UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:
Ingenieros de Sistemas

TEMA:
ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA INFORMÁTICO BASADO EN APLICACIÓN MÓVIL Y GEOLOCALIZACIÓN PARA EL REGISTRO DE BACHES EN LAS VÍAS DE LA CIUDAD DE QUITO.

AUTORES:
CRISTHIAN JAVIER CASTRO GAIBOR
JONATHAN PAUL VELASCO NAVAS

TUTOR:
RODRIGO EFRAÍN TUFIÑO CÁRDENAS

Quito, julio del 2018

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Castro Gaibor Cristhian Javier con documento de identificación Nº 1727003939 y Velasco Navas Jonathan Paul, con documento de identificación Nº 1723017669, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación con el tema: ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA INFORMÁTICO BASADO EN APLICACIÓN MÓVIL Y GEOLOCALIZACIÓN PARA EL REGISTRO DE BACHES EN LAS VÍAS DE LA CIUDAD DE QUITO, mismo que ha sido desarrollado para optar por el título de INGENIEROS DE SISTEMAS en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

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

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

CRISTHIAN JAVIER
CASTRO GAIBOR
CI: 1727003939

JONATHAN PAUL
VELASCO NAVAS
CI: 1723017669

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, con el tema: ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA INFORMÁTICO BASADO EN APLICACIÓN MÓVIL Y GEOLOCALIZACIÓN PARA EL REGISTRO DE BACHES EN LAS VÍAS DE LA CIUDAD DE QUITO, realizado por Cristhian Javier Castro Gaibor y Jonathan Paul Velasco Navas, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, julio del 2018

RODRIGO EFRAÍN TUFIÑO CÁRDENAS

CI: 1717646390

DEDICATORIA

Dedicado a la memoria de mi hermano Alex Castro, y abuelita Olivia Benavides.

Cristhian Javier Castro Gaibor

A mi madre, mi padre, mi hermano, mi hija y mi alma gemela. Quienes a pesar de los problemas y dificultades me han brindado su apoyo incondicional. Toda su abnegación y consejos para ser una mejor persona día a día, a ustedes les dedico este logro.

Jonathan Paul Velasco Navas

AGRADECIMIENTO

A mis padres Alex Castro y Yolanda Gaibor, por la confianza depositada en mí, que, con su amor, dedicación, y apoyo incondicional, son los pilares en mi vida que me permitieron lograr este objetivo. A mi hermano Lenin Castro por su compañía, y cariño hacia mí.

A mi compañero y amigo Paul Velasco, con quien compartimos el esfuerzo de haber conseguido la culminación de este trabajo.

A todos mis amigos, en especial Daniel Yangua y Victoria Alomoto por su amistad, compañerismo, y por las anécdotas vividas en nuestra vida universitaria.

Al Ingeniero Rodrigo Tufiño por la guía recibida en el transcurso del proyecto.

Cristhian Javier Castro Gaibor

Agradezco a Dios por haberme acompañado y guiado a lo largo de toda mi vida en especial durante los años de mi carrera universitaria, por ser mi fortaleza en momentos de debilidad y por brindarme una vida llena de aprendizajes experiencias y felicidad.

A mis padres Gyovanni Velasco y Carmita Navas, por apoyarme incondicionalmente en todo momento, enseñarme valores como la humildad, respeto, y la responsabilidad, por haber confiado en mí dándome la oportunidad de tener una excelente educación.

A mi hermano Byron Velasco, por brindarme ánimos y ser mi apoyo incondicional, sus oportunas palabras siempre me han hecho sentir mejor.

A mi incondicional amigo Cristhian Castro, con quien compartimos momentos gratos en la universidad y por el esfuerzo de haber conseguido la culminación de este trabajo.

A mis amigos, en especial Daniel Yangua, Victoria Alomoto y Andrés Carrera con quienes he compartido alegrías y tristezas, largas jornadas de estudio y también grandes momentos de esparcimiento. Gracias por su amistad.

A mi alma gemela Berenice Nolivos y a nuestra hija Eimy las dos siempre han sido y serán mi fuente de inspiración más grande.

A todos los profesores de la carrera de Ingeniería de Sistemas, quienes a diario aportan con un granito de arena para formarnos como excelentes profesionales

Al Ingeniero Rodrigo Tufiño, por aceptar dirigir nuestro trabajo de titulación, por todo el tiempo y esfuerzo dedicado para sacar nuestro proyecto adelante.

Jonathan Paul Velasco Navas

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente documento detalla el ciclo de vida de desarrollo del sistema informático basado en aplicación móvil y geolocalización para el registro de baches en las vías de la ciudad de Quito, mismo que se enfoca en la automatización del procedimiento que se lleva a cabo por parte de las Administraciones Zonales del Municipio del Distrito Metropolitano de Quito, para la priorización y atención de estas anomalías.

La elaboración de este proyecto se centró en la Administración Zonal "Eloy Alfaro", atravesando por el análisis de la problemática actual, definición de los requerimientos, diseño, construcción, pruebas e implementación del sistema informático.

La solución construida busca reducir el tiempo empleado en cada una de las etapas por los actores involucrados en el proceso. El sistema se encuentra conformado por un aplicativo móvil que permite el registro y geolocalización de baches mediante la captura de una fotografía en el lugar del acontecimiento, junto a esto también es posible validar la reparación de cada uno de los baches registrados, mismo que es constatado con la toma de otra imagen, por otro lado, una aplicación Web cuya función es la de visualizar, administrar, y controlar los registros hechos mediante el aplicativo móvil.

ABSTRACT

This document details the development life cycle of the computer system based on mobile application which uses geolocation for the registration of potholes in the roads of Quito city. This system focuses on the automation of the actual procedure carried out by the administration of each zone inside the Quito District Metropolitan Municipality, for the prioritization and attention of these anomalies.

The elaboration of this project was centered in the Zone Administration "Eloy Alfaro". It consisted of doing a full analysis of the current problems, definition of the requirements, design, construction, tests and implementation of the computer system.

The solution built seeks to reduce the time spent by the individuals involved on the process on each stage. The system is made up of a mobile application that allows the registration and geolocation of potholes by capturing a photograph at the site of the event. Additionally, by using this application it is possible to validate the repair of each registered pothole, which is then verified by taking another image. Finally, as part of the project, a Web application was created with the main functionality of visualize, manage, and control the records made through the mobile application.

INTRODUCCIÓN

La automatización de tareas cotidianas en nuestra vida nos permite simplificar y volver menos repetitivos ciertos procesos que llevamos a cabo con frecuencia, más aún con la masificación de la tecnología en varias áreas en las que nos desenvolvemos diariamente, resulta difícil encontrar procedimientos en los que la tecnología no pueda ser participe con el objetivo de mejorar su eficiencia y a la postre los resultados obtenidos.

Basados en esto, se busca implementar una solución tecnológica la cual permita la automatización del proceso que llevan a cabo las administraciones zonales del Municipio de Quito, para el registro y control de los baches en las vías del Distrito Metropolitano.

El tema social generado en la ciudadanía por el estado y adecuado mantenimiento de las vías ha hecho que este tema haya sido tratado con especial interés en varias ciudades alrededor del mundo, en las cuales ya han sido implementadas soluciones informáticas similares a la que se plantea con la realización de este proyecto es así como:

En la ciudad de México en el año 2016 se puso en marcha el programa Bache 24, con la finalidad de atender en un plazo no mayor a 24 horas, los reportes realizados por la ciudadanía en la red vial primaria, mediante una aplicación móvil. (Bache-24, 2018)

Así también, la ciudad de Buenos Aires en su sitio Web oficial aporta con la sección para reportar un bache en las vías, que de igual manera a la ciudad de México es un proyecto abierto a la ciudadanía. (Buenos Aires Ciudad, 2018)

En ambos casos los principales inconvenientes que surgen es la inadecuada información generada por el ciudadano. Mediante estos antecedentes, si bien resulta beneficioso hacer partícipe a la ciudadanía en el proceso, a su vez en situaciones es

también conflictivo por motivo de que la información proveniente de los ciudadanos podría ser prestada a confusión con otros incidentes diferentes a los baches, lo que causaría que el análisis y control que se tenga sobre la información generada no sea correcto.

Es por esta razón que el proyecto que se plantea a continuación tiene el fin de automatizar el proceso interno llevado a cabo por los funcionarios de la Municipalidad, resultando ser una fase previa a que en un futuro se pueda dar la apertura a la ciudadanía.

Justificación

En la actualidad el Municipio del Distrito Metropolitano de Quito, cuenta con varios moderadores que realizan el proceso de registro de baches de forma manual. Por lo tanto, para la Unidad de Informática es necesario un registro y control automatizado de los baches en las vías de la ciudad, para la solución eficiente de esta problemática.

La herramienta informática simplificará el trabajo de los moderadores registrando la ubicación geográfica, y fotografía del bache, en un proceso sencillo e intuitivo para el usuario, dejando de lado el proceso arcaico que se ha venido llevando a cabo durante mucho tiempo.

Objetivo general

Automatizar el registro y ubicación geográfica de los baches en las calles del Distrito Metropolitano de Quito.

Objetivos específicos

Analizar el proceso actual empleado por la Municipalidad y las deficiencias existentes para el registro de baches en las calles de la ciudad de Quito, además de la problemática que esto genera en la ciudadanía.

Diseñar un sistema automatizado para el registro de anomalías en las calles de la ciudad de Quito.

Construir un sistema informático que registre la ubicación geográfica automáticamente por medio de una fotografía de los baches que se presentan en las calles de la ciudad de Quito.

Llevar a cabo una prueba piloto en la Administración Zonal del Municipio de Quito "Eloy Alfaro", para evaluar el desempeño y la usabilidad del sistema informático.

Capítulo 1

Marco Teórico

El presente proyecto emplea la investigación aplicada de un problema, el cual requiere ser intervenido y posteriormente mejorarlo partiendo del conocimiento adquirido. En el manual de Frascati se rescata que este tipo de investigación implica la consideración de todos los conocimientos existentes y su profundización, en un intento de solucionar problemas específicos. También se describe que los resultados de la investigación aplicada recaen, en primer lugar, sobre un producto único o un número limitado de productos, operaciones, métodos o sistemas. La investigación aplicada desarrolla ideas y las convierte en algo operativo. (Organización para la Cooperación y el Desarrollo Económicos "OCDE", 2013)

1.1 Metodologías

En este apartado se hace mención de las bases en las que se fundamenta el progreso del proyecto, por lo que se detalla temas como metodologías de gestión y desarrollo del proyecto y los lineamientos a las que se ciñen.

La metodología global, practicada a lo largo del proyecto es Scrum, por ser un conjunto de técnicas para el trabajo conjunto a fin de conseguir los mejores resultados posibles. Ya en la fase de programación se hace una diferenciación, en dicha etapa se aplica la metodología XP. El combinar ambas metodologías no es para nada extraño como menciona Henrik Kniberg en su libro "*Scrum y Xp desde las trincheras*", Scrum se enfoca en las prácticas de organización y gestión, mientras que XP se centra más en las prácticas de programación. Esa es la razón de que funcionen tan bien juntas: tratan de áreas diferentes y se complementan entre ellas. (Kniberg, 2007)

1.1.1 Metodología Scrum

Scrum nace en el año 1986, cuando en base a sus estudios realizados en nuevas prácticas de producción Hirotaka Takeuchi e Ikujijo Nonaka, publicaron el artículo "The New Product Development Game", en este se hace mención de las nuevas formas de gestionar proyectos en donde la rapidez y flexibilidad son necesarios. Si bien Scrum nace como un conjunto de buenas prácticas enfocado a productos tecnológicos, en 1993, Jeff Sutherland lo adapta en un modelo propio de desarrollo de software en Ease/Corporation, y posteriormente en 1996 se presentaron las practicas formales usadas en el desarrollo de software que finalmente fueron incluidas en el manifiesto ágil. (Gallego, 2012)

Scrum al ser una metodología ágil emplea un desarrollo iterativo e incremental, es precisamente estas iteraciones las que definen la metodología, concretamente Scrum las llama "sprints". Por medio de la figura No.1 se explica el proceso general llevado a cabo en Scrum.

Proceso llevado a cabo en la metodología Scrum

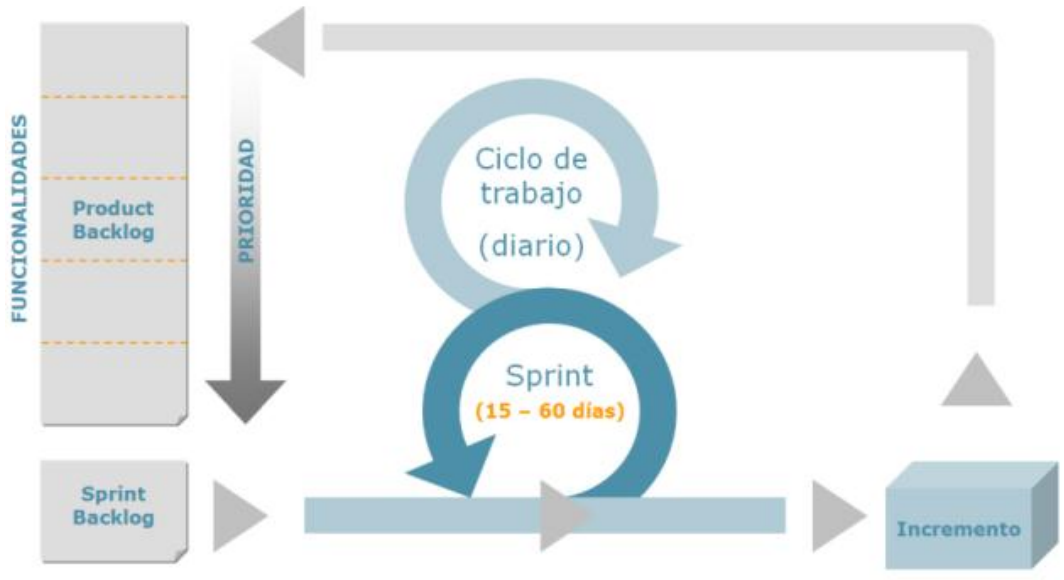

Figura 1. Proceso llevado a cabo en la ejecución de la metodología Scrum.
Elaborado por: (Palacio, 2008)

Los componentes y conceptos esenciales que presenta Scrum a lo largo de su aplicación son:

1.1.1.1 Reuniones

En este apartado destacan la planificación, seguimiento y revisión del sprint.

- **Planificación del sprint**

Esta es llevada a cabo previo al inicio de cada sprint, cuyo objetivo es definir el trabajo que será cumplido con la ejecución de esta iteración. Producto de esta reunión surge el "sprint backlog" que no es más que la lista de tareas a realizarse.

- **Seguimiento del sprint**

Se trata de una reunión diaria, entre los miembros del equipo en la cual se habla del avance que se lleva de cada tarea. En ella también se comunica posibles barreras que imposibiliten el cumplimiento de una tarea, con esto el equipo busca una solución óptima.

- **Revisión del sprint**

En esta se presentan los resultados procedentes de la elaboración de las tareas. Cabe mencionar que en ella se cuenta con la participación del cliente a fin de generar retroalimentación.

1.1.1.2 Elementos

Son los artefactos surgidos en cada una de las fases por las que atraviesa el desarrollo del proyecto.

- **Product Backlog**

Es el inventario en el cual se listan todas las funcionalidades o requisitos que el proyecto debe cumplir, es realizada por el cliente bajo la asesoría del Scrum Master, el que guiará en el coste estimado para completar cada requerimiento. (Gallego, 2012)

- **Sprint Backlog**

Es el listado obtenido de la planificación de un sprint, en el cual se enumera las tareas a realizar y el miembro del equipo asignado para su cumplimiento, también se detallan tiempos estimados y recursos necesarios en su ejecución.

1.1.1.3 Roles o responsabilidades

Existen tres actores principales que influyen dentro del proyecto. Y para lo cual Scrum defines las responsabilidades de cada uno de ellos.

- **Product Owner**

Denominado propietario del producto es la persona conocedora del negocio del cliente y la visión del producto final, como vimos es quien realiza y asigna prioridades a los requisitos a ser escritos en el "product backlog".

- **Scrum Master**

Es la persona que se encarga de la correcta ejecución de la metodología en el proyecto, y cuya función primordial es la de actuar como enlace entre el dueño del producto y el equipo de desarrollo.

- **Equipo de desarrollo**

Son equipos pequeños comúnmente de 3-9 personas, aunque esto es dependiente del tamaño del proyecto, son quienes se encargan del desarrollo de los requisitos planeados en el "product backlog", se organiza en la distribución de tareas de cada uno de los "sprints". Es siempre necesario que el equipo de desarrollo conozca el funcionamiento de Scrum.

1.1.2 Metodología XP

La Programación Extrema o "Xtreme Programming" por sus siglas en inglés, cuyos mentores son Kent Beck, Ward Cunningham y Ron Jeffries tiene su origen en 1996, está englobada dentro de las llamadas metodologías ágiles, XP enfatiza la satisfacción

del cliente. Como relata su sitio oficial, *"En lugar de ofrecerle todo lo que el cliente pueda desear en una fecha muy lejana en el futuro, este proceso ofrece el software que necesita a medida que lo necesita"*. (Xtreme Programming, 2013)

XP se centra en el trabajo en equipo, precisamente de grupos de programación pequeños comúnmente en parejas, donde la comunicación resulte más factible. Y a decir de la comunicación es un punto clave dentro de la metodología ya sea entre jefes de proyecto, programadores y sobre todo el cliente, siendo este una parte primordial y debiendo estar inmiscuido dentro del proceso.

1.1.2.1 Valores XP

De manera innata XP plantea 5 valores los cuales forman parte de su esencia, que se detallan a continuación.

- **Comunicación**

Contrario a otras metodologías, XP minimiza la documentación, en contraste a esto enfatiza la comunicación cara a cara, la interacción personal por parte de los actores que se inmiscuyen en el proceso. Lo que produce que el avance sea rápido y genere documentación estrictamente necesaria.

Así mismo, el cliente se ve involucrado en el equipo de trabajo, cuyo aporte oportunamente comunicado logra que el equipo de desarrolladores busque soluciones y enfrenten posibles cambios de manera óptima.

- **Simplicidad**

El segundo valor nace a partir de la frase "realizar solo lo necesario" y no perder tiempo en futuros requerimientos que pueden aparecer o no hacerlo. Con esto se busca que el progreso del proyecto sea rápido y seguro. (Calabria, Píriz, & Mousques, 2003)

- **Retroalimentación (Feedback)**

La participación del cliente aportando retroalimentación continua, hace que el proyecto se enfoque y dirija hacia donde el quiere, y así también los requerimientos y cambios que pudieran existir serán solventados los más pronto posible lo que reduce el tiempo de desarrollo.

- **Respeto**

Este punto se enfoca en el respeto hacia todos los integrantes del proyecto, fomentando el trabajo en equipo, donde no se tomarán decisiones repentinas sino al contrario debe haber un consenso entre todos los actores, todo el equipo debe trabajar como uno. (Borja Yolanda, 2018)

- **Coraje**

Que el equipo de desarrollo tenga la valentía para afrontar los cambios a sabiendas que pueden surgir inesperadamente, de modificar el código en cualquier momento por cualquier miembro del equipo sabiendo que no se afectará el correcto funcionamiento del sistema.

1.1.2.2 Roles y responsabilidades

Al igual que en otras metodologías XP también define roles, por lo que se procede a definir los principales.

- **Programador**

Es la persona encargada de producir el código que da funcionamiento al sistema, es también mencionable que una de sus tareas es definir las pruebas unitarias.

- **Cliente**

Su responsabilidad principal es la de escribir las historias de usuario, asigna prioridad a estas, y hace pruebas funcionales que avalen el correcto cumplimiento de los requerimientos del sistema.

- **Encargado de pruebas (Tester)**

Su trabajo reside en ejecutar las pruebas, cuyos resultados son difundidos al equipo. Por la naturaleza de su oficio brinda soporte al cliente al momento de llevar a cabo las pruebas funcionales.

- **Encargado de seguimiento (Tracker)**

Es quien lleva el seguimiento del progreso real de las iteraciones, en función de esto evalúa el porcentaje de cumplimiento de objetivos en cada una de las iteraciones, y en caso de requerirlo efectúa ajuste de tiempo y recursos que permita la finalización de las metas propuestas.

- **Entrenador (Coach)**

Es la persona conocedora del proceso y aplicación de la metodología, provee al equipo las guías que hacen que el proyecto adopte las buenas prácticas concebidas por XP.

- **Consultor**

Su función es la de dar soporte en caso de que el equipo requiera de un agente externo para la solución de un problema específico.

- **Gestor (Big boss)**

Su labor esencial es la coordinación, creando las condiciones adecuadas para el equipo, es el vínculo entre clientes y programadores.

1.1.2.3 Prácticas XP

La metodología establece un conjunto de reglas y prácticas que se pueden agrupar en:

- **Planificación**

Aquí consta la creación de "historias de usuario", originadas por el cliente y con participación de los programadores en la estimación de tiempos de desarrollo de cada una.

Historias de usuario

Por medio de este instrumento XP especifica los requisitos del sistema. De manera general en ella se describe las funcionalidades que el sistema debe cumplir. Físicamente se trata de tarjetas en las que consta fecha de creación, descripción del requerimiento, objetivos que se quieren cumplir con la realización, entre otros datos que el equipo considere relevantes.

- **Diseño**

Dentro de este grupo de reglas XP destaca entre algunos conceptos los siguientes:

Simplicidad

Un diseño simple permite ser implementado rápidamente en contraste a un diseño complejo. Es por esto por lo que XP alienta a implementar el diseño más simple que funcione.

Recodificación

La metodología XP sugiere escribir nuevamente el código sin cambiar su funcionalidad siempre que sea posible, con el objetivo de volverlo más simple, conciso y entendible.

- **Desarrollo**

Disponibilidad del cliente

En esta se especifica claramente la participación del cliente durante el desarrollo del proyecto, esto resulta ser fundamental ya que, si bien el cliente de inicio plantea las historias de usuario, estas pueden producir mal entendidos o requerir mayor detalle en explicación para la adecuada ejecución por parte de los programadores. Esto siempre en función de la esencia de XP en promover la comunicación directa antes que generar documentación.

Programación dirigida por las pruebas

Tradicionalmente las pruebas son ejecutadas una vez concluido el desarrollo del software, XP propicia una guía inversa en la cual cada funcionalidad que sea programada debe ser sujeta a pruebas unitarias definidas por el grupo de programación, cuyo resultado condiciona el proceso de desarrollo.

Integraciones permanentes

El uso adecuado del control de versiones es primordial para el desarrollo del proyecto, todos los programadores deben trabajar sobre la última versión disponible es por este motivo que siempre se debe publicar las nuevas versiones aun cuando no estén concluidas en totalidad, siempre y cuando se encuentren sin errores. El hacer mejoras en versionas antiguas causa problemas de integración al final del proyecto lo que se traduce en alargue de tiempo.

- **Pruebas**

Si consideramos que cada funcionalidad desarrollada ha sido sometida a pruebas unitarias, se puede interpretar que el proceso de pruebas no es necesario.

Sin embargo, XP plantea que cada una de las historias de usuario desarrollada debe ser objeto de pruebas de aceptación por parte del cliente para dar por terminada dicha historia de usuario. Una prueba de aceptación no es más que una comprobación en donde el cliente verifica que el módulo cumple o no todas las funcionalidades para el que fue construido, comúnmente se las conoce como pruebas de caja negra, ya que solo importa el resultado arrojado dejando de lado el método diseñado para llegar a dicho resultado.

1.2 Herramientas y lenguajes de programación

A lo largo del desarrollo del sistema se hizo uso de las siguientes herramientas, y tecnologías que fueron seleccionadas en base al ajuste a las necesidades del sistema.

Se hizo una diferenciación de lenguajes de programación por la naturaleza de las aplicaciones a construir, resultando que el aplicativo Web emplea Python como lenguaje base, y a su vez la aplicación móvil maneja Java.

1.2.1 Lenguaje de programación Python

Python es un lenguaje de programación de código abierto se desarrolla bajo una licencia aprobada por OSI, por lo que es libremente utilizable y distribuible, incluso para uso comercial. Python se ejecuta del lado del servidor, entre sus características principales están: Es un lenguaje interpretado que proporciona una sintaxis sencilla y comprensible, es multiparadigma soporta programación orientada a objetos, programación imperativa, y es multiplataforma.

Algo que enfatiza es su comunidad, que es muy activa para colaboración con el código. Conjuntamente, su documentación es minuciosamente detallada, que produce un efecto positivo en la curva de aprendizaje y uso del lenguaje de programación. (Python, 2018)

1.2.2 Framework Django

Python tiene múltiples frameworks usados para el desarrollo Web, uno de ellos es Django que dentro del proyecto ha sido empleado en la aplicación de Web.

Django es gratis y de código abierto, las ventajas de su uso es que fomenta un desarrollo rápido, diseño limpio y pragmático con esto facilita la creación de aplicativos Web, su principio fundamental es el DRY (Don’t Repeat Yourself), que promueve el re-uso de código fuente.

