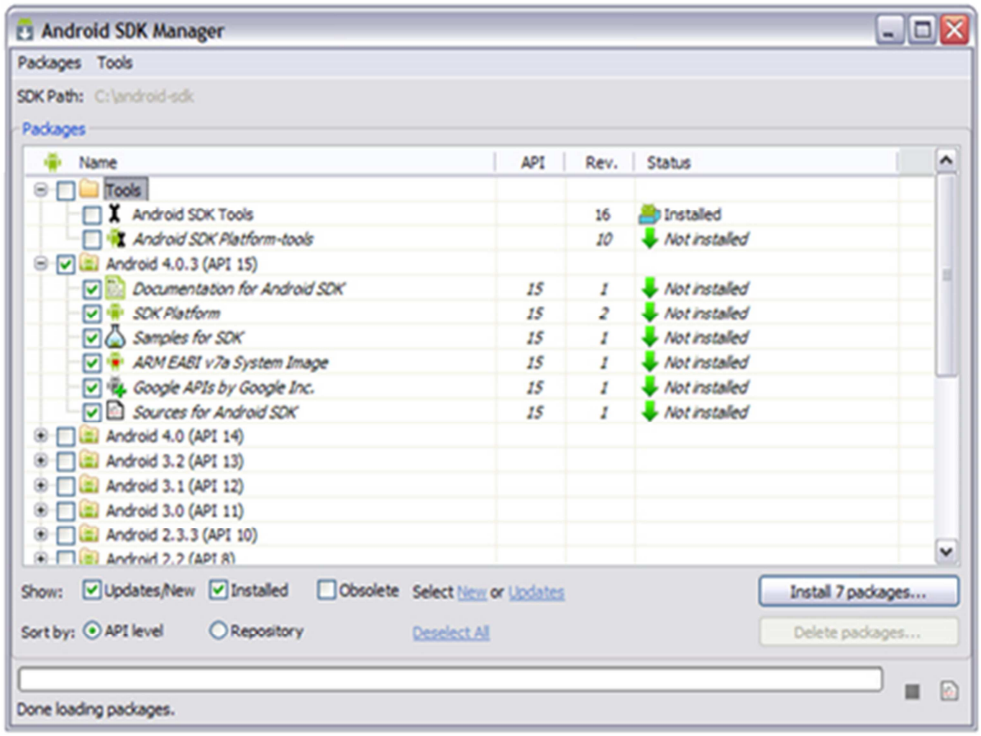

Figura 4. Pantalla de Android SDK Manager.
Fuente: (Báez, 2006, pág. 4)

2.4 Programación extrema (XP)

La programación extrema o eXtreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck. Es el más destacado de los procesos ágiles para desarrollo de software. Al igual que éstos, la programación

extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. (Flores, 2015)

2.4.1 El Proceso XP

"La programación extrema usa un enfoque orientado a objetos como paradigma preferido de desarrollo, y engloba un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades estructurales: planeación, diseño, codificación y pruebas." como se muestran en la figura 5. (Pressman, 2010, pág. 62)

Proceso XP y las tareas de cada acción estructural

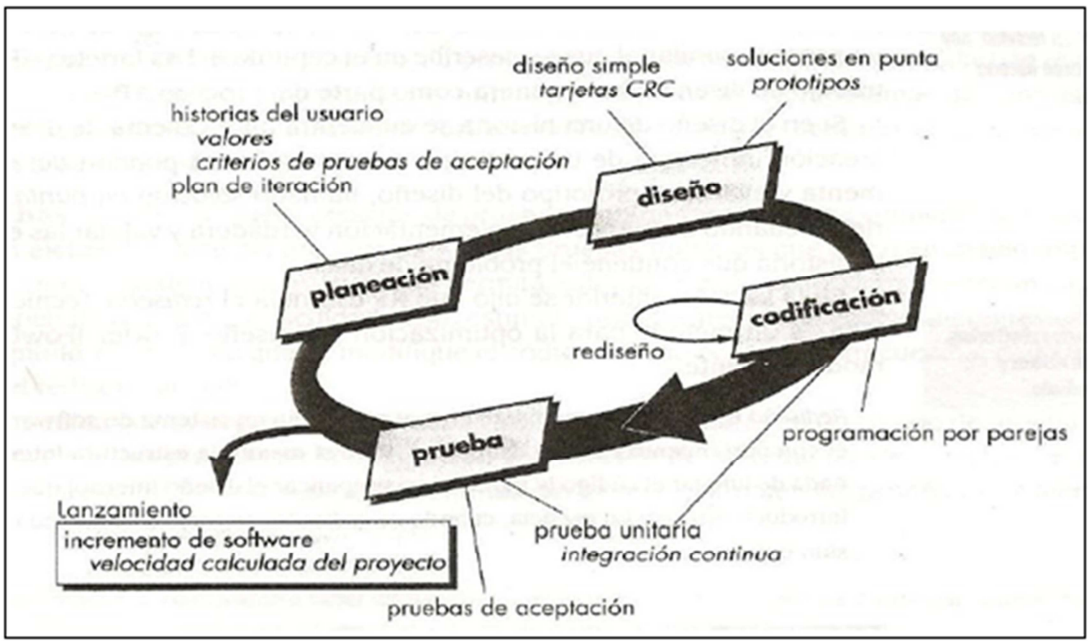

Figura 5. Muestra las tareas dentro del proceso XP.
Fuente: (Pressman, 2010, pág. 62).

Planeación: conocida como juego de planeación, esta actividad empieza escuchando para recabar la información que permite a los técnicos del equipo entender el contexto del negocio para el software, y así adquieran la información de salida, características y funcionalidad que requieren. Al escuchar se genera una historia que es similar a los casos de uso, esta es descrita por el cliente y colocada en una tarjeta indizada.

A cada historia se le asigna una prioridad (valor) que es dada por el cliente, luego los miembros del equipo XP evalúan cada historia y asignan un costo medido en semanas de desarrollo.

Los clientes y desarrolladores trabajan juntos para decidir cómo agrupar las historias para la siguiente entrega, y después del compromiso de entrega el equipo XP ordena las historias a ser desarrolladas, y después de la primera entrega se calculará la velocidad con que se trabajará, la misma que ayuda a estimar fechas de entrega y programar actividades para entregas posteriores.

Diseño: XP utiliza el principio de "mantenlo sencillo". Un diseño sencillo siempre se prefiere sobre un complejo. XP usa tarjetas CRC (Clase Responsabilidad Colaborador) para diseñar el software en un contexto orientado a objetos y son relevantes para el incremento del software. Las CRC son el único producto del trabajo de diseño dentro del proceso XP.

Si XP encuentra un diseño complicado de una historia recomienda la creación inmediata de un prototipo operativo de esa porción del diseño y así disminuye el riesgo cuando empiece la verdadera implementación.

En XP parte principal del concepto de diseño es que se realiza tanto antes como después del inicio de la codificación.

Codificación: después del diseño de las historias, no se inicia con la codificación sino que realiza varias pruebas unitarias a las historias del usuario que van a ser entregadas en el próximo incremento. Después de realizar las pruebas el desarrollador tiene más claro lo que debe implementar para pasar las pruebas. Al pasar la prueba unitaria el desarrollador obtiene una retroalimentación instantánea.

Parte esencial de la actividad de codificación de XP es la programación en parejas, quienes son las encargadas de crear el código para una historia, lo cual generara un mecanismo de solución en tiempo real y también el aseguramiento de la calidad en tiempo real debido a que el código se revisa conforme se genera.

Pruebas: cada una de las pruebas unitarias que se realizan deben ser implementadas con el uso de una estructura que permita automatizarlas.

Al organizar las pruebas unitarias individuales en un grupo de prueba universal las pruebas de integración y validación pueden realizarse a diario, esto permite al equipo tener un indicador continuo del avance y alertas si algo marcha mal. Esto ayuda a reducir los problemas al final ya que se corrige pequeños problemas cada cierto tiempo y no problemas más grandes al final del plazo. Las “Pruebas de Validación” son el proceso de la revisión del software que cumple con las especificaciones requeridas. Las Pruebas de Aceptación XP, también llamadas “Pruebas del Cliente” son especificadas por el cliente y estas se centran en la funcionalidad general del sistema que deben ser revisadas por parte del cliente. (Pressman, 2010)

A continuación las etapas del proceso XP en forma gráfica para facilitar su comprensión como se aprecia en la figura 6.

Proceso XP

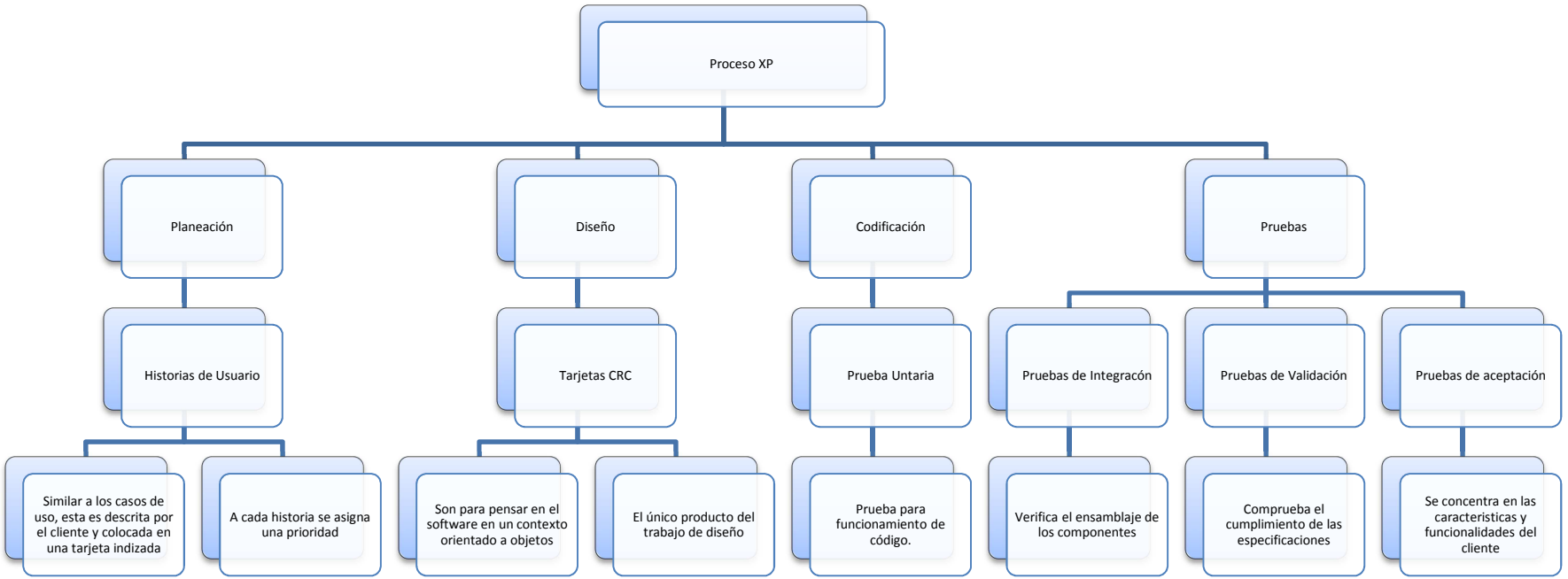

Figura 6. Descripción de las etapas del proceso XP.
Elaborado por: Andrea Martínez y Michael Flores

2.5 Lenguaje Unificado de Modelado (UML)

El Lenguaje de Modelado Unificado (UML: Unified Modeling Lenguaje) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s.UML es llamado un lenguaje de modelado, no un método. Los métodos consisten de ambos de un lenguaje de modelado y de un proceso. El UML, fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE (Booch, G. et al., 1999). (Pallioto, 2008)

UML es una de las herramientas más emocionantes en el mundo actual del desarrollo de software. Esto se debe a que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas. (Schmuller, 2001, pág. 22)

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que se requiere entre todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado. (Pallioto, 2008)

2.5.1 Diagramas UML

"La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a los cuales se les conoce como modelo. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema." (Schmuller, 2001, pág. 25)

2.5.1.1 Diagrama de clases

"Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenimiento." (Patricio, 1996)

Un diagrama de clases está compuesto por los siguientes elementos:

- "Clase es un descriptor de un conjunto de objetos que comparten estructura, comportamiento y relaciones similares. Se representa mediante un rectángulo

con hasta tres compartimientos" como se observa en la figura 7. más claramente. (Diclase, 2007)

- Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Símbolo UML de una clase

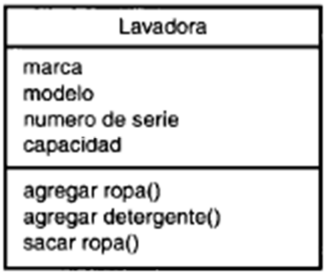

Figura 7. Representación de un diagrama de clase
Fuente: (Schmuller, 2001, pág. 26)

2.5.1.2 Diagrama de objetos

"Un diagrama de objetos es una instancia de clase (una entidad que tiene valores específicos de los atributos y acciones)" según la representación gráfica de la figura 8. (Schmuller, 2001, pág. 26)

"Los Diagramas de Objetos están vinculados con los Diagramas de Clases. Un objeto es una instancia de una clase, por lo que un diagrama de objetos puede ser visto como una instancia de un diagrama de clases. Los diagramas de objetos describen la estructura estática de un sistema en un momento particular y son usados para probar la precisión de los diagramas de clases." (Bresano, 1996, pág. 5)

El símbolo de UML de objeto

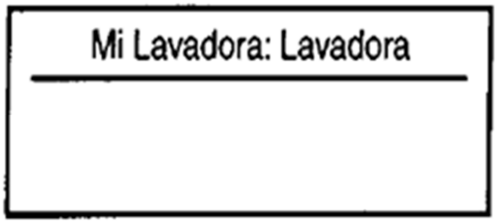

Figura 8. Descripción de un diagrama de objetos
Fuente: (Schmuller, 2001, pág. 27)

2.5.1.3 Diagrama de casos de uso

Un diagrama de caso de uso es una descripción de acciones del sistema, desde el punto de vista del usuario. Es una herramienta muy importante ya que es una técnica de aciertos y errores del sistema. Y además puede ser entendido por la gente en general no solo por los desarrolladores

Para ello se utiliza elementos como roles de usuario, actividades y tipos de conexión como se observa en la figura 9. (Schmuller, 2001, pág. 27)

Figura de casos de uso UML

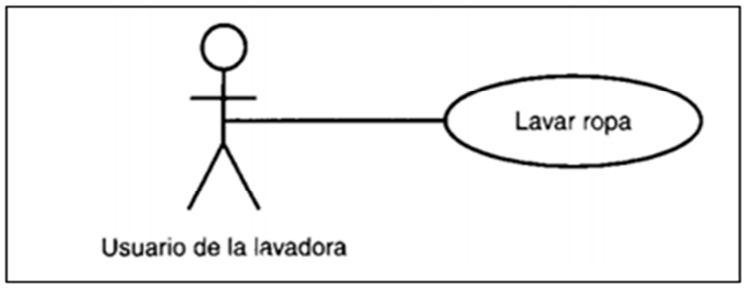

Figura 9. Diagrama de casos de uso
Fuente: (Schmuller, 2001, pág. 27)

2.5.1.4 Diagrama de estados

"El diagrama de estados representa un objeto en particular cuando está en determinado estado" (Bresano, 1996, pág. 8)

"Poe ejemplo una lavadora podrá estar en fase de remojo, lavado, enjuague, centrifugado o apagado" como se detalla en la figura 10. (Schmuller, 2001, pág. 27)

Diagrama de Estados UML

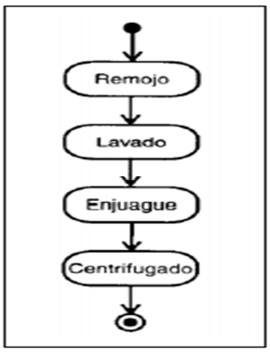

Figura 10. Representación de un diagrama de estados
Fuente: (Schmuller, 2001, pág. 28)

2.5.1.5 Diagrama de secuencia

"Los diagramas de clases y los de objetos representan información estática. No obstante, en un sistema funcional los objetos interactúan entre sí, y tales interacciones suceden con el tiempo. El diagrama de secuencia UML muestra la mecánica de la interacción con base en tiempos" como se observa en la figura 11. (Schmuller, 2001, pág. 28)

Diagrama de secuencia UML

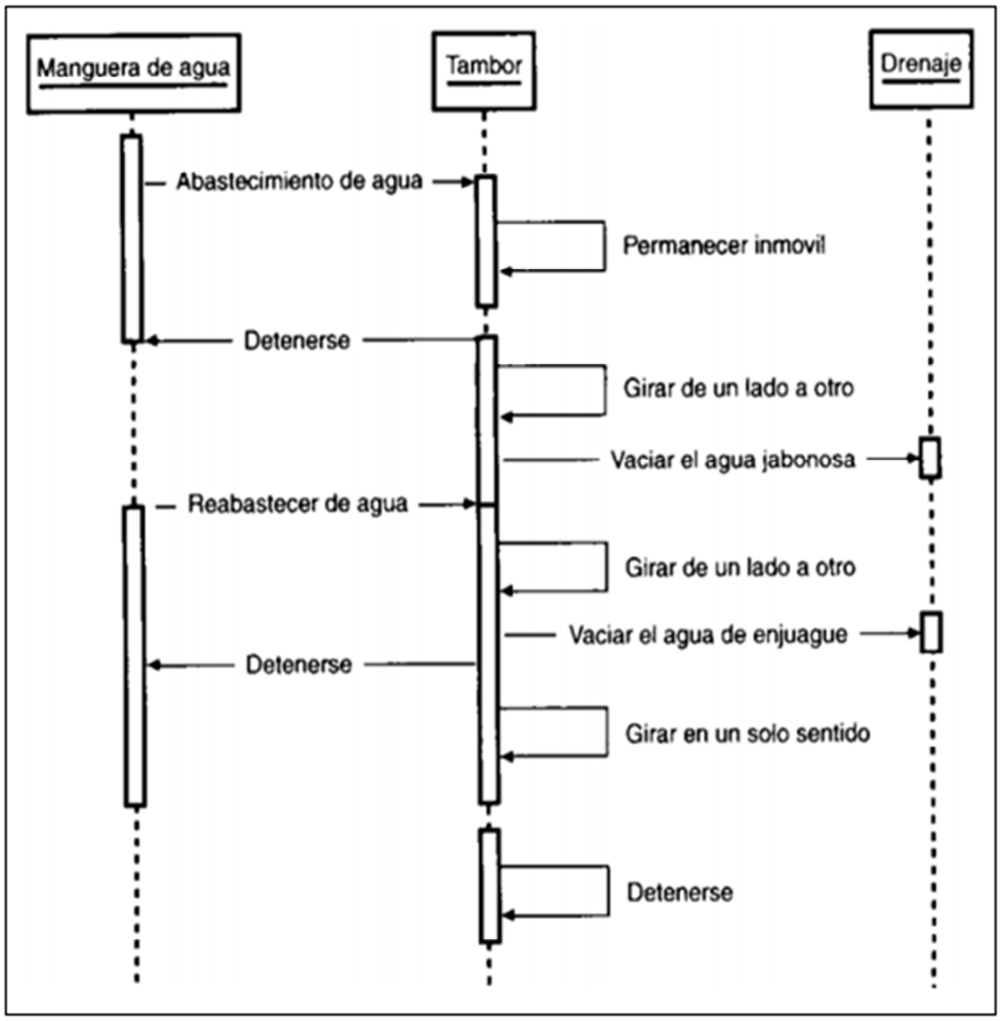

Figura 11. Descripción de un diagrama de secuencia
Fuente: (Schmuller, 2001, pág. 29)

2.5.1.6 Diagrama de despliegue

"El diagrama de despliegue UML muestra la arquitectura física de un sistema informático. Puede representar los equipos y dispositivos, mostrar sus interconexiones y el software que se encontrara en cada máquina. Cada computadora

está representada por un cubo y las interacciones entre la computadoras están representadas por líneas que conectan a los cubos", esta representación muestra en la figura 12. La interacción entre equipos (Schmuller, 2001, pág. 31)

Diagrama de despliegue UML

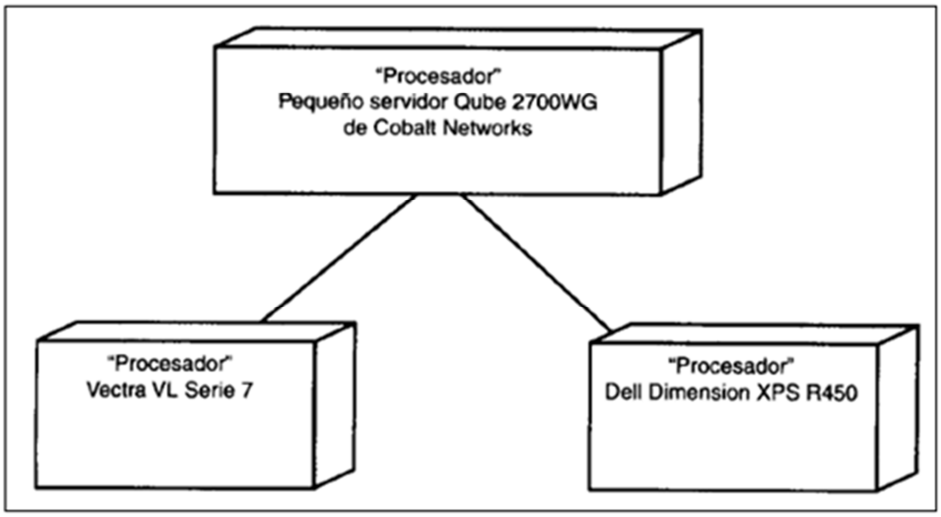

Figura 12. Ejemplo de un diagrama de despliegue
Fuente: (Schmuller, 2001, pág. 31)

2.6 Servicios Web

"Un servicio Web o Web Services es un servicio ofrecido por una aplicación que expone su lógica a clientes de cualquier plataforma mediante una interfaz accesible a través de la red utilizando protocolos estándar de internet." (Rodríguez, 2014)

Un Servicio Web es un componente software que puede ser registrado, descubierto e invocado mediante protocolos estándares de Internet.

- Permiten exponer y hacer disponibles funcionalidades (servicios) de los sistemas informáticos de las organizaciones mediante tecnologías y protocolos WEB estándar.
- Cada Servicio Web se responsabilidad de realizar un conjunto de funciones concretas y bien definidas
- Servicios Web actúan como componentes independientes que se pueden integrar para formar sistemas distribuidos complejos

Un Servicio Web (Web Service [WS]) es una aplicación software identificada por un URI (Uniform Resource Identifier), cuyas interfaces se pueden definir, describir y

descubrir mediante documentos XML. Los Servicios Web hacen posible la interacción entre agentes de software (aplicaciones) utilizando mensajes XML intercambiados mediante protocolos de Internet.

Interoperabilidad

Distintas aplicaciones, en lenguajes de programación diferentes, ejecutadas sobre cualquier plataforma, pueden utilizar los Servicios Web para intercambiar datos

- La interoperabilidad se consigue mediante el uso de estándares abiertos.
- Servicios Web se asientan sobre protocolos y estándares ya existentes y muy difundidos (HTTP, XML, etc.)
- Uso de protocolos específicos extensibles) no imponen restricciones sobre las aplicaciones a las que dan acceso ni sobre las tecnologías que las implementan (independencia de lenguaje y de plataforma)
- OASIS (estándares abiertos avanzados para la sociedad de la información) y W3C(World Wide Web Consortium): organizaciones responsables de definir la arquitectura y estándares para los Servicios Web

Para facilitar la comprensión del funcionamiento de los servicios web y su interoperabilidad se presenta la figura 13 como ejemplo.