De manera nativa el framework enfoca gran parte de su potencial en seguridad, ayuda a los desarrolladores a evitar muchos errores comunes, como la inyección de SQL, scripts entre sitios, falsificación de solicitudes y clickjacking. Su sistema de

autenticación de usuarios proporciona una forma segura de administrar cuentas de usuario y contraseñas. (Django, Django Framework, 2018)

En cuanto a la arquitectura que implementa Django, se basa en la conocida MVC (Modelo Vista Controlador), a pesar de ello posee reformas en el funcionamiento lo que produce un cambio de nomenclatura a MVT (Modelo Vista Template-Plantilla).

Para la comprensión del flujo de trabajo de la arquitectura interna de Django, se presenta la figura No.25. (Montero, 2012)

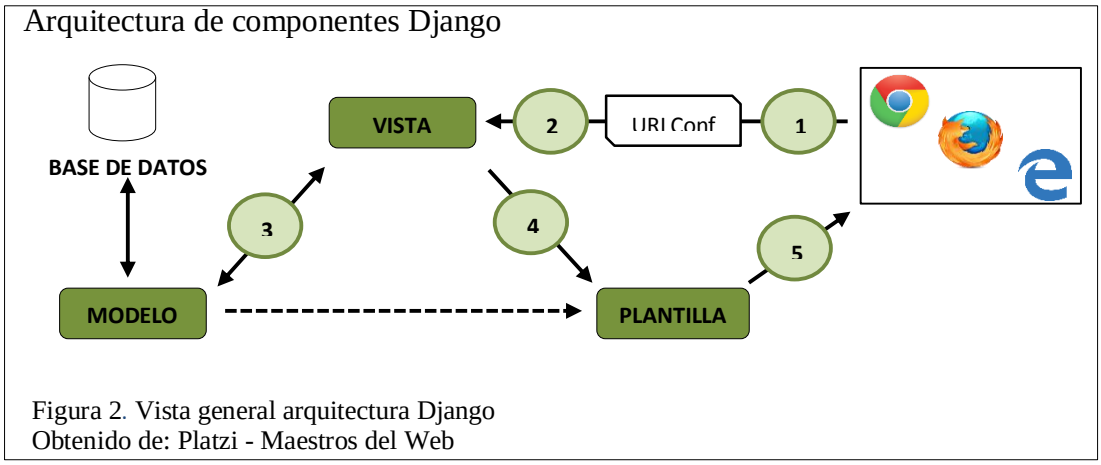

Figura 2. Vista general arquitectura Django
Obtenido de: Platzi - Maestros del Web

1. El navegador manda una solicitud.
2. El URLConf lee la solicitud y ubica la vista apropiada.
3. La vista interactúa con el modelo para obtener datos.
4. La vista llama a la plantilla.
5. La plantilla renderiza la respuesta a la solicitud del navegador.

1.2.3 RestFramework

Django frecuentemente se combina con otras aplicaciones, como es el caso de Rest Framework la que es usada para la construcción de API's Web basadas en los modelos de Django, entre las ventajas que ofrece es que admite fuentes de datos relacionales o no relacionales, también brinda un API navegable de fácil usabilidad para pruebas en ambiente de desarrollo. (DjangoRestFramework, 2018)

Rest Framework fue utilizado, para realizar las API's de servicios que permitan mantener una conexión entre las aplicaciones del sistema.

1.2.4 GeoDjango

Otra de estos complementos extra al framework es GeoDjango el que añade la posibilidad de trabajar con datos geográficos. Logrando la construcción de aplicaciones Web GIS. GeoDjango es distribuido bajo la licencia BSD que permite su libre uso. (Django, GIS, 2018)

Los modelos de datos del sistema demandaron uso de objetos del tipo geográfico, por lo que se hizo primordial usar GeoDjango para el oportuno almacenamiento y manejo de la geo información generada por el sistema.

1.2.5 Lenguaje de programación Java

Java es un lenguaje de programación orientado a objetos, empleado para el desarrollo de aplicaciones: desktop, Web y móviles, es así como una de las alternativas para Android es usar Java para la construcción de sus aplicativos. Concretamente la versión Java ME constituye un entorno sólido de seguridad e interfaces de usuario flexible para aplicaciones ejecutadas en dispositivos móviles. (Java, 2018)

1.2.6 Bootstrap

Para las interfaces gráficas de la aplicación Web, se decidió el empleo de una herramienta que nos permita abstraer la edificación de hojas de estilo desde cero.

Bootstrap es un framework front-end, gratuito para el desarrollo de páginas Web. Ofrece plantillas realizadas en base a HTML y CSS, además de un sin número de elementos que facilitan el diseño de interfaces para el usuario. Entre sus cualidades es la adaptabilidad a distintos dispositivos, y la compatibilidad con los diferentes navegadores Web actuales del mercado. (Bootstrap, 2018)

1.2.7 Editor de código Visual Studio Code

Es un editor de texto concretamente código fuente de distintos lenguajes de programación, desarrollado por Microsoft, sin embargo, es compatible para su ejecución en Linux y macOS.

Dentro de sus valores a destacar esta la gran variedad de extensiones (plugins), para potenciar su funcionamiento tales como diccionarios de sintaxis para auto completar el código, también se puede encontrar herramientas de depuración, y una de las fortalezas es la integración con GIT para el manejo de versiones y trabajo en equipo en la construcción de los proyectos, también se encuentra entre sus ventajas la consola integrada en la misma aplicación. Cabe mencionar que Visual Studio Code también es altamente personalizable cuenta con opciones de instalar temas, iconos, fuentes, y demás para hacerlo adaptable al gusto del usuario. (Visual Studio Code, 2018)

1.2.8 Android Studio

Android Studio el IDE de desarrollo oficial para aplicaciones móviles para el sistema operativo Android, es de licencia gratuita, y puede ser usado en los diferentes sistemas operativos (Windows, macOS, Linux). (Android Studio, 2018)

1.2.9 Power Designer

Power Designer, es una herramienta para la modelización de datos, que brinda diferentes técnicas para el modelamiento entre las que constan: conceptual, físico, lógico. Las ediciones actuales que provee Power Designer a sus usuarios. DataArchitect, empleada para la modelización de la BDD, BusinessAnalyst, que se centra en el modelamiento de procesos de negocios. Adicional a esto existe la EntrepriseArchitect que incluye ambas funcionalidades. (SAP, 2018)

1.2.10 StarUML

Es uno de los más populares instrumentos UML, que soporta un total de 11 tipos de diagramas UML: clases, objeto, casos de uso, componentes, despliegue, estado, entre

otros. Además, se puede crear diagramas entidad relación, usado en el diseño de base de datos. Otra de sus particularidades es el poder generar modelos a partir de sus modelos o crear uno desde código fuente mediante ingeniería inversa. Todo esto a través de extensiones, disponibles para su instalación. (Star UML, 2018)

1.2.11 JustinMind

Se trata de un instrumento para realizar prototipado de pantallas de aplicaciones móviles o Web, su interfaz permite construir las interfaces arrastrando y soltando la gran variedad de elementos UI proporcionados por la herramienta. (JustinMind, 2018)

1.2.12 PostgreSQL

PostgreSQL es un sistema gestor de base de datos relacionales de código abierto, ejecutable en los principales sistemas operativos, asimismo destaca su arquitectura, confiabilidad, integridad de datos, sólido conjunto de características, extensibilidad y su comunidad de código abierto. La herramienta oficial de administración del motor PostgreSQL, es pgAdmin que provee una interfaz instintiva y fácil de operar. (PostgreSQL, 2018)

1.2.13 PostGIS

Algo que lo caracteriza a PostgreSQL son sus complementos tan potentes como el módulo PostGIS, el que nos permite manejar BDD con carácter espacial con el fin de manipular datos geográficos dentro de ellas. (PostGIS, 2018)

1.2.14 Git

Git es un sistema de control de versiones de código abierto, generalmente es usado para la administración organizada del código de fuente evitando problemas de integración en las aplicaciones. La particularidad que distingue a Git de otras herramientas para el control de versiones es su modelo de ramificación ya que tiene

disponible la opción de crear varias sucursales independientes entre sí y de la rama principal para facilitar el trabajo en los equipos de desarrollo. (Git, 2018)

Capítulo 2

Análisis y Diseño

2.1 Análisis

Dentro de este apartado del capítulo dos, se analizará el problema que surge en el registro de baches en las vías del Distrito Metropolitano de Quito y su impacto en la ciudad, para esto se hace un desglose del proceso actual llevado a cabo por las administraciones zonales centrándose en la Administración Zonal "Eloy Alfaro".

2.1.1 Descripción del problema

En los últimos años la presencia de baches en las vías del Distrito Metropolitano de Quito se ha convertido en uno de los mayores problemas para la ciudadanía, lo cual ha provocado inconformidad e inseguridad, debido a que el deterioro de las vías es un causante de daños en los vehículos, accidentes de tránsito, e inconvenientes para los transeúntes dificultando su paso por las vías, en época de lluvias este problema se complica aún más puesto que el tamaño y número de baches aumenta en la ciudad causando mayor molestia en los ciudadanos.

El Distrito Metropolitano de Quito tiene 8.234,73 km de longitud de calzada que forman parte de más de 11 mil vías a lo largo de la ciudad, según la Empresa Pública Metropolitana de Movilidad y Obras Públicas (EPMMOP) el 80% de las vías en la ciudad requieren de rehabilitación, debido a que ya cumplieron con su vida útil, estimada entre 10 a 20 años. (EPMMOP, 2016)

A principios del 2017 se inició el "Plan de Obras Viales" el cual se centra en la rehabilitación del asfalto en los ejes principales de la ciudad, dividida en tres fases: la primera fase en la cual se ha invertido 3 millones de dólares aproximadamente para la reparación de los ejes viales, la segunda fase entre los años 2017-2018, con una inversión alrededor de 9 millones de dólares comprenderá la atención a 64 vías de

acceso a 89 barrios y la tercera fase realizará intervenciones preventivas en vías como: Interoceánica, Simón Bolívar, Ruta Viva, con el fin de ampliar su vida útil. (EPMMOP, 2017)

El proceso para la atención de los baches en la calzada comienza con la priorización de estas incidencias por parte de las administraciones zonales. En el caso de la Administración Zonal "Eloy Alfaro" (AZEA) lleva a cabo un proceso manual de registro y verificación. El cual inicia con la planificación y documentación del mapa de ruta diario, por parte del jefe de la Unidad de Informática, posterior a esto un moderador de AZEA recibe el documento a fin de recorrer el trayecto establecido, donde constata la existencia de baches o rellenos, llevando un registro en papel con fecha, calle principal, calle secundaria, parroquia y referencia; denominado matriz de mantenimiento rutinario, junto a esto se realiza la toma fotográfica como evidencia del hecho. Al terminar el recorrido el moderador, firma y hace entrega de la hoja de matriz de mantenimiento rutinario junto a los archivos fotográficos.

El jefe de la Unidad de Informática recibe estos documentos y verifica junto con el operador la correcta toma de datos además de las fotografías de cada incidente, mediante esto asigna la prioridad en función del nivel de riesgo vial e impacto a la ciudadanía, asimismo procede a digitalizar el documento haciendo uso de la herramienta Microsoft Excel.

Finalmente, el documento digitalizado es enviado a la EPMMOP para la elaboración de estudios técnicos, factibilidad y la ejecución de la obra, basado en la prioridad asignada a los incidentes.

En la figura No. 2 se refleja el proceso actual que se ejecuta por parte de AZEA.

Diagrama de proceso de reporte de baches Administración Zonal "Eloy Alfaro"

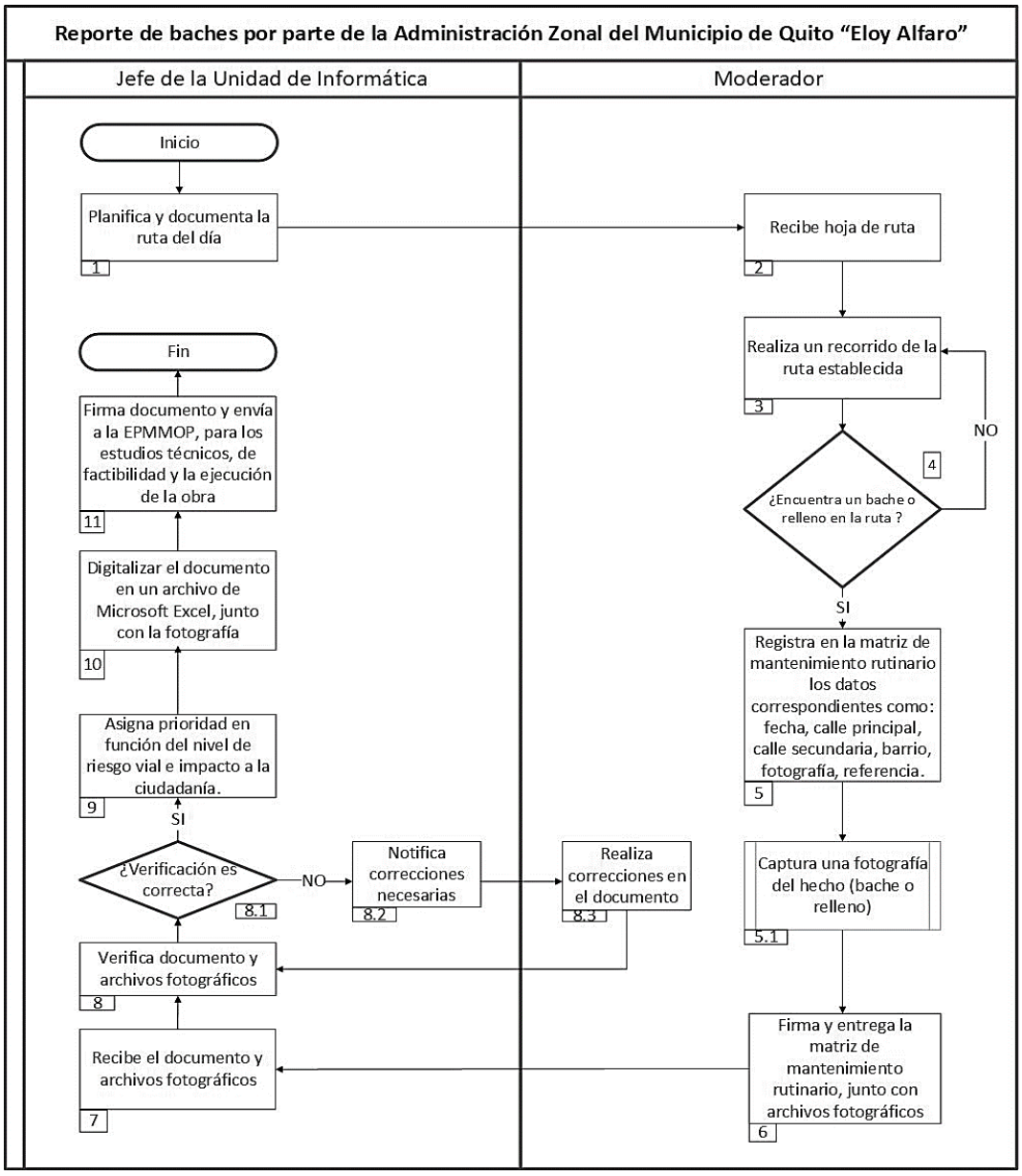

Figura 3. Diagrama del proceso de reporte de baches por parte de AZEA.
Elaborado por: Cristhian Castro y Paul Velasco

2.1.1.1 Formulación del problema

1. ¿Cómo afecta en la AZEA llevar un registro manual de baches?
2. ¿De qué modo influye a los funcionarios de AZEA el ciclo de trabajo manual?
3. ¿Qué impacto sobre el registro de baches tiene el uso ineficiente e inapropiado de las TICS?

Para AZEA el hecho de realizar un proceso manual causa retrasos en el registro, verificación y priorización de los baches pudiendo ocasionar que la acción de reparación de estas anomalías llevada a cabo por parte de la EPMOOP presente demoras en su ejecución. El proceso a causa de llevar actividades no automatizadas en su mayoría ocasiona que se torne en un trabajo engorroso para los moderadores y funcionarios de la Unidad de Informática ejecutando las mismas tareas en repetidas ocasiones y sobrecargándose de trabajo, esto provoca un uso no adecuado del tiempo debido a la demora que existe en la toma de datos de cada suceso.

Por otro lado, la falta de capacitación y el uso ineficiente de las Tecnologías de la Información y la Comunicación (TIC) por parte de los moderadores de AZEA durante el proceso de registro y validación de baches o rellenos en la ciudad de Quito, afecta el desempeño y la rapidez con la que se podría reportar estas incidencias considerando que la tecnología moderna aumenta y facilita las capacidades humanas para el cumplimiento de estas tareas. Para los funcionarios de la Unidad de Informática el no aprovechar al máximo las tecnologías digitales para la verificación, digitalización y priorización de estos incidentes provoca desperdicio de recursos y duplicación de documentos dado que los archivos de la "matriz de mantenimiento rutinario" se guardan en hojas de papel y reportes impresos de Microsoft Excel.

2.1.2 Constitución del Proyecto

En esta sección, se realizó el Acta de Constitución del Proyecto, junto al Ing. Jorge Cifuentes representante de la Administración Zonal "Eloy Alfaro", en la cual se definieron los objetivos y alcance del proyecto (Ver anexo 1).

2.1.3 Análisis de requerimientos del sistema

Para el análisis de los requerimientos del sistema, se ha hecho uso del estándar IEEE830 para la descripción de cada uno de los apartados siguientes.

2.1.3.1 Propósito

Este documento tiene como propósito concretar y evidenciar los requerimientos funcionales y no funcionales del sistema informático basado en aplicación móvil y geolocalización para el registro de baches en las vías de la ciudad. El mismo que será usado por los funcionarios del Municipio de Quito enfocándose en la Administración Zonal "Eloy Alfaro" (AZEA).

2.1.3.2 Alcance

El objetivo principal del presente trabajo es la automatización del registro y ubicación geográfica de los baches en las calles del Distrito Metropolitano de Quito (DMQ); para ello se desarrollará un sistema informático que sustituya el proceso manual llevado a cabo actualmente, brindando apoyo en las tareas de:

- Registro y verificación de baches.
- Priorización de incidencias.
- Generación de reportes.

El sistema informático se compone de dos partes principales, aplicación móvil y aplicación Web. La aplicación móvil permitirá a los moderadores del Municipio de Quito por medio de sus credenciales de usuario iniciar sesión y acceder a la aplicación para registrar los baches mediante la toma de una fotografía junto con la referencia del sitio, y de manera automática la obtención de la latitud, longitud, dirección y fecha del suceso así también brindará la posibilidad de hacer comprobaciones a los baches registrados una vez sean reparados. La aplicación Web facultará al administrador a través de sus credenciales ingresar al sistema y mantener un control de los registros y gestión por medio del estado actual del bache pudiendo ser: reportado, en curso y verificado, además se proporcionará la posibilidad de generar gráficas de reportes parametrizables por fecha, parroquia o estado de los sucesos.

Se implementará una prueba piloto en la AZEA en la cual se evaluarán los criterios de desempeño mediante pruebas de caja negra y criterios de usabilidad con la ayuda de encuestas a los operadores del sistema informático.

2.1.3.3 Personal involucrado

Tabla 1 Personal involucrado: Tutor del proyecto

Nombre	Ing. Rodrigo Tufiño
Rol	Tutor del proyecto
Categoría Profesional	Ingeniero de Sistemas
Responsabilidades	Coordinar y revisar tareas del proyecto.

- Nota: Esta tabla contiene información acerca del tutor del proyecto

Tabla 2 Personal involucrado: Representante de AZEA

Nombre	Ing. Jorge Cifuentes
Rol	Representante de AZEA
Categoría Profesional	Ingeniero de Sistemas
Responsabilidades	Definición y seguimiento de requerimientos del proyecto.

- Nota: Esta tabla contiene información acerca del representante de AZEA

Tabla 3 Personal involucrado: Desarrollador del sistema informático

Nombre	Paul Velasco
Rol	Desarrollador
Categoría Profesional	Estudiante Ingeniería de Sistemas
Responsabilidades	Diseño y desarrollo del proyecto

- Nota: Esta tabla contiene información acerca del desarrollador del proyecto

Tabla 4 Personal involucrado: Desarrollador del sistema informático

Nombre	Cristhian Castro
Rol	Desarrollador
Categoría Profesional	Estudiante Ingeniería de Sistemas
Responsabilidades	Diseño y desarrollo del proyecto.

- Nota: Esta tabla contiene información acerca del desarrollador del proyecto

2.1.3.4 Definiciones, siglas y abreviaturas

En esta sección se definirán todos los términos y abreviaturas usadas a lo largo del documento.

- **Del negocio**

Tabla 5 Cuadro de definiciones: Terminología del negocio

DMQ	Distrito Metropolitano de Quito.
AZEA	Administración Zonal Eloy Alfaro.
Moderador	Funcionario del Municipio de Quito cuya función es la de buscar y registrar baches en la ciudad.

Bache	Pequeño desnivel en el suelo o en el pavimento, producido por la pérdida o hundimiento de la capa superficial. (Real Academia Española, 2017)
Incidencia/incidente	Referente a los baches en las calles.

- Nota: Esta tabla contiene información sobre terminología del negocio.

- **Del sistema**

Tabla 6 Cuadro de definiciones: Terminología del sistema

Moderador/usuario	Persona que va a hacer uso del sistema informático, utilizando sus credenciales (usuario, contraseña).
Administrador	Persona que mediante su nombre de usuario y contraseña podrá acceder a la aplicación Web con permisos privilegiados.
RF	Funcionario del Municipio de Quito cuya función es la de buscar y registrar baches en la ciudad. Requisito Funcional
RNF	Requisito No Funcional
HU	Historia de usuario

- Nota: Esta tabla contiene información sobre terminología del sistema.

- **De tecnología**

Tabla 7 Cuadro de definiciones: Terminología de tecnología

BDD	Base de datos
Aplicación	Programa informático cuya función es la de apoyar al usuario en la realización de tareas específicas.
HTML	Lenguaje de marcado de hipertexto, permite la construcción de páginas Web.
CSS	Hojas de estilo en cascada, lenguaje de diseño gráfico utilizado para dar estilos a elementos de HTML.
Responsive	Filosofía de diseño Web, la cual se refiere a la adaptación de las interfaces de las aplicaciones independientemente del dispositivo que las ejecute.
GPS	Sistema de posicionamiento global, desarrollado con el objetivo de determinar la posición de un objeto dentro de la Tierra.

- Nota: Esta tabla contiene información sobre terminología de tecnología

2.1.3.5 Referencias

IEEE (1998). Standard IEEE 830.

2.1.4 Descripción General

Este apartado del documento será empleado para la descripción del sistema de manera global, sin detalle en los requisitos específicos d el mismo.

2.1.4.1 Perspectiva del producto

El sistema informático estará conformado por dos partes, la aplicación móvil que será diseñada para trabajar bajo el sistema operativo Android y la aplicación Web diseñada para que se ejecute en un navegador Web, las cuales trabajarán de manera conjunta para el registro, ubicación, control, y administración de los baches en las vías, además funcionará de manera autónoma sin la necesidad de interactuar con otros sistemas.

2.1.4.2 Funciones del producto

El sistema informático permitirá realizar las siguientes funciones:

- **Administración de usuarios**

Este módulo permitirá al administrador crear nuevos usuarios, que accederán al sistema informático, así mismo podrá listar los moderadores registrados en el sistema y actualizar sus datos personales, además del cambio de estado y con esto habilitar o deshabilitar al moderador.

Casos de uso módulo de administración de usuarios.

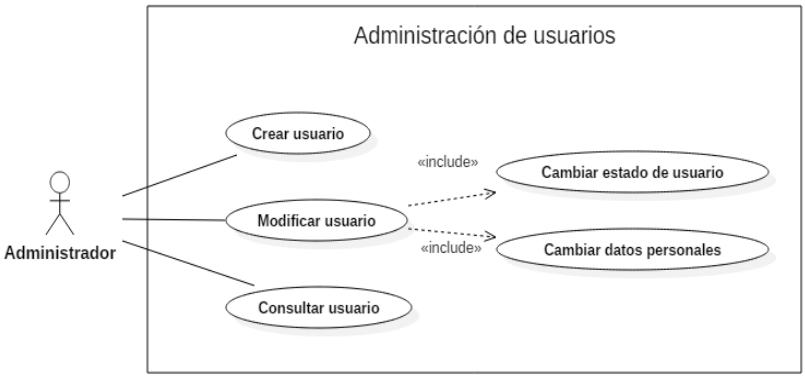

Figura 4. Diagrama de casos de uso módulo administración de usuarios.
Elaborado por: Cristhian Castro y Paul Velasco

- **Autenticación de usuarios**

Este módulo tiene como finalidad manejar el inicio de sesión dentro de las aplicaciones del sistema informático, para esto se validará las credenciales del usuario correspondiente (administrador, moderador).

Casos de uso módulo autenticación de usuarios.

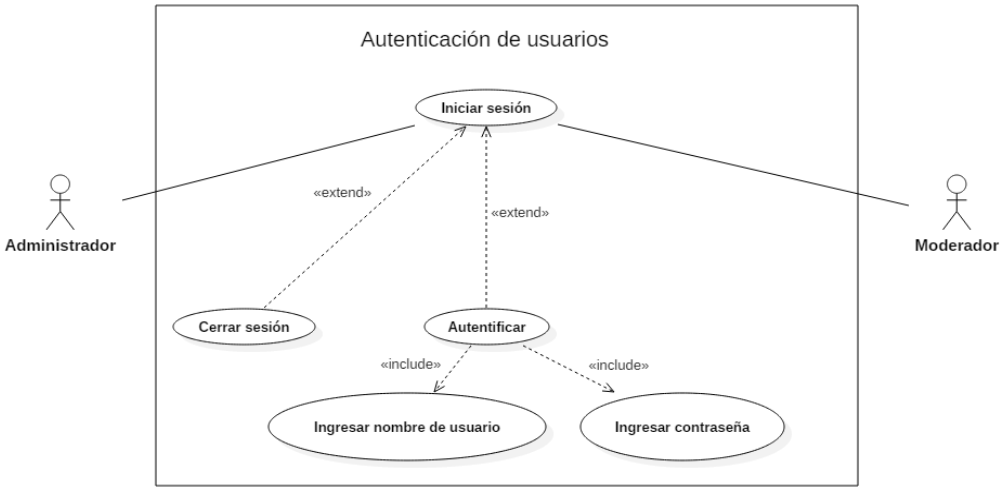

Figura 5. Diagrama de casos de uso módulo autenticación de usuarios.
Elaborado por: Cristhian Castro y Paul Velasco

- **Registro y validación**

El moderador podrá registrar los baches con el uso de la aplicación móvil en donde hará la captura de una fotografía, e ingreso de la referencia al sitio del incidente cabe destacar que la calle principal y secundaria, serán calculadas de forma automática a partir de la latitud y longitud obtenida por el GPS del dispositivo móvil, a esto se debe agregar automáticamente la fecha y moderador responsable del registro.

Casos de uso módulo registro y validación.

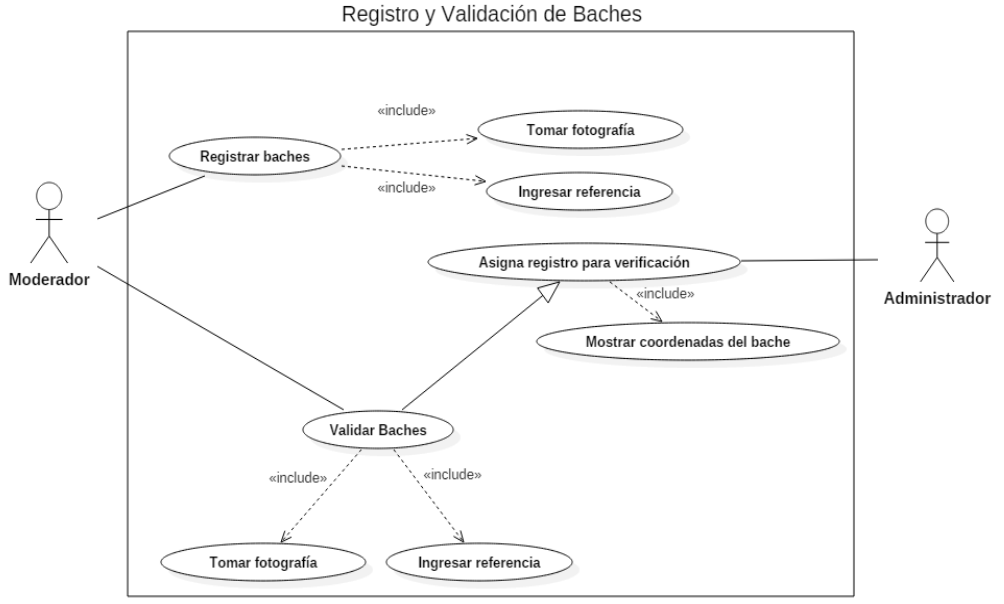

Figura 6. Diagrama de casos de uso módulo de registro y validación de baches.
Elaborado por: Cristhian Castro y Paul Velasco

- **Administración de registros:**

El administrador tendrá acceso a la ubicación exacta e información de los baches registrados por los moderadores mediante un mapa, en el cual podrá gestionar el estado actual del incidente (reportado, en curso y verificado), así mismo asignará registros de baches que estén en estado "en curso" a los moderadores para su verificación, finalmente obtendrá gráficas de reportes parametrizables por fecha de registro, parroquia o estado de las incidencias.

Casos de uso módulo administración de registros.

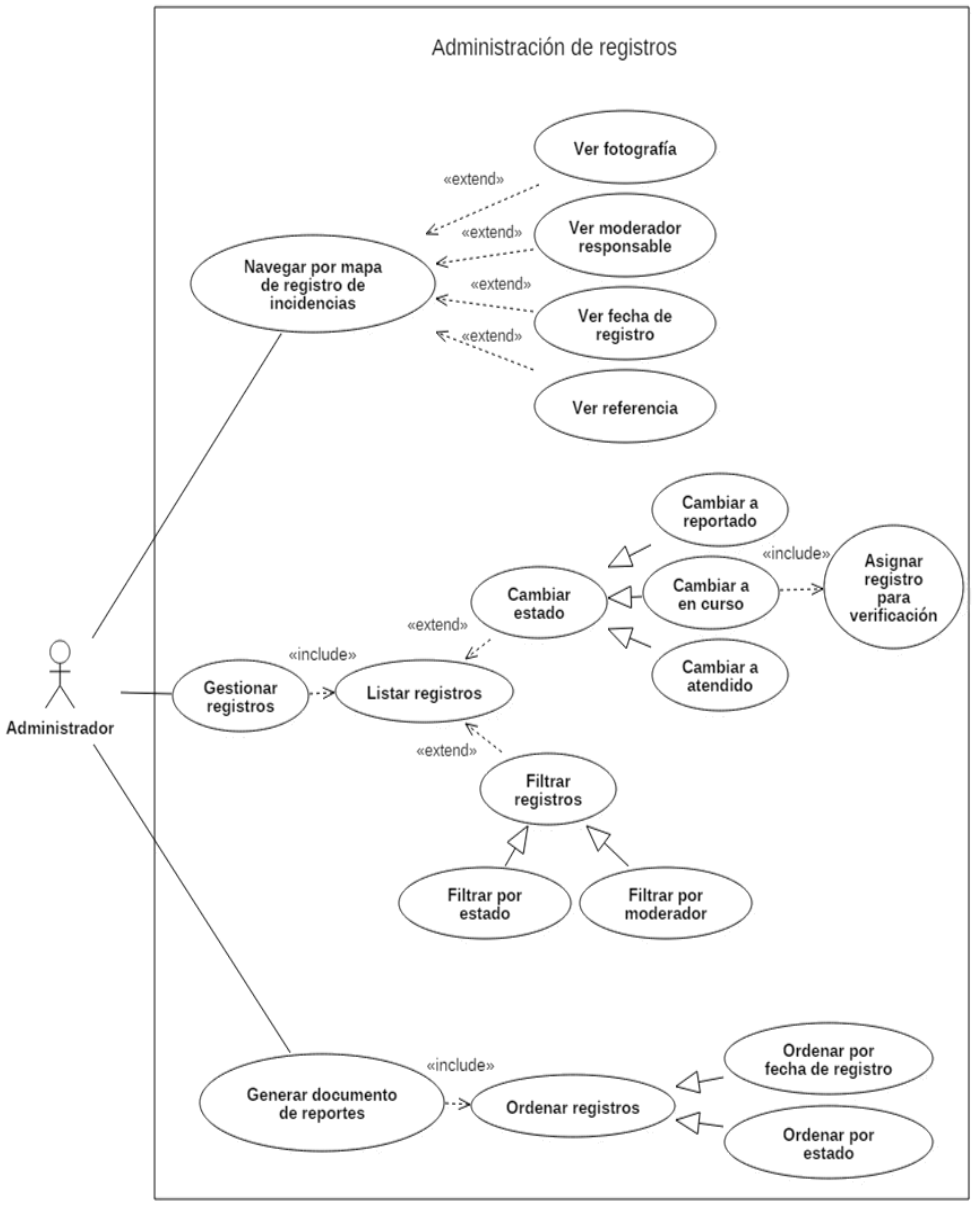

Figura 7. Diagrama de casos de uso módulo administración de registros.GG
Elaborado por: Cristhian Castro y Paul Velasco

2.1.4.3 Características del usuario

El sistema informático interactuará con dos tipos de usuario: moderador, administrador.

- **Moderador:** Persona que hará uso de la aplicación móvil, su formación en tecnología es baja, únicamente debe conocer de manera básica el uso de un smartphone, o tablet.
- **Administrador:** Usuario que empleará la aplicación Web, su formación en tecnología es alta, además debe poseer el conocimiento de cómo funciona el sistema informático en totalidad.

2.1.5 Restricciones

Aquí se incluye ciertas restricciones o limitaciones que se imponen para el funcionamiento del sistema informático.

2.1.5.1 Limitaciones de hardware y software

Debido a que el sistema informático consta de una aplicación móvil esta deberá ser ejecutada en smartphones, tablets o dispositivos con sistema operativo Android en su versión mínima 4.4 (KitKat) o superior.

2.1.5.2 Consideraciones de seguridad

Cada uno de los usuarios previo al manejo de algunos de los componentes del sistema informático conforme el rol asignado, deberá autenticarse.

2.1.6 Suposiciones y Dependencias

El uso del sistema informático supone contar con conexión a internet y la instalación de un navegador Web.

2.1.7 Requisitos Específicos

A continuación, se listan los requisitos del sistema tomados del acta de definición de requerimientos producto de la reunión con el Ing. Jorge Cifuentes representante de la

Administración Zonal Eloy Alfaro, lo cual permita a los desarrolladores entender el funcionamiento o características deseables del sistema informático.

Para esta parte del documento se ha precisado en que componente del sistema será acondicionado el requerimiento (Ver anexo 2).

2.1.7.1 Requisitos Funcionales

El sistema informático tanto su aplicación Web y móvil permitirán el uso de las distintas funcionalidades enmarcadas en los siguientes módulos:

2.1.7.1.1 Administración de usuarios

Tabla 8 Requisito Funcional 1.1: Creación de usuario en la aplicación Web.

Aplicación-Web **SRS-Especificación de Requisitos Funcionales**			
Código	**Nombre**		
RF 1.1	Creación de usuario		
Descripción	El administrador del sistema debe ingresar la información del nuevo usuario.		
Entradas	**Fuente**	**Salidas**	**Restricciones**
Datos obligatorios: - Nombres - Apellidos - CI - Contraseña - Estado **Datos opcionales:** - Dirección - Teléfono - e-mail	Formulario de ingreso de datos	- Confirmación de usuario registrado - Datos ingresados y nombre de usuario asignado - Mensajes de error	Esta acción solo podrá ser realizada por el usuario administrador.
Proceso	La aplicación mostrará el formulario para la creación de nuevos usuarios, donde el administrador completará los campos descritos en las "entradas", una vez presionada la opción guardar la aplicación validará los datos proporcionados en caso de ser correcta se almacenará en la BDD y mostrará en pantalla la información registrada, caso contrario notificará los mensajes de error. Antes de almacenar la información en la BDD la aplicación presenta al usuario administrador una pantalla con la confirmación de los datos ingresados.		
Prerrequisito	Ninguno	**Prioridad**	Alta

- Nota: Esta tabla contiene información sobre la creación de un usuario, con el uso de sus datos personales

Tabla 9 Requisito Funcional 1.2: Modificar información del usuario.

Aplicación-Web **SRS-Especificación de Requisitos Funcionales**			
Código	**Nombre**		
RF 1.2	Modificación de usuario		
Descripción	El administrador del sistema podrá actualizar la información del usuario.		
Entradas	**Fuente**	**Salidas**	**Restricciones**
- Nombres y apellidos - Contraseña - Dirección - Teléfono - e-mail - Estado	Formulario de modificación de datos	-Confirmación de cambios registrado -Datos actualizados -Mensajes de error	Esta acción solo podrá ser realizada por el usuario administrador.
Proceso	La aplicación mostrará el formulario para la actualización de datos del usuario, donde el administrador modificará los campos descritos en las "entradas", la contraseña se restablecerá por defecto al número de cédula de identidad del usuario, una vez presionada la opción guardar la aplicación validará los datos proporcionados en caso de ser correcta se almacenará en la BDD y mostrará en pantalla la información registrada, caso contrario notificará los mensajes de error. Antes de almacenar la información en la BDD la aplicación presenta al usuario administrador una pantalla con la confirmación de los datos ingresados.		
Prerrequisito	RF 1.1	**Prioridad**	Alta

- Nota: Esta tabla contiene información para actualizar la información del usuario

Tabla 10 Requisito Funcional 1.3: Búsqueda filtrada del usuario

Aplicación-Web **SRS-Especificación de Requisitos Funcionales**			
Código	**Nombre**		
RF 1.3	Búsqueda filtrada de usuario		
Descripción	El administrador del sistema podrá buscar y listar usuarios almacenados en el sistema		
Entradas	**Fuente**	**Salidas**	**Restricciones**
- Nombres y apellidos - Estado - Nombre de usuario - Cédula de identidad	Criterio de búsqueda	- Listado de usuario/s encontrados	El usuario o usuarios deben estar registrados en el sistema
Proceso	El administrador podrá seleccionar el criterio establecido en las "entradas" y llenar el campo de búsqueda para filtrar el listado de usuarios. Si el campo se encuentra vacío arrojará la lista completa de usuarios.		
Prerrequisito	RF 1.1	**Prioridad**	Baja

- Nota: Esta tabla contiene información para buscar los usuarios en la aplicación en base a criterios de búsqueda

2.1.7.1.2 Autenticación de usuarios

Tabla 11 Requisito Funcional 2.1: Inicio de sesión en la aplicación Web

Aplicación-Web, Móvil **SRS-Especificación de Requisitos Funcionales**			
Código	**Nombre**		
RF 2.1	Inicio de sesión		
Descripción	La aplicación permitirá al usuario iniciar sesión por medio de sus credenciales		
Entradas	**Fuente**	**Salidas**	**Restricciones**
- Nombre de usuario - Contraseña	Formulario de modificación de datos	- Interfaz de usuario acorde al perfil - Mensajes de error	Solo podrá ser realizada por usuarios registrados en el sistema
Proceso	La aplicación mostrará el formulario de inicio de sesión en el cual el operador ingresará sus credenciales (nombre de usuario, contraseña), seguido a esto la aplicación verificará la existencia del usuario y el ingreso correcto de la contraseña, de ser verdadero se cargará la interfaz de usuario correspondiente al perfil, caso contrario indicará un mensaje de error de credenciales.		
Prerrequisito	RF 1.1	**Prioridad**	Alta

- Nota: Esta tabla contiene información para inicio de sesión de usuarios en la aplicación Web

2.1.7.1.3 Administración de Registros

Tabla 12 Requisito Funcional 3.1: Estado de registros en la aplicación Web

Aplicación-Web **SRS-Especificación de Requisitos Funcionales**			
Código	**Nombre**		
RF 3.1	Estado de registros		
Descripción	La aplicación permitirá al administrador cambiar el estado (reportado, en curso, verificado) de los registros		
Entradas	**Fuente**	**Salidas**	**Restricciones**
- Puntos de geolocalización -Estado	Formulario de estado de registros	- Cambio de estado y color de los registros	Esta acción solo podrá ser realizada por el usuario administrador
Proceso	La aplicación mostrará el mapa de la ciudad de Quito con los puntos de geolocalización ingresados a través de la aplicación móvil, estos puntos estarán disponibles en el panel de incidentes donde el administrador podrá cambiar su estado (reportado – en curso - verificado).		
Prerrequisito	RF 4.1	**Prioridad**	Alta

- Nota: Esta tabla contiene información para el cambio de estado de un registro (reportado, en curso y validado)

Tabla 13 Requisito Funcional 3.2: Mostrar listado de registros

Aplicación-Web **SRS-Especificación de Requisitos Funcionales**

Código	**Nombre**		
RF 3.2	Mostrar listado de registros		
Descripción	La aplicación permitirá al administrador obtener un inventario de los registros de baches.		
Entradas	**Fuente**	**Salidas**	**Restricciones**
- Puntos de geolocalización - Estado - Nombre del moderador - Fecha - Parroquia	Formulario de filtrado	- Matriz de registros.	Esta acción solo podrá ser realizada por el usuario administrador
Proceso	La aplicación mostrará una matriz con los registros de baches, el cual podrá ser filtrado por alguno de los criterios detallados en las "entradas".		
Prerrequisito	RF 3.1 – RF 4.1	**Prioridad**	Media

- Nota: Esta tabla contiene información para mostrar un listado de los registros

Tabla 14 Requisito Funcional 3.3: Asignación de tareas de validación

Aplicación-Web **SRS-Especificación de Requisitos Funcionales**				
Código	**Nombre**			
RF 3.3	Asignación tareas de validación			
Descripción	La aplicación permitirá al administrador asignar a un moderador los registros en estado "en curso" para su verificación			
Entradas	**Fuente**	**Salidas**	**Restricciones**	
- Puntos de geolocalización - Estado - Nombre del moderador	Formulario de asignación de validaciones	- Confirmación por pantalla de asignación.	Esta acción solo podrá ser realizada por el usuario administrador Solo es posible asignar puntos en estado "en curso"	
Proceso	La aplicación mostrará un listado con los registros en estado "en curso", junto a esto se ubicará un campo de búsqueda dinámica para asignar la tarea de validación a un moderador. Una vez presionado el botón enviar se almacenará en el perfil del moderador los registros establecidos para su verificación.			
Prerrequisito	RF 3.1 – RF 4.1 – RF 3.2		**Prioridad**	Alta

- Nota: Esta tabla contiene información para cambiar el estado de un registro de "reportado" a "en curso"

Tabla 15 Requisito Funcional 3.4: Generar reportes en la aplicación Web

Aplicación-Web **SRS-Especificación de Requisitos Funcionales**			
Código	**Nombre**		
RF 3.4	Generar Reportes		
Descripción	La aplicación permitirá al administrador crear informes de todos los baches en sus tres diferentes estados (reportado, en curso y solucionado), siendo parametrizable por parroquia o fecha de registro.		
Entradas	**Fuente**	**Salidas**	**Restricciones**

- Puntos de geolocalización - Estado - Nombre del moderador - Fecha de registro	Formulario generación de reportes	- Matriz de mantenimiento rutinario filtrada.	Esta acción solo podrá ser realizada por el usuario administrador
Proceso	La aplicación mostrará un listado con los registros en los tres diferentes estados (reportado, en curso y verificado), donde el administrador podrá ordenar los registros por fecha o parroquia, con ello el administrador obtendrá informes sobre la cantidad de baches en una parroquia o en un tiempo establecido.		
Prerrequisito	RF 3.2-RF 3.3-RF 4.1-RF 4.2	**Prioridad**	Alta

- Nota: Esta tabla contiene información para generar reportes de los registros.

2.1.7.1.4 Reporte y validación de baches

Tabla 16 Requisito Funcional 4.1: Registro de baches desde la aplicación móvil

Aplicación-Móvil **SRS-Especificación de Requisitos Funcionales**			
Código	**Nombre**		
RF 4.1	Registro de baches		
Descripción	La aplicación permitirá al moderador de AZEA realizar registros de baches en las vías.		
Entradas	**Fuente**	**Salidas**	**Restricciones**
Datos automáticos - Fecha - Datos del moderador - Coordenadas geográficas - Calle principal **Datos manuales** - Fotografía - Referencia	Formulario de registro de bache	- Mensaje de registro correcto - Alertas de error	No es posible realizar registros sin una fotografía.
Proceso	La aplicación mostrará el formulario de registro de baches donde el moderador ingresará los datos manuales y junto a los automáticos especificados en las "entradas" se almacenará en la base de datos. De lo contrario se indicará mensajes de error.		
Prerrequisito	RF 2.1	**Prioridad**	Alta

- Nota: Esta tabla contiene información para registrar baches utilizando la aplicación móvil

Tabla 17 Requisito Funcional 4.2: Validación de baches en la aplicación móvil

Aplicación-Móvil **SRS-Especificación de Requisitos Funcionales**			
Código	**Nombre**		
RF 4.2	Validación de baches		
Descripción	La aplicación permitirá al moderador de AZEA realizar validaciones de baches en las vías, una vez que hayan sido rellenados.		
Entradas	**Fuente**	**Salidas**	**Restricciones**

Datos automáticos - Fecha - Datos del moderador (usuario) - Ubicación del bache a solucionar (latitud y longitud) - Calle principal **Datos manuales** - Fotografía - Referencia al sitio del suceso	Formulario de validación de baches	- Mensaje de validación correcta - Alertas de error	Para realizar una validación el incidente debe ser asignado por el administrador. No es posible realizar validaciones sin una fotografía.
Proceso	La aplicación mostrará una lista de incidencias asignadas por el administrador para su verificación, una vez que el moderador seleccione una de ellas la aplicación mostrará el formulario de validación de baches reparados donde se anexará los datos manuales junto a los automáticos especificados en las "entradas" se almacenará en la BDD. En caso de no completar correctamente los campos requeridos se notificará al moderador mediante alertas de error.		
Prerrequisito	RF 2.1 – RF 3.3	**Prioridad**	Alta

- Nota: Esta tabla contiene información para validar baches por parte del usuario moderador utilizando la aplicación móvil

2.1.7.2 Requisitos No Funcionales

El sistema informático tanto su aplicación Web como en su aplicación móvil contarán con diversas cualidades y características específicas, las cuales han sido listadas a continuación:

2.1.7.2.1 Disponibilidad

Tabla 18 Requisito No Funcional 1.1: Disponibilidad del sistema

Aplicación-Web, Móvil **SRS-Especificación de Requisitos No Funcionales**			
Código	**Nombre**		
RNF 1.1	Disponibilidad del sistema		
Descripción	El sistema informático debe brindar disponibilidad completa.		
Entradas	**Fuente**	**Salidas**	**Restricciones**
- Arquitectura del sistema	No aplica	No aplica	Se ve influenciado por factores externos al sistema
Proceso	El sistema informático, debe asegurar que sus aplicaciones estarán disponibles para su uso continuo a menos que se vea afectado por factores externos fuera del alcance.		
Prerrequisito	Ninguno	**Prioridad**	Alta

- Nota: Esta tabla contiene información acerca de la disponibilidad del sistema

2.1.7.2.2 Usabilidad

Tabla 19 Requisito No Funcional 2.1: Interfaces gráficas

Aplicación-Web, Móvil **SRS-Especificación de Requisitos No Funcionales**			
Código	**Nombre**		
RNF 2.1	Interfaces gráficas		
Descripción	El sistema informático debe contar con un diseño de interfaces minimalistas e intuitivas.		
Entradas	**Fuente**	**Salidas**	
- Formularios - Tipografía - Botones, iconos	CSS	Plantillas HTML, que componen la interfaz gráfica.	
Proceso	El sistema informático estará desarrollado con interfaces limpias, dinámicas y diseñadas para que al usuario le sea fácil de utilizar, además con el afán de conseguir que sienta atracción de usar las aplicaciones.		
Prerrequisito	Ninguno	**Prioridad**	Alta

- Nota: Esta tabla contiene información acerca de las interfaces gráficas.

Tabla 20 Requisito No Funcional 2.2: Diseño responsivo

Aplicación-Web, Móvil **SRS-Especificación de Requisitos No Funcionales**			
Código	**Nombre**		
RNF 2.2	Diseño responsivo		
Descripción	El sistema informático garantizará la adecuada visualización de las interfaces de usuario.		
Entradas	**Fuente**	**Salidas**	
- Interfaz de usuario	CSS	Interfaz adaptable al dispositivo.	
Proceso	Debe poseer un diseño “responsive” o adaptativo, con el fin de que las aplicaciones se ajusten a los distintos dispositivos que puedan ejecutarlas (computadora, smartphone, tablet,)		
Prerrequisito	Ninguno	**Prioridad**	Alta

- Nota: Esta tabla contiene información acerca del diseño responsive en el sistema

2.1.7.2.3 Seguridad

Tabla 21 Requisito No Funcional 3.1: Seguridad en contraseñas

Aplicación-Web **SRS-Especificación de Requisitos No Funcionales**			
Código	**Nombre**		
RNF 3.1	Seguridad en contraseñas		
Descripción	El sistema encriptadas las contraseñas asignadas a los usuarios.		
Entradas	**Fuente**	**Salidas**	
- Contraseña	Formulario de registro	Contraseña encriptada	
Proceso	En el proceso de creación de usuarios, el sistema deberá encriptar la contraseña antes de ser almacenada en la BDD, para con esto precautelar la seguridad.		
Prerrequisito	Ninguno	**Prioridad**	Alta

- Nota: Esta tabla contiene información acerca de la seguridad en el sistema

2.1.7.2.4 Portabilidad

Tabla 22 Requisito No Funcional 4.1: Portabilidad

Aplicación-Web **SRS-Especificación de Requisitos No Funcionales**			
Código	**Nombre**		
RNF 4.1	Sistema Operativo		
Descripción	La aplicación podrá ser usada en distintos sistemas operativos.		
Entradas	**Fuente**	**Salidas**	
- Arquitectura	Sistema Operativo	No aplica	
Proceso	La ejecución de la aplicación no dependerá del sistema operativo en el que se ejecute el navegador Web.		
Prerrequisito	Ninguno	**Prioridad**	Alta

- Nota: Esta tabla contiene información acerca de la portabilidad en el sistema.

2.2 Diseño

En este capítulo, se muestra el diseño de la aplicación y la arquitectura del sistema informático a través de los diagramas de base de datos, clases y componentes.