Interoperabilidad gracias a la independencia de lenguaje y de plataforma

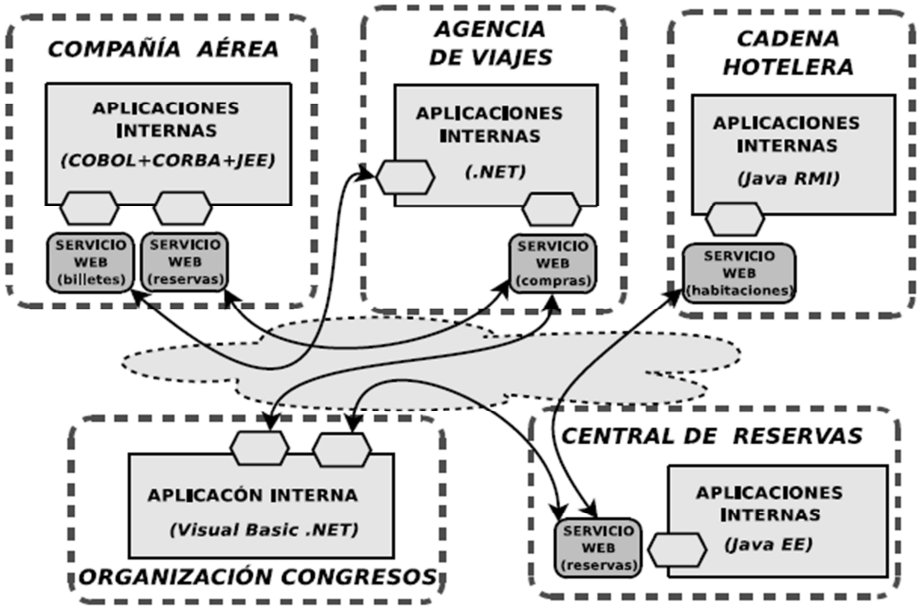

Figura 13. Ejemplo de uso de servicios web
Fuente: (Juan, 2008, pág. 2)

Elementos necesarios para la definición de Servicios Web

1. Sintaxis común para todas las especificaciones mediante el uso de la estructura XML

- XML: eXtensible Markup Language(Lenguaje de marcado extensible)
    - Estándar para la definición de lenguajes de marcas
    - Flexible y extensible
- Metalenguaje usado en Servicios Web para especificar los lenguajes y protocolos necesarios
    - Permite definición de lenguajes para: describir servicios y representar mensajes intercambiados

2. Mecanismos de interacción entre extremos mediante uso de SOAP (Simple Object Access Protocol)

- Necesidad de un formato de mensajes neutro, abierto y extensible
    - Representación de mensajes de invocación (argumentos) y respuesta (valor retorno) como documentos XML
- Especificación del modo de interacción: síncrono (RPC: petición-respuesta) y asíncrono (petición)
- Mapeo de los mensajes en el protocolo de transporte (HTTP, SMTP) Mecanismos de interacción entre extremos) uso de SOAP

3. Lenguaje común para describir los servicios) uso de WSDL (Web Service Description Language)

- Descripción de los servicios y sus interfaces de forma estándar mediante documentos XML
- Papel análogo al del IDL en middleware convencional
- Incluye toda la información necesaria para suplir la falta de un middleware común centralizado
    - Especifica cada operación disponible, con sus parámetros de entrada y de salida
    - Puede usarse para generar los stubs/skeleton y las capas intermedias necesarias para escribir: clientes que invoquen los Servicios Web servidores que los implementen
    - Especificar información sobre la localización del servicio (URIs)

4. Publicación y localización de servicios mediante el uso de UDDI (Universal Description, Discovery and Integration)
    - La descripción de los servicios (documentos WSDL) se almacena en un directorio de servicios
    - UDDI especifica cómo: se publican y descubren los servicios, cómo trabajan los directorios de servicios Web
    - Acceso al directorio UDDI mediante Servicios Web) uso de mensajes SOAP
        - Servidor da de alta de servicios (documentos WSDL + descripción)
        - Cliente “descubre” servicios (documentos WSDL)

Para resumir la arquitectura de protocolos de servicios web que se describieron se presenta la siguiente figura 14. (Juan, 2008, pág. 5)

Descripción de protocolos de servicios web

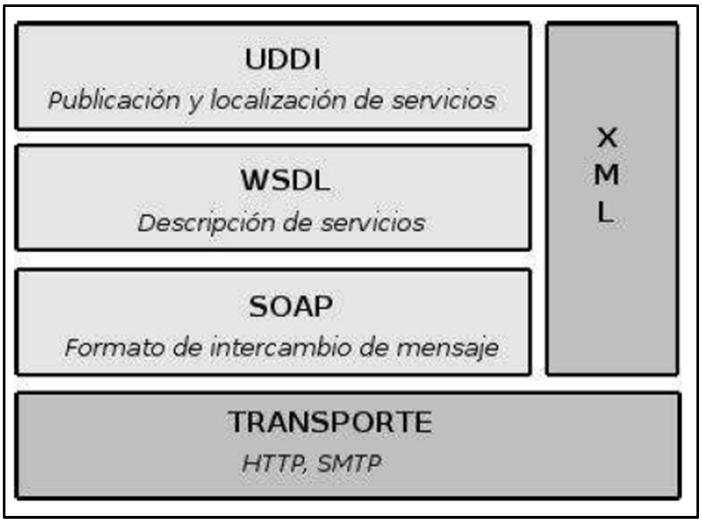

Figura 14. Arquitectura básica de protocolos de servicios web
Fuente: (Juan, 2008, pág. 4)

CAPÍTULO 3

ANÁLISIS DEL SISTEMA Y DISEÑO

3.1 Especificación de requisitos del software

En la etapa de planificación de la metodología XP nos plantea la realización del levantamiento de los requisitos de software, que se logran recolectando las historias de usuarios y tarjetas CRC (Clase Responsabilidad Colaborador) mismas que son escritas por el propio usuario que expresa sus necesidades para darnos una idea clara de los requisitos necesarios del software.

3.1.1 Definición de roles de usuario

Se describen los roles de usuario con las funciones que podrán utilizar dentro de la aplicación Android como se muestra en la Tabla 3.

Tabla 3. *Roles y requisitos de Usuario del Sistema*

Usuarios	Funciones	Requisitos mínimos
Administrador	Es el encargado de la administración del sistema como la publicación y modificación de notificaciones dentro de la aplicación, mensajes masivos	• Conocimiento intermedios de computación • Conocimiento intermedio de base de datos • Conocimiento de administración de portales web
Usuario	Es la persona encargada de utilizar la aplicación desde un dispositivo Android en el cual podrá hacer uso del chat y verificar las notificaciones publicadas	• Manejo de dispositivos móviles Android • Conocimiento básico de aplicaciones móviles

Nota. Describe las funciones y los requisitos de los usuarios para el uso de la aplicación Android.
Elaborado por: Andrea Martínez y Michael Flores

3.1.2 Levantamiento de historias de usuario

Las historias de usuario poseen una breve descripción de los requisitos del software que solicita el cliente, que es la forma más fácil de recolectar los requisitos de los usuarios.

En la tabla 4. Describe la gestión de registro del Smartphone para acceder a la aplicación considerando los flujos normales y flujos alternativos

Tabla 4. *Historia de Usuario Gestión de Registro de usuario*

Nombre:	Gestión de registro de usuario(Smartphone)
Autores:	Andrea Martínez y Michael Flores
Fecha:	07/11/2014
Descripción: Permite el registro de los usuarios en el sistema	
Precondiciones: Se necesita que el usuario tenga la aplicación en su dispositivo móvil Android.	
Flujo normal: Registro usuarios a Ingresa los datos del registro para el nuevo usuario b Se pulsa registrar para que la información sea guardada	
Flujo alternativo: El sistema comprueba si los datos ingresados son correctos y de no serlos manda alertas de información mal ingresada	
Post condiciones: La información ingresada correctamente es guardada en el sistema	

Nota. Se describe el flujo del registro dentro de la aplicación Android.
Elaborado por: Andrea Martínez y Michael Flores

En la tabla 5. Se detalla la lógica de la gestión de notificaciones del Smartphone para enviar y recibir notificaciones, señalando el flujo normal y el flujo alternativo de este proceso

Tabla 5. *Historia de Usuario Gestión de Notificaciones*

Nombre:	Gestión de notificaciones (Smartphone)
Autores:	Andrea Martínez y Michael Flores
Fecha:	07/11/2014
Descripción: Permite gestionar las notificaciones del sistema	
Precondiciones: Se necesita que el usuario sea el administrador y debe estar autentificado en el sistema	
Flujo normal: Ingreso de notificaciones a Ingresa los datos para la nueva notificación b Se guarda la notificación dependiendo del tipo de notificación Envía notificación a- Se selecciona una notificación y se envía la información b- Se envía la información a todos los usuarios que posean la aplicación móvil Revisa notificación a- Los usuarios revisan la notificación que llega a la aplicación móvil b- Se revisa la información enviada en la notificación.	

Flujo alternativo: El sistema comprueba si el ingreso es como administrador, de no serlo no podrá ingresar al sistema y gestionar ningún tipo de notificación desde el portal web.
Post condiciones: Si el ingreso al sistema es como administrador el sistema permitirá la gestión de todo tipo de notificaciones desde el portal web. Y el usuario las revisara desde su aplicación móvil.

Nota. Se describe el flujo de la gestión de notificaciones dentro de la aplicación Android
Elaborado por Andrea Martínez y Michael Flores

En la tabla 6. Se detalla la lógica de la gestión de chat del Smartphone para realizar el envío y recepción de mensajes de texto, señalando el flujo normal y el flujo alternativo que debe tener el proceso.

Tabla 6. *Historia de usuario gestión de chat*

Nombre:	Gestión de chat (Smartphone)
Autores:	Andrea Martínez y Michael Flores
Fecha:	07/11/2014
Descripción: Permite acceder al chat	
Precondiciones: El usuario debe estar autenticado en el sistema	
Flujo normal: Ingreso al chat a- El usuario dentro del sistema escoge la opción de chat b- Dentro del chat puede buscar un contacto por medio del nombre, correo o número celular c-Se envía la solicitud al contacto y de aceptar puede empezar un chat Envío de mensajes dentro del chat a- Se selecciona el contacto y se envía un mensaje b- Se envía el mensaje al contacto y empieza el proceso de sincronización para el envío y recepción de los mensajes con el contacto seleccionado	
Flujo alternativo: El sistema verifica la conexión de no haberla no se podrá enviar ni recibir los mensajes	
Post condiciones: De tener conexión con el servidor web se pueden enviar y recibir los mensajes desde el chat	

Nota. Se describe el flujo de la gestión de chat dentro de la aplicación Android
Elaborado por Andrea Martínez y Michael Flores

En la tabla 7. Se detalla la lógica de la gestión de sincronización del Smartphone, para permitir el envío y recepción de notificaciones, mensajes, datos de contacto y demás datos para el correcto funcionamiento de la aplicación, también se detalla el flujo normal y el flujo alternativo del proceso.

Tabla 7. *Historia de usuario sincronización con el servidor*

Nombre:	Sincronización con el servidor (Smartphone)
Autores:	Andrea Martínez y Michael Flores
Fecha:	07/11/2014
Descripción: Permite realizar una sincronización con el servidor desde el dispositivo móvil	
Precondiciones: Se necesita la conexión con el servidor desde el dispositivo móvil para lo cual los servicios del servidor deben estar levantados, el dispositivo tiene que tener acceso Wireless a red de la UPS.	
Flujo normal: Sincronizar a- El usuario está listo con su dispositivo móvil para la sincronización b- El servidor está listo para recibir los datos desde el dispositivo móvil Enviar información a- El dispositivo móvil envía la información hacia el servidor b- El servidor recibe la información y la guarda en la base de datos Recibir información a- El dispositivo móvil recibe la información desde el servidor b- El servidor envía la información hacia el dispositivo móvil	
Flujo alternativo: El sistema comprueba la conectividad, y de no existir no realiza la sincronización	
Post condiciones: Se realiza el proceso de sincronización con el servidor	

Nota. Se describe el flujo de la sincronización con el servidor dentro de la aplicación Android
Elaborado por Andrea Martínez y Michael Flores

La tabla 8 se detalla el proceso de acceso al portal web por el usuario administrador

Tabla 8. *Historia de usuario acceso al portal web*

Nombre:	Acceso al portal Web
Autores:	Andrea Martínez y Michael Flores
Fecha:	07/11/2014
Descripción: Permite el acceso del usuario administrador a la plataforma web	
Precondiciones: Se necesita una conexión a internet, la conexión con el portal web se puede realizar desde cualquier dispositivo que tenga acceso a un browser de internet, el dispositivo tiene que tener acceso a red de la UPS	
Flujo normal: acceso	

a- el usuario administrador coloca su usuario y contraseña b- presiona enviar para validar la información e ingresa al portal web
Flujo alternativo: el usuario administrador coloca datos incorrectos dentro de los casilleros de usuarios y contraseña a- Presiona enviar b- Aparece un mensaje de error en autentificación y solicita que ingrese nuevamente los datos
Post condiciones: se ingresa al portal web

Nota. Se describe el flujo del acceso al portal web
Elaborado por Andrea Martínez y Michael Flores

Tabla 9. *Historia de usuario gestión de registro y envío de notificaciones*

Nombre:	Gestión de registro y envío de notificaciones por el portal web
Autores:	Andrea Martínez y Michael Flores
Fecha:	07/11/2014
Descripción: Permite registrar y enviar las notificaciones.	
Precondiciones: Se necesita estar autentificado con el usuario administrador, se necesita cualquier dispositivo que tenga un browser de internet, el dispositivo tiene que tener acceso a la red de la UPS.	
Flujo normal: Registro a.- El administrador se coloca en la pestaña de notificaciones b.-El administrador selecciona la carrera, el tipo de notificación, el título de la notificación y detalla la notificación c.- El administrador presiona guardar y enviar	
Flujo alternativo: si el administrador no está autentificado no puede registrar notificación, si falta datos el mensaje no se enviara.	
Post condiciones: Se realiza el proceso de registro y envío de la notificación	

Nota. Se describe el flujo de registro y envío de notificaciones
Elaborado por Andrea Martínez y Michael Flores

3.1.3 Tarjetas CRC (Clase Responsabilidades Colaboradores)

En las tarjetas CRC se distinguen las clases y como una clase colabora con otra para cumplir sus responsabilidades y asociaciones con las mismas. Las tarjetas CRC son un insumo para determinar los requisitos finales de la aplicación como se detalla en las tablas a continuación.

La tabla 10. Muestra las funciones que debe tener las clases con relación al usuario

Tabla 10. *Tarjeta CRC clase usuario*

Nombre de la clase: Usuario **Descripción:** Esta Clase detalla todo lo relacionado a las funciones que realizara el usuario dentro de la aplicación	
Responsabilidades: • Valida los datos de usuario • Valida password • Crea el usuario • Cambia password	**Colaboradores:** • Contacto • Chat • Notificación

Nota. Describe las clases y colaboradores de la clase Usuario
Elaborado por: Andrea Martínez y Michael Flores

La tabla 11. Muestra las funciones que debe tener las clases con relación al contacto

Tabla 11. *Tarjeta CRC clase contacto*

Nombre de la clase: Contacto (Smartphone) **Descripción:** Esta clase detalla todas las características que se podrán tener en los contactos	
Responsabilidades: • Búsqueda de contactos • Envío de solicitud de contactos • Ver datos del contacto • Visualizar lista de contactos • Acepta solicitud de contactos	**Colaboradores:** • Usuario • Chat

Nota. Describe las clases y colaboradores de la clase *Contacto*
Elaborado por: Andrea Martínez y Michael Flores

En la tabla 12. Nos detalla las funciones de la clase Chat y todas las funciones que se deberían poder realizar dentro de esta clase

Tabla 12. *Tarjeta CRC clase chat*

Nombre de la clase: Chat (Smartphone) **Descripción:** Clase donde se realiza las conversaciones del Chat	
Responsabilidades: • Enviar mensaje • Recibir mensaje • Eliminar un mensaje • Eliminar un listado de mensajes	**Colaboradores:** • Contacto • Usuario

Nota. Describe las clases y colaboradores de la clase *Chat*
Elaborado por: Andrea Martínez y Michael Flores

En la tabla 13. Nos muestra el detalle de las funciones que debe tener las Notificaciones dentro de la aplicación.

Tabla 13. *Tarjeta CRC clase notificación*

Nombre de la clase: Notificación (Smartphone) **Descripción:** Muestra el detalle de las notificaciones	
Responsabilidades: • Recibir notificaciones • Visualizar detalle de notificación • Eliminar notificación • Visualizar listado de notificaciones recibidas	**Colaboradores:** • Usuario

Nota. Describe las clases y colaboradores de la clase *notificación*
Elaborado por: Andrea Martínez y Michael Flores

La tabla 14. Muestra las funciones que debe tener las clases con relación al registro y envío de notificaciones

Tabla 14. *Tarjeta CRC clase registro y envío de notificaciones*

Nombre de la clase: Registro y envío de notificaciones **Descripción:** Esta clase detalla todas las características que se podrán tener en el portal web que debe permitir el registro y el envío de notificaciones	
Responsabilidades: • Verificar el acceso del usuario Administrador • Registro de las notificaciones por tipo y carrera • Envío de notificaciones	**Colaboradores:** • Usuario web

Nota. Describe las clases y colaboradores de la clase *registro y envío de notificaciones*
Elaborado por: Andrea Martínez y Michael Flores

3.2 Diseño del Sistema

3.2.1 Diseño de la base de datos

Considerando que PostgresSql es un sistema de gestión de bases de datos objeto-relacional cliente/servidor, distribuido bajo licencia BSD y con su código fuente disponible libremente. Que es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales se optó por utilizarlo como núcleo de almacenamiento de todo el proyecto.

Del mismo modo se evaluó el volumen de datos con los que trabaja la aplicación Android y aprovechando que dentro el sistema operativo Android viene una base de datos integrada denominada SQLite que permite un almacenamiento de hasta 1 terabyte y que la velocidad de comunicación con la aplicación es más eficiente debido a que SQLite se enlaza directamente con la aplicación sin procesos en segundo plano se optó por trabajar con SQLite.

Ya habiendo definido las base de datos y sus tipo se procede a realizar el diseño de las bases de datos de la aplicación Android denominada UPSY y de la base de datos del servidor central, para ello se utilizó la herramienta de modelado Sybase Power Designer, que cumple con los estándares: UML 2.1, XML, Business Process

Modeling Notation y Document Type Definition. Permitiéndonos diseñar modelos de negocio, modelos de datos, modelos empresariales, etc.

Utilizando la herramienta anteriormente mencionada se diseñaron los modelos físicos y conceptuales de la base de datos SQLite que estará alojada dentro del dispositivo móvil (Smartphone) embebida en el sistema operativo Android y de la base de datos PostgresSql donde se almacenan los datos de la aplicación de todos los usuarios.

3.2.1.1 Diseño conceptual de las base de datos

Se describe gráficamente el diseño conceptual de la base de datos SQLite para el dispositivo Smartphone en la figura 15. Mostrando la estructura de las relaciones y tablas. Del mismo modo en la figura 16. Se muestra el modelo conceptual de la base de datos PostgresSql que constituye el núcleo de almacenamiento de todo el proyecto con la estructura de las relaciones y tablas.

Diseño conceptual de la base de datos SQLite

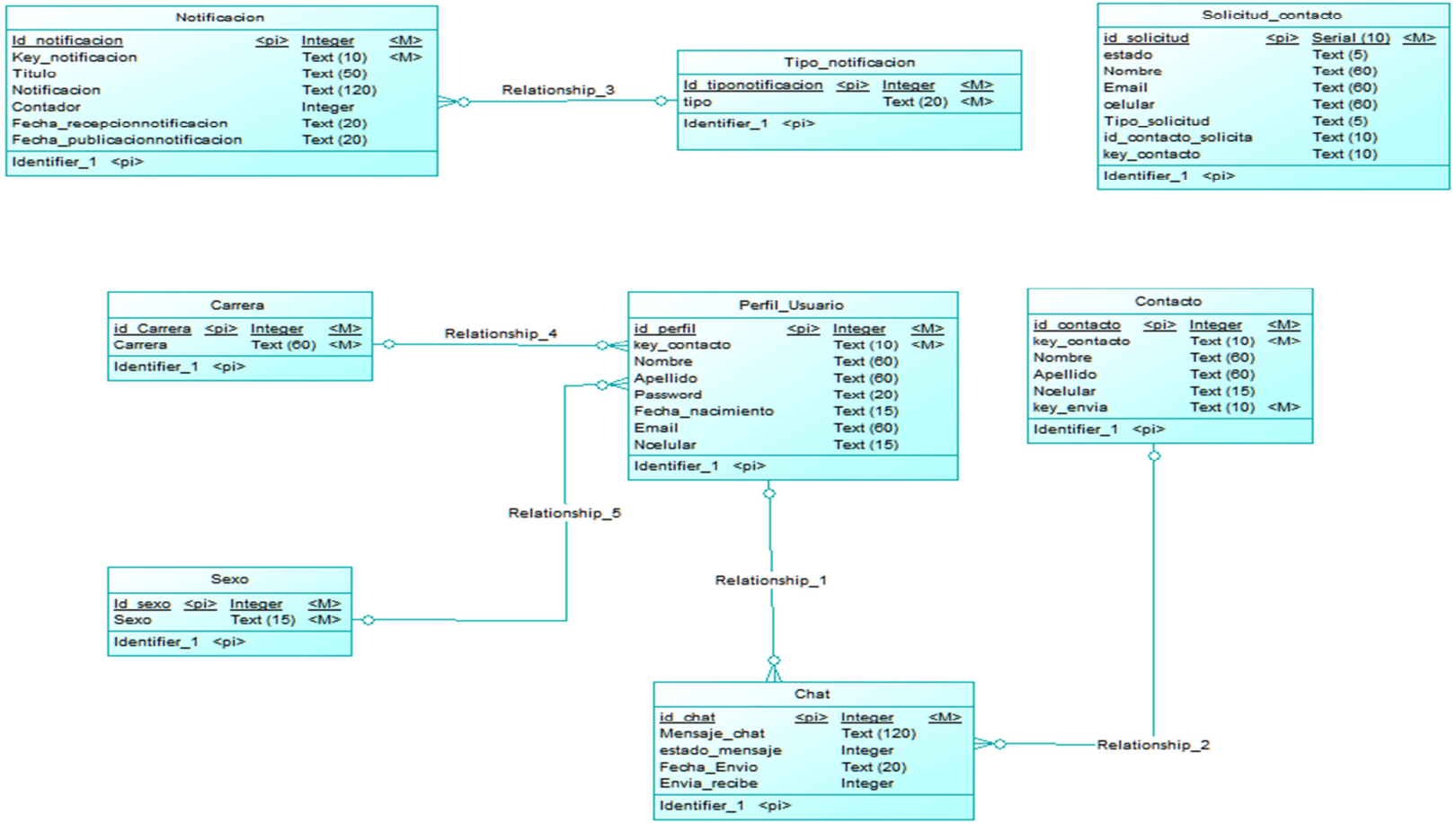

Figura 15. Diseño conceptual de la base de datos SQLite

Elaborado por: Andrea Martínez y Michael Flores

Diseño conceptual de la base de datos PostgresSql

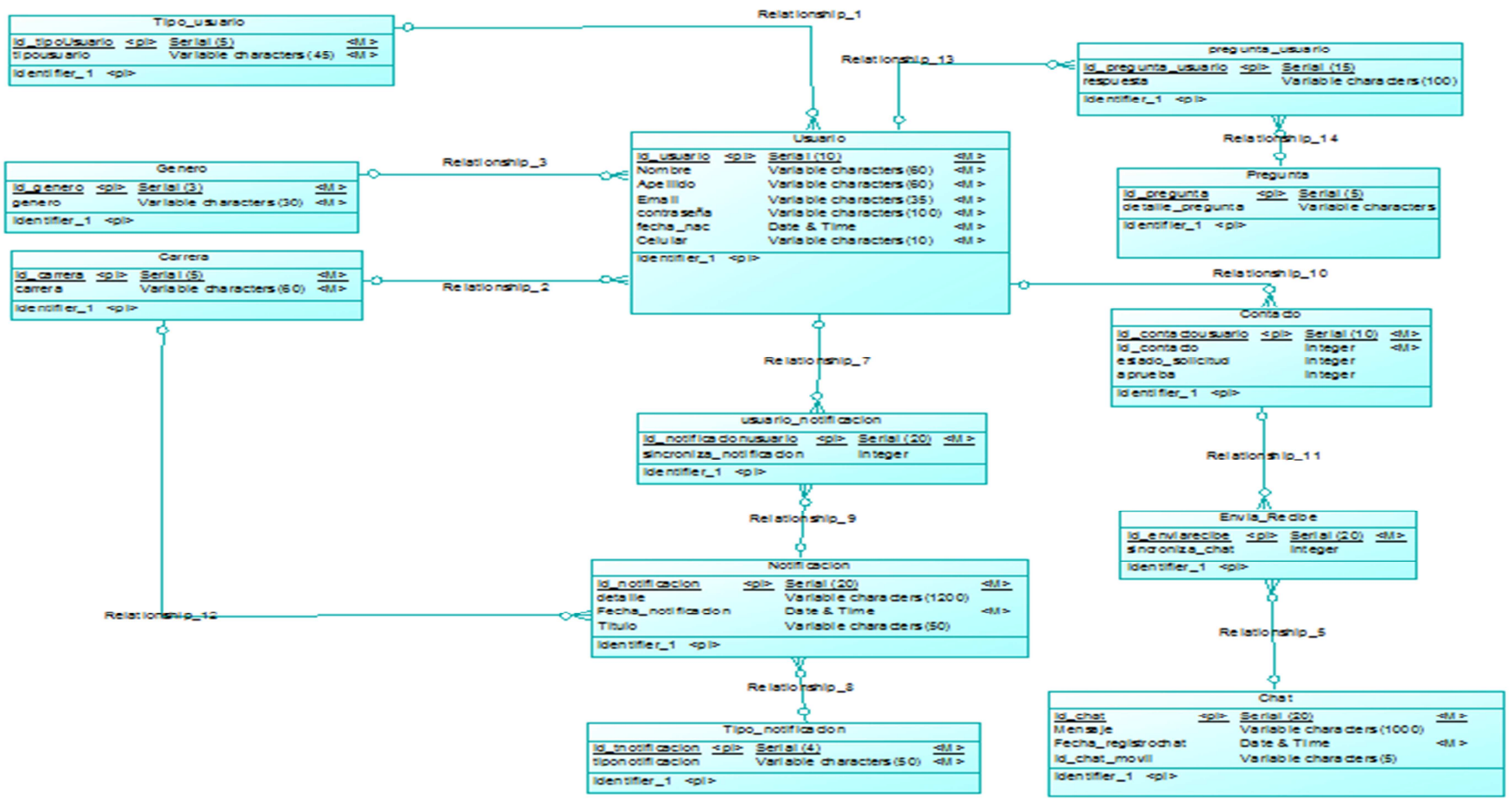

Figura 16. Diseño conceptual de la base de datos en PostgresSql
Elaborado por: Andrea Martínez y Michael Flores

3.2.1.2 Diseño físico de las bases de datos

El diseño físico nos muestra los campos definitivos con los que cada tabla estará armada, esto quiere decir que, obtendremos un detalle total de los campos con la descripción de: tipo del campo, claves principales, claves foráneas y otros. Esto se puede observar en la figura 17. Del diseño físico de la base de datos SQLite para el dispositivo Smartphone y en la figura.18 el diseño físico de la base de datos PostgresSql núcleo del proyecto graficados a continuación.