2.2.1 Diagrama de base de datos

Se ha establecido la necesidad de que el sistema informático cuente con dos bases de datos, la BDD principal del sistema será con la que interaccione con la aplicación Web, en esta se almacenará toda la información concerniente a los usuarios, validación, registro de baches (fotografías, puntos de localización, información adicional) la data será generada por la aplicación móvil. Además, se contará con una BDD local para el dispositivo móvil con el objetivo de mantener en ella la información temporalmente en el caso de que no se cuente con una conexión activa a internet.

2.2.1.1 Diagrama de base de datos aplicación Web

La figura No. 7 refleja la organización de tablas y relaciones de la base de datos que en un principio será manejada por el sistema informático esta puede estar sujeta a cambios propios de la fase de desarrollo del sistema, se puede apreciar que el diagrama cuenta con las tablas de usuario, administración, perfil, reporte, estado, parroquia, validación, fotografía, las cuales han sido definidas de la siguiente manera.

- **Administración:** Para almacenar las distintas administraciones zonales del Municipio.
- **Perfil:** Se registrarán los diferentes perfiles de usuario que el sistema permite.
- **Usuario:** Contendrá los datos personales, credenciales de los usuarios (moderador, administrador) que trabajan con el sistema, por lo que mantiene relación con el perfil asignado a cada uno de ellos, y la administración zonal a la cual pertenece.
- **Estado:** Se guarda los distintos estados (reportado, en curso, verificado) en los que un bache se encuentra.
- **Fotografía:** En ella se registra la ruta en el servidor donde se deposita la fotografía del reporte de un bache.
- **Parroquia:** Tabla en la que se cargarán por defecto las parroquias asociadas a cada administración zonal del Municipio.
- **Reporte:** Es la tabla principal dentro del diagrama, su función es la de contener los registros de cada reporte de un bache geolocalizado en la ciudad por medio de los campos latitud y longitud, tiene dependencia con la tabla usuario para la asignación de un moderador responsable del reporte, junto con el correspondiente parroquia del bache reportado contenido en la tabla "parroquia", así mismo posee relación con el estado en el que se encuentra el bache, y posee una única fotografía afín al incidente.
- **Validación:** Se registrarán las tareas de validación de baches asignados por el administrador a cada uno de los moderadores. Es por ello se encuentra ligada a la tabla "usuario" con la finalidad de inscribir el moderador responsable de la verificación del incidente y el registro designado de la tabla "reporte", de igual modo estará sujeto a la existencia de una fotografía que avale el cumplimiento de la tarea.

Modelo conceptual base de datos aplicación Web.

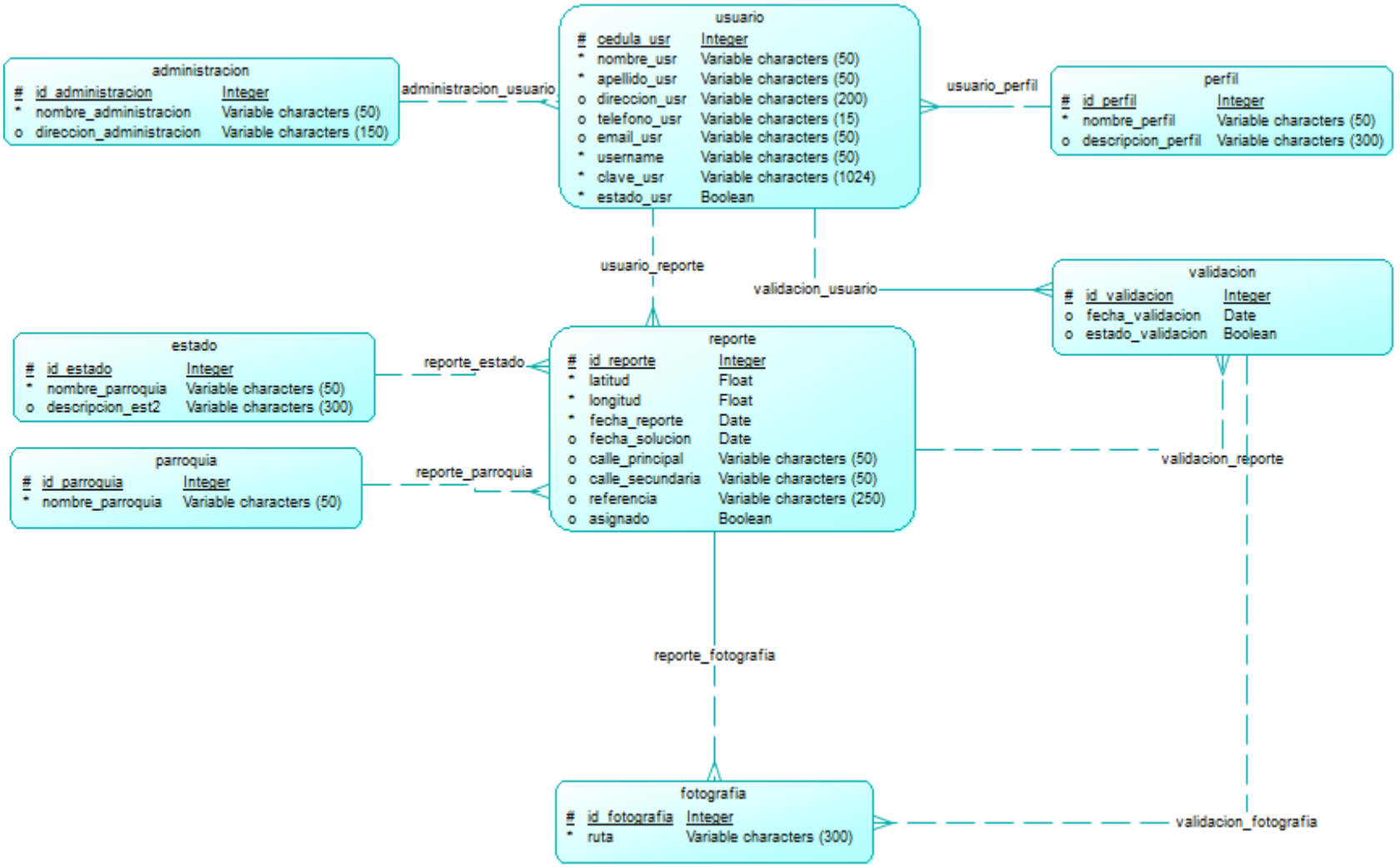

Figura 8. Diagrama conceptual de base de datos de la aplicación Web
Elaborado por: Cristhian Castro y Paul Velasco

2.2.1.2 Diagrama de base de datos aplicación móvil

Por medio de la figura No. 8, se manifiesta el esquema de base de datos con la cual interactúa la aplicación móvil, tomando en cuenta que esta BDD tiene el sentido de preservar datos temporales, se usa la abreviación “temp” junto con el nombre de cada una de las tablas. Al igual que la base de datos de la aplicación Web hay que mencionar que este diagrama es tentativo para el desarrollo mas no el final. El criterio para la creación de las tablas es el siguiente:

- **Usuario_temp:** Tabla creada con el fin de servir de apoyo para el almacenamiento de los datos del moderador que inicia sesión en la aplicación móvil.
- **Reporte_temp:** Guarda la información momentánea del reporte de un bache.
- **Fotografía_temp:** Archiva transitoriamente la ruta de las fotografías.
- **Tareas_asignada:** En ella se cargarán los reportes que fueron fijados al moderador para su verificación.
- **Validación_temp:** El momento de realizar una validación y no contar con conexión a internet, en esta tabla serán recopilados los datos capturados, para su envío una vez tener una conexión activa.

Modelo conceptual base de datos aplicación móvil.

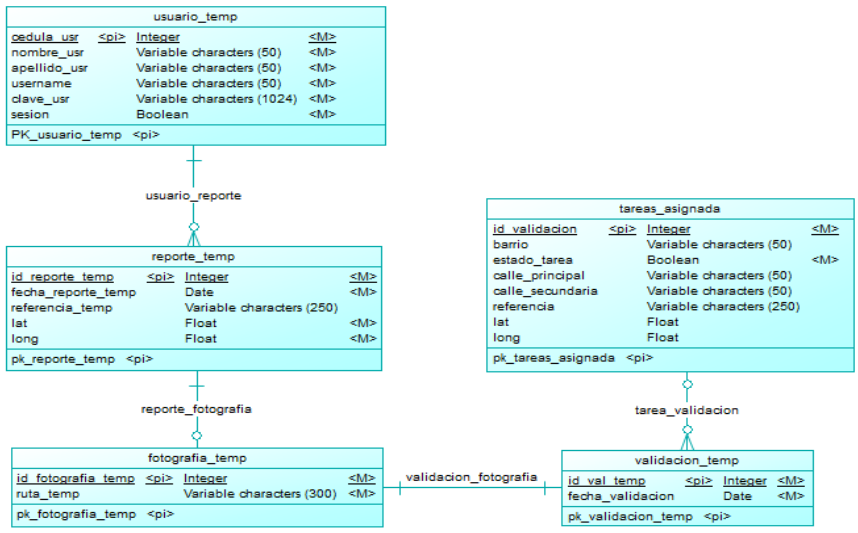

Figura 9. Diagrama conceptual de base de datos de la aplicación móvil
Elaborado por: Cristhian Castro y Paul Velasco

2.2.2 Diagrama de clases

A continuación se muestran los diagramas de clases de la aplicación Web y de la aplicación móvil para el sistema informático donde se define la estructura del sistema a partir de sus clases, atributos y relaciones, en consecuencia cada clase muestra la información de usuarios, perfiles, parroquias de la ciudad de Quito, administraciones zonales, reportes y validaciones las cuales permiten la interacción entre todos los objetos del sistema informático representando las clases que finalmente serán programadas.

2.2.2.1 Diagrama de clases de la aplicación Web

En la figura No. 9 se presenta el diagrama que muestra las clases necesarias para el desarrollo de la aplicación Web, tomando en cuenta las funcionalidades que cada clase va a tener acorde a las necesidades del sistema informático.

Diagrama de clases aplicación Web.

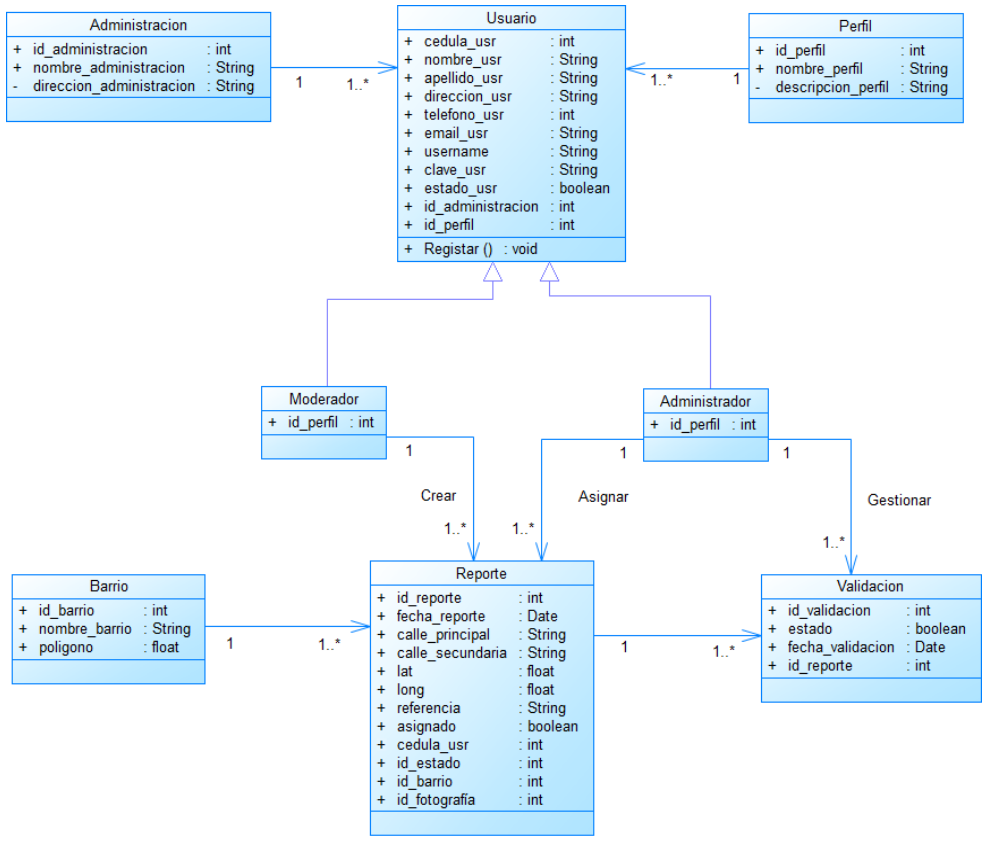

Figura 10. Diagrama de clases aplicación Web.
Elaborado por: Cristhian Castro y Paul Velasco

A continuación, se describe la función de cada clase perteneciente a la aplicación Web:

Tabla 23 Diagrama de Clase No 1: Administración

CLASE No 1:	**Administración**	
Descripción	Contiene las diferentes administraciones zonales existentes en la ciudad de Quito.	
Atributo	**Tipo**	**Descripción**
id_administracion	int	Identificador de la administración zonal
nombre_administracion	String	Nombre de la administración zonal
direccion_administracion	String	Dirección de la administración zonal

- Nota: Esta tabla contiene información acerca de los atributos en la clase Administración.

Tabla 24 Diagrama de Clase No 2: Perfil

CLASE No 2:	**Perfil**	
Descripción	Almacena información correspondiente al perfil del usuario (administrador o moderador)	
Atributo	**Tipo**	**Descripción**
id_perfil	int	Identificador del tipo de perfil
nombre_perfil	String	Nombre del perfil
descripcion_perfil	String	Descripción de las funciones del perfil

- Nota: Esta tabla contiene información acerca de los atributos en la clase Perfil.

Tabla 25 Diagrama de Clase No 3: Usuario

CLASE No 3:	**Usuario**	
Descripción	Incluye la información, datos personales, nombre de usuario y contraseña del personal que usará el sistema informático ya sea este un administrador o un moderador	
Atributo	**Tipo**	**Descripción**
cedula_usr	int	Cédula del usuario
nombre_usr	String	Nombre del usuario
apellido_usr	String	Apellido del usuario
direccion_usr	String	Dirección domiciliaria del usuario
telefono_usr	int	Número telefónico del usuario
email_usr	String	Correo electrónico del usuario
username	String	Nombre de usuario para acceder al sistema
clave_usr	String	Clave de usuario para acceder al sistema
estado_usr	boolean	Estado de usuario (Activo, inactivo)
id_administracion	int	Identificador de administración
id_perfil	int	Identificador de perfil
Operaciones	**Función**	**Descripción**
Registrar ()	void	Permite almacenar el usuario en el sistema

- Nota: Esta tabla contiene información acerca de los atributos en la clase Usuario.

Tabla 26 Diagrama de Clase No 4: Administrador

CLASE No 4:	**Administrador**	
Descripción	Abarca al usuario de con el perfil de administrador, heredando los atributos de la clase usuario.	
Atributo	**Tipo**	**Descripción**
id_perfil	int	Identificador del usuario con perfil de administrador

- Nota: Esta tabla contiene información acerca de los atributos en la clase Administrador.

Tabla 27 Diagrama de Clase No 5: Moderador

CLASE No 5:	**Moderador**	
Descripción	Abarca usuario de con el perfil de moderador, heredando los atributos de la clase usuario.	
Atributo	**Tipo**	**Descripción**
id_perfil	int	Identificador del usuario con perfil de moderador

- Nota: Esta tabla contiene información acerca de los atributos en la clase Moderador.

Tabla 28 Diagrama de Clase No 6: Parroquia

CLASE No 6:	**Parroquia**	
Descripción	Engloba las parroquias en la ciudad de Quito.	
Atributo	**Tipo**	**Descripción**
id_parroquia	int	Identificador de la parroquia de Quito
nombre_parroquia	String	Nombre de la parroquia de Quito

- Nota: Esta tabla contiene información acerca de los atributos en la clase Parroquia.

Tabla 29 Diagrama de Clase No 7: Reporte

CLASE No 7:	**Reporte**	
Descripción	Contiene información correspondiente a los baches en la ciudad como fotografía, fecha del reporte, coordenadas, referencia, calle principal y calle secundaria, junto con la parroquia en la cual se encuentra el bache	
Atributo	**Tipo**	**Descripción**
id_reporte	int	Identificador del reporte
fecha_reporte	Date	Fecha en el que se realizó el reporte
calle_principal	String	Calle principal del reporte
calle_secundaria	String	Calle secundaria del reporte
lat	float	Coordenadas de latitud del reporte
long	float	Coordenadas de longitud del reporte
referencia	String	Referencia o comentario acerca del reporte
asignado	boolean	Asignación de validación a un moderador
cedula_usr	Int	Cédula del moderador
id_estado	int	Identificador de estado del bache (reportado, en curso, validado)
id_parroquia	int	Identificador de parroquia
id_fotografia	int	Identificador de fotografía

- Nota: Esta tabla contiene información acerca de los atributos en la clase Reporte.

Tabla 30 Diagrama de Clase No 8: Validación

CLASE No 8:	**Validación**	
Descripción	Comprende información correspondiente a las validaciones (reparaciones) de los baches	
Atributo	**Tipo**	**Descripción**
id_validacion	int	Identificador para la validación de un bache
estado	boolean	Muestra el estado del bache
fecha_validacion	Date	Fecha de validación de un bache
id_reporte	int	Identificador de reporte

- Nota: Esta tabla contiene información acerca de los atributos en la clase Validación.

2.2.2.2 Diagrama de clases de la aplicación móvil

En este apartado se muestra el diagrama de clases correspondiente a la aplicación móvil, donde se definen las características de cada una de las clases, atributos, y relaciones de dependencia, seguidamente se procede a describir a cada clase que interviene en el desarrollo de la aplicación móvil:

Diagrama de clases aplicación móvil.

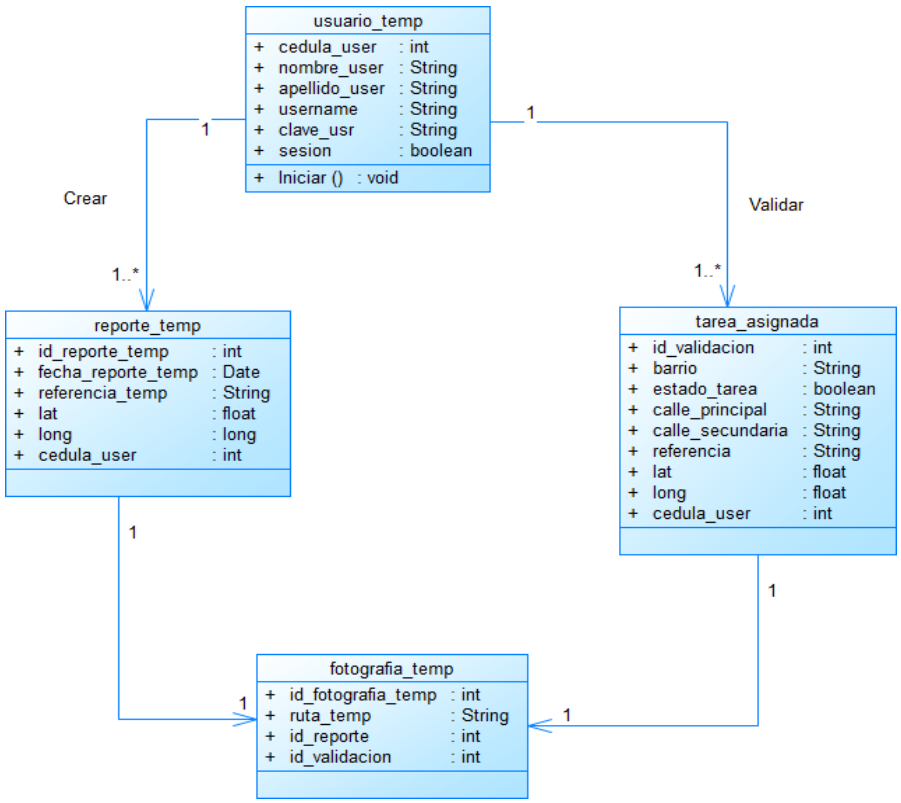

Figura 11. Diagrama de clases aplicación móvil.
Elaborado por: Cristhian Castro y Paul Velasco

Tabla 31 Diagrama de Clase No 1: usuario_temp

CLASE No 1:	**usuario_temp**	
Descripción	Contiene información, datos personales, nombre de usuario y contraseña de los usuarios moderadores, encargados de registrar baches en la ciudad de Quito	
Atributo	**Tipo**	**Descripción**
cedula_user	int	Cédula del usuario
nombre_user	String	Nombre del usuario
apellido_user	String	Apellido del usuario
username	String	Nombre de usuario para acceder a la aplicación móvil
clave_usr	String	Clave de usuario para acceder a la aplicación móvil
sesion	boolean	Estado de sesión del usuario

Operaciones	Función	Descripción
Iniciar()	Void	Permite al usuario iniciar sesión el usuario en la aplicación móvil

- Nota: Esta tabla contiene información acerca de los atributos en la clase usuario_temp.

Tabla 32 Diagrama de Clase No 2: reporte_temp

CLASE No 2:	**reporte_temp**	
Descripción	Incorpora información correspondiente a los baches en la ciudad como coordenadas (latitud y longitud) del incidente, fecha, referencia al sitio donde se encuentra el bache.	
Atributo	**Tipo**	**Descripción**
id_reporte_temp	int	Identificador del reporte
fecha_reporte_temp	Date	Fecha en el que se realiza el reporte
lat	float	Coordenadas de latitud del reporte
long	float	Coordenadas de longitud del reporte
referencia_temp	String	Referencia al sitio donde se encuentra el bache.
cedula_usr	Int	Cédula del moderador que reporta el incidente

- Nota: Esta tabla contiene información acerca de los atributos en la clase reporte_temp.

Tabla 33 Diagrama de Clase No 3: fotografía_temp

CLASE No 3:	**fotografia_temp**	
Descripción	Contiene las fotografías de los baches, capturadas por los usuarios moderadores, dentro de la aplicación móvil.	
Atributo	**Tipo**	**Descripción**
id_fotografia_temp	int	Identificador de la fotografía.
ruta_temp	String	Localización exacta del archivo fotográfico.
id_reporte	int	Identificador del reporte
Id_validacion	int	Identificador de validación

- Nota: Esta tabla contiene información acerca de los atributos en la clase fotografía_temp.

Tabla 34 Diagrama de Clase No 4: tarea_asignada

CLASE No 4:	**tarea_signada**	
Descripción	Engloba una serie de atributos los cuales el usuario moderador deberá ingresar para poder validar los reportes de baches anteriores como baches reparados.	
Atributo	**Tipo**	**Descripción**
id_validacion	int	Identificador de validación
estado_tarea	boolean	Estado de la tarea asignada
calle_principal	String	Calle principal de validación
calle_secundaria	String	Calle secundaria de validación
referencia	String	Referencia al sitio de validación
lat	float	Coordenadas de latitud de validación
long	float	Coordenadas de longitud de validación
cedula_user	int	Cédula del usuario

- Nota: Esta tabla contiene información acerca de los atributos en la clase tarea_asignada.

2.2.3 Diagrama de componentes

Se ha dividido al sistema informático en cuatro ambientes principales:

- **Servidor de base de datos:** Contiene la BDD, que en este caso se usa el sistema gestor de base de datos “PostgreSQL” y el módulo “Postgis” que nos permitirá hacer el manejo de datos espaciales en nuestra BDD.
- **Dispositivo móvil:** Se concentra en la aplicación construida en “Android” y la BDD temporal con la que operará.
- **Navegador Web:** Su función es la de contener los archivos HTML edificados con el apoyo de “Bootstrap” para la parte del CSS, y “JavaScript”.
- **Servidor de aplicaciones:** Dentro de ella está encasillada la aplicación realizada bajo el lenguaje de programación “Python” y el framework para desarrollo Web “Django” acoplado con el módulo “GeoDjango” para las funcionalidades geográficas del aplicativo, este marco de trabajo posee una arquitectura interna que se denomina MVP o modelo, vista, plantilla, en la cual el llamado “django model” es el encargado del mapeo de una tabla de la BDD por lo que es quien nos permite tener un enlace al servidor de base de datos, las “Django Views” encapsulan los métodos o lógica del negocio y están relacionado con los “Django templates” que son quienes presentan la información al usuario mediante el navegador Web. Adicionalmente otro de los componentes es el “Web Service” es la tecnología que permitirá el intercambio de datos con la aplicación móvil.

Mediante la figura No. 11 se observa la interacción de los componentes del sistema informático.

Diagrama de componentes sistema informático

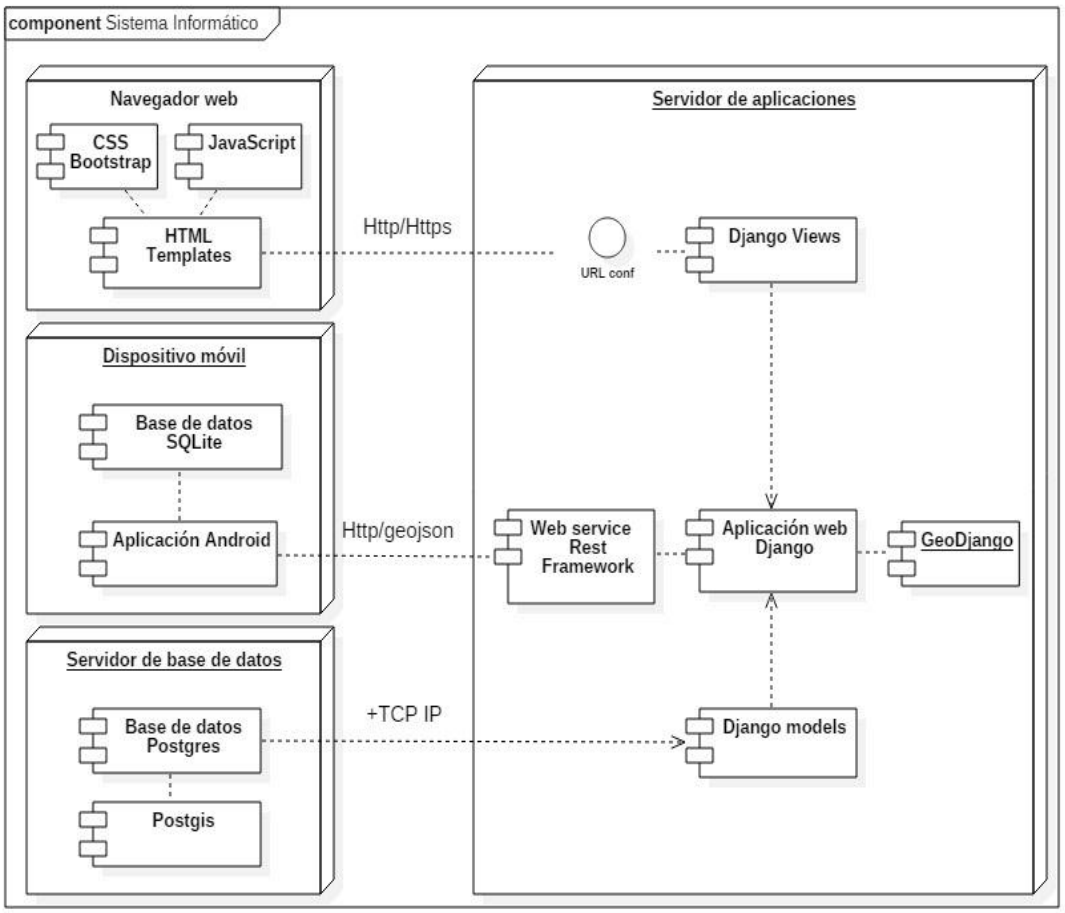

Figura 12. Diagrama de componente sistema informático.
Elaborado por: Cristhian Castro y Paul Velasco

2.2.4 Interfaces gráficas

A continuación, se exhiben los prototipos de interfaces gráficas del usuario tanto de la aplicación Web como móvil, las cuales han sido diseñadas con la herramienta JustinMind.

2.2.4.1 Aplicación Web

Las distintas interfaces aquí representadas han sido divididas mediante los distintos módulos desarrollados en la aplicación.

- **Módulo login**

Esta pantalla será la de inicio de la aplicación, cuyo objetivo es permitir al usuario administrador el ingreso de sus credenciales para acceder al aplicativo.

Prototipo de la interfaz gráfica módulo login (aplicación Web)

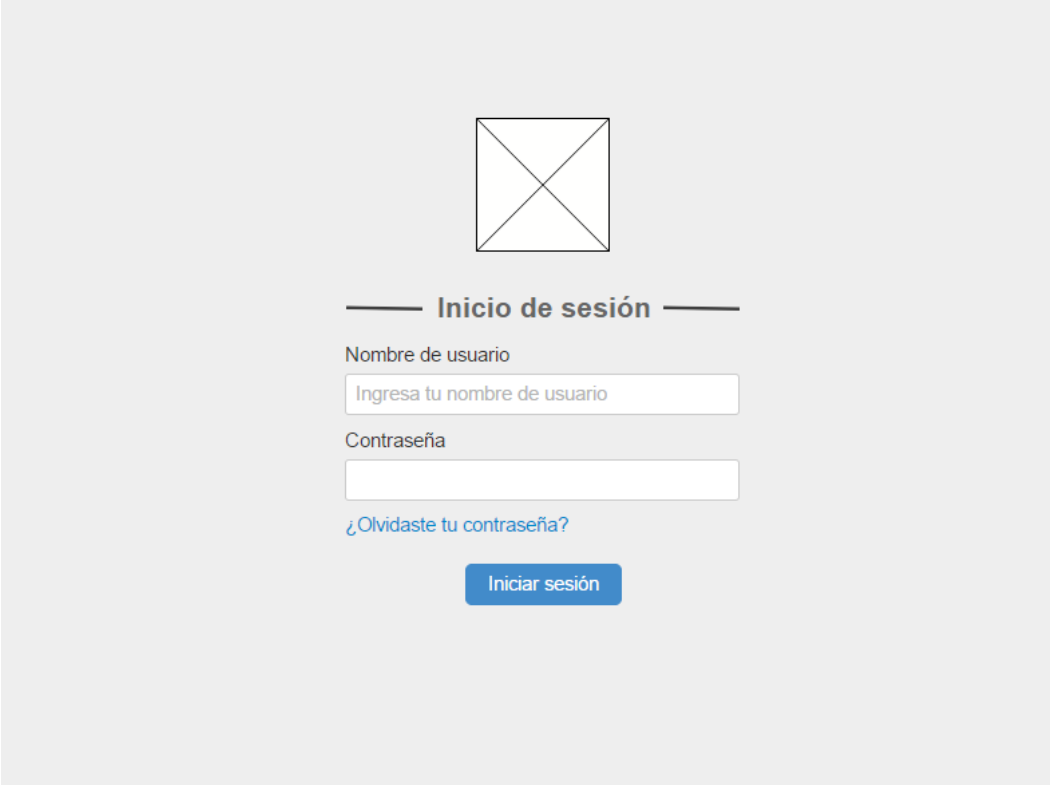

Figura 13. Interfaz gráfica módulo login en la aplicación Web
Elaborado por: Cristhian Castro y Paul Velasco

- **Módulo administración de registros**

Este módulo ha sido dividido en tres pantallas con las que el usuario interactuará, la primera se refiere al mapa donde el administrador podrá navegar y acceder a los detalles de cada bache registrado e incluso modificar su estado, la segunda corresponde a un listado de todos los baches registrados en el sistema con lo cual el administrador pueda hacer cambios en sus estados por medio de acciones en lote e individuales además de brindar la posibilidad de asignar los baches a un moderador para su validación y de igual forma se ejecutará acciones grupales a los registros, finalmente la tercera concierne a la creación de los reportes parametrizados.

Prototipo de la interfaz gráfica módulo administración de registros (Mapa)

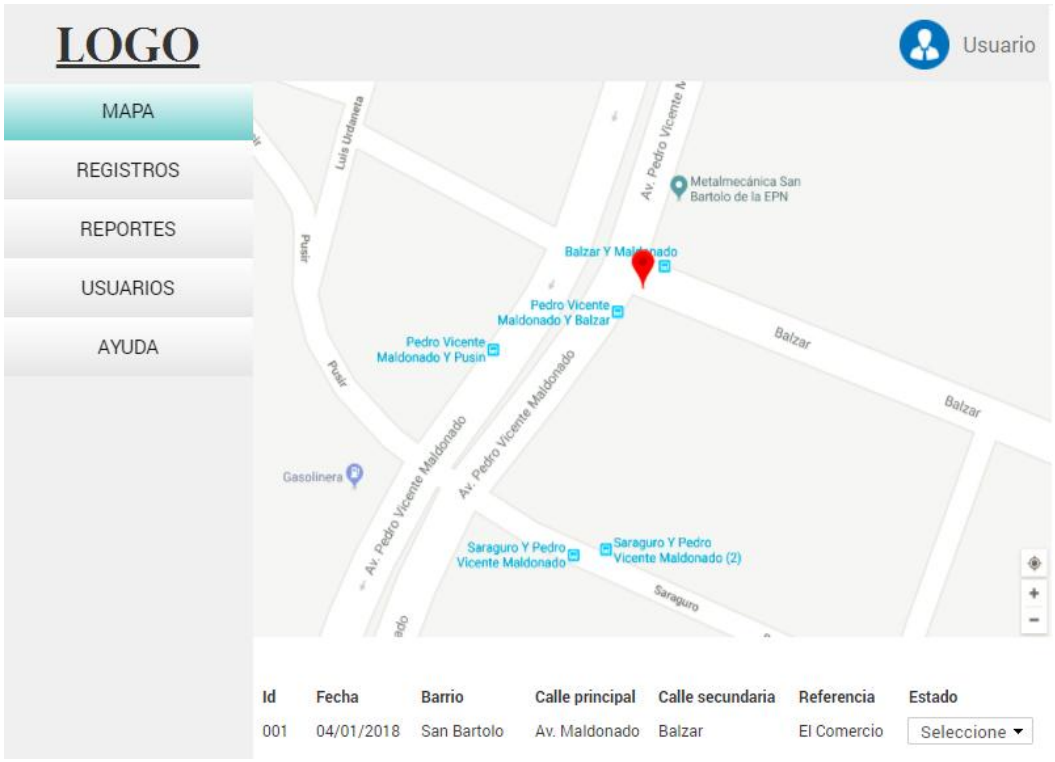

Figura 14. Interfaz gráfica módulo de administración de registros (Mapa).
Elaborado por: Cristhian Castro y Paul Velasco

Prototipo de la interfaz gráfica módulo administración de registros (Lista)

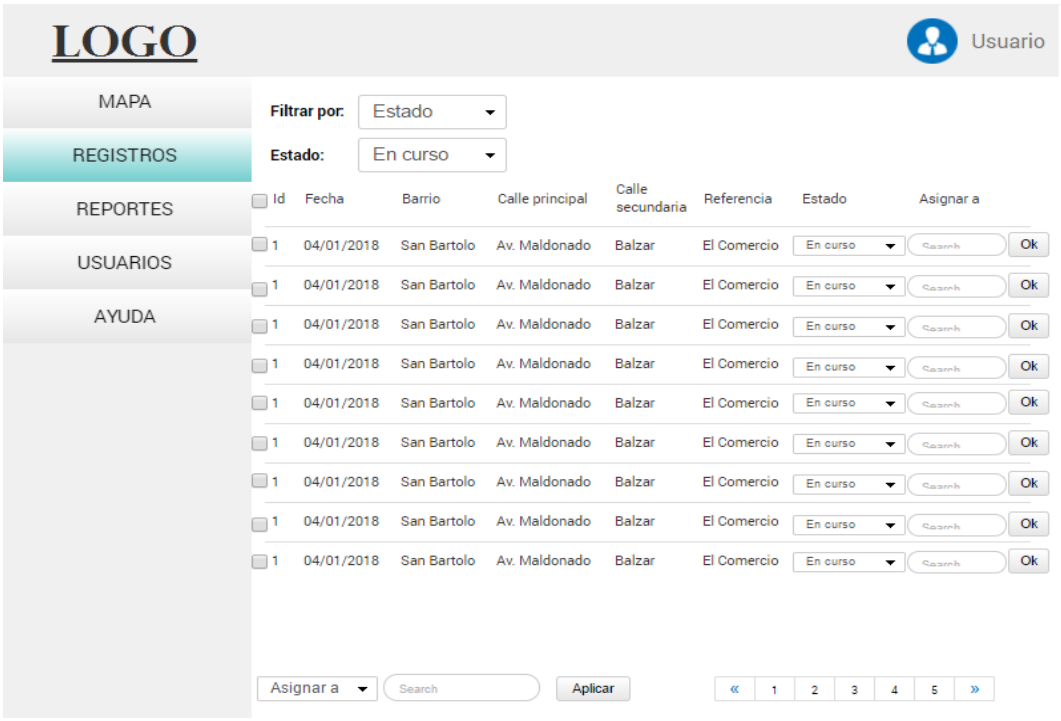

Figura 15. Interfaz gráfica módulo de administración de registros (Lista).
Elaborado por: Cristhian Castro y Paul Velasco

Prototipo de la interfaz gráfica módulo administración de registros (reportes)

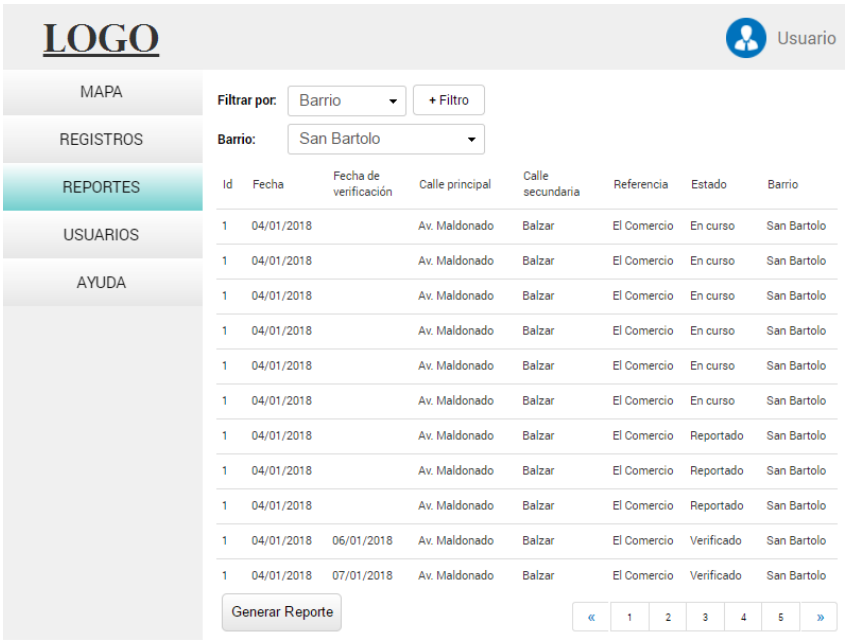

Figura 16. Interfaz gráfica módulo de administración de registros (Reportes).
Elaborado por: Cristhian Castro y Paul Velasco

- **Módulo administración de usuarios**

Las interfaces gráficas con las que el administrador puede trabajar en este módulo son las de listado, creación y modificación de usuarios, estos dos últimos serán presentados por medio de formularios dentro de ventanas emergentes (modales).

Prototipo de la interfaz gráfica módulo administración de usuarios (ingresar)

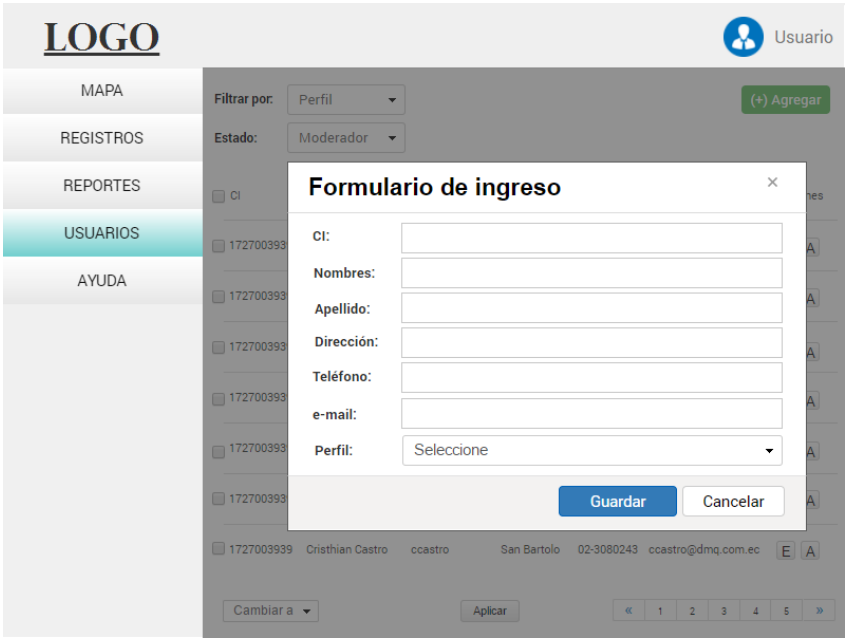

Figura 17. Interfaz gráfica módulo de administración de usuarios (Ingreso).
Elaborado por: Cristhian Castro y Paul Velasco

Prototipo de la interfaz gráfica módulo administración de usuarios (modificar)

Figura 18. Interfaz gráfica módulo de administración de usuarios (Modificación).
Elaborado por: Cristhian Castro y Paul Velasco

2.2.4.2 Aplicación móvil

El prototipo de la aplicación móvil se trata de una versión inicial del programa que funcionará en teléfonos inteligentes, tablets y otros dispositivos móviles, el mismo que muestra conceptos y opciones de diseño agradables para el usuario de tal manera que se pueda entender el producto final.

- **Módulo inicio de sesión**

Se trata de la interfaz donde el usuario moderador ingresa sus credenciales que están almacenadas en la base de datos, donde el ingreso de estas credenciales permite o niegan el acceso a la aplicación móvil.

Prototipo de la interfaz gráfica inicio de sesión (aplicación móvil)

Figura 19. Prototipo navegacional - Inicio de sesión en la aplicación móvil.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Módulo Mapa**

Esta interfaz muestra la pantalla principal de la aplicación móvil, el cual es un mapa de la ciudad de Quito con 2 botones esenciales el primer botón es para activar el GPS del dispositivo móvil y así mostrar la ubicación exacta del dispositivo, el segundo botón extiende un menú desplazable que contiene todos los módulos de la aplicación móvil.

Prototipo de la interfaz gráfica mapa inicial (aplicación móvil)

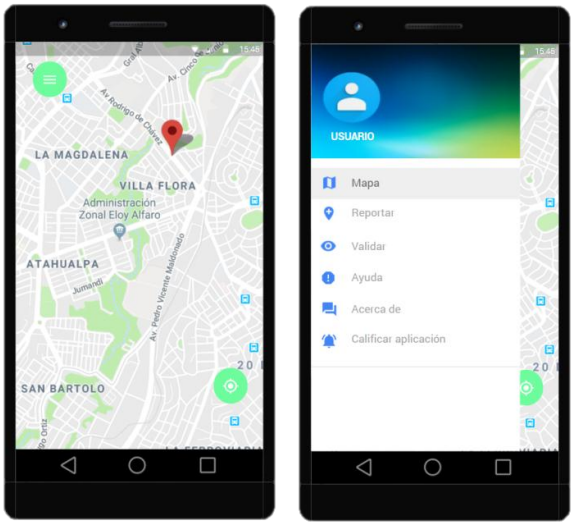

Figura 20. Prototipo navegacional - Módulo mapa en la aplicación móvil.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Módulo Reportar**

La siguiente pantalla muestra el formulario que permite al usuario moderador realizar un reporte de baches en la ciudad de Quito, donde se debe tomar las coordenadas del GPS del dispositivo móvil, ingresar una referencia al lugar del incidente, tomar o adjuntar una fotografía del bache y por último ingresar la fecha del incidente.

Prototipo de la interfaz gráfica reporte de bache (aplicación móvil)

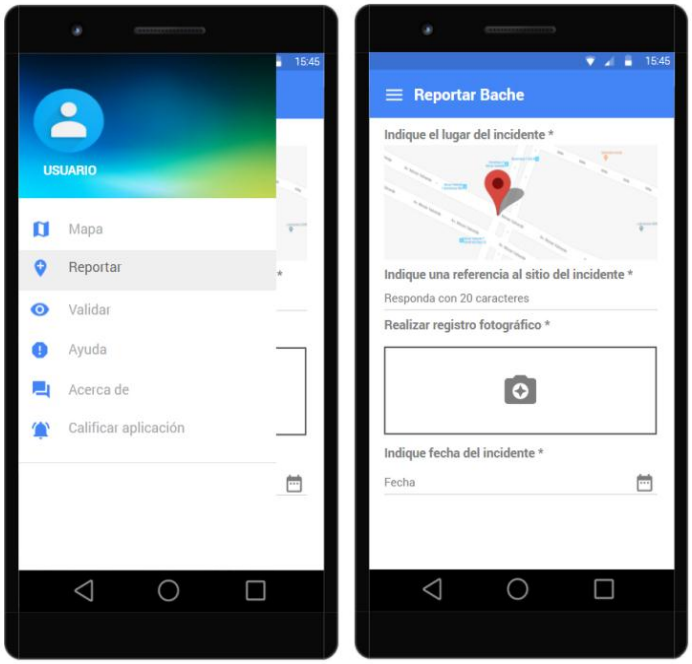

Figura 21. Prototipo navegacional - Módulo reportar en la aplicación móvil.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Módulo Validar**

En este módulo se muestra la pantalla que permite al usuario moderador realizar validaciones de los baches que han sido reparados, donde el moderador realiza un recorrido por puntos establecidos por el administrador del sistema para verificar si los baches han sido reparados o no han sido reparados.

Prototipo de la interfaz gráfica validación de bache (aplicación móvil)

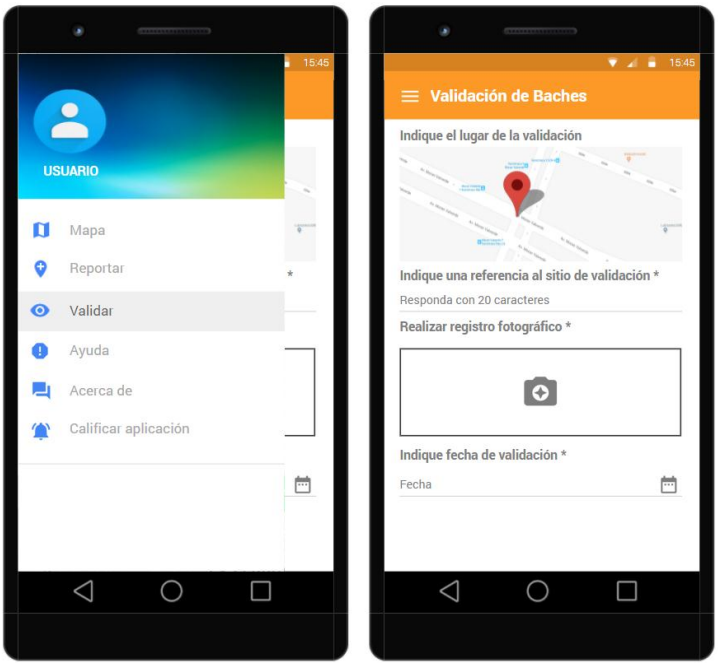

Figura 22. Prototipo navegacional - Módulo validar en la aplicación móvil.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Módulo Ayuda**

La siguiente pantalla muestra un módulo de ayuda que disipara de forma puntual las dudas que tenga el usuario acerca de cómo usar la aplicación móvil.

Prototipo de la interfaz gráfica pantall de ayuda (aplicación móvil)

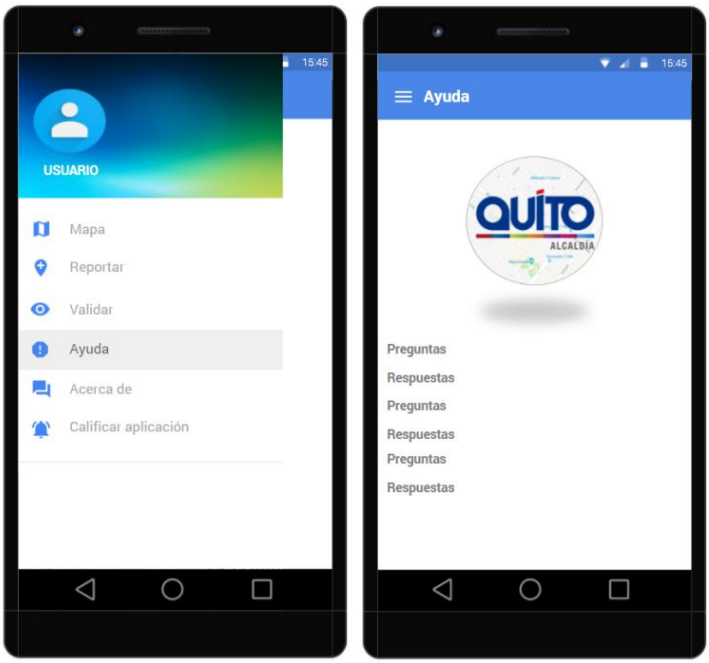

Figura 23. Prototipo navegacional - Módulo ayuda en la aplicación móvil.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Módulo Acerca de**

En este módulo se muestra la pantalla que contiene el propósito de haber desarrollado el sistema informático, así mismo se da información al usuario acerca de los autores intelectuales y entidades que fueron parte de la creación del sistema.

Prototipo de la interfaz gráfica pantalla acerca de (aplicación móvil)

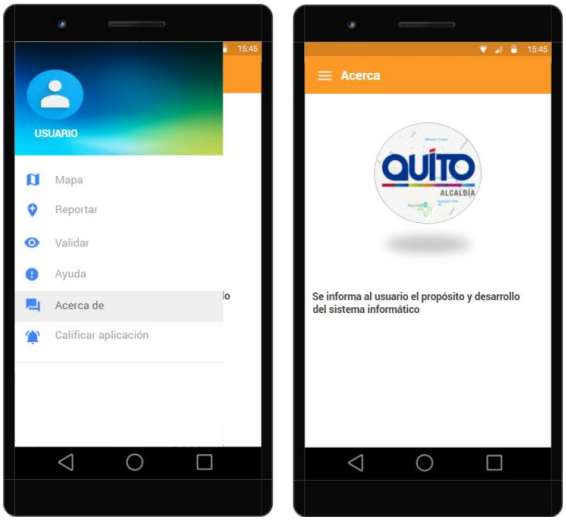

Figura 24. Prototipo navegacional - Módulo acerca de en la aplicación móvil.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Módulo Calificar aplicación**

Esta interfaz contiene un enlace directo hacia la tienda de Google (Play Store), donde el usuario podrá calificar a la aplicación móvil, y comentar si es de su agrado o no.