Diseño físico de la base de datos SQLite

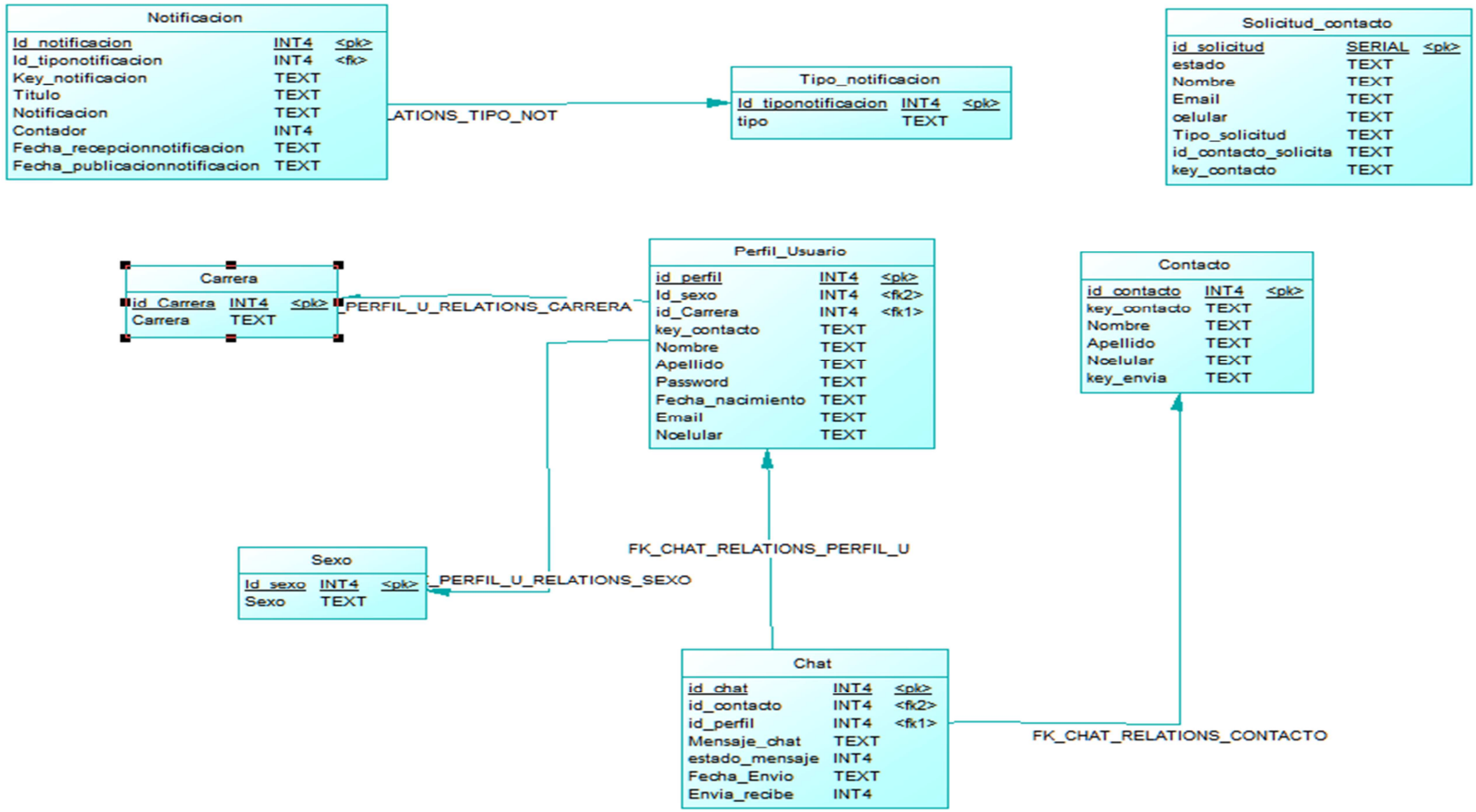

Figura 17. Diseño físico de la base de datos SQLite
Elaborado por: Andrea Martínez y Michael Flores

Diseño físico de la base de datos PostgresSql

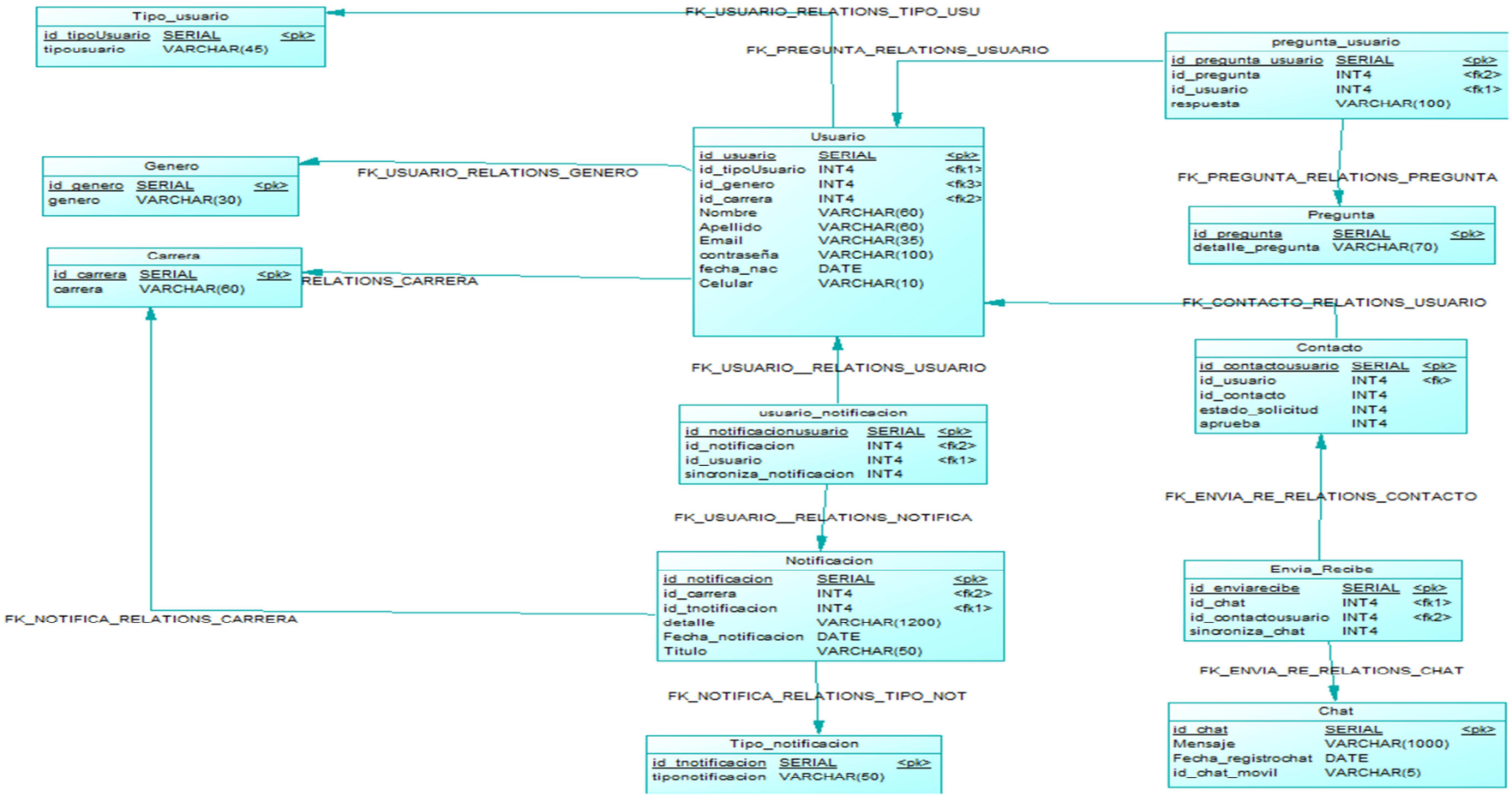

Figura 18. Diseño físico de la base de datos en PostgresSql
Elaborado por: Andrea Martínez y Michael Flores

3.2.1.3 Diccionario de la base de datos del servidor

El Diccionario de la base de datos describe los campos de las tablas incluyendo las características que poseen y la función que cumplen dentro de la aplicación, de igual manera es una guía para el programador en la etapa del desarrollo.

Los mensajes de notificación son almacenados dentro de la tabla 15.

Tabla 15. *Tabla notificación de la base de datos PostgresSql*

NOTIFICACION					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_notificacion	int	4	S	Primary Key	Id clave principal de tabla notificacion
id_tnotificacion	int	4	S	Foreign Key	Id clave foranea de tabla Tipo_notificacion
Titulo	varchar	50	N		Titulo o encabezado de la notificacion
detalle	varchar	60	N		Mensaje de notificacion
Fecha_notificacion	datetime		N		Fecha de creacion de notificacion
Fecha_actualiza_notificacion	datetime		N		Fecha de publicacion de la notificacion
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE notificacion ADD CONSTRAINT pk_notificacion PRIMARY KEY(id_notificacion);	Primary Key				
ALTER TABLE notificacion ADD CONSTRAINT fk_notifica_relations_tipo_not FOREIGN KEY (id_tnotificacion)	Foreign Key	REFERENCES tipo_notificacion (id_tnotificacion) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla NOTIFICACION de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

A fin de clasificar por tipo, las notificaciones de la base de datos PostgresSql se presenta la tabla 16.

Tabla 16. *Tabla tipo_notificacion de la base de datos PostgresSql*

TIPO_NOTIFICACION					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_tnotificacion	int	4	S	Primary Key	Id clave principal de tabla Tipo_notificacion
tiponotificacion	varchar	50	S		Tipo de notificacion
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE tipo_notificacion ADD CONSTRAINT pk_tipo_notificacion PRIMARY KEY(id_tnotificacion);	Primary Key				

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla TIPO_NOTIFICACION de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

La tabla 17. Es una tabla intermediaria también conocida como impositor entre el usuario y la notificación para concatenar al usuario con una o varias notificaciones.

Tabla 17. *Tabla Usuario_Notificación de la base de datos PostgresSql*

USUARIO_NOTIFICACION					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_notificacionusuario	int		S	Primary Key	Id clave principal de tabla notificacionusuario
id_usuario	int		S	Foreign Key	Id clave foranea de la tabla usuario
id_notificacion	int		S	Foreign Key	Id clave foranea de la tabla notificacion
contador	int		N		Verifica si el usuario reviso la notificacion
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE usuario_notificacion ADD CONSTRAINT pk_usuario_notificacion PRIMARY KEY(id_notificacionusuario);	Primary Key				
ALTER TABLE usuario_notificacion ADD CONSTRAINT fk_usuario__relations_notifica FOREIGN KEY (id_notificacion)	Foreign Key	REFERENCES notificacion (id_notificacion) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			
ALTER TABLE usuario_notificacion ADD CONSTRAINT fk_usuario__relations_usuario FOREIGN KEY (id_usuario)	Foreign Key	REFERENCES usuario (id_usuario) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla USUARIO_NOTIFICACION de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

Todos los datos de los usuarios de la aplicación Android se almacenan dentro de la tabla Usuario como se aprecia en la tabla 18.

Tabla 18. *Tabla Usuario de la base de datos PostgresSql*

USUARIO					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_usuario	int	4	S	Primary Key	Id clave principal de tabla usuario
id_tipoUsuario	int	4	S	Foreign Key	Id clave foranea de la tabla tipousuario
id_genero	int	4	S	Foreign Key	Id clave foranea de la tabla genero
id_carrera	int	4	S	Foreign Key	Id clave foranea de la tabla Carrera
Nombre	varchar	60	N		Es el nombre de Usuario del sistema
Apellido	varchar	60	N		Apellido del usuario
Email	varchar	35	S		correo electronico del usuario para su acceso
contraseña	varchar	25	S		Contraseña de ingreso al sistema
fecha_nac	datetime		S		Fecha de nacimiento del usuario
Celular	varchar	10	S		Numero celular del usuario
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE usuario ADD CONSTRAINT pk_usuario PRIMARY KEY(id_usuario);	Primary Key				
ALTER TABLE usuario ADD CONSTRAINT fk_usuario_relations_carrera FOREIGN KEY (id_carrera)	Foreign Key	REFERENCES carrera (id_carrera) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			
ALTER TABLE usuario ADD CONSTRAINT fk_usuario_relations_genero FOREIGN KEY (id_genero)	Foreign Key	REFERENCES genero (id_genero) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			
ALTER TABLE usuario ADD CONSTRAINT fk_usuario_relations_tipo_usu FOREIGN KEY (id_tipousuario)	Foreign Key	REFERENCES tipo_usuario (id_tipousuario) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla USUARIO de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

La tabla 19 es la tabla encargada de concatenar el mensaje de chat con el usuario al que le corresponde.

Tabla 19. *Tabla Envia_Recibe de la base de datos PostgresSql*

ENVIA_RECIBE					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_enviarecibe	int	4	S	Primary Key	Id clave principal de tabla envia recibe
id_usuario	int	4	S	Foreign Key	Id clave foranea de la tabla Usuario
id_chat	int	4	S	Foreign Key	Id clave foranea de la tabla Chat
id_destino	int	4	S	Foreign Key	Id clave foranea de la tabla Usuario como destinatario
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE envia_recibe ADD CONSTRAINT pk_envia_recibe PRIMARY KEY(id_enviarecibe);	Primary Key				
ALTER TABLE envia_recibe ADD CONSTRAINT fk_envia_re_relations_chat FOREIGN KEY (id_chat)	Foreign Key	REFERENCES chat (id_chat) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			
ALTER TABLE envia_recibe ADD CONSTRAINT fk_envia_re_relations_usuario FOREIGN KEY (id_usuario)	Foreign Key	REFERENCES usuario (id_usuario) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla ENVIA_RECIBE de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

El mensaje del chat es almacenado en la tabla 20.

Tabla 20. *Tabla chat de la base de datos PostgresSql*

CHAT					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_chat	int 4	4	S	Primary Key	Id clave principal de la tabla Chat
Mensaje	varchar	120	N		Cuerpo del mensje
Fecha_registro	datetime		N		Fecha de creacion del mensaje
Fecha_update	datetime		N		Fecha de rebicion del mensaje por parte del destinatario
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE chat ADD CONSTRAINT pk_chat PRIMARY KEY(id_chat);	Primary Key				

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla CHAT de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

Para diferenciar el rol de usuario (normal o Administrador) se tiene la tabla 21.

Tabla 21. *Tabla tipo_usuario de la base de datos PostgresSql*

TIPO_USUARIO					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_tipoUsuario	int		S	Primary Key	Id clave principal de la tabla Tipo de usuario
tipousuario	varchar	45	S		Detalle de tipo de usuario del sistema
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE tipo_usuario ADD CONSTRAINT pk_tipo_usuario PRIMARY KEY(id_tipousuario);	Primary Key				

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla TIPO_USUARIO de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

El listado de las carreras que pueden usar la aplicación se encuentra almacenado en la tabla 22.

Tabla 22. *Tabla Carrera de la base de datos PostgresSql*

CARRERA					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_carrera	int		S	Primary Key	Id clave principal de la tabla carrera
carrera	varchar	60	S		Nombre de la carrera Universitaria
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE carrera ADD CONSTRAINT pk_carrera PRIMARY KEY(id_carrera);	Primary Key				

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Carrera de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

El sexo o género del usuario se encuentra en la tabla 23.

Tabla 23. *Tabla Género de la base de datos PostgresSql*

GENERO					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_genero	int		S	Primary Key	Id clave principal de la tabla Genero
genero	varchar	30	S		Nombre de tipo de genero
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE genero ADD CONSTRAINT pk_genero PRIMARY KEY(id_genero);	Primary Key				

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Género de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

La tabla que relaciona a los usuarios para su comunicación es la tabla 24.

Tabla 24. *Tabla Contacto de la base de datos PostgresSql*

CONTACTO					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_contactousuario	int		S	Primary Key	Id clave principal de la tabla Contacto
id_contacto	int		S		id del contacto relacionado
estado_solicitud	int		S		muestra el esatdo de la solicitud
aprueba	int		S		muestra si fue confirmado o no la solicitud de amistad
Constraints					
Nombre del constraint	**Tipo de constraint**	**Definición**			**Descripción/ Comentario**
ALTER TABLE contacto ADD CONSTRAINT pk_contacto PRIMARY KEY(id_contactousuario);	Primary Key				
ALTER TABLE contacto ADD CONSTRAINT fk_contacto_relations_usuario FOREIGN KEY (id_usuario)	Foreign Key	REFERENCES usuario (id_usuario) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT;			

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Contacto de la base de datos PostgresSql que utiliza los servicios web
Elaborado por: Andrea Martínez y Michael Flores

3.2.1.4 Diccionario de la base de datos de la aplicación Android "UPSY"

El listado de las carreras que pueden usar la aplicación Android se encuentra almacenado en la tabla 25.

Tabla 25. *Tabla Carrera de la base de datos SQLite*

CARRERA					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_Carrera	int4	4	S	Primary Key	Id clave principal de tabla carrera
Carrera	Text	60	S		Nombre de la carrera

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Carrera de la base de datos SQLite que utiliza la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

El sexo o género del usuario se encuentra en la tabla 26.

Tabla 26. *Tabla Sexo de la base de datos SQLite*

SEXO					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_sexo	int4	4	S	Primary Key	Id clave principal de tabla Sexo
Sexo	Text	15	S		Nombre de genero

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Sexo de la base de datos SQLite que utiliza la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

En la tabla 27. Se encuentran almacenados todos los contactos del usuario principal de la aplicación Android

Tabla 27. *Tabla Contacto de la base de datos SQLite*

CONTACTO					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_contacto	int4	4	S	Primary Key	Id clave principal de tabla contacto
key_contacto	Text	10	S		Id clave de usuario de todo el sistema
Nombre	Text	60	S		Nombre del usuario externo llamado contacto
Apellido	Text	60	S		Apellido del usuario externo llamado contacto
Ncelular	Text	15	S		Numero celular del contacto

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Contacto de la base de datos SQLite que utiliza la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

Los mensajes entre el usuario y el contacto se encuentran almacenados en la Tabla 28.

Tabla 28. *Tabla Chat de la base de datos SQLite*

CHAT					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_chat	int4	4	S	Primary Key	Id clave principal de tabla chat
id_contacto	int5	4	S	Foreign Key	Id clave foranea de tabla contacto
id_perfil	int6	4	S	Foreign Key	Id clave foranea de tabla perfil
Mensaje_chat	Text	120	S		cuerpo del mensaje
Contador_chat	int4	4	N		permite verificar si se reviso el mensaje
Fecha_Envio	Text	20	S		Fecha de envio del mensaje
Fecha_recepcion	Text	20	S		Fecha de recepcion del mensaje
Fecha_sincronizacion	Text	20	S		Fecha de la ultimasincronizacion con el servidor

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Chat de la base de datos SQLite que utiliza la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

Los datos del usuario principal del dispositivo Android se encuentran registrados en la tabla 29.

Tabla 29. *Tabla Perfil_Usuario de la base de datos SQLite*

PERFIL_USUARIO					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_perfil	int4	4	S	Primary Key	Id clave principal de tabla perfil
id_sexo	int4	4	S	Foreing Key	Id clave foranea de tabla sexo
id_Carrera	int4	4	S	Foreing Key	Id clave foranea de tabla Carrera
key_contacto	Text	10	S		Identificador de usuario de todo el sistema
Nombre	Text	60	S		Nonbre usuario local
Apellido	Text	60	S		Apellido usuario local
Password	Text	20	S		Contraseña de acceso al sistema
Fecha_nacimiento	Text	15	S		Fecha de nacimiento de usuario
Email	Text	50	S		E-mail de usuario
Ncelular	Text	15	S		numero de celular del usuario

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Perfil_Usuario de la base de datos SQLite que utiliza la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

El detalle de las notificaciones recibidas desde el servidor se almacena en la Tabla 30.

Tabla 30. *Tabla Notificación de la base de datos SQLite*

NOTIFICACION					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_notificacion	int4	4	S	Primary Key	Id clave principal de tabla Notificacion
id_tiponotificacion	int4	4	S	Foreing Key	Id clave foranea de tabla Tipo de notificacion
Key_notificacion	Text	10	S		Identificador de notificacion en todo el sistema
Titulo	Text	50	S		Titulo o encabezado de la notificacion
Notificacion	Text	120	S		Mensaje de notificacion
Contador	int4	4	N		Verificador de lectura de mensaje
Fecha_recepcionnotificacion	Text	20	S		Fecha de recepcion de notificacion
Fecha_publicacionnotificacion	Text	20	S		Fecha de publicacion de notificacion

Nota: En la tabla se encuentran los datos con los que está compuesta la tabla Notificación de la base de datos SQLite que utiliza la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

Para clasificar las notificaciones del servidor se considera el tipo de notificación que se almacena en la tabla 31.

Tabla 31. *Tabla Tipo_Notificación de la base de datos SQLite*

TIPO_NOTIFICACION					
Nombre	**Tipo de Dato**	**Longitud**	**Obligatorio S/N**	**Restricción /Constraint**	**Descripción/ Comentario**
id_tiponotificacion	int4	4	S	Primary Key	Id clave principal de tabla Tipo de notificacion
tipo	Text	20	S		Tipo de notificacion

Nota. En la tabla se encuentran los datos con los que está compuesta la tabla Tipo_Notificación de la base de datos SQLite que utiliza la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

3.2.2 Diagrama de clases del proyecto

Debido a la arquitectura del proyecto que está formado por un portal web para el acceso del usuario Administrador para el registro de notificaciones, por un conjunto de servicios web y una aplicación Android que se encontrara instalada en el dispositivo Android.

Se desarrollaron tres diagramas de clases que muestran las clases de las cuales están formadas las aplicaciones anteriormente mencionadas.

3.2.2.1 Diagrama de clases de aplicación Android

El desarrollo de la aplicación Android “UPSY” está compuesto de clases que interactúan entre sí para lograr comunicarse con una interfaz gráfica del dispositivo Android, esta interacción entre las clases de la aplicación Android se muestra en la figura 19.

Diagrama de clases de aplicación Android “UPSY”

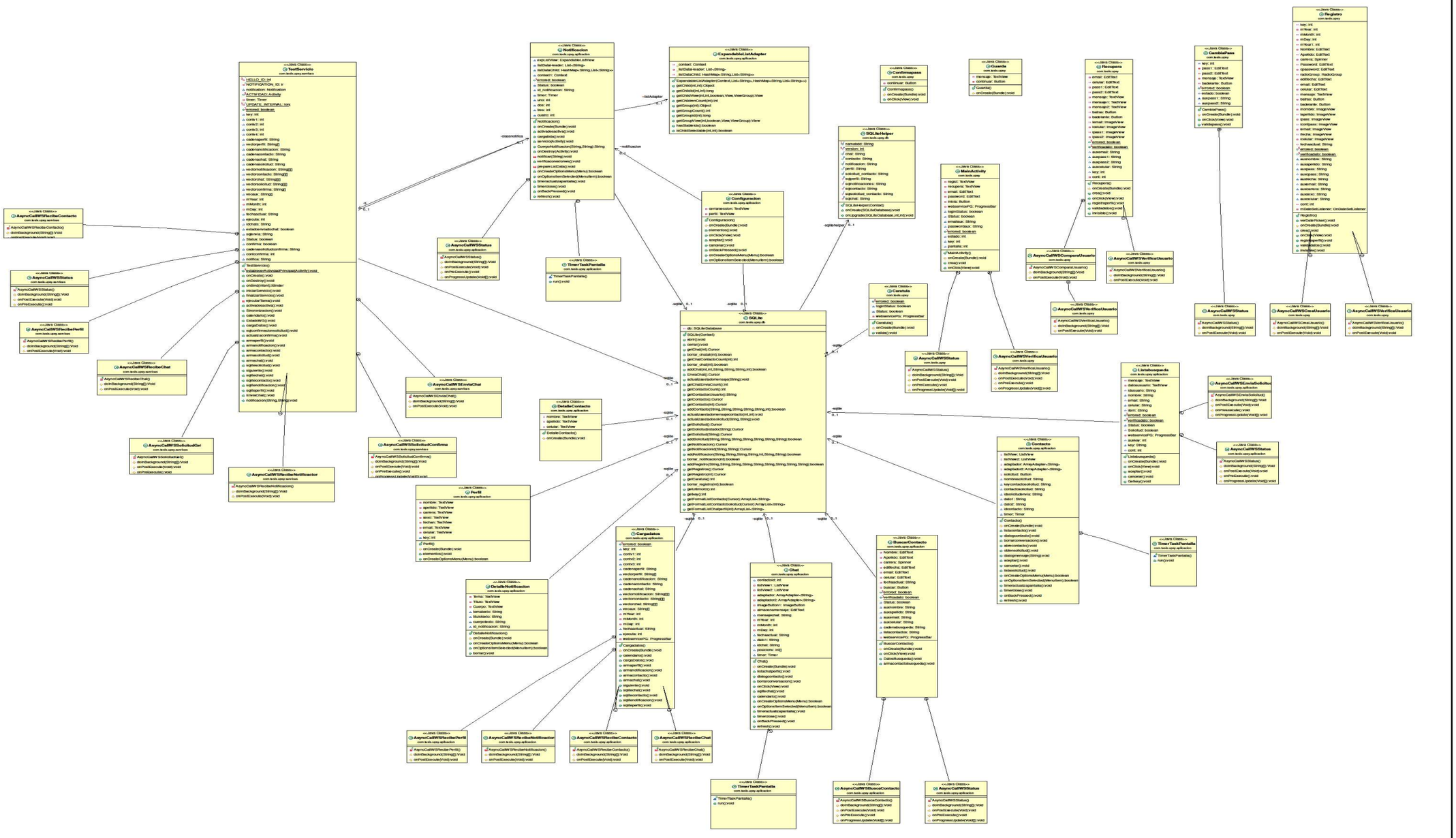

Figura 19. Diagrama de clases aplicación Android “UPSY”
Elaborado por: Andrea Martínez y Michael Flores

3.2.2.2 Diagrama de clases de los servicios web

El diagrama del desarrollo de los servicios web se muestra en la figura 20.

Diagrama de clases del servicios web

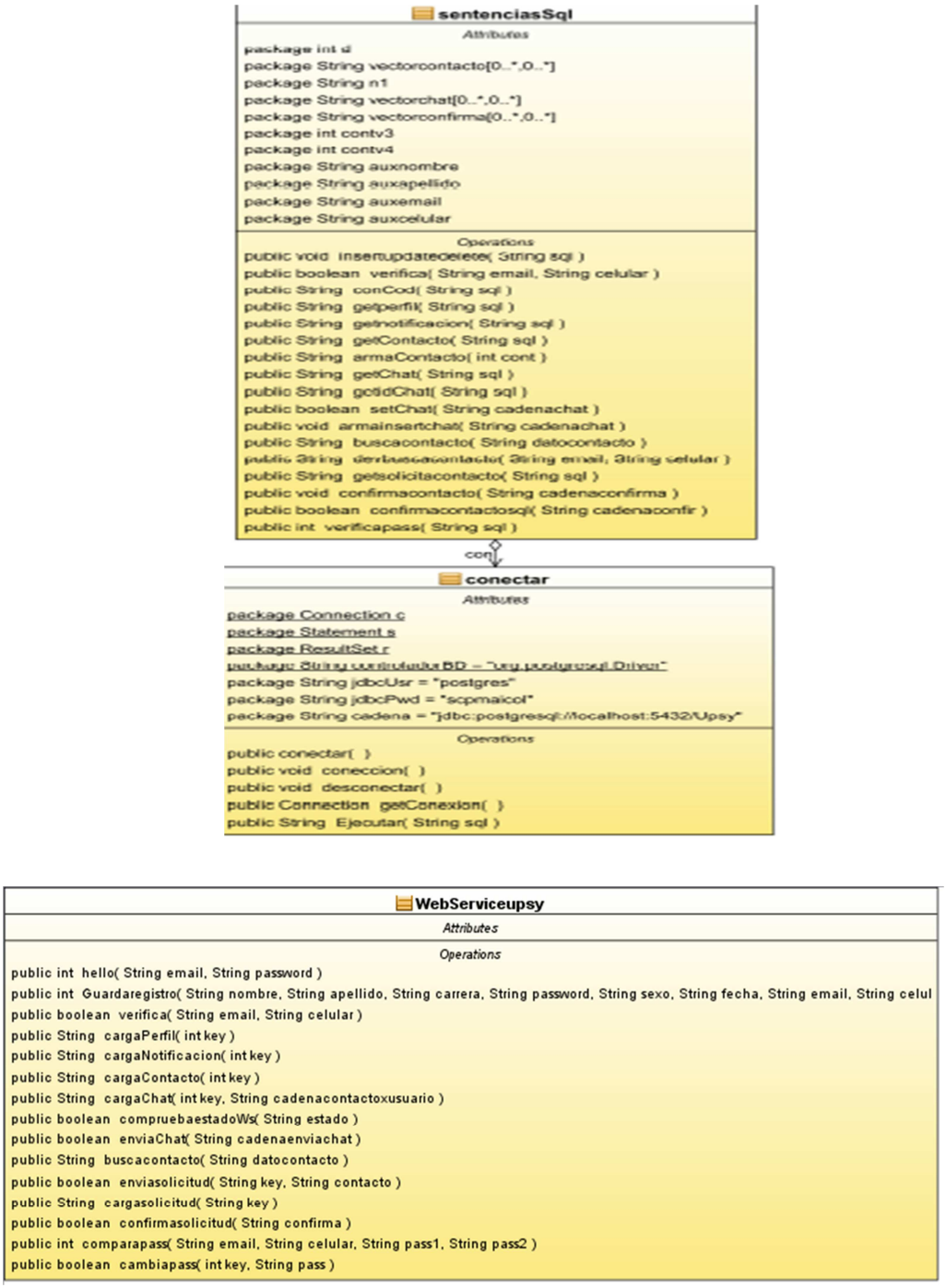

Figura 20. Diagrama de clases de los servicios web
Elaborado por: Andrea Martínez y Michael Flores

3.2.2.3 Diagrama de clases del portal web

Se describe el diagrama de clases del portal web en la figura 21.

Diagrama de clases del portal web

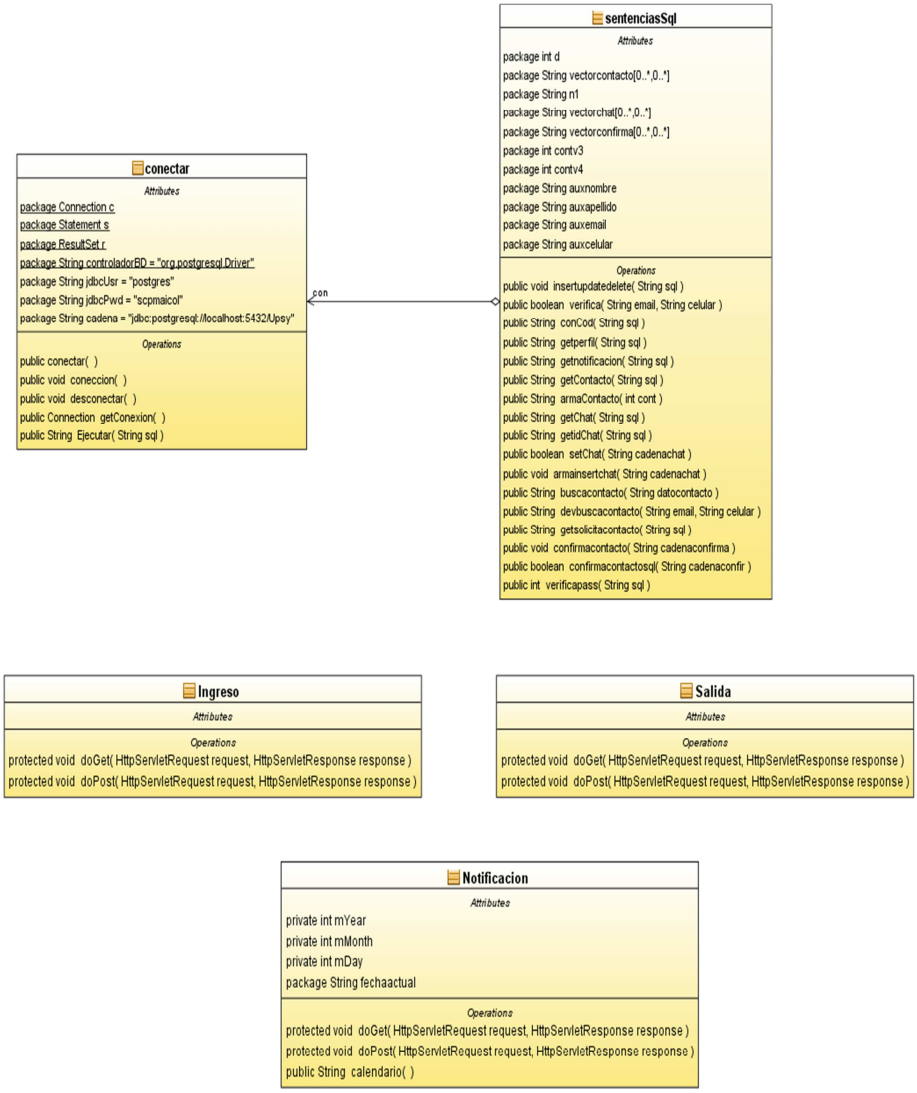

Figura 21. Diagrama de clases del portal web
Elaborado por: Andrea Martínez y Michael Flores

3.2.3 Diccionario de clases

El diccionario de clases describe los atributos, operaciones de cada clase, el nombre de variables, junto con código, tipo de dato de entrada, tipo de dato de retorno, visibilidad de variables e interfaz, valores iniciales, especificación de solo lectura, métodos abstractos, métodos finales, o métodos estáticos que fueron utilizados para cumplir con los objetivos planteados.

Debido a que este proyecto está realizado en dos ambientes el primero para un dispositivo móvil que funciones con Android y el segundo un portal web con una colección de servicios web detallamos las clases de ambos a continuación.

3.2.3.1 Diccionario de clases aplicación Android

Clase CambiaPass

En la clase CambiaPass se encuentra la conexión con la interfaz gráfica de la pantalla de cambio de contraseña, el proceso de verificación y el proceso de cambio de contraseña que se realiza en conjunto con la clase WebService, el detalle de los componentes de la clase CambiaPass se aprecia en la tabla 32. Clase Cambia.

Tabla 32. *Clase CambiaPass*

Atributos							
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial			Solo Lectura
key	key	int	public				
pass1	pass1	EditText	private				
pass2	pass2	EditText	private				
mensaje	mensaje	TextView	private				
badelante	badelante	Button	private				
errored	errored	boolean	public	false			
estado	estado	boolean	public				
auxpass1	auxpass1	String	public				
auxpass2	auxpass2	String	public				
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
onClick	onClick	void	public				
validapass	validapass	void	public				

Nota. En la tabla se encuentran los componentes dela clase cambiapass
Elaborado por: Andrea Martínez y Michael Flores

Clase Carátula

Realiza la comprobación de sesión de usuario iniciado en el dispositivo, si existe una sesión abierta inicia la aplicación en la pantalla de chat o notificación caso contrario

muestra la pantalla de registro o de solicitud de inicio de sesión, el detalle de los componentes de la clase Caratula se muestran en la Tabla 33. Clase Carátula.

Tabla 33. *Clase Carátula*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
sqlite errored loginStatus Status webservicePG	sqlite errored loginStatus Status webservicePG	SQLite boolean boolean boolean ProgressBar		private public public public private		false	
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle) valida	onCreate(Bundle) valida	void vid	protected public				

Nota. En la tabla se encuentran los componentes de la clase carátula
Elaborado por: Andrea Martínez y Michael Flores

Clase Confirmapass

Esta clase interactúa directamente con la clase CambiaPass ya que se muestra siempre y cuando el proceso de validación para el cambios de contraseña es correcto, después de la verificación aparece la pantalla de ingreso de nuevo password y se solicita una confirmación de cambio de password, el detalle de los componentes de la clase Confirmapass se muestra en la tabla 34. Confirmapass.

Tabla 34. *Clase Confirmapass*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
continuar	continuar	Button		private			
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle) onClick(View)	onCreate(Bundle) onClick(View)	void void	protected public				

Nota. En la tabla se encuentran los componentes de la clase confirmapass
Elaborado por: Andrea Martínez y Michael Flores

Clase Guarda

Esta clase recibe los datos validados de la clase registro de usuario y procede a crear el usuario en el servidor externo de base de datos PostgresSql mediante la clase Webservices, además muestra la confirmación de usuario creado, dirigiéndonos a la pantalla principal para el inicio de sesión de usuario, el detalle de los componentes de la clase Guarda se muestra en la tabla 35. Clase Guarda.

Tabla 35. *Clase Guarda*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
mensaje continuar	mensaje continuar	TextView Button		private private			
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle) onClick(View)	onCreate(Bundle) onClick(View)	void void	protected public				

Nota. En la tabla se encuentran los componentes de la clase guarda
Elaborado por: Andrea Martínez y Michael Flores

Clase Recupera

La clase recupera realiza el proceso de validación de datos para la recuperación de contraseña, si los datos son confirmados en su totalidad se procede a el cambio de contraseña caso contrario aparece el mensaje de error en datos, el detalle de los componentes se muestra en la tabla 36.

Tabla 36. *Clase Recupera*

Atributos						
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial		Solo Lectura
email	email	EditText	private			
celular	celular	EditText	private			
pass1	pass1	EditText	private			
pass2	pass2	EditText	private			
mensaje	mensaje	TextView	private			
mensaje1	mensaje1	TextView	private			
mensaje2	mensaje2	TextView	private			
batras	batras	Button	private			
badelante	badelante	Button	private			
iemail	iemail	ImageView	private			
icelular	icelular	ImageView	private			
errored	errored	boolean	public	false		
verificadato	verificadato	boolean	public			
auxemail	auxemail	String	public			
auxpass1	auxpass1	String	public			
auxpass2	auxpass2	String	public			
auxcelular	auxcelular	String	public			
key	key	int	public			
cont	cont	int	private			
Operaciones						
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
onClick	onClick	void	public				
crea	crea	void	public				
registraperfil	registraperfil	void	public				
validadatos	validadatos	void	public				
invisible	invisible	void	public				

Nota. En la tabla se encuentran los componentes de la clase recupera
Elaborado por: Andrea Martínez y Michael Flores

Clase Registro

Se encarga de solicitar los datos para la creación del perfil del usuario nuevo, estos datos son validados para poder proseguir al proceso de creación de perfil que tiene que ver con la clase Guarda como lo vimos anteriormente, el detalle de los componentes de la clase Registro se encuentra en la tabla 37.

Tabla 37. *Clase Registro*

Atributos						
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial		Solo Lectura
key	key	int	private			
mYear	mYear	int	private			
mMonth	mMonth	int	private			
mDay	mDay	int	private			
Nombre	Nombre	EditText	private			
Apellido	Apellido	EditText	private			
carrera	carrera	Spinner	private			
Password	Password	EditText	private			
cpassword	cpassword	EditText	private			
radioGroup	radioGroup	RadioGroup	private			
editfecha	editfecha	EditText	private			
email	email	EditText	private			
celular	celular	EditText	private			
mensaje	mensaje	TextView	private			
batras	batras	Button	private			
badelante	badelante	Button	private			
inombre	inombre	ImageView	private			
iapellido	iapellido	ImageView	private			
ipass	ipass	ImageView	private			
iconfpass	iconfpass	ImageView	private			
iemail	iemail	ImageView	private			
ifecha	ifecha	ImageView	private			
icelular	icelular	ImageView	private			
fechaactual	fechaactual	String	private			
errored	errored	boolean	public	false		
verificadato	verificadato	boolean	public			
auxnombre	auxnombre	String	public			
auxapellido	auxapellido	String	public			
auxpass	auxpass	String	public			
auxcpass	auxcpass	String	public			
auxfecha	auxfecha	String	public			
auxemail	auxemail	String	public			
auxcarrera	auxcarrera	String	public			
auxsexo	auxsexo	String	public			
auxcelular	auxcelular	String	public			
cont	cont	int	private			
mDateSetListener	mDateSetListener	OnDateSetListener	private			
Operaciones						
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
verDatePicker()	verDatePicker()	void	public				
onCreate(Bundle)	onCreate(Bundle)	void	protected				
crea	crea	void	public				
onClick(View)	onClick(View)	void	public				
registraperfil()	registraperfil()	void	public				
validadatos	validadatos	void	public				
invisible	invisible	void	public				

Nota. En la tabla se encuentran los componentes de la clase registro
Elaborado por: Andrea Martínez y Michael Flores

Clase BuscarContacto

La búsqueda de contacto para su posterior vinculación se realiza por medio de la presente clase, que realiza una verificación para encontrar similitudes con los datos

de búsqueda, si encuentra contactos con los datos buscados se procede al envió de una solicitud de amistad o contacto para poder vincular los usuarios, el detalle de los componentes se encuentran detallados en la tabla 38. Clase BuscarContacto.

Tabla 38. *Clase BuscarContacto*

Atributos					
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura
Nombre	Nombre	EditText	private		
Apellido	Apellido	EditText	private		
carrera	carrera	Spinner	private		
editfecha	editfecha	EditText	private		
email	email	EditText	private		
celular	celular	EditText	private		
fechaactual	fechaactual	String	private		
buscar	buscar	Button	private		
errored	errored	boolean	public	false	
verificadato	verificadato	boolean	public		
Status	Status	boolean	public		
auxnombre	auxnombre	String	public		
auxapellido	auxapellido	String	public		
auxemail	auxemail	String	public		
auxcelular	auxcelular	String	public		
cadenabusqueda	cadenabusqueda	String	public		
listacontactos	listacontactos	String	public		
webservicePG	webservicePG	ProgressBar	private		

Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
onClick(View)	onClick(View)	void	public				
DatosBusqueda	DatosBusqueda	void	public				
armacontactobusqueda()	armacontactobusqueda()	void	public				
launchRingDialog()	launchRingDialog()	void	public				

Nota. En la tabla se encuentran los componentes de la clase buscar contacto
Elaborado por: Andrea Martínez y Michael Flores

Clase Cargadatos

Este es el proceso inicial después del inicio de sesión del usuario, donde se realiza una petición de datos del usuario al servidor de base de datos PostgresSql, para almacenar en el dispositivo Smartphone, el detalle de los componentes de la clase Cargadatos se encuentran en la tabla 39.

Tabla 39. *Clase Cargadatos*

Atributos					
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura
errored	errored	boolean	private	false	
key	key	int	private		
contv1	contv1	int	private		
contv2	contv2	int	private		
contv3	contv3	int	private		
cadenaperfil	cadenaperfil	String	private		
vectorperfil	vectorperfil	String[]	private		
cadenanotificacion	cadenanotificacion	String	private		
cadenacontacto	cadenacontacto	String	public		
cadenachat	cadenachat	String	public		
vectornotificacion	vectornotificacion	String[][]	public		
vectorcontacto	vectorcontacto	String[][]	public		
vectorchat	vectorchat	String[][]	public		
vecaux	vecaux	String[]	public		
sqlite	sqlite	SQLite	private		
mYear	mYear	int	private		
mMonth	mMonth	int	private		
mDay	mDay	int	private		
fechaactual	fechaactual	String	public		
ejecuta	ejecuta	int	public		
webservicePG	webservicePG	ProgressBar	private		

Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
calendario()	calendario()	void	public				
cargaDatos()	cargaDatos()	void	public				
armaperfil()	armaperfil()	void	public				
armanotificacion()	armanotificacion()	void	public				
armacontacto()	armacontacto()	void	public				
armachat()	armachat()	void	public				
siguiente()	siguiente()	void	public				
sqlitechat()	sqlitechat()	void	public				
sqlitecontacto()	sqlitecontacto()	void	public				
sqlitenotificacion()	sqlitenotificacion()	void	public				
sqliteperfil()	sqliteperfil()	void	public				

Nota. En la tabla se encuentran los componentes de la clase cargadatos
Elaborado por: Andrea Martínez y Michael Flores

Clase Chat

En la clase chat es la interfaz de presentación al usuario de los mensajes enviados y recibidos por los contactos, se instancia con la selección del contacto de este modo no se confunden las conversaciones, el detalle de los componentes de la clase de chat se encuentran en la tabla 40.

Tabla 40. *Clase Chat*

Atributos					
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura
contactoid	contactoid	int	int		
sqlite	sqlite	SQLite	private		
listView1	listView1	ListView	private		
listView2	listView2	ListView	private		
adaptador	adaptador	ArrayAdapter<String>	private		
adaptador2	adaptador2	ArrayAdapter<String>	private		
imageButton1	imageButton1	ImageButton	private		
almacenamensaje	almacenamensaje	EditText	private		
mensajechat	mensajechat	String	public		
mYear	mYear	int	private		
mMonth	mMonth	int	private		
mDay	mDay	int	private		
fechaactual	fechaactual	String	public		
dato1	dato1	String	public		
idchat	idchat	String	public		
posicionv	posicionv	int[]	public		
timer	timer	Timer	public		

Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
listachatperfil()	listachatperfil()	void	public				
dialogcontacto()	dialogcontacto()	void	public				
borrarconversacion()	borrarconversacion()	void	public				
onClick(View)	onClick(View)	void	public				
sqlitechat()	sqlitechat()	void	public				
calendario()	calendario()	void	public				
onCreateOptionsMenu(Menu)	onCreateOptionsMenu(Menu)	boolean	public				
onOptionsItemSelected(MenuItem)	onOptionsItemSelected(MenuItem)	boolean	public				
timeractualizapantalla()	timeractualizapantalla()	void	public				
timerclose()	timerclose()	void	public				
onBackPressed()	onBackPressed()	void	public				
refresh()	refresh()	void	public				

Nota. En la tabla se encuentran los componentes de la clase chat
Elaborado por: Andrea Martínez y Michael Flores

Clase Configuración

En la clase configuración se encuentran parámetros de verificación del perfil, cambios en la aplicación Android, opciones de cierre de sesión y otros parámetros de configuración, el detalle de los componentes de la clase configuración en la tabla 41.

Tabla 41. *Clase Configuración*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
cerrarsession	email	TextView		private			
perfil	celular	TextView		private			
sqlite	pass1	SQLite		private			
notificacion	cerrarsession	Notificacion		public			
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
elementos()	elementos()	void	public				
onClick(View)	onClick(View)	void	public				
aceptar()	aceptar()	void	public				
cancelar()	cancelar()	void	public				
onCreateOptionsMenu(Menu)	onCreateOptionsMenu(Menu)	boolean	public				
onOptionsItemSelected(MenuItem)	onOptionsItemSelected(MenuItem)	boolean	public				

Nota. En la tabla se encuentran los componentes de la clase configuración
Elaborado por: Andrea Martínez y Michael Flores

Clase Contacto

Enlista los contactos confirmados, la lista de solicitud de contactos y la búsqueda de contactos, se detallan los componentes de la clase contacto en la tabla 42.

Tabla 42. *Clase Contacto*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
listView	listView	ListView		private			
listView2	listView2	ListView		private			
adaptador	adaptador	ArrayAdapter<String>		private			
adaptador2	adaptador2	ArrayAdapter<String>		private			
sqlite	sqlite	SQLite		private			
solicitud	solicitud	Button		private			
nombresolicitud	nombresolicitud	String		public			
keycontactosolicitud	keycontactosolicitud	String		public			
contactosolicitud	contactosolicitud	String		public			
idsolicitudenvia	idsolicitudenvia	String		public			
dato1	dato1	String		public			
dato2	dato2	String		public			
idcontacto	idcontacto	String		public			
timer	timer	Timer		public			
Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
listacontacto()	listacontacto()	void	public				
listacontacto()	listacontacto()	void	public				
borrarconversacion()	borrarconversacion()	void	public				
abrecontacto()	abrecontacto()	void	public				
obtensolicitud()	obtensolicitud()	void	public				
dialogmensaje(String)	dialogmensaje(String)	void	public				
aceptar()	aceptar()	void	public				
cancelar()	cancelar()	void	public				
listasolicitud()	listasolicitud()	void	public				
onCreateOptionsMenu(Menu)	onCreateOptionsMenu(Menu)	boolean	public				
onOptionsItemSelected(MenuItem)	onOptionsItemSelected(MenuItem)	boolean	public				
timeractualizapantalla()	timeractualizapantalla()	void	public				
timerclose()	timerclose()	void	public				
onBackPressed()	onBackPressed()	void	public				
refresh()	refresh()	void	public				

Nota. En la tabla se encuentran los componentes de la clase contacto
Elaborado por: Andrea Martínez y Michael Flores

Clase DetalleContacto

Los datos del contacto se muestran en esta pantalla, como nombre, apellido, teléfono, etc. El detalle de los componentes de la clase DetalleContacto se encuentran en la tabla 43.

Tabla 43. *Clase DetalleContacto*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
sqlite	sqlite	SQLite		private			
nombre	nombre	TextView		private			
apellido	apellido	TextView		private			
celular	celular	TextView		private			
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				

Nota. En la tabla se encuentran los componentes de la clase detallecontacto
Elaborado por: Andrea Martínez y Michael Flores

Clase DetalleNotificacion

El cuerpo de las notificaciones son desplegadas en esta pantalla, permitiendo verificar el detalle de su mensaje, la descripción de los componentes de la clase DetalleNotificacion en la tabla 44.

Tabla 44. *Clase DetalleNotificacion*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
Tema	Tema	TextView		private			
Titulo	Titulo	TextView		private			
Cuerpo	Cuerpo	TextView		private			
tematexto	tematexto	String		public			
titulotexto	titulotexto	String		public			
cuerpotexto	cuerpotexto	String		public			
id_notificacion	id_notificacion	String		public			
sqlite	sqlite	SQLite		private			
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				

Nota. En la tabla se encuentran los componentes de la clase detallenotificacion
Elaborado por: Andrea Martínez y Michael Flores

Clase ExpandableListAdapter

Esta clase se utiliza para construir un componente de despliegue de las notificaciones, esta clase es encargada de establecer la estructura de las notificaciones, el tipo de notificaciones y las notificaciones agrupadas que están dentro de este tipo, el detalle de los componentes de la clase ExpandableListAdapter se encuentra en la tabla 45.

Tabla 45. *Clase ExpandableListAdapter*

Atributos							
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura		
_context	_context	Context	private				
_listDataHeader	_listDataHeader	List<String>	private				
_listDataChild	_listDataChild	HashMap<String, List<String>>	private				
Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
getChild(int, int)	getChild(int, int)	Object	public				
getChildId(int, int)	getChildId(int, int)	long	public				
getChildId(int, int)	getChildId(int, int)	long	public				
getChildView(int, int, boolean, View, ViewGroup)	getChildView(int, int, boolean, View, ViewGroup)	View	public				
getChildrenCount(int)	getChildrenCount(int)	int	public				
getGroup(int)	getGroup(int)	Object	public				
getGroupCount()	getGroupCount()	int	public				
getGroupId(int)	getGroupId(int)	long	public				
getGroupView(int, boolean, View, ViewGroup)	getGroupView(int, boolean, View, ViewGroup)	View	public				
hasStableIds()	hasStableIds()	boolean	public				
isChildSelectable(int, int)	isChildSelectable(int, int)	boolean	public				

Nota. En la tabla se encuentran los componentes de la clase ExpandableListAdapter
Elaborado por: Andrea Martínez y Michael Flores

Clase Listabusqueda

Esta clase se utiliza para mostrar los contactos encontrados después de la búsqueda de contactos, si existen coincidencias muestra un mensaje para enviar una solicitud de contacto caso contrario muestra un mensaje de error, el detalle de los componentes de la clase Listabusqueda se muestra en la tabla 46.