Prototipo de la interfaz gráfica pantalla de calificación (aplicación móvil)

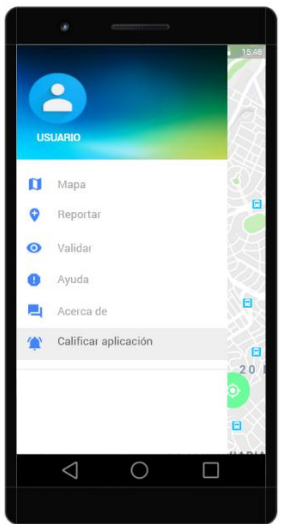

Figura 25. Prototipo navegacional - Módulo calificar aplicación.
Elaborado por: Cristhian Castro y Paul Velasco.

Capítulo 3

Construcción y Pruebas

3.1 Construcción

En este apartado se exponen detalles inherentes a la fase de desarrollo de los componentes del sistema, por lo que el punto de partida es el diagrama y diccionario de base de datos, junto a esto se puntualizan las historias de usuario y las iteraciones surgidas a partir de la aplicación de la metodología XP en la etapa de programación, por último, se presenta fragmentos de código descritos, que resultan esenciales, para el funcionamiento del sistema. También se incluyen aplicaciones que ofrecieron soporte al momento de especificar, diseñar, construir, y documentar el sistema.

3.1.1 Historias de usuario

Al usar una metodología ágil tal como XP para el desarrollo del sistema es prescindible detallar las historias de usuario derivadas a partir de los requerimientos planteados del proyecto. Así mismo se describe las iteraciones resultantes del desarrollo de los objetivos planteados por cada historia de usuario.

3.1.1.1 Administración de usuarios

Tabla 35 Historia de usuario No 1: Administración de usuarios

Historia de usuario		**21/02/2018**
N.º de historia	1	
Nombre historia	Administración de usuarios	
Responsable	Cristhian Castro	
Descripción		
Al acceder a este módulo, se despliega el listado de usuarios registrados en el sistema a cada uno de los cuales se brinda la posibilidad de editar datos personales, activar o desactivar, y la opción de reestablecer contraseña. A su vez el administrador puede ingresar nuevos usuarios mediante un formulario en el que se llena datos personales del moderador.		
Objetivos	**1.** Crear formulario de ingreso de datos del usuario para su registro en el sistema. **2.** Crear formulario de modificación de datos del usuario. **3.** Desarrollar un botón que permita reestablecer a la contraseña por defecto.	Por desarrollar Por desarrollar Por desarrollar

	4. Incorporar un botón que permita cambiar el estado actual del usuario.	Por desarrollar
Razón	Para que los usuarios puedan hacer uso de las aplicaciones del sistema. Para que el administrador pueda modificar los datos de los usuarios en caso de error de ingreso o se requiera cambiar el estado del usuario. Para que el administrador pueda hacer uso de la acción, en el caso de que los usuarios olviden su contraseña. Realizar la acción de cambio de estado de forma independiente a la modificación de los datos personales de cada usuario.	
Observación		
Este proceso es llevado a cabo en la aplicación Web.		

- Nota: Esta tabla contiene los objetivos a desarrollarse en la HU: Administración de usuarios

- **Primera iteración**

Tabla 36 Historia de usuario No 1: Administración de usuarios (Iteración 1)

N.º de historia	1	**Fecha**	**27/02/2018**
Para la creación de los formularios de ingreso y modificación de usuarios, se decidió mostrarlos en ventanas modales, con esto se buscó evitar el exceso de pantallas, y que el administrador no abandone la lista desplegada de usuarios del sistema. En el proceso de ingreso de un nuevo usuario al sistema el administrador deberá completar los datos personales, además se definió que el "username" sea creado automáticamente por el sistema formado por la primera letra del nombre junto al apellido así mismo se debe certificar que no existan duplicidades en los "usernames" generados, así también que la contraseña por defecto de cada nuevo usuario sea su cédula de identidad. En la ventana de modificación de usuario, se presenta el mismo formulario del ingreso, pero en esta ocasión se encuentra parametrizado con los datos de cada usuario, con el fin de que el administrador cambie los datos necesarios, incluyendo el estado.			

- Nota: Esta tabla muestra el desarrollo de la HU: Administración de usuarios, iteración 1.

- **Segunda iteración**

Tabla 37 Historia de usuario No 1: Administración de usuarios (Iteración 2)

N.º de historia	1	**Fecha**	**01/03/2018**

Partiendo de los objetivos de la historia de usuario 1, en la segunda iteración se trabajó en el desarrollo de dos botones, de los cuales el primero permita al administrador reestablecer la contraseña del moderador a la generada por defecto en el sistema, y el segundo botón permitirá que se active o desactive los usuarios, cuyo funcionamiento resulte independiente del formulario de edición de datos personales, Esta iteración se centró en la presentación del listado de usuarios del sistema, añadiendo un color identificativo según el estado de cada uno, además se agregó filtros por estado, nombre, perfil, con el propósito de facilitar la administración de los usuarios registrados en el sistema.

- Nota: Esta tabla muestra el desarrollo de la HU: Administración de usuarios, iteración 2

- **Resultado Final**

Tabla 38 Objetivos cumplidos en el módulo Administración de usuarios

N.º de historia 1	**Objetivos**	**Estado**
Objetivos	**1.** Crear formulario de ingreso de datos del usuario para su registro en el sistema.	Cumplido
	2. Crear formulario de modificación de datos del usuario.	Cumplido
	3. Desarrollar un botón que permita reestablecer a la contraseña por defecto.	Cumplido
	4. Incorporar un botón que permita cambiar el estado actual del usuario.	Cumplido

- Nota: Esta tabla muestra el cumplimiento de todos los objetivos de la HU: Administración de usuarios.

3.1.1.2 Autenticación de usuarios

Tabla 39 Historia de usuario No 2: Autenticación de usuarios

Historia de usuario		**03/03/2018**
N.º de historia	2	
Nombre historia	Autenticación de usuarios	
Responsable	Cristhian Castro y Paul Velasco	
Descripción		
En esta parte se avala que los usuarios registrados en el sistema accedan al sistema informático por medio de las credenciales asignadas. El sistema controla que el acceso a las aplicaciones sea según el perfil asignado, siendo "moderador" aplicación móvil, "administrador" aplicación Web. El sistema garantiza la integridad de nombre de usuario y contraseña en el proceso de autenticación.		
Objetivos	**1.** Crear formulario de inicio de sesión en la aplicación Web.	Por desarrollar
	2. Crear formulario de inicio de sesión en la aplicación móvil.	Por desarrollar
	3. Asegurar el acceso por parte de la aplicación móvil a los datos de los usuarios moderadores almacenados en el servidor.	Por desarrollar

	4. Implementar las seguridades necesarias en las acciones de autenticación.	Por desarrollar
Razón	Con el fin de que los usuarios administradores puedan hacer uso de la aplicación Web. Para que los usuarios moderadores accedan a la aplicación móvil. Permitir que la aplicación móvil tenga acceso mediante un API a los datos de moderadores alojados en el servidor. Con esto se busca que el método de autenticación usado por la aplicación móvil sea seguro evitando posibles vulnerabilidades.	
Observación		
Este proceso se lo debe destinar tanto a la aplicación Web como la móvil.		

- Nota: Esta tabla contiene los objetivos a desarrollarse en la HU para el módulo de Autenticación de usuarios

Primera iteración

Tabla 40 Historia de usuario No 2: Autenticación de usuarios (Iteración 1)

N.º de historia	2	**Fecha**	**09/03/2018**
El acceso por parte de los usuarios a las aplicaciones del sistema se lo hace mediante autenticación con las credenciales concedidas a cada uno de ellos, tal como se detalló en los requerimientos, el sistema controla el perfil asignado al usuario en función de eso concede permiso de acceso a cada una de las aplicaciones. Para esto se construyó un formulario de inicio de sesión en cada una de los aplicativos. Para el proceso de ingreso a la aplicación móvil se desarrolló un API de autenticación la cual consume los datos guardados en el servidor, por lo que el usuario para este procedimiento debe contar con conexión a internet.			

- Nota: Esta tabla muestra el desarrollo de la HU: Autenticación de usuario, iteración 1

- **Segunda iteración**

Tabla 41 Historia de usuario No 2: Autenticación de usuarios (Iteración 2)

N.º de historia	2	**Fecha**	**14/03/2018**
En esta iteración surgió como propuesta el cambio de las interfaces graficas de los formularios de inicio de sesión, por lo que se rediseñó los estilos, para lograr un mejor efecto en la mezcla de colores, fuentes, iconos y otros elementos de la pantalla. El método de autenticación de usuarios empleado por la API en la aplicación móvil cumplía la funcionalidad de verificar e iniciar sesión, sin embargo, no implementaba procedimientos que resguarden y aseguren el acceso al sistema lo que pudiese comprometer la información almacenada.			

- Nota: Esta tabla muestra el desarrollo de la HU: Autenticación de usuario, iteración 2

- **Tercera iteración**

Tabla 42 Historia de usuario No 2: Autenticación de usuarios (Iteración 3)

N.º de historia	2	**Fecha**	**20/03/2018**
En esta iteración se agregó el tipo de autenticación por token, lo que significa que cada una de las peticiones HTTP que se hagan al servidor deberán incluir en su cabecera un token generado automáticamente por el sistema al momento de crear al usuario. Por este motivo el proceso tuvo un ligero cambio en su funcionamiento, ya que primero se hace una petición enviando las credenciales al sistema en caso de ser correctas la respuesta es el token asignado a ese usuario. Este token es utilizado para consumir otros servicios brindados por el sistema informático.			

- Nota: Esta tabla muestra el desarrollo de la HU: Autenticación de usuario, iteración 3

- **Cuarta iteración**

Tabla 43 Historia de usuario No 2: Autenticación de usuarios (Iteración 4)

N.º de historia	2	**Fecha**	**21/03/2018**
Se centró en la aplicación móvil, adaptando la funcionalidad de que el proceso de autenticación sea realizado una única vez a pesar de que el aplicativo sea cerrado. Para esto se utilizó la técnica de almacenar el estado en caso de ya haber hecho el inicio de sesión, este valor se comprueba al ejecutar la aplicación.			

- Nota: Esta tabla muestra el desarrollo de la HU: Autenticación de usuario, iteración 4

- **Resultado Final**

Tabla 44 Objetivos cumplidos en el módulo Autenticación de usuarios

N.º de historia 2	**Objetivos**	**Estado**
Objetivos	**1.** Crear formulario de inicio de sesión en la aplicación Web.	Cumplido
	2. Crear formulario de inicio de sesión en la aplicación móvil.	Cumplido
	3. Asegurar el acceso por parte de la aplicación móvil a los datos de los usuarios moderadores almacenados en el servidor.	Cumplido
	4. Implementar las seguridades necesarias en las acciones de autenticación.	Cumplido

- Nota: Esta tabla muestra el cumplimiento de todos los objetivos de la HU: Autenticación de usuarios

3.1.1.3 Administración de registros

Tabla 45 Historia de usuario No 3: Administración de registros

Historia de usuario		**23/03/2018**
N.º de historia	3	
Nombre historia	Administración de registros	
Responsable	Cristhian Castro	
Descripción		
Su fin es el de administrar todos los baches registrados mediante la aplicación móvil. En donde tendrá la posibilidad de visualizar el punto en el mapa, fotografía del incidente		

y a la vez el administrador podrá hacer cambios en el estado del registro, asignaciones de validación a los moderadores, y generar graficas de reportes parametrizables.		
Objetivos	**1.** Listar los registros almacenados en la base de datos.	Por desarrollar
	2. Visualizar un mapa donde se encuentren registrados los baches clasificados por su estado.	Por desarrollar
	3. Observar el punto individual geolocalizado en un mapa.	Por desarrollar
	4. Mostrar fotografía asociada con el incidente ya sea de reporte o validación.	Por desarrollar
	5. Añadir la funcionalidad para que el administrador pueda cambiar de estado reportado a en curso a los registros.	Por desarrollar
	6. Incorporar un botón con el cual se pueda asignar registros de baches a moderadores para su validación.	Por desarrollar
	7. Permitir al administrador realizar acciones masivas (cambiar estado, asignación) sobre grupos de registros seleccionados.	Por desarrollar
	8. Generar graficas de reportes parametrizables por fecha, estado, administración zonal, parroquia, de los incidentes.	Por desarrollar
Razón	Para que el administrador tenga acceso a los registros almacenados en la base, los cuales puedan ser filtrados facilitando su seguimiento y administración. Que el administrador pueda ver en un mapa la localización exacta del bache. Acceso a un mapa de la ciudad con todos los baches reportados y el estado en el que se encuentran. Para que al administrador vea la imagen asociada al registro del bache o a su validación una vez reparado. Para que el administrador cambie de estado reportado a en curso a los registros que han sido enviados para su reparación. Para asignar a moderadores los registros reparados para su validación. Para facilitar al administrador las acciones de cambio de estado y asignación de reportes. Para mantener un control y facilitar la toma de decisiones.	
Observación		
Las acciones correspondientes a esta historia se lo realizan en la aplicación Web.		

- Nota: Esta tabla posee los objetivos a cumplirse en la HU Administración de registros

- **Primera iteración**

Tabla 46 Historia de usuario No 3: Administración de registros (Iteración 1)

N.º de historia	3	**Fecha**	**25/03/2018**
La primera iteración se enfocó en el listado de registros, en donde se desplegó información relacionada tales como fecha del suceso, moderador responsable, dirección, parroquia y administración zonal. Para esta parte se puso énfasis en la construcción de la tabla la cual contiene filtros, acciones de ordenamiento, y paginación que facilitan la gestión y control de los registros. Con el fin de aportar al listado mayor dinamismo se implementó un código de colores relacionado al estado.			

- Nota: Esta tabla muestra el desarrollo de la HU: Administración de registros, iteración 1

- **Segunda iteración**

Tabla 47 Historia de usuario No 3: Administración de registros (Iteración 2)

N.º de historia	3	**Fecha**	**28/03/2018**
En cuanto a las acciones individuales que se posibilitan en cada registro de bache están, la visualización de la locación exacta mediante un mapa centrado en el punto a manera de croquis. Otra acción es la de la visualización de la fotografía relacionada al incidente, en los casos de: registro en el cual el administrador observa una foto y validación donde se construyó una galería para ver el antes y después de la reparación del bache. En este módulo también es posible cambiar de estado reportado a en curso individualmente a cada registro, por lo que se incorporó un botón para cumplir esta funcionalidad. Para la determinación del usuario al que será asignado se tomó en cuenta las tareas de validación pendientes que pudiese tener, el moderador con menos tareas será elegido.			

- Nota: Esta tabla muestra el desarrollo de la HU: Administración de registros, iteración 2

- **Tercera iteración**

Tabla 48 Historia de usuario No 3: Administración de registros (Iteración 3)

N.º de historia	3	**Fecha**	**31/03/2018**
En cuanto a las acciones individuales que se posibilitan en cada registro de bache están, la visualización de la locación exacta mediante un mapa centrado en el punto a manera de croquis. Otra acción es la de la visualización de la fotografía relacionada al incidente, en los casos de: registro en el cual el administrador observa una foto y validación donde se construyó una galería para ver el antes y después de la reparación del bache. En este módulo también es posible cambiar de estado reportado a en curso individualmente a cada registro, por lo que se incorporó un botón para cumplir esta funcionalidad. Para la determinación del usuario al que será asignado se tomó en cuenta las tareas de validación pendientes que pudiese tener, el moderador con menos tareas será elegido.			

- Nota: Esta tabla muestra el desarrollo de la HU: Administración de registros, iteración 3

- **Cuarta iteración**

Tabla 49 Historia de usuario No 3: Administración de registros (Iteración 4)

N.º de historia	3	**Fecha**	**02/04/2018**
Con el motivo de facilitar la acción del administrador sobre los registros, a parte de la funcionalidad individual de cambio de estado y asignación, se desarrolló botones que permitan realizar estas acciones a un grupo de registros, con el fin de disminuir el tiempo que pudiera transcurrir al hacerlo uno por uno. La funcionalidad es sencilla el administrador mediante un checkbox selecciona los registros a los que quiere aplicar la respectiva acción y ejecuta el botón.			

- Nota: Esta tabla muestra el desarrollo de la HU: Administración de registros, iteración 4

- **Quinta iteración**

Tabla 50 Historia de usuario No 3: Administración de registros (Iteración 5)

N.º de historia	3	**Fecha**	**08/04/2018**
Como siguiente punto en este módulo se incluyó un mapa en el que se visualizan todos los puntos registrados en la ciudad, cada uno de los cuales se representa por un color según el estado en el que se encuentran. Adicional al hacer clic en cada punto se desplegará una ventana emergente, con información acerca del bache, fotografía, fecha de reporte, reportador y en el caso de ser un bache ya reparado adicional a los datos anteriores se añade la fotografía de la reparación, fecha de validación y el validador. Dentro del mapa se cuenta con un filtro por el que se elige que tipos de registros desea ver, en función del estado.			

- Nota: Esta tabla muestra el desarrollo de la HU: Administración de registros, iteración 5

- **Sexta iteración**

Tabla 51 Historia de usuario No 3: Administración de registros (Iteración 6)

N.º de historia	3	**Fecha**	**14/04/2018**
En esta iteración se trabajó los reportes que genera el sistema. Los criterios son "registros por administración zonal", "registros por parroquia", en donde se muestran los baches categorizados por estado. La administración zonal es proveniente a la que se encuentra ligado el usuario administrador, de igual manera en base a esto se cargan las parroquias de las cuales se puede obtener el historial. En si el reporte es una serie de gráficos que permiten visualizar dinámicamente los criterios antes mencionados, es también configurable el rango de fechas que se emitirá el reporte, así también el tipo de grafico pudiendo ser barras o pastel. Finalmente, los gráficos resultantes de aplicar los distintos parámetros se pueden descargar en formato pdf, jpg, png o svg.			

- Nota: Esta tabla muestra el desarrollo de la HU: Administración de registros, iteración 5

- **Resultado Final**

Tabla 52 Objetivos cumplidos en el módulo Administración de registros

N.º de historia 3	Objetivos	Estado
Objetivos	**1.** Listar los registros almacenados en la base de datos.	Cumplido
	2. Visualizar un mapa donde se encuentren registrados los baches clasificados por su estado.	Cumplido
	3. Observar el punto individual geolocalizado en un mapa.	Cumplido
	4. Mostrar fotografía asociada con el incidente ya sea de reporte o validación.	Cumplido
	5. Añadir la funcionalidad para que el administrador pueda cambiar de estado reportado a en curso a los registros.	Cumplido
	6. Incorporar un botón con el cual se pueda asignar registros de baches a moderadores para su validación.	Cumplido
	7. Permitir al administrador realizar acciones masivas (cambiar estado, asignación) sobre grupos de registros seleccionados.	Cumplido
	8. Generar graficas de reportes parametrizables por fecha, estado, administración zonal, parroquia, de los incidentes.	Cumplido

- Nota: Esta tabla muestra el cumplimiento de todos los objetivos de la HU: Administración de registros.

3.1.1.4 Reporte y validación de baches

Tabla 53 Historia de usuario No 4: Reporte y Validación

Historia de usuario	**20/04/2018**
N.º de historia	4
Nombre historia	Reporte y Validación
Responsable	Cristhian Castro y Paul Velasco
Descripción	
Esta sección tiene como propósito que los usuarios moderadores puedan reportar y validar baches en la ciudad de Quito, con el uso de la aplicación móvil donde se usarán parámetros como una toma fotográfica o carga de imagen con su dispositivo móvil, la ubicación geográfica con las coordenadas de latitud y longitud, la dirección de la calle principal y una referencia al sitio donde se encuentra el bache o relleno, de manera automática el identificador del usuario moderador se enviará junto con los parámetros anteriores. El barrio y parroquia y la fecha de envió se calcularán en la aplicación Web de forma automatizada.	

Objetivos	**1.** Construir interfaz de reporte, que abarque todos los parámetros de envió de información en una sola pantalla	Por desarrollar
	2. Mostrar un mapa interactivo en el cual el usuario moderador pueda visualizar su ubicación en tiempo real, en la pantalla de reporte y validación	Por desarrollar
	3. Tomar una fotografía, o cargar desde la galería y mostrar la fotografía en la pantalla de reporte y validación	Por desarrollar
	4. Determinar de forma automática la calle principal con las coordenadas de latitud/longitud, en la pantalla de reporte y validación	Por desarrollar
	5. Cambiar la imagen a formato BASE64 y comprimirla.	Por desarrollar
	6. Enviar todos los parámetros en formato JSON hacia la aplicación Web, tanto en la pantalla de reporte y validación	Por desarrollar
	7. Crear una lista de validaciones pendientes que el usuario pueda visualizar en la aplicación móvil	Por desarrollar
Razón	Para que el usuario moderador pueda reportar la existencia de baches. Que el usuario moderador pueda ver en un mapa la localización exacta del bache o del bache reparado, todo en tiempo real Para que quede constancia forma gráfica el reporte del bache, o la validación del mismo Para que el proceso de verificar la dirección de la calle sea un proceso automático, y también para constatar que el usuario moderador en verdad se dirige a los sitios donde existan baches o rellenos. Para poder enviar la imagen convertida en un String, tanto en la pantalla de reporte y validación Para que la aplicación Web reciba todos los datos en un formato menos detallado con valores de clave/valor. Para que el usuario moderador se dirija al lugar del suceso y verifique si el bache ha sido reparado.	
Observación		
Las acciones correspondientes a esta historia se lo realizan en la aplicación móvil.		

- Nota: Esta tabla posee los objetivos a cumplirse en la HU Administración de registros

- **Primera iteración**

Tabla 54 Historia de usuario No 4: Reporte y Validación (Iteración 1)

N.º de historia	4	**Fecha**	**29/04/2018**

Para la creación de la interfaz de reporte, se decidió mostrar la información en 2 fragmentos diferentes, un fragmento pequeño que mostrará un mapa de la ciudad de Quito, junto con un marcador de posición en tiempo real y el otro fragmento que contendrá al fragmento anterior y mostrará los parámetros de envió de información como tomar una fotografía o cargar una imagen, ingresar una referencia, además se definió que la dirección de la calle principal donde se encuentra el bache sea de forma automática. Con este diseño se buscó evitar el exceso de pantallas para que el usuario moderador se enfoque en realizar reportes sistemáticos.

- Nota: Esta tabla muestra el desarrollo de la HU: Reporte y Validación, iteración 1.

- **Segunda iteración**

Tabla 55 Historia de usuario No 4: Reporte y Validación (Iteración 2)

N.º de historia	4	**Fecha**	**06/05/2018**
En cuanto al marcador de posición en el mapa se mejoró esta premisa para que funcione a medida que el usuario se moviliza, de tal manera que el marcador se visualice en la aplicación en tiempo real. Al dar clic sobre el marcador de posición este mostrará las coordenadas de latitud y de longitud, para evidenciar el cambio de coordenadas Otra acción fue la de implementar un imagebutton que ofrezca la posibilidad al usuario de tomar la fotografía o cargar una imagen desde su galería y mostrar en la pantalla de manera didáctica.			

- Nota: Esta tabla muestra el desarrollo de la HU: Reporte y Validación, iteración 2.

- **Tercera iteración**

Tabla 56 Historia de usuario No 4: Reporte y Validación (Iteración 3)

N.º de historia	4	**Fecha**	**13/05/2018**
Como siguiente punto se incluyó el cálculo de la dirección de la calle principal donde se encuentre el bache o relleno, en base a las coordenadas de latitud y longitud la aplicación muestra la calle principal, para que esta pueda ser enviada junto con los parámetros de referencia, fotografía, coordenadas geográficas (latitud y longitud). De la misma manera la dirección de la calle principal se actualiza conforme cambia el marcador de posición y las coordenadas dentro del mapa de Quito.			

- Nota: Esta tabla muestra el desarrollo de la HU: Reporte y Validación, iteración 3.

- **Cuarta iteración**

Tabla 57 Historia de usuario No 4: Reporte y Validación (Iteración 4)

N.º de historia	4	**Fecha**	**20/05/2018**
Con el motivo de facilitar la acción de enviar reportes o validaciones, se decidió por cambiar el formato de la imagen a BASE64, donde la imagen se convierte en un conjunto de caracteres ASCII, que al ser enviados por internet la aplicación Web decodifica estos caracteres y muestra la imagen resultante.			

Así mismo se optó por redimensionar el tamaño de la imagen antes de codificarla en BASE64, para que la carga de información sea más ligera y el envío de información se mucho más eficaz.

- Nota: Esta tabla muestra el desarrollo de la HU: Reporte y Validación, iteración 4.