Tabla 46. *Clase Listabusqueda*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
mensaje	mensaje	TextView		private			
datosusuario	datosusuario	TextView		private			
idusuario	idusuario	String		public			
nombre	nombre	String		public			
email	email	String		public			
celular	celular	String		public			
item	item	String		public			
errored	errored	boolean		public			
verificadato	verificadato	boolean		public			
Status	Status	boolean		public			
Solicitud	Solicitud	boolean		public			
webservicePG	webservicePG	ProgressBar		private			
auxkey	auxkey	int		public			
key	key	String		public			
sqlite	sqlite	SQLite		private			
Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
onClick(View)	onClick(View)	void	public				
aceptar()	aceptar()	void	public				
cancelar()	cancelar()	void	public				
Getkey()	Getkey()	void	public				

Nota. En la tabla se encuentran los componentes de la clase listabusqueda
Elaborado por: Andrea Martínez y Michael Flores

Clase Notificación

Es la pantalla principal de la aplicación donde se encuentra el listado de las notificaciones brindándonos acceso a todas las opciones de la aplicación. El contenido de la Clase Notificación se muestra en la tabla 47.

Tabla 47. *Clase Notificacion*

Atributos					
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura
s	s	TestServicio	private		
listAdapter	listAdapter	ExpandableListAdapter	public		
expListView	expListView	ExpandableListView	public		
listDataHeader	listDataHeader	List<String>	public		
listDataChild	listDataChild	HashMap<String, List<String>>	public		
sqlite	sqlite	SQLite	private		
context1	context1	Context	public		
errored	errored	boolean	public	false	
Status	Status	boolean	public		
id_notificacion	id_notificacion	String	public		
timer	timer	Timer	private		

Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	public				
activadesactiva()	activadesactiva()	void	public				
cargalista()	cargalista()	void	public				
servicio(Activity)	servicio(Activity)	void	public				
CuerpoNotificacion(String, String)	CuerpoNotificacion(String, String)	void	public				
onDestroy(Activity)	onDestroy(Activity)	void	public				
notificar(String)	notificar(String)	void	private				
verificaconexionws()	verificaconexionws()	void	public				
prepareListData()	prepareListData()	void	private				
onCreateOptionsMenu(Menu)	onCreateOptionsMenu(Menu)	void	public				
onOptionsItemSelected(MenuItem)	onOptionsItemSelected(MenuItem)	void	public				
timeractualizapantalla()	timeractualizapantalla()	void	public				
timerclose()	timerclose()	void	public				
onBackPressed()	onBackPressed()	void	public				
refresh()	refresh()	void	public				

Nota. En la tabla se encuentran los componentes de la clase Notificacion
Elaborado por: Andrea Martínez y Michael Flores

Clase Perfil

Esta pantalla muestra los datos con los que se creó el usuario, el detalle de los componentes de la Clase Perfil se encuentran en la tabla 48. Clase Perfil.

Tabla 48. *Clase Perfil*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
nombre	nombre	TextView		private			
apellido	apellido	TextView		private			
carrera	carrera	TextView		private			
sexo	sexo	TextView		private			
fechan	fechan	TextView		private			
email	email	TextView		private			
celular	celular	TextView		private			
key	key	int		public			
sqlite	sqlite	SQLite		private			
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
onCreate(Bundle)	onCreate(Bundle)	void	protected				
elementos()	elementos()	void	public				
onCreateOptionsMenu(Menu)	onCreateOptionsMenu(Menu)	void	public				
onOptionsItemSelected(MenuItem)	onOptionsItemSelected(MenuItem)	void	public				

Nota. En la tabla se encuentran los componentes de la clase perfil de usuario
Elaborado por: Andrea Martínez y Michael Flores

Clase SQLite

Esta clase es una de las más importantes, debido a que contiene todos los métodos para el ingreso, consulta, actualización y borrado de datos de la base de datos SQLite, los componentes de la Clase SQLite se muestran en la tabla 49. Clase SQLite.

Tabla 49. *Clase SQLite*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
sqlitehelper	sqlitehelper	SQLiteHelper		private			
db	db	SQLiteDatabase		private			
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
abrir	abrir	void	public				
cerrar	cerrar	void	public				
getChat	getChat	Cursor	public				
borrar_chatall	borrar_chatall	boolean	public				
getChatContactoCount	getChatContactoCount	int	public				
borrar_chat	borrar_chat	boolean	public				
addChat	addChat	boolean	public				
EnviaChat	EnviaChat	Cursor	public				
actualizaestadomensaje	actualizaestadomensaje	void	public				
getChatEnviaCount	getChatEnviaCount	int	public				
getContactoCount	getContactoCount	int	public				
getContactoxUsuario	getContactoxUsuario	String	public				
getContacto	getContacto	Cursor	public				
getContacto	getContacto	Cursor	public				
addContacto	addContacto	boolean	public				
actualizaestadosolicitud	actualizaestadosolicitud	void	public				
getSolicitud	getSolicitud	Cursor	public				
getSolicitudestado	getSolicitudestado	Cursor	public				
getSolicitud	getSolicitud	Cursor	public				
addSolicitud	addSolicitud	boolean	public				
getNotificacion	getNotificacion	Cursor	public				
getNotificacionid	getNotificacionid	Cursor	public				
addNotificacion	addNotificacion	boolean	public				
borrar_notificacion	borrar_notificacion	boolean	public				
addRegistro	addRegistro	boolean	public				
getRegistros	getRegistros	Cursor	public				
getRegistro	getRegistro	Cursor	public				
getCaratula	getCaratula	int	public				
borrar_registro	borrar_registro	boolean	public				
getUltimoID	getUltimoID	int	public				
getFormatListContacto	getFormatListContacto	ArrayList<String>	public				
getFormatListContactoSolicitud	getFormatListContactoSolicitud	ArrayList<String>	public				
getFormatListChatperfil	getFormatListChatperfil	ArrayList<String>	public				

Nota. En la tabla se encuentran los componentes de la clase Sqlite
Elaborado por: Andrea Martínez y Michael Flores

Clase SQLiteHelper

En esta clase se realiza la conexión con la base de datos SQLite, el detalle de los componentes de la Clase SQLiteHelper se muestran en la tabla 50. Clase SQLiteHelper.

Tabla 50. *Clase SQLiteHelper*

Atributos							
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura		
namebdd	namebdd	String	public	Upsybdd			
version	version	int	private	13			
chat	chat	String	public	chat			
contacto	contacto	String	public	contacto			
notificacion	notificacion	String	public	notificacion			
perfil	perfil	String	public	perfil			
solicitud_contacto	solicitud_contacto	String	public	solicitud_contacto			
sqlperfil	sqlperfil	String	private				
sqlnotificaciones	sqlnotificaciones	String	private				
sqlcontacto	sqlcontacto	String	private				
sqlsolicitud_contacto	sqlsolicitud_contacto	String	private				
sqlchat	sqlchat	String	private				
Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
SQLiteHelper(Context)	SQLiteHelper(Context)		public				
onCreate(SQLiteDatabase)	onCreate(SQLiteDatabase)	void	public				
onUpgrade(SQLiteDatabase, int, int)	onUpgrade(SQLiteDatabase, int, int)	void	public				

Nota. En la tabla se encuentran los componentes de la clase Sqlitehelper
Elaborado por: Andrea Martínez y Michael Flores

Clase TestServicio

La encargada de la sincronización de las notificaciones, mensajes, aceptación de contactos y demás actualizaciones de información es la clase TestServicio, se realiza mediante la conformación de una servicio el cual es creado por la clase anteriormente mencionada, esta clase se ejecuta en segundo no tiene una interfaz gráfica, solo se muestra su interacción con la aplicación al recibir y enviar información. El detalle de los componentes de la Clase TestServicio se encuentra en la tabla 51.

Tabla 51. *Clase TestServicio*

Atributos					
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura
HELLO_ID	HELLO_ID	int	private	1	
NOTIFICATION_ID	NOTIFICATION_ID	int	private	1010	
notification	notification	Notification	public		
ACTIVIDAD	ACTIVIDAD	Activity	public		
timer	timer	Timer	private	null	
UPDATE_INTERVAL	UPDATE_INTERVAL	long	private	5000	
errored	errored	boolean	public	false	
key	key	int	public		
contv1	contv1	int	public		
contv2	contv2	int	public		
contv3	contv3	int	public		
contv4	contv4	int	public		
cadenaperfil	cadenaperfil	String	public		
vectorperfil	vectorperfil	String[]	public		
cadenanotificacion	cadenanotificacion	String	public		
cadenacontacto	cadenacontacto	String	public		
cadenachat	cadenachat	String	public		
cadenasolicitud	cadenasolicitud	String	public		
vectornotificacion	vectornotificacion	String[][]	public		
vectorcontacto	vectorcontacto	String[][]	public		
vectorchat	vectorchat	String[][]	public		
vectorsolicitud	vectorsolicitud	String[][]	public		
vectorconfirma	vectorconfirma	String[]	public		
vecaux	vecaux	String[]	public		
sqlite	sqlite	SQLite	private		
classnotifica	classnotifica	Notificacion	private		
mYear	mYear	int	private		
mMonth	mMonth	int	private		
mDay	mDay	int	private		
fechaactual	fechaactual	String	public		
ejecuta	ejecuta	int	public		
idchats	idchats	String	public		
estadoenviadochat	estadoenviadochat	boolean	public		
sqlenvia	sqlenvia	String	public	contacto	
Status	Status	boolean	public	notificacion	
confirma	confirma	boolean	public	perfil	
cadenasolicitudconfirma	cadenasolicitudconfirma	String	public	solicitud_contacto	
contconfirma	contconfirma	int	private		
notifica	notifica	String	private		

Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
establecerActividadPrincipal(Activity)	establecerActividadPrincipal(Activity)	void	public			X	
onCreate()	onCreate()	void	public				
onDestroy()	onDestroy()	void	public				
onBind(Intent)	onBind(Intent)	IBinder	public				
iniciarServicio()	iniciarServicio()	void	public				
finalizarServicio()	finalizarServicio()	void	public				
ejecutarTarea()	ejecutarTarea()	void	private				
activadesactiva()	activadesactiva()	void	public				
Sincronizacion()	Sincronizacion()	void	public				
calendario()	calendario()	void	public				
EstadoWS()	EstadoWS()	void	public				
cargaDatos()	cargaDatos()	void	public				
sqlconfirmacionsolicitud()	sqlconfirmacionsolicitud()	void	public				
actualizaconfirma()	actualizaconfirma()	void	public				
armaperfil()	armaperfil()	void	public				
armanotificacion()	armanotificacion()	void	public				
armacontacto()	armacontacto()	void	public				
armasolicitud()	armasolicitud()	void	public				
armachat()	armachat()	void	public				
sqlitesolicitud()	sqlitesolicitud()	void	public				
siguiente()	siguiente()	void	public				
sqlitechat()	sqlitechat()	void	public				
sqlitecontacto()	sqlitecontacto()	void	public				
sqlitenotificacion()	sqlitenotificacion()	void	public				
sqliteperfil()	sqliteperfil()	void	public				
EnviaChat()	EnviaChat()	void	public				
notificacion(String, String)	notificacion(String, String)	void	public				
AsyncCallWSRecibePerfil	AsyncCallWSRecibePerfil	class	private				
AsyncCallWSRecibeNotificacion	AsyncCallWSRecibeNotificacion	class	private				
AsyncCallWSRecibeContacto	AsyncCallWSRecibeContacto	class	private				
AsyncCallWSRecibeChat	AsyncCallWSRecibeChat	class	private				
AsyncCallWSEnviaChat	AsyncCallWSEnviaChat	class	private				
AsyncCallWSStatus	AsyncCallWSStatus	class	private				
AsyncCallWSSolicitudGet	AsyncCallWSSolicitudGet	class	private				
AsyncCallWSSolicitudConfirma	AsyncCallWSSolicitudConfirma	class	private				

Nota. En la tabla se encuentran los componentes de la clase testservicio
Elaborado por: Andrea Martínez y Michael Flores

Clase WebService

La clase WebService contiene los métodos para comunicarse la aplicación con la base de datos externa, también se encuentra el direccionamiento IP del servidor web donde se encuentran alojados los servicios web, el detalle de los componentes de la Clase WebService se encuentra en la tabla 52. Clase WebService.

Tabla 52. *Clase WebService*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
NAMESPACE	NAMESPACE	String		private			
URL	URL	String		private			
SOAP_ACTION	SOAP_ACTION	String		private			
Operaciones							
Nombre	Código	Tipo Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
invokeWS	invokeWS	boolean	public			X	
invokeLoginWS	invokeLoginWS	int	public			X	
verificaWS	verificaWS	boolean	public			X	
guardaregistroWS	guardaregistroWS	int	public			X	
cargaPerfilWS	cargaPerfilWS	String	public			X	
cargaNotificacionWS	cargaNotificacionWS	String	public			X	
cargaContactoWS	cargaContactoWS	String	public			X	
cargaChatWS	cargaChatWS	boolean	public			X	
enviaChatWS	enviaChatWS	String	public			X	
buscaContactoWS	buscaContactoWS	int	public			X	
enviasolicitudWS	enviasolicitudWS	boolean	public			X	
solicitudcontactoWS	solicitudcontactoWS	String	public			X	
confirmasolicitudWS	confirmasolicitudWS	boolean	public			X	
comparapassWS	comparapassWS	int	public			X	
cambiapassWS	cambiapassWS	boolean	public			X	

Nota. En la tabla se encuentran los componentes de la clase webservice
Elaborado por: Andrea Martínez y Michael Flores

3.2.3.2 Diccionario de clases servicios web

Clase WebServiceupsy

En esta clase se encuentran los servicios web vigentes para el funcionamiento de la aplicación, el detalle de los componentes de la clase WebServiceupsy se muestra en la tabla 53.

Tabla 53. *Clase WebServiceupsy*

Atributos					
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura
a	a	boolean	public		
aux	aux	int	public	0	
sen	sen	sentenciasSql	public		
genero	genero	int	public		
carreras	carreras	int	public		
key	key	int	public	0	
aux	aux	boolean	public	true	
aux	aux	String	public	111	
correcta	correcta	boolean	public		
auxsolicita	auxsolicita	String	public		
auxconfirma	auxconfirma	boolean	public		
auxcompara	auxcompara	int	public		
confpass	confpass	boolean	public	true	

Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
hello	hello	int	public				
Guardaregistro	Guardaregistro	int	public				
verifica	verifica	boolean	public				
cargaPerfil	cargaPerfil	String	public				
cargaNotificacion	cargaNotificacion	String	public				
cargaContacto	cargaContacto	String	public				
cargaChat	cargaChat	String	public				
compruebaestadoWs	compruebaestadoWs	boolean	public				
enviaChat	enviaChat	boolean	public				
buscacontacto	buscacontacto	String	public				
enviasolicitud	enviasolicitud	boolean	public				
cargasolicitud	cargasolicitud	String	public				
comparapass	comparapass	int	public				
cambiapass	cambiapass	boolean	public				

Nota. En la tabla se encuentran los componentes de la clase WebServiceupsy
Elaborado por: Andrea Martínez y Michael Flores

Clase conectar

La clase conectar establece la conexión entre el servidor de servicios web y la base de datos PostgresSql, el detalle de los componentes de la clase conectar se encuentran en la tabla 54.

Tabla 54. *Clase conectar*

Atributos					
Nombre	Código	Tipo de Dato	Visibilidad	Valor Inicial	Solo Lectura
c	c	Connection	public		
s	s	Statement	public		
r	r	ResultSet	public		
controladorBD	controladorBD	String	public		
jdbcUsr	jdbcUsr	String	public		
jdbcPwd	jdbcPwd	String	public		
cadena	cadena	String	public		

Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
coneccion	hello	void	public				
desconectar	desconectar	void	public				
Ejecutar	Ejecutar	String	public				

Nota. En la tabla se encuentran los componentes de la clase conectar
Elaborado por: Andrea Martínez y Michael Flores

Clase sentenciasSql

Esta clase permite insertar, consultar, borrar y actualizar la información en la base de datos, la clase mencionada contiene todos los procesos que permiten el manejo de la base de datos principal, el detalle de los componentes de la clase sentenciasSql se encuentran en la tabla 55. Clase sentenciasSql.

Tabla 55. *Clase sentenciasSql*

Atributos							
Nombre	Código	Tipo de Dato		Visibilidad		Valor Inicial	Solo Lectura
d	d	int		public			
con	con	conectar		public			
vectorcontacto	vectorcontacto	String		public			
n1	n1	String		public			
vectorchat	vectorchat	String		public			
vectorconfirma	vectorconfirma	String		public			
contv3	contv3	int		public			
contv4	contv4	int		public			
auxnombre	auxnombre	String		public			
auxapellido	auxapellido	String		public			
auxemail	auxemail	String		public			
auxcelular	auxcelular	String		public			
Operaciones							
Nombre	Código	Tipo de Retorno	Visibilidad	Abstracto	Final	Estático	Solo Lectura
insertupdatedelete	insertupdatedelete	void	public				
verifica	verifica	boolean	public				
conCod	conCod	String	public				
getperfil	getperfil	String	public				
getnotificacion	getnotificacion	String	public				
armaContacto	armaContacto	String	public				
getChat	getChat	String	public				
getidChat	getidChat	String	public				
setChat	setChat	boolean	public				
armainsertchat	armainsertchat	void	public				
buscacontacto	buscacontacto	String	public				
devbuscacontacto	devbuscacontacto	String	public				
getsolicitacontacto	getsolicitacontacto	String	public				
confirmacontacto	confirmacontacto	void	public				
confirmacontactosql	confirmacontactosql	boolean	public				
verificapass	verificapass	int	public				

Nota. En la tabla se encuentran los componentes de la clase sentenciasSql
Elaborado por: Andrea Martínez y Michael Flores

3.2.4 Diagrama de casos de uso

Los diagramas de caso de uso están compuestos por gráficos que describen el comportamiento del sistema, su funcionalidad así como la interacción con los actores que participan en el proceso. Los casos de uso para el sistema están divididos en dos

partes por un lado los diagramas del servidor web y por el otro los diagramas del cliente Android.

3.2.4.1 Diagramas de caso de uso del servidor web

Las actividades que se deben realizar para poder acceder al portal web por parte del Administrador de la aplicación web se muestran en la figura 22.

Caso de uso Acceso al Portal Web

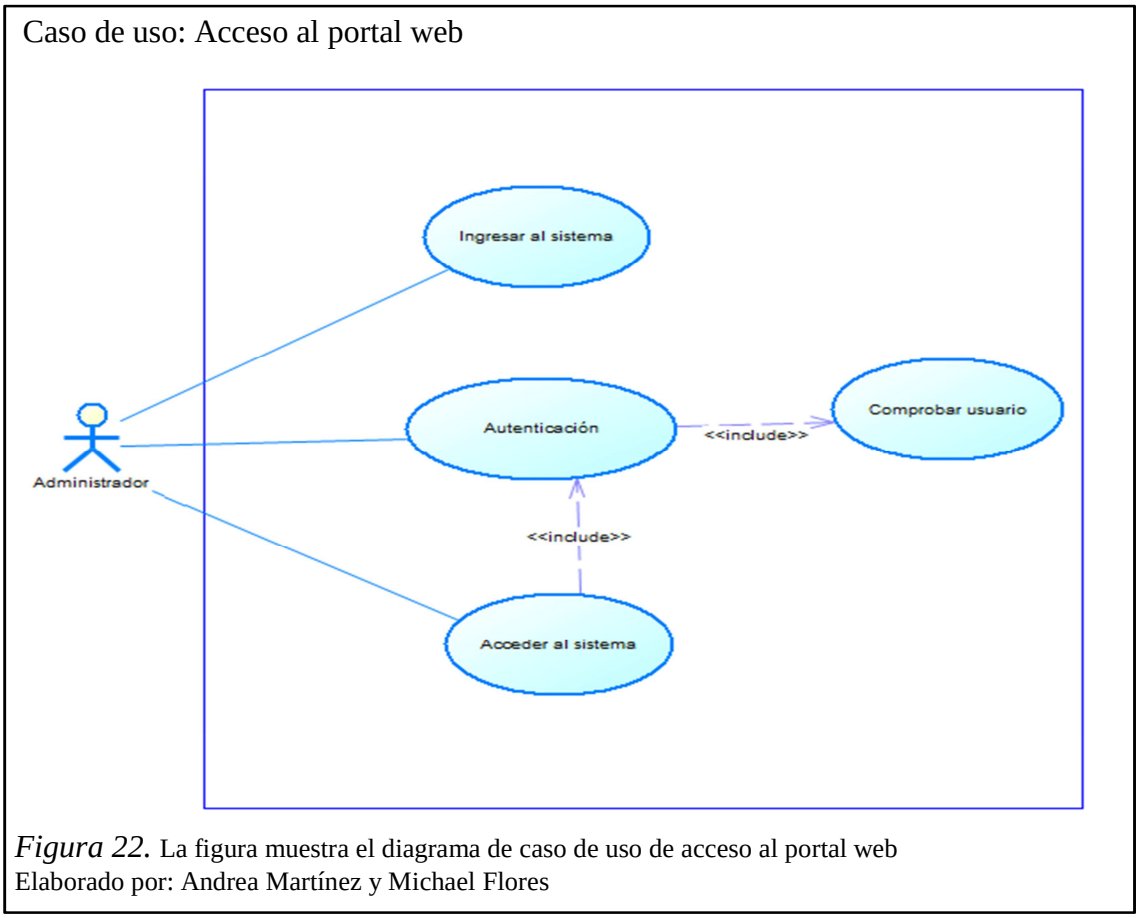

Figura 22. La figura muestra el diagrama de caso de uso de acceso al portal web
Elaborado por: Andrea Martínez y Michael Flores

Caso de uso: acceso al portal web

Actor Principal: administrador

Precondiciones: el sistema web recibe un login para verificar el acceso al mismo

Escenario:

Administrador: ingresa al sistema Web.

Administrador: introduce el usuario y contraseña.

Administrador: ingresa para proceder a utilizarla.

Excepciones:

El administrador no tiene acceso

El usuario y contraseña ingresados son incorrectos

Prioridad: esencial debe implementarse para el acceso al sistema web

Frecuencia de uso: dependiendo del número de veces de acceso al sistema

Canal para el Actor: a través del sistema web

Gestión de notificaciones

Las opciones en que se pueden realizar dentro de la gestión de notificaciones se observan en la figura 23.

Caso de Uso: gestión de notificaciones

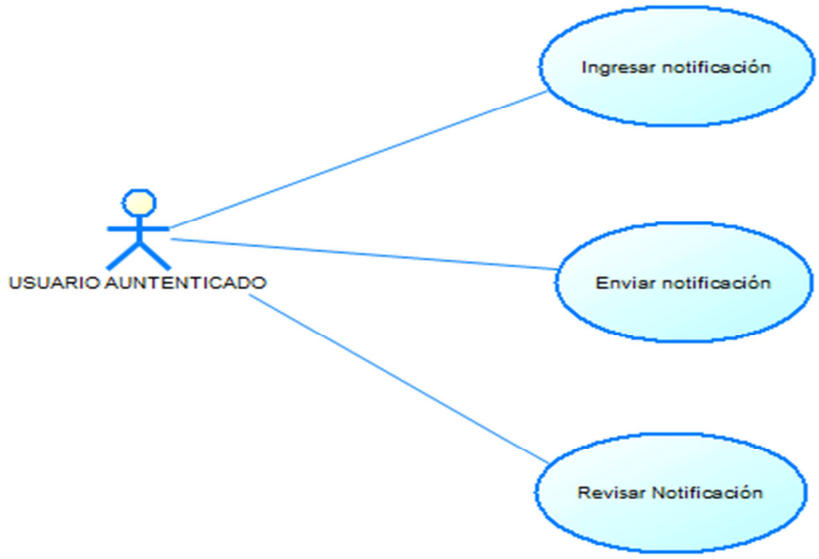

Figura 23. La figura muestra el caso de uso de la gestión de
Elaborado por: Andrea Martínez y Michael Flores

Caso de uso: gestión de notificaciones

Actor Principal: administrador

Precondiciones: el sistema web recibe un login para verificar el acceso al mismo

Escenario:

Administrador: ingresa al sistema Web.

Administrador: introduce el usuario y contraseña.

Administrador: ingresa y realiza el ingreso, envío y revisión de notificaciones

Excepciones:

El administrador no tiene acceso.

El usuario y contraseña ingresados son incorrectos.

Prioridad: esencial debe implementarse para la gestión de notificaciones

Frecuencia de Uso: dependiendo del número de veces de acceso al sistema

Canal para el Actor: a través del sistema web

3.2.4.2 Diagramas de caso de uso del cliente Android

Ingreso a la aplicación Android

Los procesos que se deben cumplir para permitir el inicio de sesión en la aplicación Android se detallan en la figura 24.

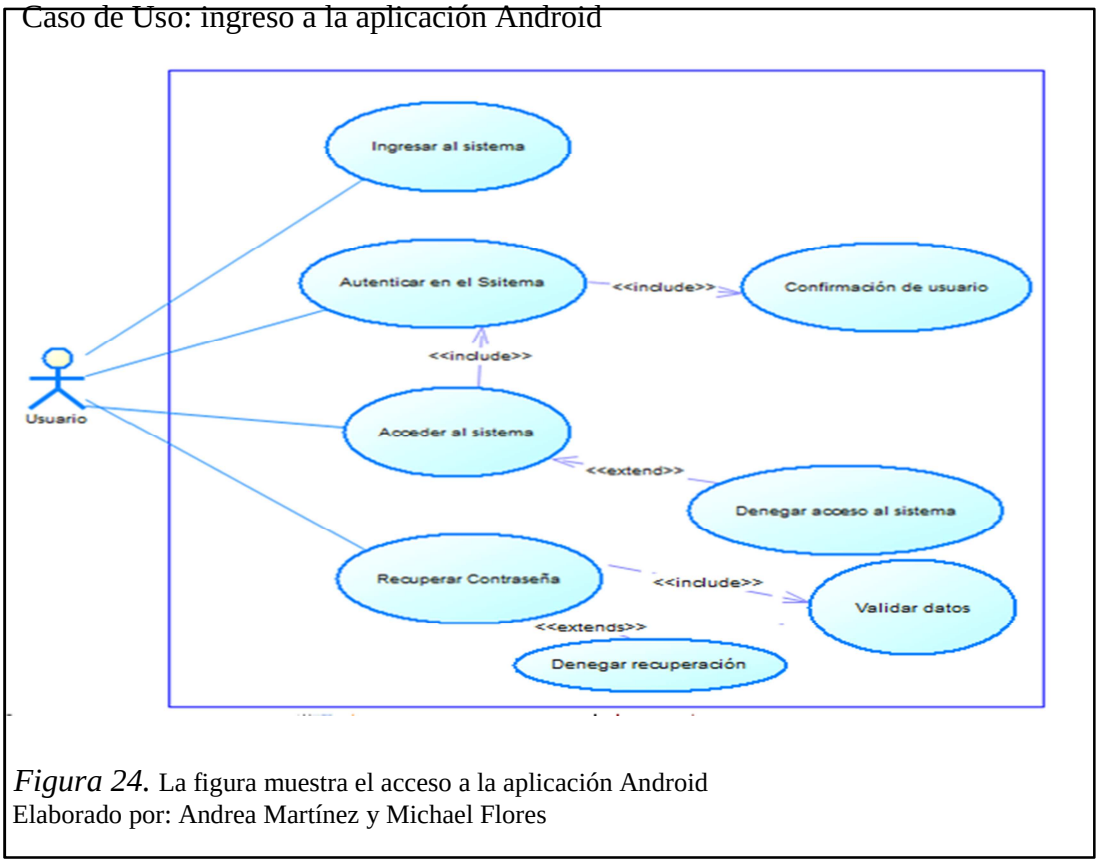

Figura 24. La figura muestra el acceso a la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

Caso de uso: Ingreso a la aplicación móvil

Actor Principal: usuario (Estudiante)

Precondiciones: la aplicación móvil recibe un login para verificar el acceso al mismo

Escenario:

Usuario (Estudiante): abre la aplicación en su celular.

Usuario (Estudiante): introduce el usuario y contraseña para autentificar en el sistema.

Usuario (Estudiante): accede a la aplicación para proceder a utilizarla.

Usuario (Estudiante): de no recordar la contraseña puede recuperar la contraseña

Teléfono: valida los datos y de ser correctos entrega una nueva clave caso contrario no la entrega.

Excepciones:

El teléfono se encuentra conectado a la red para acceder a la aplicación.

El usuario y contraseña ingresados son incorrectos.

Validación de datos incorrectos.

Prioridad: esencial debe implementarse para el acceso a la aplicación

Frecuencia de Uso: dependiendo del número de veces de acceso al sistema

Canal para el Actor: a través de la aplicación móvil

Actores Secundarios: teléfono celular

Consulta Notificaciones

La forma como se debe acceder y visualizar a las notificaciones por parte del usuario autentificado se muestra en la figura 25.

Caso de Uso: consulta notificaciones

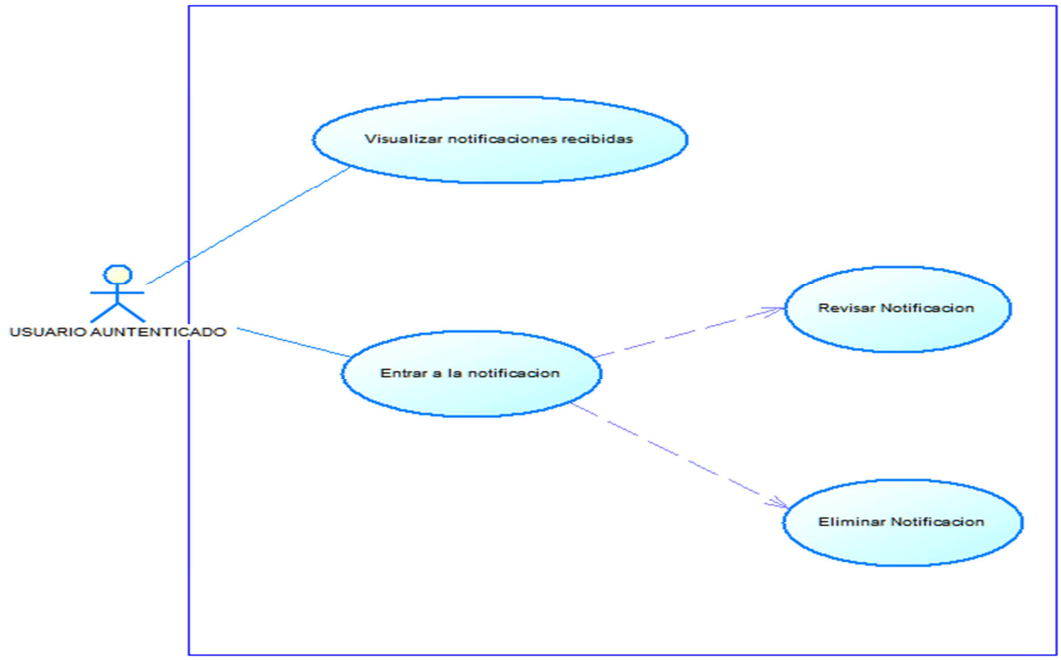

Figura 25. Caso de uso consultar notificaciones
Elaborado por: Andrea Martínez y Michael Flores

Caso de uso: consulta notificaciones aplicación móvil

Actor Principal: usuario (Estudiante)

Precondiciones: el usuario (Estudiante) debe tener acceso al servidor para poder recibir las actualizaciones de las notificaciones y poder revisarlas.

Escenario:

Usuario (Estudiante): ingresa en la aplicación al módulo de notificaciones

Usuario (Estudiante): selecciona la notificación

Teléfono: carga la notificación seleccionada

Usuario (Estudiante): revisa la información de la notificación seleccionada

Usuario (Estudiante): puede eliminar la notificación.

Excepciones:

El usuario o tiene acceso al servidor y no cargan las noticias de las notificaciones

Prioridad: esencial debe implementarse para uso de aplicación móvil

Frecuencia de Uso: limitada, dependiendo del número de veces que el usuario revise las notificaciones

Canal para el Actor: a través de la aplicación móvil

Actores Secundarios: teléfono celular

Chat

Las actividades que se pueden realizar dentro del chat se describen en la figura 26.

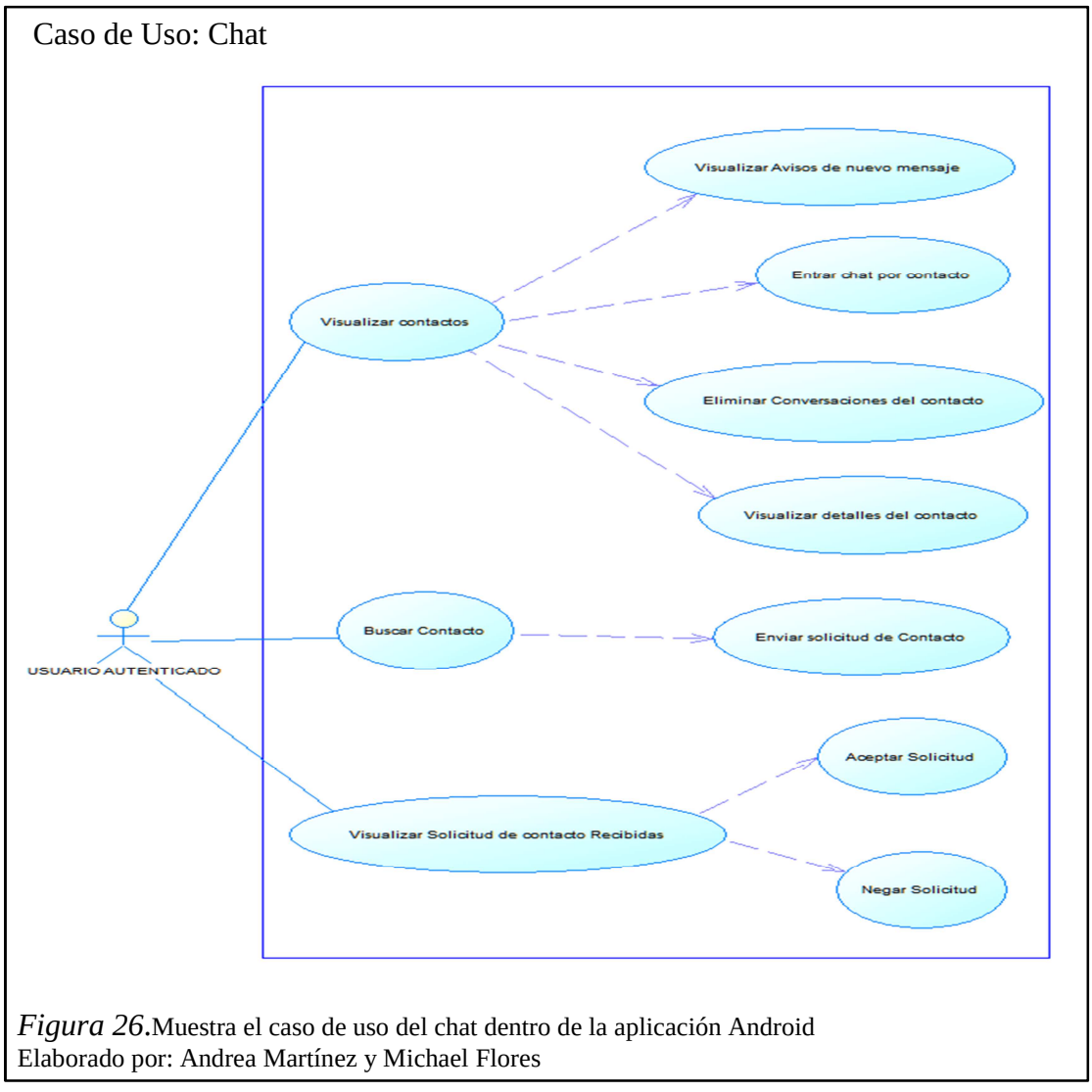

*Figura 26.*Muestra el caso de uso del chat dentro de la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

Caso de uso: Chat

Actor Principal: usuario (Estudiante)

Descripción: ingresa a la opción del chat para poder acceder a revisar los mensajes

Precondición: sesión de usuario iniciada.

Escenario:

Usuario: ingresa al chat visualiza sus contactos.

Usuario: al visualizar a sus contactos tiene la opción de ver los detalles de sus contactos, puede ingresar al chat por contacto y a su vez eliminar conversaciones.

Usuario: puede realizar la búsqueda de un nuevo contacto y enviarle una solicitud de contacto.

Usuario: puede revisar las solicitudes de contacto recibidas las cuales puede aceptar o denegar

Prioridad: esencial para uso del chat

Frecuencia de Uso: dependiendo del número de veces de acceso que el usuario acceda al chat.

Canal para el Actor: a través de la aplicación móvil.

Actores Secundarios: teléfono celular.

Conversación

La interacción entre los mensajes enviados, recibidos y visualizados se puede apreciar en la figura 27.

Caso de Uso: Conversación

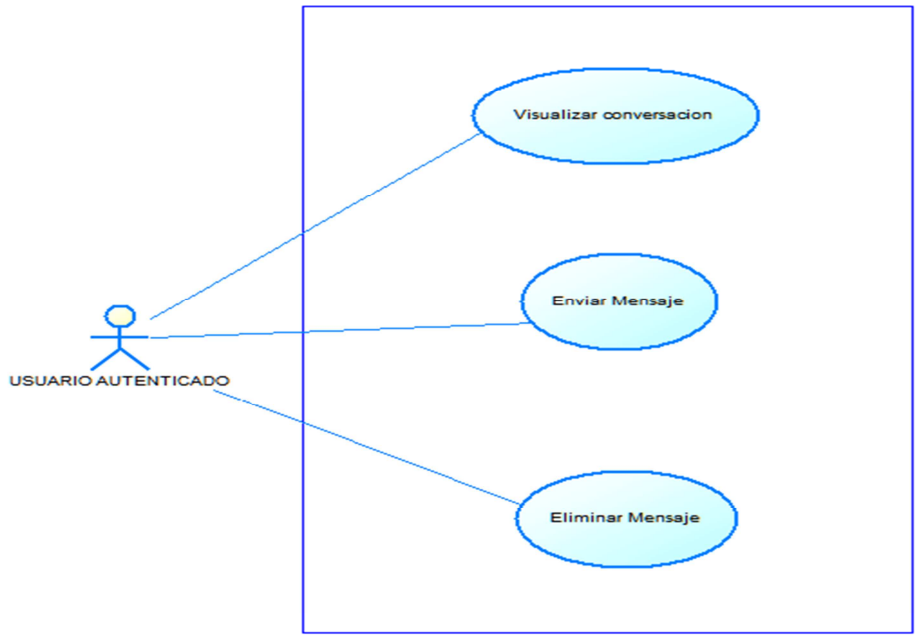

Figura 27. Muestra el caso de uso en el que el usuario realiza una conversación
Elaborado por: Andrea Martínez y Michael Flores

Caso de uso: Conversación

Actor Principal: usuario (Estudiante)

Enviar Mensaje:

Descripción: envía el mensaje que el usuario ingresó en chat

Precondición: sesión de usuario iniciada.

Escenario:

1. Usuario selecciona contacto
2. Usuario envía mensaje al contacto seleccionado

Prioridad: esencial para uso en la aplicación móvil

Frecuencia de Uso: dependiendo del número de veces que el usuario quiera hacer una conversación y enviar un mensaje

Canal para el Actor: a través de la aplicación móvil

Actores Secundarios: teléfono celular

Recibir Mensaje:

Descripción: recibe un mensaje de otro contacto de chat para mostrar al usuario.

Precondición: sesión de usuario iniciada.

Escenario:

1. Sistema recibe Mensaje
2. Sistema muestra a Usuario el contacto y el mensaje

Prioridad: esencial para uso en la aplicación móvil.

Frecuencia de Uso: dependiendo del número de veces que el usuario reciba un mensaje y lo revise dentro de la aplicación.

Canal para el Actor: a través de la aplicación móvil.

Actores Secundarios: teléfono celular.

3.2.5 Diagrama de secuencia

Los diagramas de secuencia permiten modelar un conjunto de objetos, donde se puede identificar los módulos y clases que forman parte del sistema a través del tiempo. En este caso está dividido en dos partes la primera los diagramas de secuencia del servidor web y la segunda parte los diagramas de secuencia del aplicación Android.

3.2.5.1 Diagramas de secuencia servidor web

Acceso al portal web

La interacción entre los eventos generados por el usuario para acceder al portal web se detalla en la figura 28**.**

Diagrama de secuencia acceso al portal web

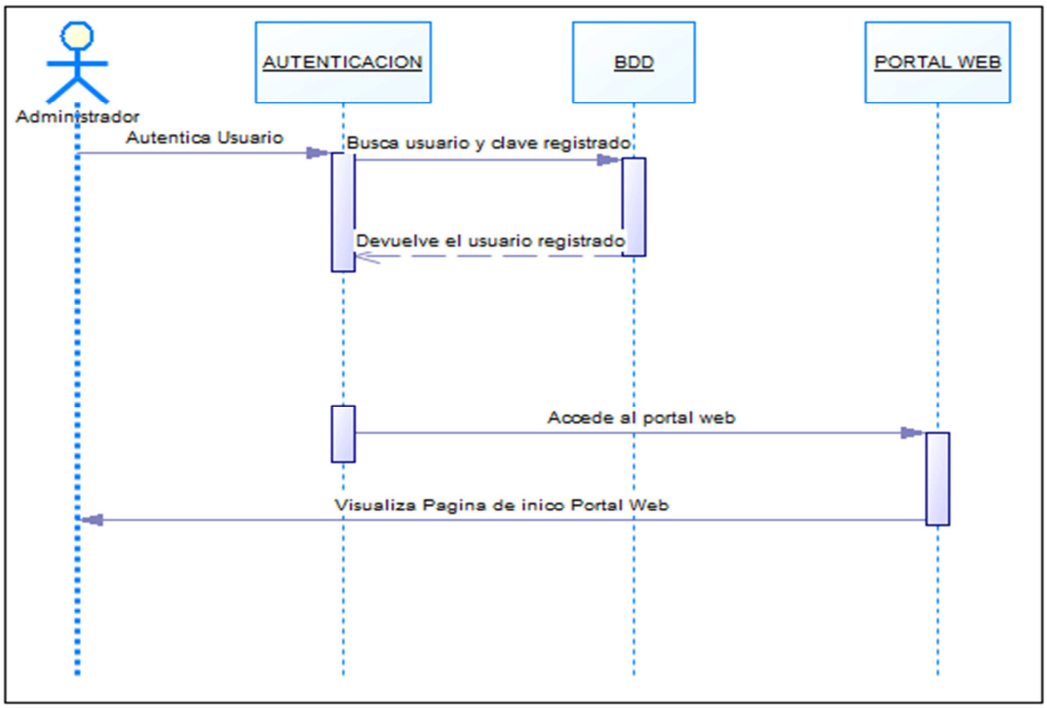

Figura 28. Diagrama de secuencia acceso al portal web
Elaborado por: Andrea Martínez y Michael Flores

Gestión de Notificaciones

La secuencia de eventos que se generan entre los componentes del portal web para el ingreso y envío de notificaciones se muestran en la figura 29.

Diagrama de Secuencia Gestión de Notificaciones

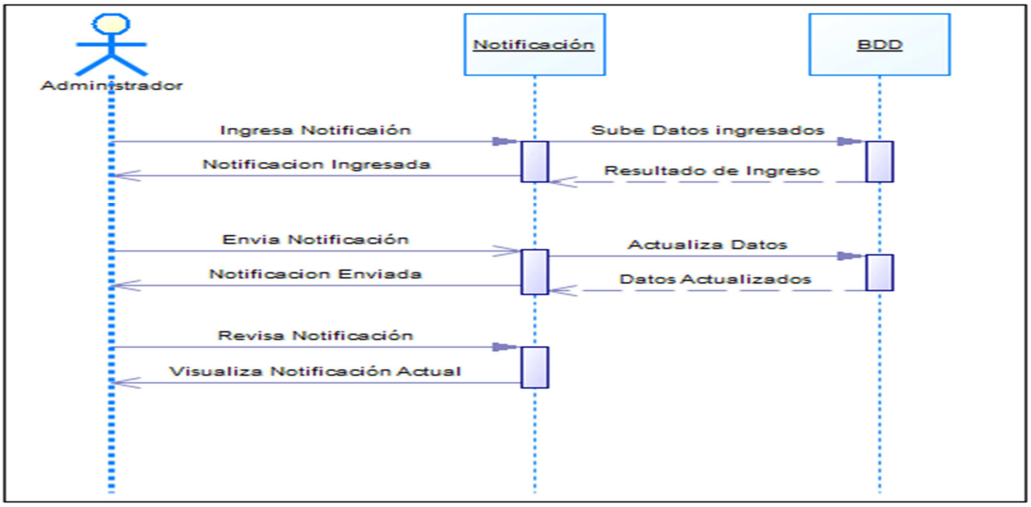

Figura 29. Diagrama de secuencia de la gestión de notificaciones
Elaborado por: Andrea Martínez y Michael Flores

3.2.5.2 Diagramas de secuencia aplicación Android

Ingreso aplicación Android

La sucesión de eventos entre los componentes que se generan para el acceso o inicio de sesión en la aplicación Android se muestran en la figura 30.

Diagrama de secuencia ingreso aplicación móvil

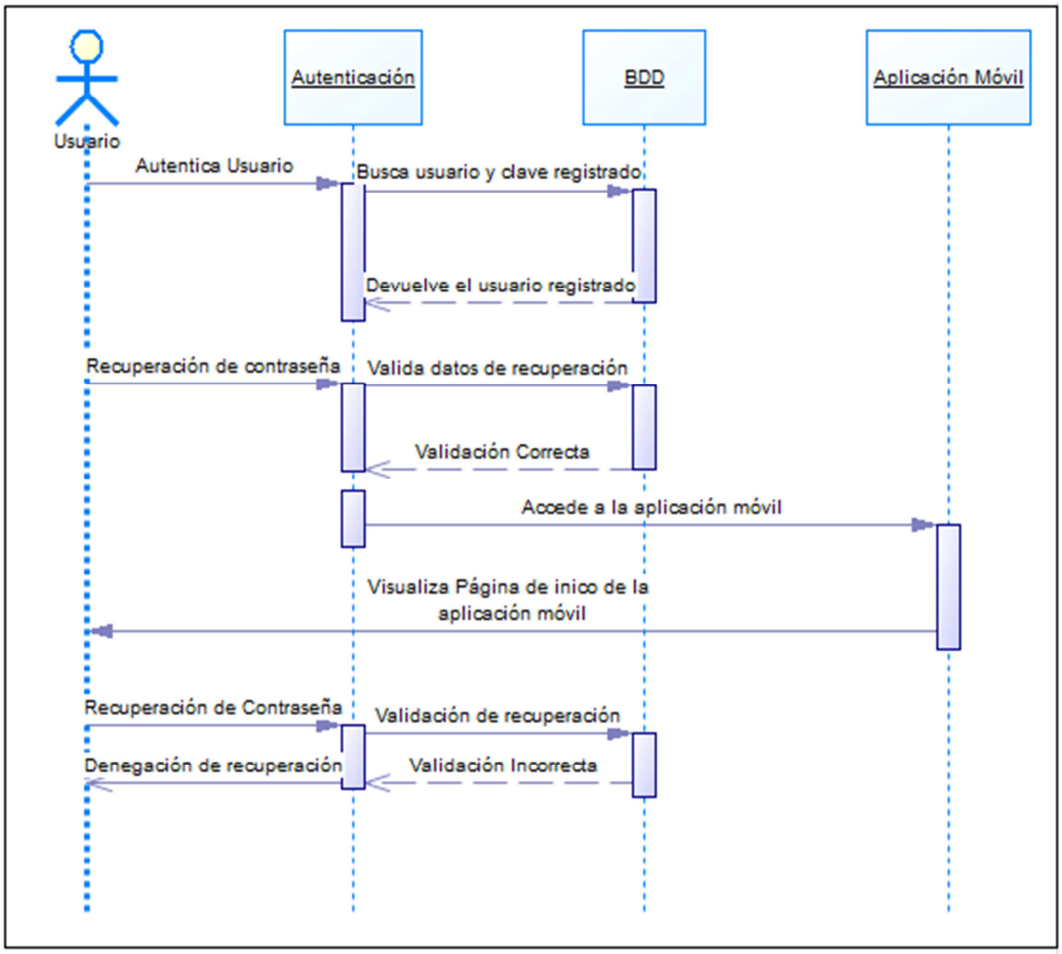

Figura 30. Diagrama de secuencia del ingreso a la aplicación Android
Elaborado por: Andrea Martínez y Michael Flores

Consulta de notificaciones

Para poder visualizar las notificaciones se siguen una cadena de eventos entre los componentes de la aplicación Android esto se puede observar en la figura 31.

Diagrama de secuencia consulta de notificaciones

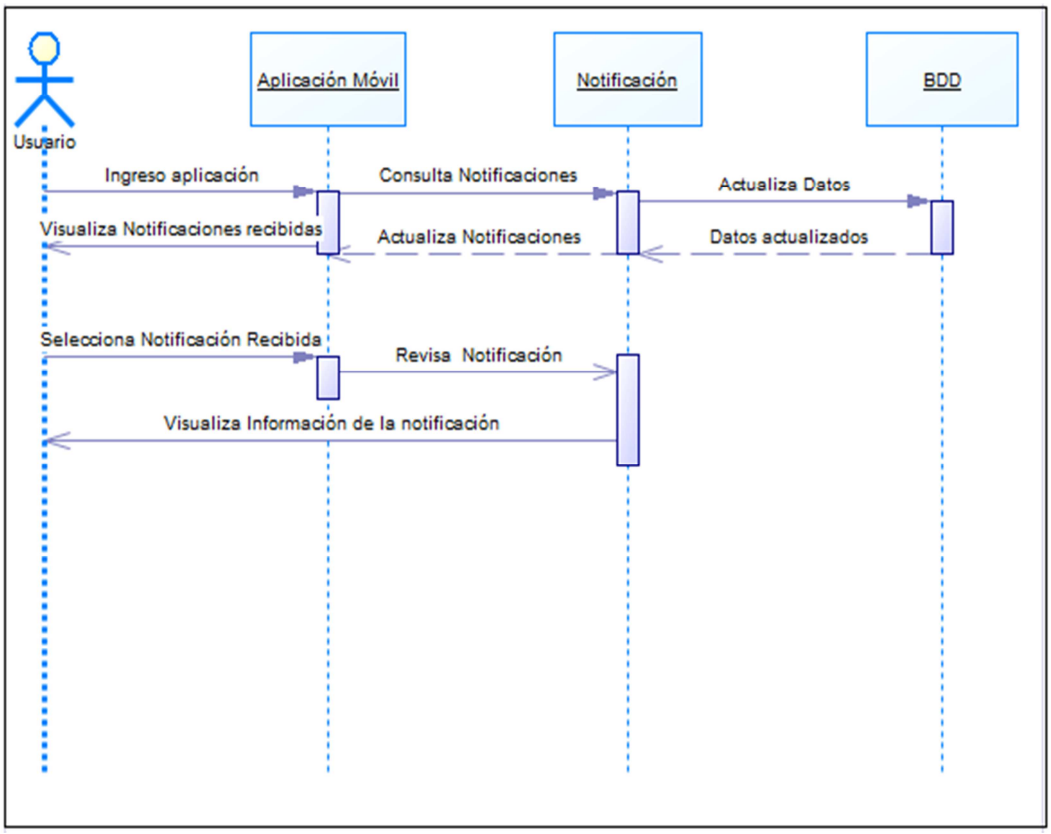

Figura 31. Diagrama de secuencia consulta de notificaciones
Elaborado por: Andrea Martínez y Michael Flores

Chat

el manejo del módulo de chat se puede observar mediante el seguimiento de los eventos secuenciales que se generan al realizar una actividad dentro del chat esta descripción se aprecia en la figura 32.