- **Quinta iteración**

Tabla 58 Historia de usuario No 4: Reporte y Validación (Iteración 5)

N.º de historia	4	**Fecha**	**27/05/2018**
En esta iteración se trabajó en el formato de envió de información como coordenadas geográficas, dirección de la calle principal, imagen convertida a BASE64, y lugar de referencia, desde la aplicación móvil hacia la aplicación Web y se optó por utilizar el formato de intercambio de información JSON, donde la conversión de objetos java a formato JSON se la realiza en la aplicación móvil y la que recepta y decodifica la información en formato JSON es la aplicación Web. De la misma manera la parroquia y barrio se calcula en la aplicación Web por medio de las coordenadas de latitud y longitud enviadas desde la aplicación móvil.			

- Nota: Esta tabla muestra el desarrollo de la HU: Reporte y Validación, iteración 5.

- **Sexta iteración**

Tabla 59 Historia de usuario No 4: Reporte y Validación (Iteración 6)

N.º de historia	4	**Fecha**	**03/06/2018**
En esta iteración se trabajó en crear la lista de validaciones pendientes las cuales son obtenidas desde la aplicación Web para posteriormente almacenar estas tareas en la base de datos local del dispositivo móvil, y mostrarlas en la interfaz de validación, de tal manera que el usuario pueda cumplir con sus tareas de validación. Finalmente, las tareas de validación enviadas desde la aplicación móvil son reflejadas en la aplicación Web, con todos los parámetros de envió de información como fotografía, coordenadas geográficas (latitud/longitud), dirección de la calle principal y referencia.			

- Nota: Esta tabla muestra el desarrollo de la HU: Reporte y Validación, iteración 6.

3.1.2 Diseño Físico Base de Datos

A continuación, se exhibe y detalla los diseños de base de datos empleados por cada una de las aplicaciones del sistema informático.

3.1.2.1 Base de datos de la aplicación Web

Diagrama físico de la base de datos aplicación Web.

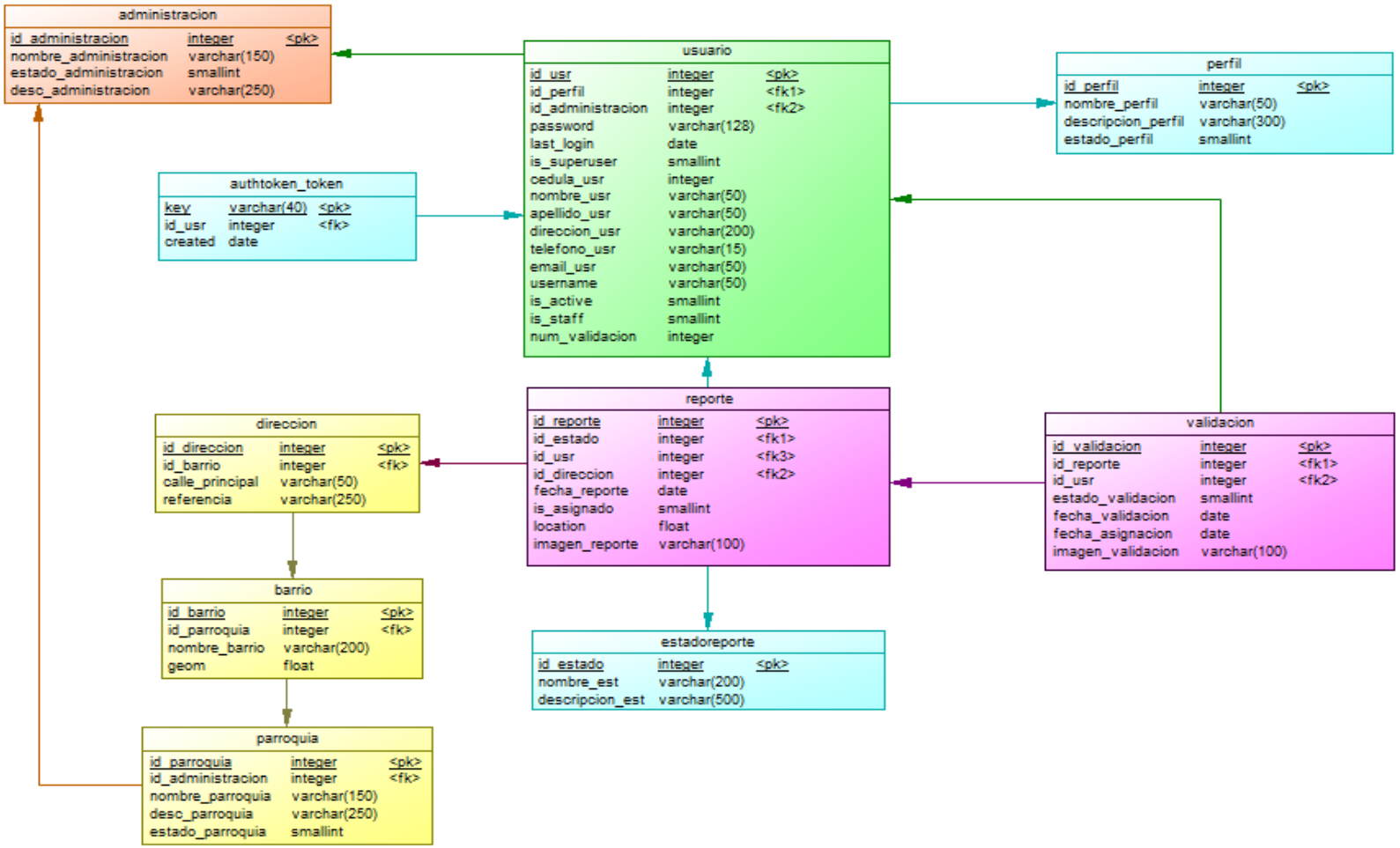

Figura 26. Modelo físico base de datos aplicación Web.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Diccionario de datos aplicación Web**

Con la descripción de cada una de las tablas pertenecientes a la base de datos antes presentada, se busca dar explicación a la utilidad de cada uno de los campos por lo cual se hace uso del esquema siguiente.

Tabla 60 Representación de la tabla Perfil

Tabla	**Perfil**				
Descripción	Contiene la información concerniente a los perfiles de usuario dentro del sistema.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_perfil	integer	5	Identificador de perfil.	Si	PK
nombre_perfil	varchar	50	Campo de nombre de perfil.	Si	
descripcion_perfil	varchar	300	Campo de la descripción del perfil.	No	
estado_perfil	boolean		Campo que indica el estado del perfil	Si	

- Nota: Esta tabla muestra información sobre la tabla perfil.

Tabla 61 Representación de la tabla Administración

Tabla	**Administración**				
Descripción	Contiene la información concerniente a las administraciones zonales del Municipio de Quito.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_administracion	integer	5	Identificador de administración zonal.	Si	PK
nombre_administracion	varchar	150	Campo de nombre de administración zonal.	Si	
desc_administracion	varchar	250	Campo de la descripción de la administración zonal.	No	
estado_administracion	boolean		Campo que indica el estado de la administración zonal.	Si	

- Nota: Esta tabla muestra información sobre la tabla administración.

Tabla 62 Representación de la tabla Usuario

Tabla	**Usuario**				
Descripción	Contiene la información concerniente a los usuarios del sistema.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_usr	integer	5	Identificador de usuario, auto incremental.	Si	PK
id_perfil	integer	5	Identificador de perfil de usuario.	Si	FK
id_administracion	integer	5	Identificador de administración de usuario.	Si	FK
password	varchar	128	Campo de contraseña del usuario.	Si	
last_login	date		Campo que almacena fecha del último acceso al sistema.	No	

is_superuser	boolean		Campo que indica si un usuario tiene privilegios de administrador.	Si	
cedula_usr	integer	10	Campo de cédula de identidad del usuario.	Si	
nombre_usr	varchar	50	Campo de nombre(s) del usuario.	Si	
apellido_usr	varchar	50	Campo de apellido(s) del usuario.	Si	
dirección_usr	varchar	200	Campo de dirección domiciliaria del usuario.	No	
telefono_usr	varchar	15	Campo de teléfono del usuario.	No	
email_usr	varchar	50	Campo de correo electrónico del usuario.	Si	
username	varchar	50	Campo de nombre de usuario asignado.	Si	
is_active	boolean		Campo que indica el estado del usuario.	Si	
is_staff	boolean		Campo que identifica si un usuario es permitido ingresar con permisos de super administrador.	Si	
num_validacion	integer	5	Campo que almacena el número de tareas pendientes de validación de baches.	Si	

- Nota: Esta tabla muestra información sobre la tabla usuario.

Tabla 63 Representación de la tabla Auth_token

Tabla	**Authtoken_token**				
Descripción	Contiene la información concerniente a los tokens generados para cada usuario, los cuales son usados para el acceso a las API's que provee el sistema informático.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
key	varchar	40	Campo donde se almacena el token generado.	Si	PK
id_user	integer	5	Identificador de usuario.	Si	FK
created	date		Campo de fecha de creación de token	Si	

- Nota: Esta tabla muestra información sobre la tabla authtoken_token.

Tabla 64 Representación de la tabla Reporte

Tabla	**Reporte**				
Descripción	Contiene la información concerniente a los registros de baches generados a partir de la aplicación móvil.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_reporte	integer	5	Identificador de reporte.	Si	PK
id_estado	integer	5	Identificador de estado del reporte.	Si	FK
id_usr	integer	5	Identificador del usuario reportador.	Si	FK
id_direccion	integer	5	Identificador de la dirección del acontecimiento.	Si	FK
fecha_reporte	date	150	Campo de fecha cuando se produjo el reporte.	Si	
is_asignado	boolean	250	Campo donde se indica si el reporte ha sido asignado para su validación.	No	

location	geometry (point)		Campo geográfico que almacena el punto (latitud/longitud) de la localización del reporte.	Si	
imagen_reporte	varchar	100	Campo que almacena la ruta del archivo fotográfico asociado con el reporte.	Si	

- Nota: Esta tabla muestra información sobre la tabla reporte.

Tabla 65 Representación de la tabla Estado reporte

Tabla	**Estado reporte**				
Descripción	Contiene la información concerniente a los estados por los que a traviesa el reporte de un bache.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_estado	integer	5	Identificador del estado del reporte.	Si	PK
nombre_est	varchar	200	Campo del nombre del estado.	Si	
descripcion_est	varchar	500	Campo que almacena la descripción del estado.	No	

- Nota: Esta tabla muestra información sobre la tabla estado reporte.

Tabla 66 Representación de la tabla Dirección

Tabla	**Dirección**				
Descripción	Contiene la información concerniente a la dirección donde se suscitó el reporte de un bache.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_direccion	integer	5	Identificador de dirección.	Si	PK
id_barrio	integer	5	Identificador del barrio descrito en la dirección.	Si	FK
calle_principal	varchar	50	Campo donde almacena la calle principal del reporte.	No	
referencia	varchar	250	Campo donde se almacena una referencia asociada a la dirección donde se registró el bache.	No	

- Nota: Esta tabla muestra información sobre la tabla dirección.

Tabla 67 Representación de la tabla Barrio

Tabla	**Barrio**				
Descripción	Contiene la información concerniente a los múltiples barrios de la ciudad de Quito.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_barrio	integer	5	Identificador del barrio.	Si	PK
id_parroquia	integer	5	Identificador de la parroquia a la que pertenece el barrio.	Si	FK
nombre_barrio	varchar	200	Campo del nombre del barrio.	Si	
geom	geometry (polygon)		Campo geográfico que almacena el polígono del barrio.	Si	

- Nota: Esta tabla muestra información sobre la tabla barrio.

Tabla 68 Representación de la tabla Parroquia

Tabla	**Parroquia**				
Descripción	Contiene la información concerniente a las parroquias en las que está dividida la ciudad de Quito.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**

id_parroquia	integer	5	Identificador de la parroquia.	Si	PK
id_administracion	integer	5	Identificador de la administración zonal a la que pertenece la parroquia.	Si	FK
nombre_parroquia	varchar	150	Campo del nombre de la parroquia.	Si	
desc_parroquia	varchar	250	Campo que almacena una breve descripción de la parroquia.	No	
estado_parroquia	boolean		Campo que indica el estado de la parroquia	Si	

- Nota: Esta tabla muestra información sobre la tabla parroquia.

Tabla 69 Representación de la tabla Validación

Tabla	**Validación**				
Descripción	Contiene la información concerniente a la asignación de validación de un bache.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK-FK**
id_validacion	integer	5	Identificador de validación.	Si	PK
id_reporte	integer	5	Identificador del reporte asociado para la validación.	Si	FK
id_usr	integer	5	Identificador del usuario validador.	Si	FK
estado_validacion	boolean		Campo que indica el estado de la validación.	Si	
fecha_validación	date		Campo de fecha cuando se produjo la validación.	No	
fecha_asignacion	date		Campo de fecha cuando se asignó el bache para su validación.	No	
imagen_reporte	varchar	100	Campo que almacena la ruta del archivo fotográfico asociado con la validación del bache.	Si	

- Nota: Esta tabla muestra información sobre la tabla validación.

3.1.2.2 Base de datos aplicación móvil

Diagrama físico de la base de datos aplicación Web.

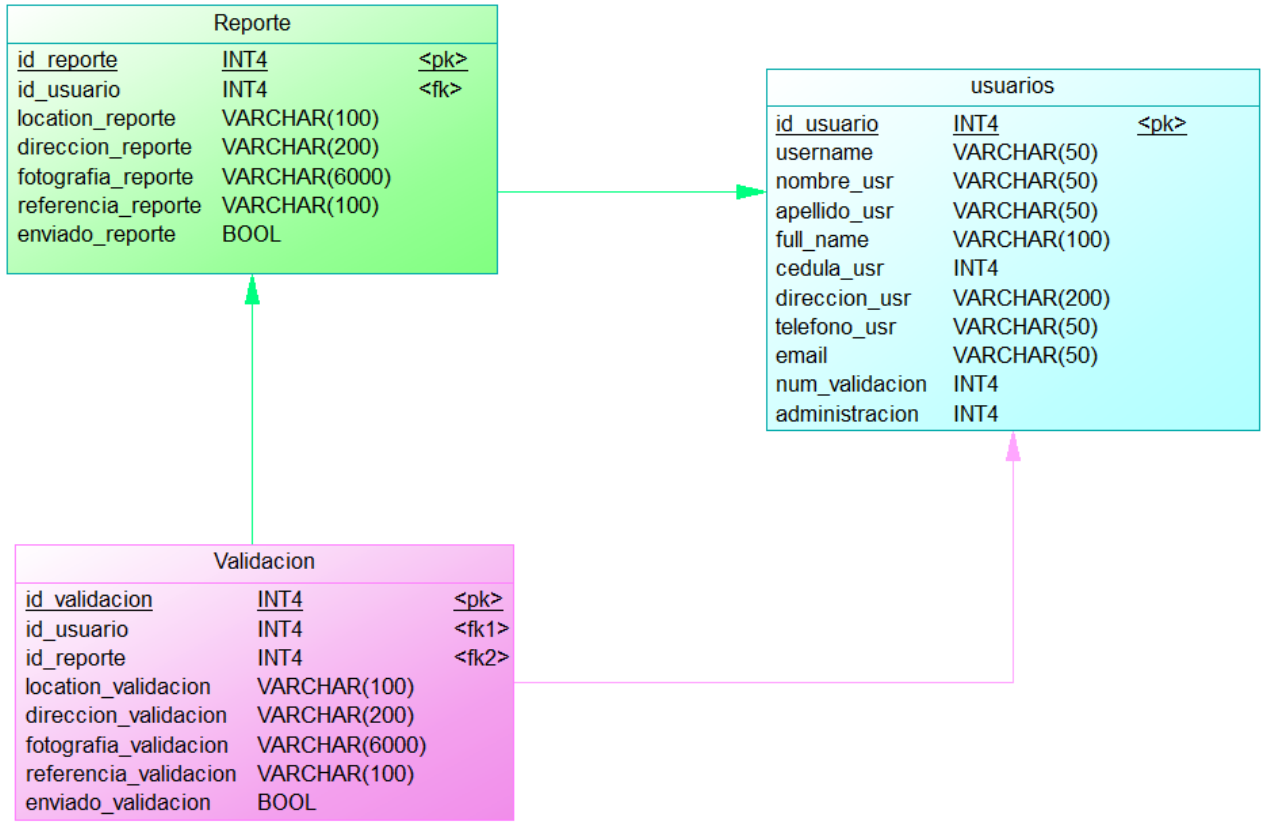

Figura 27. Modelo físico base de datos aplicación Móvil.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Diccionario de datos aplicación móvil**

El siguiente apartado señala una breve descripción de cada una de las tablas de la base de datos, al mismo tiempo se especifican los datos que utiliza la aplicación móvil mostrados en el siguiente esquema:

Tabla 70 Representación de la tabla Usuario

Tabla	**Usuario**				
Descripción	Contiene la información referente a los usuarios de tipo moderador que accederán a la aplicación móvil.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_usr	integer	4	Identificador de usuario, auto incremental.	Si	PK
username	varchar	50	Campo de nombre de usuario asignado.	Si	
nombre_usr	varchar	50	Campo de nombre(s) del usuario.	Si	
apellido_usr	varchar	50	Campo de apellido(s) del usuario.	Si	
fullname	varchar	100	Campo de nombre y apellido del usuario	Si	
cedula_usr	integer	10	Campo de cédula de identidad del usuario.	Si	
dirección_usr	varchar	200	Campo de dirección domiciliaria del usuario.	No	
telefono_usr	varchar	15	Campo de teléfono del usuario.	No	
email	varchar	50	Campo de correo electrónico del usuario.	No	
num_validacion	integer	4	Campo que almacena el número de tareas pendientes de validación de baches.	Si	
administracion	integer	4	Campo que contiene el ID de administración a la que pertenece el usuario	No	

- Nota: Esta tabla muestra información sobre la tabla usuario

Tabla 71 Representación de la tabla Reporte

Tabla	**Reporte**				
Descripción	Contiene la información concerniente a los registros de baches generados a partir de la aplicación móvil y que serán enviados a la aplicación Web				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_reporte	integer	4	Identificador de reporte.	Si	PK
id_usuario	integer	4	Identificador del usuario reportador.	Si	FK
location_reporte	varchar	100	Campo geográfico que almacena el punto (latitud/longitud) de la localización del reporte.	Si	
Dirección_reporte	varchar	200	Campo que almacena la calle principal del reporte	Si	
fotografia_reporte	varchar	6000	Campo que almacena el archivo fotográfico convertido a String en formato BASE64 del reporte	Si	

referencia_reporte	varchar	100	Campo que almacena una referencia al sitio del reporte	Si	
enviado_validacion	Boolean		Campo que almacena valores True o False, para verificar si el reporte ha sido enviado o no.	Si	

- Nota: Esta tabla muestra información sobre la tabla reporte

Tabla 72 Representación de la tabla Validación

Tabla	**Validación**				
Descripción	Contiene la información concerniente a la validación de un bache reparado.				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Requerido**	**Clave PK- FK**
id_validacion	integer	4	Identificador de validación.	Si	PK
id_usuario	integer	4	Identificador del usuario que realiza el proceso de validación.	Si	FK
id_reporte	integer	4	Identificador del reporte asociado para la validación.	Si	FK
location_validacion	varchar	100	Campo geográfico que almacena el punto (latitud/longitud) de la localización de la validación.	Si	
Dirección_validacion	varchar	200	Campo que almacena la calle principal de la validación	Si	
fotografia_ validacion	varchar	6000	Campo que almacena el archivo fotográfico convertido a String en formato BASE64 de la validación	Si	
referencia_ validacion	varchar	100	Campo que almacena una referencia al sitio de validación	Si	
enviado_validacion	Boolean		Campo que almacena valores True o False, para verificar si la validación ha sido enviada o no.	Si	

- Nota: Esta tabla muestra información sobre la tabla validación

3.1.3 Código

En esta sección se expone y relata el código fuente elaborado para el cumplimiento de las funcionalidades del sistema. Se escogió porciones de código que resultan esenciales tanto de la aplicación Web como en la móvil.

3.1.3.1 Aplicación Web

Este apartado se encuentra divido en los elementos relevantes que deben ser programados para la utilización del framework Django.

- **Settings.py**

Es el archivo de configuración de Django se puede parametrizar todo lo concerniente al proyecto, como conexión a base de datos, rutas estáticas de archivos, o inicialización de complementos adicionales, todo esto a base de definición de variables o diccionarios de Python. En figura No. 28 se exhibe la configuración de base de datos empleada en fase de desarrollo.

Archivo de configuración Django.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.contrib.gis.db.backends.postgis',
    'NAME': 'GeoBache',
    'USER': 'postgres',
    'PASSWORD': '1992',
    'HOST': 'localhost',
    'PORT': 5432,
  }
}
```

Figura 28. Parámetros de configuración conexión base de datos.
Elaborado por: Cristhian Castro y Paul Velasco.

- **Models.py**

Como se ha mencionado en Django se escribe modelos los cuales describen a un objeto, en el que se definen la estructura de los datos que se almacenarán, puede contener tipos de campo y atributos como tamaño, o valores nulos, todo esto independiente del sistema gestor de base de datos con el que se trabaje.

Para ejemplificar se presenta el modelo realizado para el almacenamiento de un usuario dentro de la aplicación.

Archivo de modelos de Django.

```
class Usuario(AbstractBaseUser, PermissionsMixin):
    perfil=models.ForeignKey(Perfil, on_delete=models.CASCADE)
    administracion=models.ForeignKey(Administracion,
    on_delete=models.CASCADE)
    cedula_usr = models.CharField(max_length=10, blank=False)
    nombre_usr = models.CharField(max_length=100, blank=False)
    apellido_usr = models.CharField(max_length=100, blank=False)
    direccion_usr = models.TextField(max_length=300, blank=True)
    telefono_usr = models.CharField(max_length=10, blank=True)
    email = models.EmailField(max_length=100, blank=True)
    username =  models.CharField(max_length=50, blank=False, unique=True)
    is_active = models.BooleanField(default=True, verbose_name="Estado")
    is_staff = models.BooleanField(default=False)
    num_validacion = models.IntegerField(default=0)

    USERNAME_FIELD = 'username'
```

Figura 29. Modelo Usuario Django
Elaborado por: Cristhian Castro y Paul Velasco.

Django de manera nativa provee un modelo con atributos básicos de Usuario, el cual ha sido modificado en el caso de este proyecto, con el fin de aumentar campos a dicho modelo, por esta razón se hace una extensión de "AbstractBaseUser".

Las relaciones definidas que surgen se las efectúa haciendo una asociación entre los modelos relacionados. Estas representaciones pueden ser:

Varios a varios "ManyToManyField", varios a uno " ForeignKey", uno a uno " OneToOneField".

- **Forms.py**

Como parte de la organización de Django es posible definir formularios a partir de los modelos generados, los cuales son implementados en la capa de presentación del usuario. El siguiente es el formulario desarrollado para la creación, y edición de una administración zonal.

Archivo de formularios Django.