Diagrama de Secuencia Chat

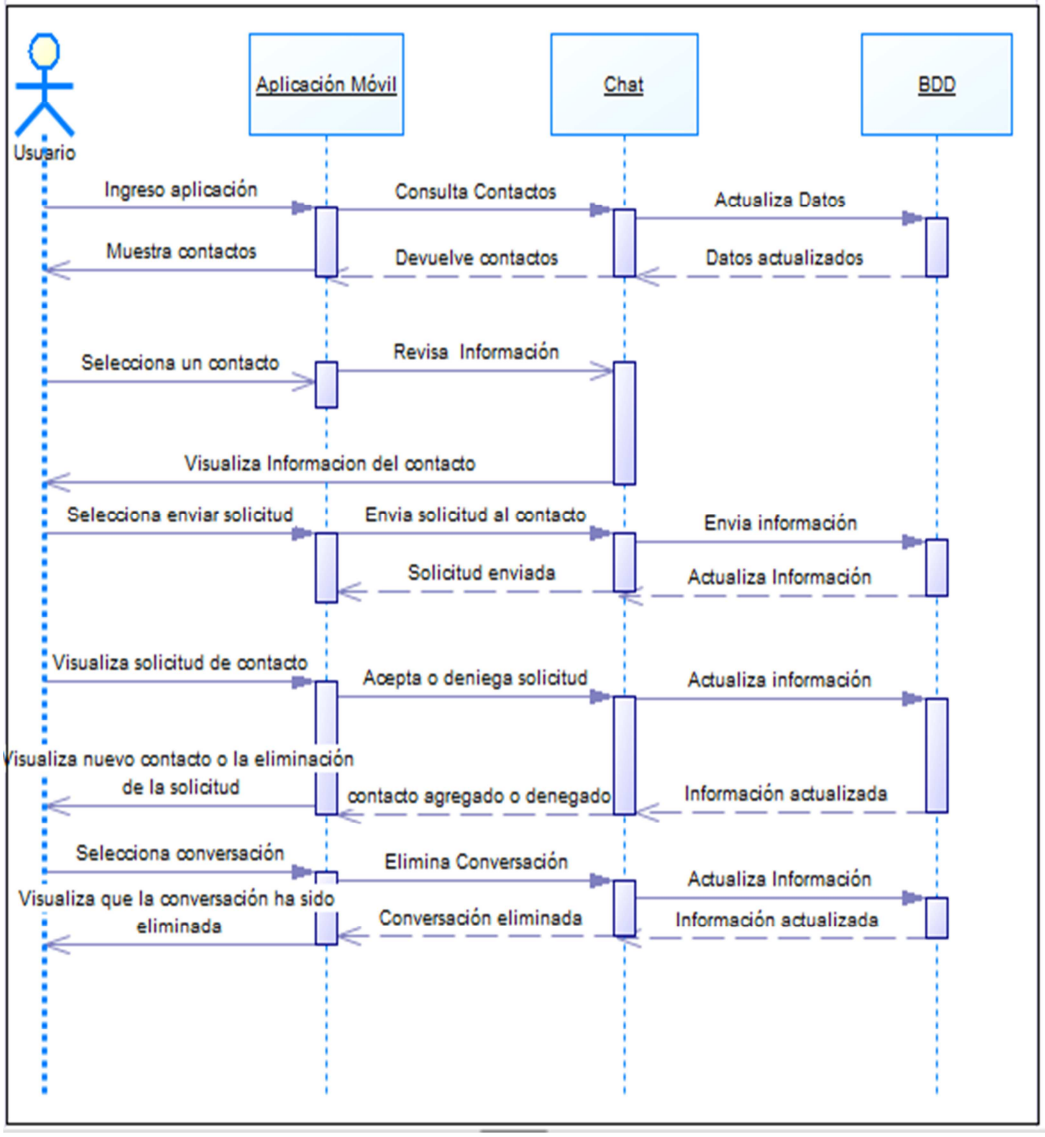

Figura 32. Diagrama de secuencia del chat
Elaborado por: Andrea Martínez y Michael Flores

Conversaciones

La serie de eventos secuenciales que se necesitan para poder entablar una conversación se aprecian en la figura 33.

Diagrama de secuencia conversaciones

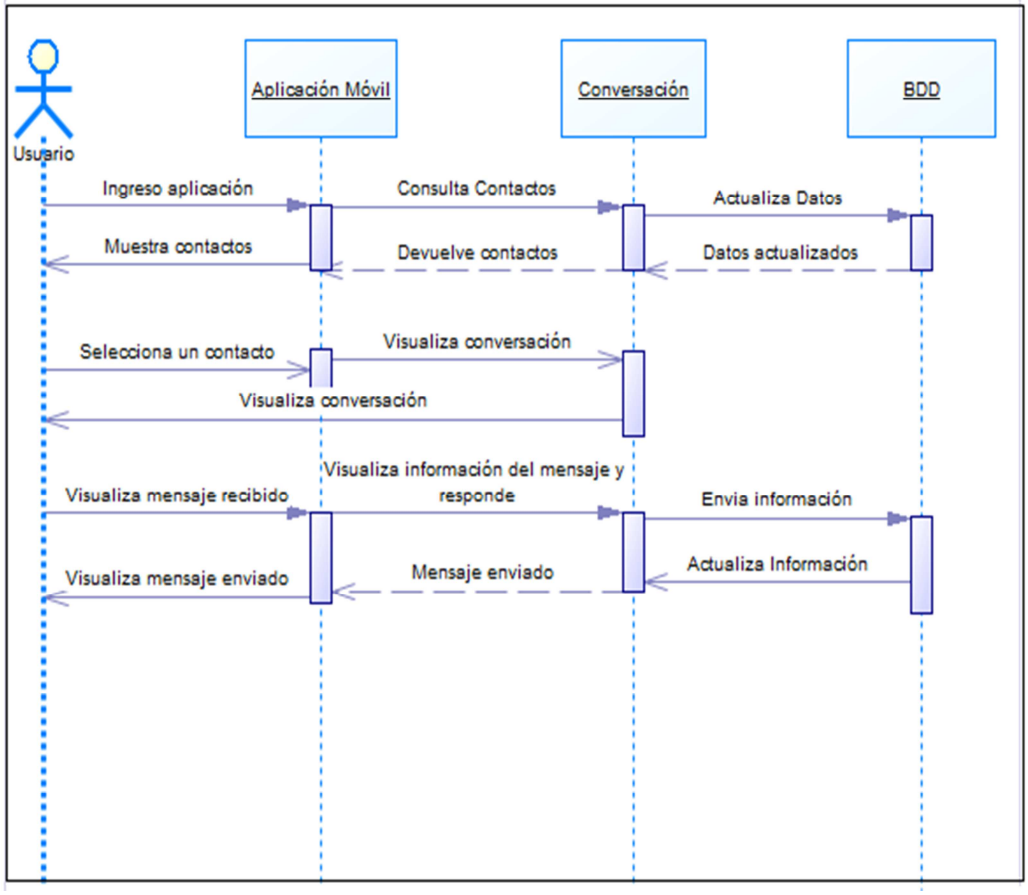

*Figura 33.*Diagrama de secuencia de conversaciones
Elaborado por: Andrea Martínez y Michael Flores

3.2.6 Diagrama de componentes

El diagrama de componentes muestra las dependencias entre los componentes. Permite visualizar con más facilidad la estructura general del sistema. El sistema está dividido por 5 componentes de software que se relacionan para el funcionamiento de cada módulo como se puede observar en la figura 34.

Diagrama de componentes

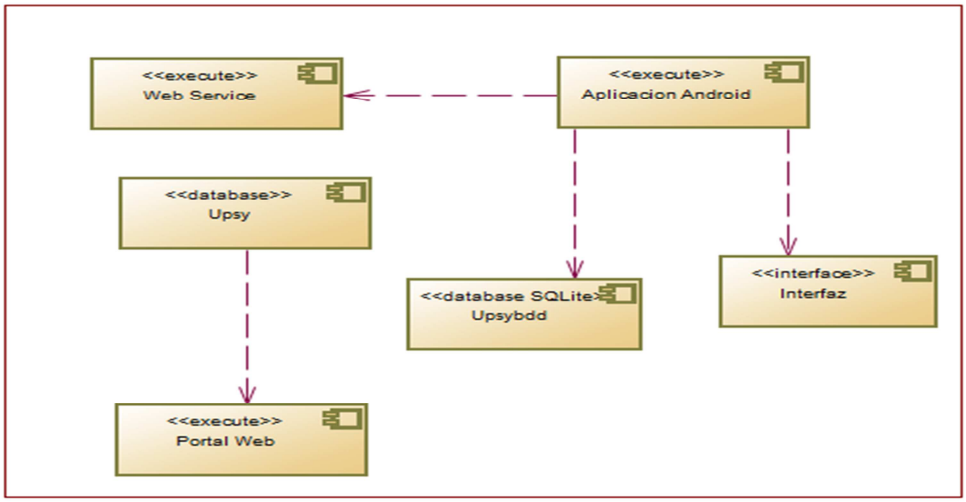

Figura 34. Diagrama de componentes
Elaborado por: Andrea Martínez y Michael Flores

3.2.7 Diagrama de despliegue

El diagrama de despliegue modela la arquitectura del proyecto Android, que se utilizó para modelar el hardware utilizado en las implementaciones del Proyecto Android, como se aprecia en la figura 35.

Diagrama de Despliegue

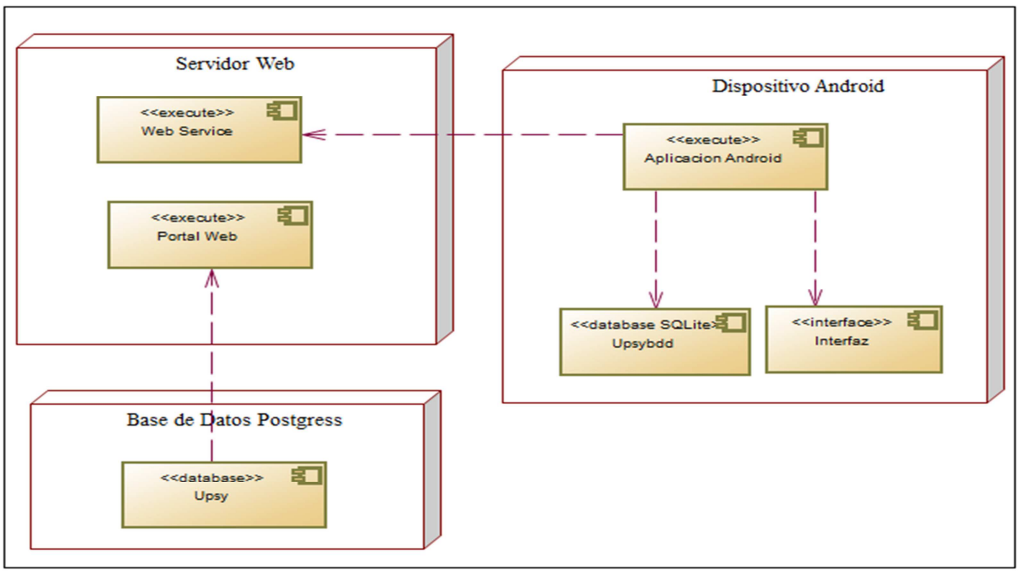

Figura 35. Muestra el diagrama de despliegue de toda la aplicación Android desarrollada.
Elaborado por: Andrea Martínez y Michael Flores

CAPÍTULO 4
ETAPA DE PRUEBAS

4.1 Introducción etapa de pruebas

Para garantizar el desempeño óptimo de la aplicación y cumpliendo una parte fundamental dentro del desarrollo de software, entramos a la fase o etapa de pruebas que nos ayudara a asegurarnos que cada parte de la aplicación desempeñe la función o muestre el comportamiento para el que fue diseñada (Pressman, 2010, pág. 385)

El objetivo de realizar un plan de pruebas es determinar la metodología oportuna que permitan verificar como actúa la aplicación Android y sus correspondientes servicios web siendo estos dos el enfoque principal del proyecto.

Para realizar dichos hallazgos nos planteamos realizar pruebas sobre los siguientes puntos:

- Funcionalidad
- Tiempo de respuesta
- Compatibilidad

Las pruebas de funcionalidad, que clarifiquen los procesos y el funcionamiento de las partes que contiene la aplicación dentro del Smartphone y demás características que se comprueben en detalle cuando se entre de lleno a la etapa de pruebas de funcionalidad.

Por otro lado tenemos las pruebas de tiempo de respuesta que nos orientan en la velocidad de reacción y desenvolvimiento en la transmisión de datos del Smartphone y el servidor, que contendrá los servicios web, debido a que en su mayor parte el funcionamiento de la aplicación móvil se la realiza en un servidor remoto que alimenta a la aplicación.

4.2 Tipos de pruebas realizadas

4.2.1 Hardware para pruebas

Para realizar las pruebas de la aplicación desarrollada se utilizó un dispositivo Android con las características mostradas en la tabla 56.

Tabla 56.*Características equipo de pruebas Android*

EQUIPO DE PRUEBAS MOVIL (SMARTPHONE)		
NRO	**TIPO**	**CARACTERISTICAS**
1	Dimensiones	146 x 72 x 7,3 mm
2	Peso	152 gramos
3	Pantalla	5,2", Full HD (1920 x 1080 píxeles)
4	Interior	Google Android 4.4 (Kitkat)
		Qualcomm Snapdragon de cuatro núcleos a 2,5 GHz
5	Memoria y almacenamiento	3 GB de RAM Memoria flash de hasta 16 GB microSD™ de hasta 128 GB (ranura para tarjeta; compatible con SDXC)
6	Conectividad	Wi-Fi® y función de punto de conexión Wi-Fi

Nota. La tabla muestra todas las características del equipo móvil en el que se realizaron las pruebas
Elaborado por: Andrea Martínez y Michael Flores.

Para las pruebas realizadas se tomó en utilizo un equipo que actúa como servidor web y de base de datos, a continuación se detalla las características del servidor como se observa en la tabla 57.

Tabla 57. *Características equipo de pruebas del servidor*

EQUIPO DE PRUEBAS DEL SERVIDOR (WEBSERVICES)		
NRO	**TIPO**	**CARACTERISTICAS**
1	Procesador	Intel Core I5-3450 CPU 3,10 GHz 3,10 GHz
2	Memoria Ram	4 GB
3	Disco Duro	3 TB
4	Video	Nvidia GeForce GT 610
5	Sonido	Nvidia High Definition Audio
6	Unidad Optica	HL-DT-ST-DVDRAM GH24NS90
7	Adaptador de red	10/100 fast Ethernet

Nota. La tabla muestra las características del servidor web y de base de datos para pruebas
Elaborado por: Andrea Martínez y Michael Flores

4.2.2 Pruebas de funcionalidad

Las pruebas de funcionalidad tienen como objetivo comprobar que los sistemas desarrollados funcionan cumpliendo las especificaciones funcionales y los requisitos. Permitiendo detectar los posibles defectos generados en la fase de programación.

Para el desarrollo y verificación de las pruebas, se utiliza la metodología de pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisaran por completo todos los requerimientos funcionales para un programa (PRESSMAN, 2010).

Dichas pruebas se basan en los modelos de casos de uso los cuales son: acceso al sistema, notificaciones, contactos y chat. Que conforman la aplicación Android; para validar las operaciones se han examinado diferentes entradas de datos por parte del usuario buscando verificar la consistencia y estabilidad de la aplicación.

Las pruebas se dividirán en:

Pruebas del módulo acceso al sistema

- Registro de usuario
- Recuperación de contraseña
- Inicio de sesión

Pruebas del módulo Notificaciones

- Notificaciones

Pruebas del módulo de Contacto

- Visualizar contacto
- Buscar Contacto
- Visualizar solicitudes recibidas

Pruebas del módulo Conversación

- Conversación

Las pruebas del módulo de acceso al sistema de la aplicación Android enfocadas en el funcionamiento del registro de usuario se muestran en la tabla 58.

Tabla 58. *Pruebas acceso al sistema 1*

Prueba 1. pruebas de acceso al sistema			
prueba	**Entrada o Acción de usuario**	**resultado esperado del sistema**	**confirmación**
P 1.1	**Registro de usuario**		
P 1.1.1	Todos los datos de registro nulos	La aplicación muestra los campos nulos y solicita su verificación	SI
P 1.1.2	El nombre nulo	La aplicación muestra mensaje de error	SI
P 1.1.3	El nombre con números o caracteres especiales	La aplicación muestra mensaje de error	SI
P 1.1.4	El nombre sin números o caracteres especiales	La aplicación no muestra error en nombre	SI
P 1.1.5	El apellido nulo	La aplicación muestra mensaje de error	SI
P 1.1.6	El apellido con números o caracteres especiales	La aplicación muestra mensaje de error	SI
P 1.1.7	El apellido sin números o caracteres especiales	La aplicación no muestra error en apellido	SI
P 1.1.8	Carrera por defecto	La aplicación no muestra error en carrera	SI
P 1.1.9	Carrera elegida por el usuario	la aplicación muestra opciones y obtiene la carrera que el usuario elige y no muestra error en carrera	SI
P 1.1.10	El password nulo y confirma Password con datos	la aplicación muestra un error de contraseña y confirma password	SI
P 1.1.11	El password nulo y confirma Password nulo	la aplicación muestra un error de contraseña y confirma password	SI
P 1.1.12	El password con datos y confirma Password con datos distintos de password	la aplicación muestra un error de contraseña y confirma password	SI
P 1.1.13	El password con datos y confirma Password con datos iguales	La aplicación no muestra error en password	SI
P 1.1.14	Genero por defecto	La aplicación no muestra error en genero	SI
P 1.1.15	Genero elegida por el usuario	la aplicación muestra opciones y obtiene el género que el usuario elige y no muestra error en genero	SI
P 1.1.16	no se coloca la fecha de nacimiento	la aplicación muestra un mensaje de error en la fecha de nacimiento	SI
P 1.1.17	se coloca la fecha actual	la aplicación muestra un mensaje de error en la fecha de nacimiento	SI
P 1.1.18	se coloca fecha de nacimiento menor de 16 años	la aplicación muestra un mensaje de error en la fecha de nacimiento	SI
P 1.1.19	se coloca una fecha de nacimiento con edad mayor de 16 años	la aplicación no muestra mensaje de error de fecha de nacimiento	SI
P 1.1.20	email con valor nulo	la aplicación muestra error en correo electrónico	SI
P 1.1.21	email con valor sin formato de correo electrónico	la aplicación muestra error en correo electrónico	SI
P 1.1.22	email con valor con formato de correo electrónico	la aplicación no muestra error en email	SI
P 1.1.23	celular con valor nulo	la aplicación muestra error en celular	SI
P 1.1.24	celular con valor diferente a diez alfanumérico o caracteres	la aplicación muestra error en celular	SI
P 1.1.25	celular con 10 valores solo numero	la aplicación no muestra error en celular	SI
P 1.1.26	Todos los datos de registro llenos correctamente con usuario repetido	la aplicación muestra mensaje de error Usuario ya existente	SI
P 1.1.27	todos los datos de registro correcto	la aplicación pasa a otra pantalla confirmando registro y bienvenida al usuario y abre la pantalla de inicio de sesión	SI

Nota. Se muestra todas las pruebas de acceso al sistema que se realizaron dentro de la aplicación Android.
Elaborado por: Andrea Martínez y Michael Flores

Las pruebas del módulo de acceso al sistema de la aplicación Android enfocadas en el funcionamiento de la recuperación de contraseña se muestran en la tabla 59.

Tabla 59. *Pruebas acceso al sistema 2*

Prueba 1. pruebas de acceso al sistema			
prueba	**Entrada o Acción de usuario**	**resultado esperado del sistema**	**confirmación**
P 1.2	**Recuperación de contraseña**		
P 1.2.1	todos los campos vacíos	la aplicación muestra error y pide verificaciones de datos	SI
P 1.2.2	email con valor nulo	la aplicación muestra error en correo electrónico	SI
P 1.2.3	email con valor sin formato de correo electrónico	la aplicación muestra error en correo electrónico	SI
P 1.2.4	email con valor con formato de correo electrónico	la aplicación no muestra error en email	SI
P 1.2.5	celular con valor nulo	la aplicación muestra error en celular	SI
P 1.2.6	celular con valor diferente a diez alfanumérico o caracteres	la aplicación muestra error en celular	SI
P 1.2.7	celular con 10 valores solo numero	la aplicación no muestra error en celular	SI
P 1.2.8	password 1 con valor nulo	la aplicación muestra error en password 1 y solicita se ingrese posible password 1	SI
P 1.2.9	password 1 con valor	la aplicación no muestra error en password 1	SI
P 1.2.10	password 2 con valor nulo	la aplicación muestra error en password 2 y solicita se ingrese posible password 2	SI
P 1.2.11	password 2 con valor	la aplicación no muestra error en password 2	SI
P 1.2.12	todos los datos correctos y el usuario no registrados	la aplicación muestra mensaje de usuario no registrado	SI
P 1.2.13	todos los datos correctos y el usuario registrado	validación de los datos del usuario con posibles password registrado pasa a pantalla de registro de nueva contraseña	SI
P 1.2.14	Todos los datos de nuevo password nulos	la aplicación muestra mensaje de error en datos de nueva contraseña	SI
P 1.2.15	password con datos y confirma password nulo	la aplicación muestra mensaje de error en datos de nueva contraseña	SI
P 1.2.16	Todos los datos de nuevo password con valor diferente a confirma password	la aplicación muestra mensaje de error en datos de nueva contraseña	SI
P 1.2.17	Todos los datos de nuevo password y confirma password llenos y con los mismos valores	aplicación no muestra error y pasa a la pantalla de mensaje éxito de cambio de password y vuelve a la pantalla de inicio	SI

Nota. La tabla muestra todas las pruebas de acceso al sistema en la recuperación de contraseña
Elaborado por: Andrea Martínez y Michael Flores

Las pruebas del módulo de acceso al sistema de la aplicación Android enfocadas en el funcionamiento del inicio de sesión de usuario se muestran en la tabla 60.

Tabla 60. *Pruebas acceso al sistema 3*

Prueba 1. pruebas de acceso al sistema			
prueba	**Entrada o Acción de usuario**	**resultado esperado del sistema**	**confirmación**
P 1.3	**Inicio de Sesión**		
P 1.3.1	Email correcto y password incorrecto	la aplicación no permite inicio de sesión y me muestra un mensaje de Email o contraseña con error	SI
P 1.3.2	Email correcto y password nulo	la aplicación no permite inicio de sesión y me muestra un mensaje de Email o contraseña con error	SI
P 1.3.3	Email incorrecto y password correcto	la aplicación no permite inicio de sesión y me muestra un mensaje de Email o contraseña con error	SI
P 1.3.4	Email nulo y password correcto	la aplicación no permite inicio de sesión y me muestra un mensaje de Email o contraseña con error	SI
P 1.3.5	Email incorrecto y password incorrecto	la aplicación no permite inicio de sesión y me muestra un mensaje de Email o contraseña con error	SI
P 1.3.6	Email nulo y password nulo	la aplicación no permite inicio de sesión y me muestra un mensaje de Email o contraseña con error	SI
P 1.3.7	Email correcto y password correcto	la aplicación permite el acceso al inicio de sesión mostrando la pantalla de notificaciones	SI

Nota. La tabla muestra todas las pruebas de acceso al sistema en el inicio de sesión de usuario
Elaborado por: Andrea Martínez y Michael Flores

Las pruebas del módulo de notificaciones de la aplicación Android enfocadas en el funcionamiento de las notificaciones se encuentran en la tabla 61.

Tabla 61. *Pruebas de interfaz de notificaciones*

Prueba 2. Notificaciones			
prueba	**Entrada o Acción de usuario**	**resultado esperado del sistema**	**confirmación**
P 2.1	se selecciona la lista desplegable de notificaciones donde no se visualiza notificaciones recibidas	la lista no se abre y no muestra ningún dato	SI
P 2.2	se selecciona la lista desplegable de notificaciones donde se visualiza notificaciones	la lista se abre y muestra notificaciones recibidas	SI
P 2.2.1	se selecciona una notificación con datos	la aplicación abre la pantalla de la notificación seleccionada	SI
P 2.2.2	se presiona botón eliminar	la aplicación me muestra mensaje de confirmación de eliminación	SI
P 2.2.3	se cancela eliminación	la aplicación vuelve a la notificación abierta	SI
P 2.2.4	se confirma eliminación	la aplicación elimina la notificación y regresa a la pantalla principal	SI

Nota. La tabla muestra todas las pruebas realizadas a la interfaz de notificaciones dentro de la aplicación.
Elaborado por: Andrea Martínez y Michael Flores

Las pruebas del módulo de contactos de la aplicación Android enfocadas en el funcionamiento y manejo de los contactos de usuario se muestran en la tabla 62.

Tabla 62. *Pruebas de interfaz de contactos 1*

Prueba 3. Contactos			
prueba	**Entrada o Acción de usuario**	**resultado esperado del sistema**	**confirmación**
P 3.1	**Visualizar contactos**		
P 3.1.1	se selecciona un contacto	la aplicación abre la pantalla de chat del contacto	SI
P 3.1.2	se selecciona un contacto y se lo mantiene presionado por tres segundos	la aplicación muestra un mensaje de selección de: borrar conversaciones y ver contacto	SI
P 3.1.3	selecciona la opción borrar conversaciones	la aplicación muestra un mensaje de solicitud de confirmación de eliminación de conversación	SI
P 3.1.4	se selecciona confirmar borrar conversación	la aplicación borra todas las conversaciones almacenadas	SI
P 3.1.5	se presiona cancelar solicitud	la aplicación no borra las conversaciones almacenadas	SI
P 3.1.6	se selecciona la opción contacto	la aplicación muestra un mensaje de solicitud de confirmación demostrar datos de contacto	SI
P 3.1.7	se selecciona aceptar mostrar datos del contacto	la aplicación muestra los datos del contacto	SI
P 3.1.8	se presiona cancelar solicitud	la aplicación vuelve a los contactos y no muestra datos del contacto	SI

Nota. La tabla muestra todas las pruebas realizadas a la interfaz de visualización de contactos
Elaborado por: Andrea Martínez y Michael Flores

Las pruebas del módulo de contactos de la aplicación Android enfocadas en el funcionamiento de la búsqueda de contactos se muestran en la tabla 63.

Tabla 63. *Pruebas de interfaz de contactos 2*

Prueba 3. Contactos			
prueba	**Entrada o Acción de usuario**	**resultado esperado del sistema**	**confirmación**
P 3.2	**Buscar Contacto**		
P 3.2.1	se ingresa e-mail nulo y celular correcto	la aplicación busca contacto por celular y me dirige a una pantalla de respuesta de contacto encontrado o no encontrado	SI
P 3.2.2	se ingresa e-mail incorrecto y celular correcto	la aplicación busca contacto por celular y me dirige a una pantalla de respuesta de contacto encontrado o no encontrado	SI
P 3.2.3	se ingresa e-mail correcto y celular correcto	la aplicación busca contacto por celular y me dirige a una pantalla de respuesta de contacto encontrado o no encontrado	SI
P 3.2.4	se ingresa e-mail nulo y celular correcto	la aplicación busca contacto por celular y me dirige a una pantalla de respuesta de contacto encontrado o no encontrado	SI
P 3.2.5	si encontró nuevo contacto se selecciona el contacto encontrado	la aplicación muestra un mensaje de confirmación de envió de solicitud de contacto	SI
P 3.2.6	se presiona confirmar envió de solicitud de contacto	la aplicación manda una solicitud de mensaje y regresa a la pantalla de contacto	SI
P 3.2.7	se cancela la solicitud	la aplicación regresa a la pantalla contacto	SI

Nota. La tabla muestra todas las pruebas realizadas a la interfaz de búsqueda de contactos
Elaborado por: Andrea Martínez y Michael Flores

Las pruebas del módulo de contactos de la aplicación Android enfocadas en el funcionamiento y manejo de las solicitudes de contactos se muestran en la tabla 64.

Tabla 64. *Pruebas de interfaz de contactos 3*

Prueba 3. Contactos			
prueba	**Entrada o Acción de usuario**	**resultado esperado del sistema**	**confirmación**
P 3.3	**Visualizar solicitudes recibidas**		
P 3.3.1	se selecciona la solicitud de contacto recibida	la aplicación muestra un mensaje de captar o cancelar la solicitud	SI
P 3.3.2	se presiona rechazar la solicitud	la aplicación borra la solicitud de la lista de solicitudes	SI
P 3.3.3	se presiona aceptar solicitud	se acepta la solicitud y se agrega automáticamente el nuevo contacto	SI

Nota. La tabla muestra todas las pruebas realizadas a la interfaz de solicitudes recibidas
Elaborado por: Andrea Martínez y Michael Flores

Las pruebas del módulo de conversación de la aplicación Android enfocadas en el funcionamiento la interfaz de chat o conversación se muestran en la tabla 65.

Tabla 65. *Pruebas de interfaz de conversación*

Prueba 4. Conversación			
prueba	**Entrada o Acción de usuario**	**resultado esperado del sistema**	**confirmación**
P 4.1	se presiona en la caja de texto y se escribe el mensaje a envía	la aplicación muestra en la lista el mensaje enviado con los datos del usuario y la fecha	SI
P 4.2	se presiona en la caja de texto y se envía un mensaje nulo	la aplicación muestra en la lista el mensaje enviado con los datos del usuario y la fecha	SI
P 4.3	se selecciona un mensaje	la aplicación muestra un mensaje de eliminar conversación	SI
P 4.4	se presiona aceptar	la aplicación elimina el mensaje y quita de la lista de conversación automáticamente	SI
P 4.5	se cancela la eliminación	la aplicación vuelve a la lista de mensajes sin eliminar el mensaje	SI

Nota. La tabla muestra todas las pruebas realizadas a la interfaz de chat o conversación
Elaborado por: Andrea Martínez y Michael Flores

Después de realizar las pruebas de funcionalidad verificamos que los objetivos planteados para el proyecto son cumplidos satisfactoriamente según los casos de uso planteados, dejando como hallazgo las tablas de verificación de funcionalidad con los resultados obtenidos, evidenciando que la aplicación Android funciona con

normalidad sin presentar fallos en la programación ni alterar procesos dentro del Smartphone.

4.2.3 Pruebas de tiempo de respuesta

4.2.3.1 Herramientas utilizadas

Para el desarrollo de las pruebas se utilizaron las siguientes herramientas.

La primera herramienta que se usó para determinar la velocidad de respuesta fue la aplicación Jmeter (jakarta-jmeter-2.5.1) La aplicación de escritorio Jmeter es un software de código abierto, una aplicación Java puro 100% diseñado para cargar la conducta funcional de prueba y medir el rendimiento. Originalmente fue diseñado para probar las aplicaciones web, pero desde entonces se ha expandido a otras funciones de prueba. Jmeter puede ser utilizado para probar el rendimiento tanto en recursos estáticos y dinámicos (archivos, lenguajes web dinámicos – PHP, Java, ASP.NET, etc. -, objetos Java, Bases de datos y consultas, servidores FTP y más). Se puede utilizar para simular una carga pesada en un servidor, grupo de servidores, la red o el objeto para probar su resistencia o para analizar el rendimiento general bajo diferentes tipos de carga. Se puede utilizar para hacer un análisis gráfico de rendimiento o para probar su comportamiento / objeto de servidor / script bajo carga pesada concurrente. (Apache Software Foundation, 2014).

Jmeter en su versión 2.5.1 es una versión totalmente estable que a diferencia de versiones anteriores trabaja con tipos de pruebas para Webservices a través del envío de archivos con extensión .xml con estructura SOAP (Simple Object Access Protocol) permite la comunicación por medio de intercambios de datos en formato simple, a continuación una imagen de la interfaz gráfica de Jmeter como se observa en la figura 36.

Aplicación Jmeter

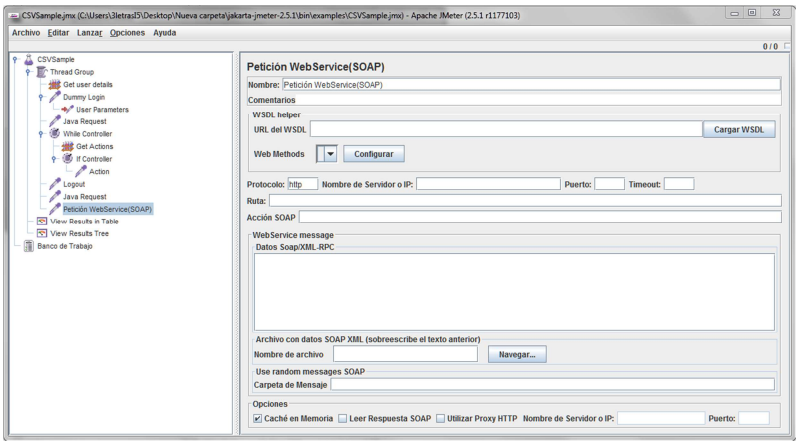

Figura 36. Aplicación Jmeter
Elaborado por: Andrea Martínez y Michael Flores

SoapUi

Para la simulación de los archivos con estructura SOAP utilizamos la aplicación SoapUI (SoapUI-x32-5.0.0) SoapUI es una solución multiplataforma de pruebas funcionales libre y de código abierto. Con unas características fáciles de usar interfaz gráfica, y de clase empresarial, SoapUI le permite crear y ejecutar pruebas funcionales, de regresión, de cumplimiento y de carga automatizadas con facilidad y rapidez. En un solo entorno de prueba. La interfaz gráfica de la aplicación SoapUi se detalla en la figura 37.

Aplicación SoapUI

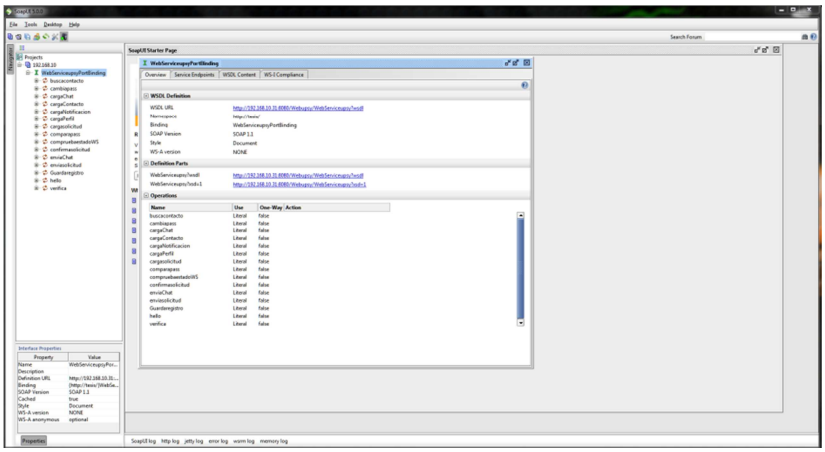

Figura 37. Aplicación SoapUI
Elaborado por: Andrea Martínez y Michael Flores

4.2.3.2 Pruebas de estrés

Para evaluar la robustez, confiabilidad y disponibilidad de la aplicación web, se realizarán pruebas de estrés, enfocadas en los tiempos de respuesta de los servicios disponibles a la aplicación Android. Considere dichos servicios trabajan como intermediarios entre la base de datos y la aplicación Android.

Las pruebas consisten en el envío, ejecución de peticiones y condiciones de forma masiva para determinar cuál es su tolerancia al uso extremo del usuario, prácticamente el objetivo es sobrecargar la aplicación móvil hasta determinar un punto de quiebre donde los servicios web muestren Bug(defectos de la programación) riesgosos para el usuario y su información.

Desarrollo de las pruebas de estrés

Las pruebas de estrés se realizaron con los siguientes datos registrados en Jmeter:

- El número propuesto de usuarios que acceden a los servicios web es de 500 usuarios
- El tiempo para la prueba es de 60 segundos
- El número de iteraciones repetición de la prueba será de 5 repeticiones por prueba
- El servicio web que se utilizara para la prueba será el servicio de verificación de servicio, este servicio se encarda de informar el estado de los servicios web a la Aplicación Android, informándole si existe problemas en los servicios y no están disponibles o que se encuentran en funcionamiento correcto
- La direccin del webservices para las pruebas es: http://192.168.10.31:8080/Webupsy/WebServiceupsy?wsdl

Se colocan los datos anteriormente mencionados en la aplicación Jmeter para realizar las pruebas dando como resultados:

Un rendimiento de 2950.259/minutos en un total de 5000 muestras que implican solicitud y respuesta del servicio web, con un margen de desviación de 988 demoras en el servicio web, una media de 1891 y una mediana de 1641 exitos en en el envio y recepcion de datos de parte del servicio web. dando como resultado un rendimiento

óptimo para un número de 500 usuarios en 60 segundos simultáneos, esto se puede observar en la figura 38.

Pruebas de rendimiento y estrés.

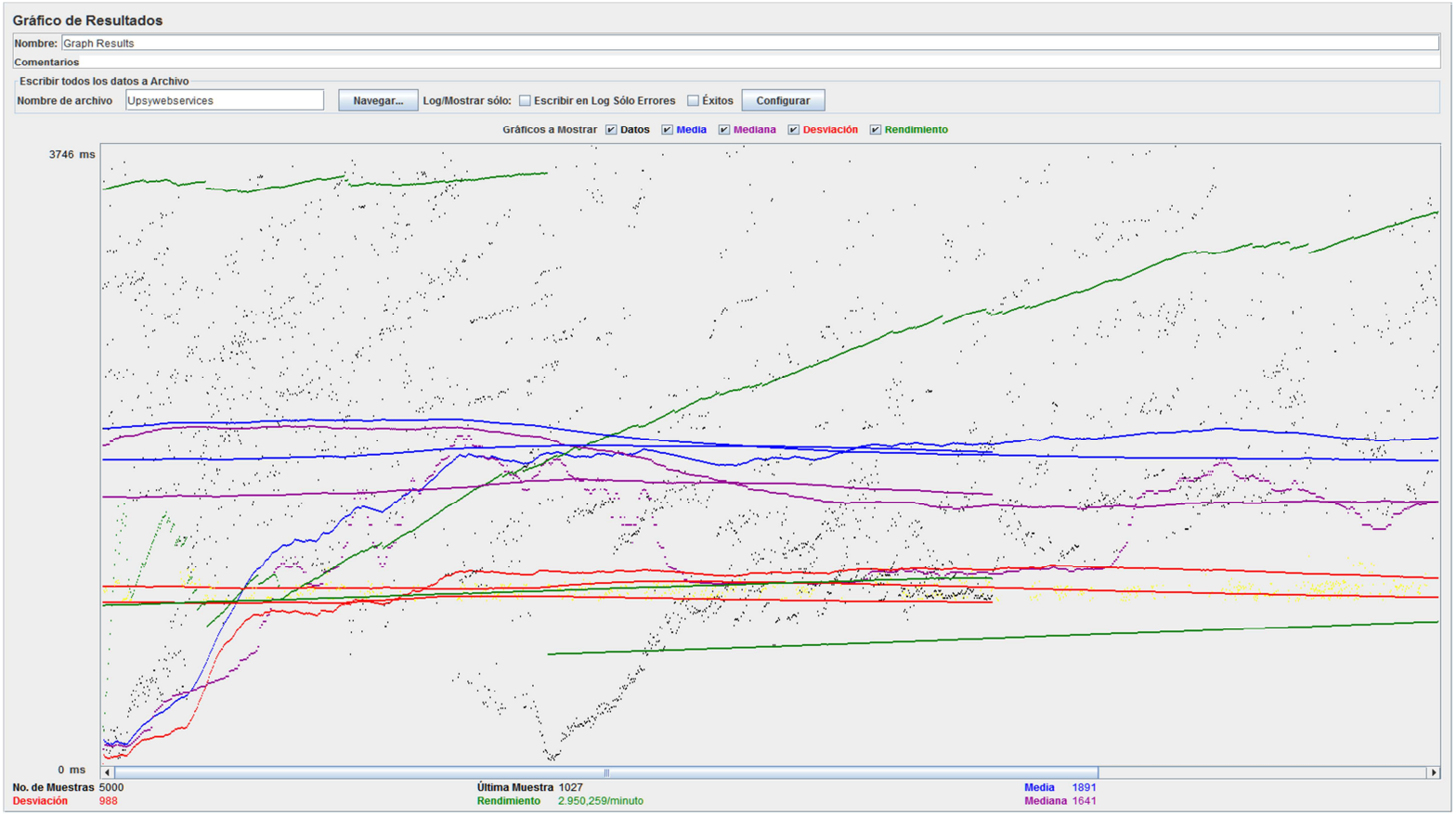

Figura 38. Gráfico en Jmeter de pruebas de rendimiento
Elaborado por: Andrea Martínez y Michael Flores

En la figura 39 se puede observar las respuestas exitosas en color rojo versus las respuestas con alerta o error en color morado que se le hicieron al servicio web todo en milisegundos. En un ambiente de prueba hostil de 83 usuarios accediendo al servicio web por segundo con un total de 500 usuarios en un periodo de 60 segundos con un total de 5000 muestras

Tiempo de respuesta del servicio web

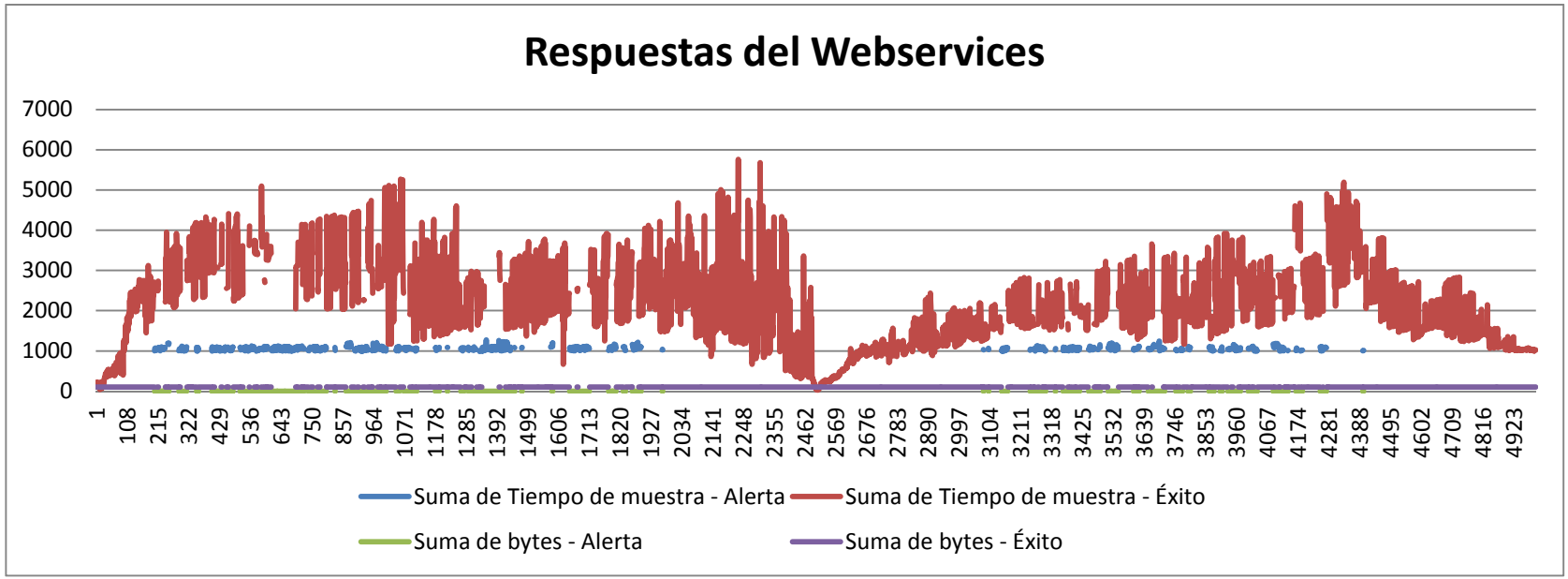

Figura 39. Muestra los resultados de los tiempos de respuesta del WebService
Elaborado por: Andrea Martínez y Michael Flores

4.2.4 Pruebas de compatibilidad

Estas pruebas se realizan con el fin de comprobar la compatibilidad de la aplicación desarrollada denominada “UPSY” con el Sistema operativo Android que use el SmartPhone. Para que la aplicación sea considerada compatible con las versiones de Android que ha publicado y de esta manera el usuario final podrá usar la aplicación Móvil de una forma efectiva sin dificultades por incompatibilidad. Considerando que Android en cada versión publicada, presenta características significativas, que necesita la aplicación Móvil para funcionar. A continuación se detallan las pruebas y los resultados obtenidos con las versiones publicadas de Android.

Para ello se instaló la aplicación “UPSY” en diferentes dispositivos con versiones diferentes dándonos como resultado la Tabla 66. Esta contiene un resumen de las versiones de Android y la compatibilidad con la aplicación “UPSY”.

Tabla 66. *Pruebas de compatibilidad*

Pruebas de compatibilidad			
Versión	**Codename**	**Api**	**Compatibilidad**
1.6	Donut	4	NO
2.1	Eclair	7	NO
2.2	Froyo	8	NO
2.3-2.3.2	GingerBread	9	NO
2.3.3-2.3.7		10	NO
3.1	Honeycomb	12	NO
3.2		13	NO
4.0.3-4.0.4	Ice Cream sandwich	15	SI
4.1	Jelly Bean	16	SI
4.2x		17	SI
4.3		18	SI
4.4x	KitKat	19	SI
5.0	Lollipop	20	SI
5.x		21	SI

Nota. La tabla muestra la compatibilidad de la aplicación Android desarrollada con las diferentes versiones de Android existentes.
Elaborado por: Andrea Martínez y Michael Flores

Las pruebas entregan como resultado que la aplicación móvil no puede funcionar en versiones menores a la 4.0.3 de Android con el API 15 debido a las funciones que se usan en la aplicación, que fueron lanzadas a partir del API 15, sin estas características el funcionamiento de la aplicación no sería apto para un uso constante y normal.

CONCLUSIONES

- Se cumple con el objetivo general planteado, permitiendo enviar y recibir mensajes entre usuarios de tipo texto además de recibir notificaciones institucionales.
- El sistema operativo Android demostró total estabilidad en el desarrollo de aplicaciones para Smartphone, facilitando el procesamiento de código y garantizando su funcionamiento.
- XP es una metodología apropiada para el desarrollo de aplicaciones móviles, debido a, que nos permite ir determinando el funcionamiento y avances de la aplicación en cortos plazos.
- En el mercado mundial de ventas de smartphones se observa un crecimiento liderado por Android que impide que esta y otras aplicaciones queden obsoletas, por esta razón es viable la migración de aplicaciones en diferentes lenguajes de programación, para el sistema operativo Android.
- El procesamiento en segundo plano o uso de tares asíncronas para trabajo con servicios web a permitido la interconexión entre la aplicación Android cliente y el servidor web, sin sufrir daños en el traslado de información, dejando de lado problemas por latencia, tiempo de respuesta, velocidad entre otros.

RECOMENDACIONES

- Se recomienda para la implementación de aplicaciones Android el uso de páginas oficiales de Android, donde se encuentra la información oficial de versiones y características de las mismas.
- La aplicación desarrollada funciona bajo el sistema operativo Android desde su versión de Api 15 y sistema operativo 4.0.3 ICE CREAM SANDWICH, ofreciendo todas sus características, al conectar el Smartphone Android a la red inalámbrica donde se encuentre el servidor de webservices.
- Para el funcionamiento adecuado de la aplicación móvil se debe recalcar que debe haber la conexión con el servidor que solo funcionará dentro de la Universidad
- Se debe realizar las pruebas en un dispositivo móvil, ya que el simulador no puede mostrar el funcionamiento real de la aplicación móvil.
- Es necesario documentar el proyecto siguiendo las recomendaciones que presenta la ingeniería de software.
- Se recomienda promover el uso de aplicaciones móviles dentro del medio estudiantil, así se expondrían los beneficios de uso dentro de la universidad.

LISTAS DE REFERENCIAS

Android Curso UPV. (2011). *www.androidcurso.com*. Recuperado el 2015 de 02 de 22, de http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/146-las-versiones-de-android-y-niveles-de-api

Apache Software Foundation. (2014). *jmeter.apache.org*. Recuperado el 23 de 02 de 2015, de http://jmeter.apache.org/

Báez, M. (2006). *Introduccion a Android*. Recuperado el 09 de 06 de 2014, de Introduccion a Android: http://pendientedemigracion.ucm.es/info/tecnomovil/documentos/android.pdf

Beck, K. (2012). *Una explicación de la Programación Extrema.* Addison-Wesley.

Bresano, M. (1996). *www.clubdelsuran.com.a*. Recuperado el 15 de 10 de 2014, de http://www.clubdelsuran.com.ar/site/materiales/proyecto/diagramas_del_uml.pdf

Diclase. (2007). *Diclase.* Recuperado el 20 de 11 de 2014, de http://lsi.ugr.es/~mvega/docis/diaclase.pdf

Flores, E. (2015). *Ingenieria de Software*. Recuperado el 24 de 09 de 2014, de http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html

Joskowicz. (2008). *Joskowicz*. Recuperado el 07 de 06 de 2014, de http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf

Manuel Baez, Á. B. (s.f.). *Introducción a Android.* Recuperado el 2015 de 01 de 08, de http://pendientedemigracion.ucm.es/info/tecnomovil/documentos/android.pdf

Motyka, J. (s.f.). *https://www.descargarandroid.com/historia-android/*. Recuperado el 20 de 02 de 2015, de https://www.descargarandroid.com/historia-android/

Ochando, F. A. (2003). *Descargar Android*. Recuperado el 2014 de 12 de 12, de https://www.descargarandroid.com/historia-android/

Pallioto, D. (2008). *http://www.docirs.cl/*. Recuperado el 15 de 11 de 2014, de http://www.docirs.cl/uml.htm

Patricio, S. (30 de 10 de 1996). *users.dcc.uchile.cl*. Recuperado el 15 de 10 de 2014, de http://users.dcc.uchile.cl/~psalinas/uml/modelo.html

Pérez, E. (2011). *elandroidelibre*. Recuperado el 2015 de 01 de 10, de http://www.elandroidelibre.com/2014/06/la-historia-de-android-en-imagenes-desde-sus-inicios-hasta-ahora.html

Pressman, R. S. (2010). *Ingenieria del Software Un enfque práctico.* Mexico: Mc Graw Hill.

Rodriguez, M. (2014). *www.ehu.eus*. Recuperado el 13 de 01 de 2015, de http://www.ehu.es/mrodriguez/archivos/csharppdf/ServiciosWeb/WebServices.pdf

Schmuller, J. (2001). *Aprendiendo UML en 24 Horas.* Recuperado el 22 de 01 de 2015, de https://mega.co.nz/#!eEUVXRCQ!BKIdSQezXdhJBrBxDlQY5ElwnccRMGC0Pce02xyTMcM

SmartBear Software. (2005). *www.soapui.org*. Recuperado el 23 de 02 de 2015, de http://www.soapui.org/

Universidad Politécnica Salesiana, Estatuto de la Federación de Estudiates. (2011). *Estatuto de la Federación de Estudiates.* Cueca.

Wells, D. (2013). *Extreme Programing*. Recuperado el 07 de 06 de 2014, de Extreme Programing: http://www.extremeprogramming.org/values.html