```
class AdministracionForm(forms.ModelForm):
  class Meta:
        model = Administracion
        fields = [
      'nombre_administracion','direccion_administracion',
      'estado_administracion',
    ]
        labels = {
      'nombre_administracion': 'Administración Zonal',
      'direccion_administracion': 'Dirección',
      'estado_administracion': 'Estado',
    }
        widgets = {
      'nombre_administracion':forms.TextInput(attrs={
                  'class':'form-control col-md-7 col-xs-12',
                  'placeholder':'Ingresa el nombre de la administración
                  zonal','required':'true'
            }),     'direccion_administracion':forms.TextInput(attrs={
                  'class':'form-control col-md-7 col-xs-12',
                  'placeholder':'Ingresa la dirección','required':'true'
            }),
      'estado_administracion':forms.Select(attrs={
                  'class':'form-control col-md-7 col-xs-12',
                  'required':'true'
            }),
    }
```

Figura 30. Formulario del modelo administración
Elaborado por: Cristhian Castro y Paul Velasco.

En esta clase se precisa el modelo al que pertenece el formulario, los campos que serán visualizados, sus etiquetas, y se establece los atributos de los campos de un formulario de igual forma como se lo hace en un documento HTML.

- **Views.py**

Las vistas dentro de Django son usadas para encapsular la lógica del negocio, aquí es donde se programan los métodos relacionados a las funcionalidades que el sistema debe realizar.

A fin de ejemplificar y detallar las funciones relacionadas con un modelo se plantea explicar todo lo concerniente a un proceso CRUD (Create, Read, Update, Delete).

Siguiendo el enfoque planteado se muestra mediante la figura No 30 el método usado para la lectura de los registros almacenados en base de datos.

Archivo de funciones Django

```
def AdminListar(request):
  Administraciones=Administracion
            .objects.all().exclude(id=0).order_by("id")
  return render(request, 'administrador/config/config_index.html', {
            'user': request.user,'Administraciones':Administraciones
            })
```

Figura 31. Método de lectura de administraciones zonales
Elaborado por: Cristhian Castro y Paul Velasco.

Básicamente lo que se hace es hacer una consulta al modelo "Administración" por medio del método "objects.all()", el resultado de la consulta es enviado a la plantilla correspondiente para su lectura y visualización.

A continuación, se presenta el método para la creación de una administración zonal.

Archivo de funciones Django

```
def AdminCrear(request):
  data=dict()
  if request.method == 'POST':
        form = AdministracionForm(request.POST)
    if form.is_valid():
            form.save()
            data['form_is_valid']=True
            Administraciones=Administracion
                  .objects.all().exclude(id=0).order_by("id")
            data['html_user_list'] = render_to_string(
                  'administrador/config/config_list.html',{
                  'Administraciones':Administraciones,
                  'user': request.user
            })
    else:
            data['form_is_valid']=False
  else:
        form = AdministracionForm()
    context = {'form':form}
    data ['html_form'] = render_to_string(
            'administrador/config/config_form_crear.html',
            context, request=request)
  return JsonResponse (data)
```

Figura 32. Método de creación de administración zonal.
Elaborado por: Cristhian Castro y Paul Velasco.

El funcionamiento que engloba el método expuesto anteriormente hace uso de una petición en caso de ser POST la función interpreta que se trata de una solicitud para almacenamiento de una nueva administración zonal por lo que parametriza el formulario del modelo administración y utiliza el método "save()" para guardar la información enviada a través del formulario. En caso de ser el tipo de petición GET inicializa el formulario del modelo y es enviado al archivo HTML correspondiente el que lo renderiza y muestra al usuario.

Otro de las funciones comunes dentro del proyecto es la de editar un registro almacenado en la base de datos para ello se hace uso de la siguiente función.

Archivo de funciones Django

```
def AdminEditar(request, pk):
  data=dict()
    administracion = get_object_or_404(Administracion, pk=pk)
  if request.method == 'POST':
        form = AdministracionForm(request.POST, instance=administracion)
    if form.is_valid():
            form.save()
            data['form_is_valid']=True
           Administraciones=Administracion
                  .objects.all()
                  .exclude(id=0)
                  .order_by("id")
            data['html_user_list'] = render_to_string(
                  'administrador/config/config_list.html',{
           'Administraciones':Administraciones,
           'user': request.user
      })
    else:
            data['form_is_valid']=False
  else:
        form = AdministracionForm(instance=administracion)
        context = {'form':form}
        data ['html_form'] = render_to_string(
            'administrador/config/config_form_editar.html',
            context,
            request=request
    )
  return JsonResponse (data)
```

Figura 33. Método de edición de una administración zonal.
Elaborado por: Cristhian Castro y Paul Velasco.

Contrario a la función que permite crear una administración zonal, esta requiere que sea enviado como parámetro el id perteneciente al objeto que se desea editar el cual es buscado dentro de los registros almacenados en base de datos. A partir de aquí la funcionalidad es similar al método de creación, se evalúa el tipo de petición si es POST interpreta que se va a modificar algún campo por lo que se utiliza el formulario del modelo el que en caso de ser correcto es guardado usando el método "save()", si es otro tipo de petición en este caso GET se inicializa el formulario del modelo instanciado con los datos de la administración zonal perteneciente al id que es enviado por el usuario.

Para la complementación del CRUD del modelo esta la función de eliminar un objeto el cual es construido de la siguiente manera.

Archivo de funciones Django

```
def AdminEliminar(request, pk):
  data=dict()
    administracion = get_object_or_404(Administracion, pk=pk)
  if request.method == 'POST':
        administracion.delete()
        data['form_is_valid']=True
       Administraciones=Administracion
            .objects.all()
            .exclude(id=0)
            .order_by("id")
        data['html_user_list'] = render_to_string(
            'administrador/config/config_list.html',{
      'Administraciones':Administraciones,
      'user': request.user
    })
  else:
        context = {'administracion':administracion}
        data ['html_form'] = render_to_string(
            'administrador/config/config_form_eliminar.html',
            context,
            request=request
    )
  return JsonResponse (data)
```

Figura 34. Método de edición de una administración zonal.
Elaborado por: Cristhian Castro y Paul Velasco.

Su funcionamiento requiere que como parámetro se envíe el id de la administración a eliminarse, a partir de ello busca el registro en la base de datos, y en el caso de ser una petición POST el registro es eliminado mediante el método “delete()”.

- **Serializers.py**

Los “serializers” permiten que los datos complejos, como los conjuntos de consultas a los modelos, se puedan procesar fácilmente en JSON o XML.

Por ello Django Rest Framework los utiliza para la construcción de API’s Rest. Su definición requiere hacer uso de los modelos Django, como se puede apreciar en la figura No 34, en la cual se expone el “serializer” del modelo reporte el que se emplea para el funcionamiento del API para reportar un bache.

Archivo serializers Django Rest Framework

```
class RegistroSerializer(ModelSerializer):
  direccion = DireccionSerializer(),fotografia = FotografiaSerializer()
  class Meta:
    model=Reporte
  def create (self,validated_data):
    direccion_data = validated_data.pop('direccion')
        fotografia_data = validated_data.pop('fotografia')
        fotografia = Fotografia.objects.create(**fotografia_data)
    point_data = validated_data.pop('location')
        barrio = Barrio.objects.get(geom__contains=point_data)
    direccion = Direccion
            .objects.create(barrio_id=barrio.id,**direccion_data)
        administracion = Barrio
            .objects.select_related('parroquia')
            .values('parroquia_id__administracion_id')
            .filter(pk=barrio.id)
        fecha = timezone.now().date()
    reporte = Reporte.objects.create(
            direccion_id=direccion.id,fecha_reporte=fecha,
            estado_id=1,location=point_data,
            administracion=administracion[0]
                  ['parroquia_id__administracion_id'],
            is_asignado=False,fotografia_id=fotografia.id,
                  **validated_data)
    return reporte
```

Figura 35. Método “serializer” del modelo reporte.
Elaborado por: Cristhian Castro y Paul Velasco.

En este método se debe precisar el modelo al que este asociado en el caso de la figura No 34 es reporte. Al ser este un modelo que tiene relacionados otros modelos como dirección y fotografía se puede hacer mención de los "serializers" propios de aquellos modelos. Para el caso su uso esta empleada en la inserción de datos por lo cual se escribe el sub-método "create", básicamente es el que obtiene los datos enviados a través de la petición los divide y crea el nuevo objeto que será almacenado en la base de datos.

Por último, es importante considerar la función API especificada en las vistas de Django, que propiamente es quien maneja las solicitudes enviadas por el cliente.

Archivo de funciones Django.

```
@api_view(['POST'])
@permission_classes((IsAuthenticated, ))
def RegistroAPI(request):
  if request.method == 'POST':
        serializer = RegistroSerializer(data=request.data)
    if serializer.is_valid():
          serializer.save()
      return Response(
                {"success": "Registro almacenado correctamente"},
                status=status.HTTP_201_CREATED)
    else:
      return Response(status=status.HTTP_400_BAD_REQUEST)
```

Figura 36. Método API para el registro de un bache.
Elaborado por: Cristhian Castro y Paul Velasco

Es necesario mediante el decorador "@api_view" definir los métodos que son permitidos en el API, para el caso solo POST, de igual forma "@permission_classes" indica que para el acceso al API debes estar autenticado. Ya para la funcionalidad mismo, se hace una llamada al "serializer" realizado con en la figura No 34 enviando como parámetro los datos que vienen en la petición. En caso de ser válido los datos se implementa el método "save()" para guardar el nuevo registro en la base de datos, y la respuesta enviada al cliente es el estado 201, caso contrario si el "serializer" no es válido se devuelve el estado 400.

3.1.3.2 Aplicación móvil

En este apartado se detalla el código fuente que se utilizó en la aplicación móvil, explicando las configuraciones esenciales y código más importante para el funcionamiento de la aplicación móvil.

3.1.3.2.1 Archivo de Configuración

Las configuraciones en toda aplicación son esenciales, debido a que una correcta configuración comprometida hacia las necesidades del desarrollador permitirá el funcionamiento óptimo de la aplicación con el sistema Android.

- **AndroidManifest.xml**

Es el archivo de configuración de la aplicación móvil, donde se cita al activity principal, también se definen los activities secundarios de la aplicación y se establecen permisos a los que la aplicación puede acceder sobre el dispositivo móvil, tales como acceso a internet, uso del GPS, disposición de almacenamiento interno y acceso a la cámara del dispositivo, estos permisos y la invocación a la pantalla principal se los puede apreciar en la figura No.37.

Permisos a los que la aplicación puede acceder sobre un dispositivo móvil

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
       package="com.example.paul.baches4">
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
<uses-permission android:name=
       "android.permission.AUTHENTICATE_ACCOUNTS"/>
<uses-permission android:name="android.permission.USE_CREDENTIALS"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name=
       "android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name=
       "android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name=
       "android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name=
       "android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name=
       "android.permission.LOCATION_HARDWARE"/>
<uses-permission android:name=
```

```
        "android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.CAMERA"/>
```

Invocación al activity principal y activities secundarios de la aplicación móvil

```
<application android:allowBackup="true"
   android:icon="@mipmap/ic_launcher"
   android:label="@string/app_name"
   android:roundIcon="@mipmap/ic_launcher_round"
   android:supportsRtl="true"
   android:theme="@style/AppTheme">
   <activity android:name=".Interfaces.Login"android:noHistory="true">
       <intent-filter>
          <action android:name="android.intent.action.MAIN"/>
          <category android:name="android.intent.category.LAUNCHER"/>
       </intent-filter>
   </activity>
   <activity android:name=".Interfaces.Principal"
       android:label="@string/title_activity_principal"
       android:theme="@style/hola" />
```

Figura 37. Parámetros de configuración y permisos para acceder a las interfaces
Elaborado por: Cristhian Castro y Paul Velasco.

3.1.3.2.2 Módulo Inicio de Sesión

Este módulo permite al usuario moderador iniciar sesión, conectándose al API de la aplicación Web, mediante el ingreso de su usuario, contraseña y un token de seguridad generado por el sistema; la primera vez el usuario ingresa sus credenciales donde la aplicación móvil almacena de forma temporal la información del usuario para el inicio y cierre de sesión dentro de una pequeña base de datos creada en SQLite.

- Usuario.java

Establecido en el patrón de diseño MVC, se crea el modelo de datos del usuario, el cual contiene los getters and setters para establecer y obtener la información de un usuario.

Código para establecer y obtener la información del usuario

```
public class Usuario {
   @SerializedName("id")
   private int id;
   @SerializedName("username")
   private String username;
   @SerializedName("nombre_usr")
   private String nombre_usr;
   @SerializedName("apellido_usr")
   private String apelllido_usr;
   @SerializedName("full_name")
```

```
    private String full_name;
    @SerializedName("cedula_usr")
    private String cedula_usr;
    @SerializedName("direccion_usr")
    private String direccion_usr;
    @SerializedName("telefono_usr")
    private String telefono_usr;
    @SerializedName("email")
    private String email;
    @SerializedName("num_validacion")
    private int num_validacion;
    @SerializedName("administracion")
    private int administracion;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
```

Figura 38. Clase Usuario.java establece y obtiene información del usuario
Elaborado por: Cristhian Castro y Paul Velasco.

- Login.java

El código que se muestra a continuación se conecta al API para obtener las credenciales del usuario y almacenarlas en la base de datos SQLite, en caso de que las credenciales ingresadas no sean las correctas el sistema emitirá un mensaje de error, para que el usuario vuelva a ingresar sus credenciales.

Código para obtener y almacenar las credenciales del usuario

```
private void getUser(final String token,final String username){
    Call<List<Usuario>> call = JsonAPI.getAllPosts(token,username);
    call.enqueue(new Callback<List<Usuario>>() {
        @Override
        public void onResponse(Call<List<Usuario>> call,
                Response<List<Usuario>> response){
            if(response.isSuccessful()) {
                List<Usuario> postsList = response.body();
                mPosts = new ArrayList<Usuario>();
                for (Usuario post : postsList) {
                    mPosts.add(post);
                    preferences.edit().putString("id_admin", ""
                         +post.getAdministracion()).commit();
                    preferences.edit().putString("user_id", ""
                         +post.getId()).commit();
                }
                conn.insertar(mPosts);
                Intent ListSong = new Intent(Login.this, Principal.class);
                startActivity(ListSong);
            }
        }
```

```
        @Override
        public void onFailure(Call<List<Usuario>> call, Throwable t){
            Toast temp = Toast.makeText(Login.this,
                 t.getMessage(), Toast.LENGTH_SHORT);
            temp.show();
        }
    });
}
```

Figura 39. Clase Login.java, método getUser obtiene credenciales de usuario
Elaborado por: Cristhian Castro y Paul Velasco.

- SQLite.java

Esta clase muestra el almacenamiento de información en la base de datos SQLite, esta fue previamente obtenida en la clase Login.java; adicional a ello se creó un método adicional para verificar la existencia de un usuario y de cumplirse esta afirmación, el sistema no volverá a almacenar la información de este usuario.

Código para almacenar información dentro de la base de datos SQLite

```
public void insertar(ArrayList<Usuario> c){
   db =this.getWritableDatabase();
   ContentValues values =new ContentValues();
   if (verificar_registro(c) ==false){
       values.put(columna_id, c.get(0).getId());
       values.put(columna_username, c.get(0).getUsername());
       values.put(columna_nombre, c.get(0).getNombre_usr());
       values.put(columna_apellido, c.get(0).getApelllido_usr());
       values.put(columna_full_name, c.get(0).getFull_name());
       values.put(columna_cedula, c.get(0).getCedula_usr());
       values.put(columna_direccion, c.get(0).getDireccion_usr());
       values.put(columna_telefono, c.get(0).getTelefono_usr());
       values.put(columna_email, c.get(0).getEmail());
       values.put(columna_num_validacion, c.get(0).getNum_validacion());
       values.put(columna_administracion, c.get(0).getAdministracion());
       db.insert(nombre_tabla,null,values);
       db.close();
   }
}
```

Figura 40. Clase SQLite.java, método insertar, guarda datos del usuario.
Elaborado por: Cristhian Castro y Paul Velasco.

3.1.3.2.3 Módulo de Ubicación

En este apartado, se examinarán las secciones de código más relevantes, que utiliza la aplicación para obtener la ubicación tiempo real del dispositivo móvil, a través de la ubicación se obtiene la dirección de la calle donde se encuentra el dispositivo móvil.

- Mapa_fragment.java

Esta clase permite al usuario obtener la ubicación geográfica mediante las coordenadas de latitud y longitud provistas por el GPS, y mostrar su ubicación en un mapa interactivo.

Código para obtener la ubicación en tiempo real

```
public void miUbicacion(){
       if (ActivityCompat.checkSelfPermission(getActivity(),
        Manifest.permission.ACCESS_FINE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED &&
        ActivityCompat.checkSelfPermission(getActivity(),
        Manifest.permission.ACCESS_COARSE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED){
               return;
        }
        LocationManager locationManager = (LocationManager)
        getActivity().getSystemService(Context.LOCATION_SERVICE);
        Location location = locationManager
               .getLastKnownLocation(LocationManager.GPS_PROVIDER);
        actualizarUbicacion(location);
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVID
               ER,1200,0,loclistener);
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVID
               ER,1200,0,loclistener);
}
```

Código para mostrar la calle a partir de sus coordenadas de latitud y longitud.

```
public void setLocation() {
   LatLng coordenadas = new LatLng(lat, lng);
   if (lat != 0.0 && lng != 0.0) {
       try {
           List<Address> list = geocoder.getFromLocation(lat, lng, 1);
           if (!list.isEmpty()) {
               Address DirCalle = list.get(0);
               String temporal;
               temporal = ""+DirCalle.getAddressLine(0).toString();
               txt.setText(temporal.substring(0, temporal.indexOf(",")));
           }
       } catch (IOException e) {
           e.printStackTrace();
       }
   }
}
```

Figura 41. Parámetros de para mostrar la ubicación en tiempo real.
Elaborado por: Cristhian Castro y Paul Velasco.

3.1.3.2.4 Módulo de Reportar

Este módulo es considerado el más importante debido a que contiene el código donde el usuario puede enviar reportes de baches existentes en las vías de la ciudad de Quito.

- **Reporte_fragment.java**

Esta clase posee los métodos para obtener la ubicación del dispositivo, mostrar la dirección de la calle en la que se encuentra, tomar una fotografía y agregar una referencia al sitio del incidente, en formato JSON, todo esto con el fin reportar este tipo de incidencias y enviar los datos a la aplicación Web.

Código para enviar la fotografía, referencia, ubicación, dirección y el identificador del usuario en formato JSON

```
enviar.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       Reportar reporte = new Reportar();
       Fotografia foto = new Fotografia();
       Direccion dir = new Direccion();

       foto.setImagen(convertirImgString());
       dir.setCalle_principal(tcallep.getText().toString().trim());
       dir.setReferencia(treferencia.getText().toString().trim());
       reporte.setLocation("POINT("
                +mapa_dos_fragment.lng+" "+mapa_dos_fragment.lat+")");
       reporte.setFotografia(foto);
       reporte.setDireccion(dir);
       reporte.setUsuario(preferences.getString("user_id",""));
       sendPost(reporte);

   }
});
```

Código para convertir imágenes en formato Base64 y almacenar la imagen en un String

```
private String convertirImgString(){
   ByteArrayOutputStream array = new ByteArrayOutputStream();
   bitmap.compress(Bitmap.CompressFormat.JPEG,50,array);
   byte [] imagenByte= array.toByteArray();
   String imagenString = Base64.encodeToString(
                                imagenByte,Base64.DEFAULT);
   return imagenString;
}
```

Figura 42. Parámetros de enviar un reporte
Elaborado por: Cristhian Castro y Paul Velasco.

3.1.3.2.5 Módulo de Validar

Este módulo, contiene el código que permite a la aplicación indicar al usuario donde se encuentra el bache y el usuario debe dirigirse al sitio del incidente y verificar si este ha sido rellenado o no.

- **Validar_fragment.java**

Con el fin de constatar el trabajo de relleno de baches esta clase posee los métodos para adjuntar un registro fotográfico, referencia, coordenadas geográficas de ubicación y el identificador del usuario que realiza el proceso de validación, donde esta información se envía hacia la aplicación Web en formato JSON.

Código para adjuntar el registro fotográfico desde la galería del teléfono, o tomar la fotografía directamente desde la aplicación.

```
@Override
public void onActivityResult(int requestCode,int resultCode,Intent data){
   super.onActivityResult(requestCode, resultCode, data);
   if (resultCode == RESULT_OK ){
       switch (requestCode){
           case COD_SELECCIONADO:
               Uri miPath = data.getData();
               imagen.setImageURI(miPath);
               try{
                   bitmap= MediaStore.Images.Media.getBitmap(
                        getActivity().getApplicationContext()
                        .getContentResolver(), miPath);
                   imagen.setImageBitmap(bitmap);
               }catch (IOException e){
                   e.printStackTrace();
               }
               break;
           case  COD_TOMAR:
MediaScannerConnection.scanFile(getActivity().getApplicationContext(),
new String[]{path}, null,
           new MediaScannerConnection.OnScanCompletedListener() {
                   @Override
                   public void onScanCompleted(String s, Uri uri){
                        Log.i("Ruta de almacenamiento","Path: "+path);
                   }
           });
           bitmap = BitmapFactory.decodeFile(path);
           imagen.setImageBitmap(bitmap);
           break;
       }
   }
}
```

Figura 43 Método onActivityResult, permite cargar y mostrar imágenes
Elaborado por: Cristhian Castro y Paul Velasco.

Código para redimensionar el tamaño de la imagen, al cambiar el tamaño de la imagen esta se enviará de forma más vertiginosa hacia la aplicación Web.

```
Private Bitmap redimensionarImagen(Bitmap bitmap, float anchoNuevo, float
alto Nuevo){
       int ancho = bitmap.getWidth();
       int alto = bitmap.getHeight();
       if (ancho > anchoNuevo || alto > altoNuevo){
```

```
        float escalaAncho = anchoNuevo/ancho;
        float escalaAlto = altoNuevo/alto;
        Matrix matrix = new Matrix();
        Matrix.postScale(escalaAncho, escalaAlto);
        Return Bitmap.createBitmap(bitmap,0,0,ancho,alto,matrix,false);
        }else{
        Return bitmap;
        }
}
```

Figura 44 Método redimensionarImagen, cambiar el ancho y alto de la imagen.
Elaborado por: Cristhian Castro y Paul Velasco.

3.2 Pruebas

En esta sección se evidencia los resultados y análisis, referentes a las pruebas a las que fue sometido el sistema, con el objetivo de conocer la capacidad del mismo, cumplimiento de funcionalidades, y desempeño. Se determinó la realización de cuatro tipos de pruebas: unitarias, carga, estrés, y contra requerimientos.

3.2.1 Pruebas unitarias

Para la realización de las pruebas unitarias se tomaron en cuenta los métodos principales que el sistema maneja, obteniendo el tiempo de ejecución de cada uno de ellos. En este conjunto de pruebas se utilizó el módulo "UnitTest" provisto por Django. Por la similitud de funcionamiento en los métodos implementados para cada modelo de la aplicación, solo se presentan los resultados de la evaluación de las funciones CRUD del modelo usuario.

Tabla 73 Ejecución de pruebas unitarias

No.	Método	Entrada	Tiempo	Env. (bytes)	Resp
1	UsuarioIndex (Listar)	No aplica	13 ms	No aplica	OK
2	UsuarioCrear	Nombre_usr Apellido_usr Email_usr Cedula_usr Id_administracion Id_perfil	18 ms	333	OK
3	UsuarioEditar	Id_usuario	5 ms	101	OK
4	UsuarioEliminar	Id_usuario	9 ms	101	OK

- Nota: Esta tabla muestra los resultados arrojados por los métodos CRUD del modelo usuario.

Adicional el modelo usuario presenta métodos complementarios que también fueron sometidos a pruebas.

Tabla 74 Ejecución de pruebas unitarias

No.	Método	Entrada	Tiempo	Env. (bytes)	Resp
1	UsuarioActivar	Id_usuario	5 ms	101	OK
2	UsuarioReestablecerPass	Id_usuario	5 ms	101	OK
3	GenerarUserNames	Nombre_usr Apellido_usr	7 ms	214	OK

- Nota: Esta tabla muestra los resultados arrojados por métodos complementarios del modelo usuario.

Asimismo, la siguiente tabla detalla los resultados obtenidos en las pruebas unitarias de métodos concernientes a la administración de registros de baches en el sistema.

Tabla 75 Ejecución de pruebas unitarias

No.	Método	Entrada	Tiempo	Env. (bytes)	Resp
1	AsignacionReporte	Id_reporte	5 ms	101	OK
2	GetPoints	Id_administracion	20 ms	101	OK
3	CargarMapa	No aplica	15 ms	No aplica	OK

- Nota: Esta tabla muestra los resultados arrojados por métodos del modelo reporte.

3.2.2 Pruebas de estrés

Para la ejecución de esta prueba se envían múltiples solicitudes al API de registro de baches para evaluar su rendimiento. Como herramienta se hizo uso de Apache JMeter la que trabaja independientemente del lenguaje de programación y diseñada para este tipo de experimentos.

- **Escenario 1**

Escenario en el que se simula el reporte concurrente de 50 baches en 1 segundo por parte de los moderadores del Municipio de Quito.

Para esto se hizo una petición POST al API, enviando en el cuerpo de la solicitud un archivo JSON con los datos del bache a reportar.

Resultado

Tabla 76 Ejecución de prueba de estrés escenario 1 (Primera Parte)

Etiqueta	# Muestras	Media	Min	Max	% Error
HTTP Request	50	303 ms	80 ms	494 ms	0.00%

- Nota: Esta tabla muestra los resultados arrojados por la simulación del escenario 1.

En la tabla No. 76 se aprecia que el tiempo máximo de respuesta a una petición fue de 0.494 segundos, y en promedio tomó un tiempo de 0.303 segundos hacer el reporte de un bache.

Tabla 77 Ejecución de prueba de estrés escenario 1 (Segunda Parte)

# Muestras	Latencia	T. conexión	Envío Bytes	Resp Bytes
50	301.4 ms	0.5 ms	175087 bytes	369 bytes

- Nota: Esta tabla muestra los resultados arrojados por la simulación del escenario 1.

Cada petición al servidor envío datos con un peso de 175087 bytes (175.08 Kbytes), conteniendo una fotografía de dimensiones 800 x 600 px con un tamaño aproximado de 128 Kbytes, a su vez la respuesta fue de 369 bytes.

Con estas condiciones el servidor tuvo un promedio de latencia en 0.3014 segundos, cuyo valor es medido justo antes de enviar la solicitud hasta justo después de que se recibió la primera respuesta. El tiempo que llevó establecer la conexión en promedio fue de 0.0005 segundos.

- **Escenario 2**

Escenario en el que se simula el reporte concurrente de 100 baches en 1 segundo por parte de los moderadores del Municipio de Quito.

Para esto se hizo una petición POST al API, enviando en el cuerpo de la solicitud un archivo JSON con los datos del bache a reportar.

Resultado

Los resultados arrojados para esta primera prueba se resumen en la siguiente tabla:

Tabla 78 Ejecución de prueba de estrés escenario 2 (Primera Parte)

Etiqueta	# Muestras	Media	Min	Max	% Error
HTTP Request	100	1689 ms	310 ms	2226 ms	0.00%

- Nota: Esta tabla muestra los resultados arrojados por la simulación del escenario 2.

En la tabla No. 78 se aprecia que el tiempo máximo de respuesta a una petición fue de 2.2 segundos, y en promedio tomó un tiempo de 1.6 segundos hacer el reporte de un bache.

Tabla 79 Ejecución de prueba de estrés escenario 2 (Segunda Parte)

# Muestras	Latencia	T. conexión	Envío Bytes	Resp Bytes
100	1666,84 ms	0.95 ms	175087 bytes	369 bytes

- Nota: Esta tabla muestra los resultados arrojados por la simulación del escenario 2.

Cada petición al servidor envío datos con un peso de 175087 bytes (175.08 Kbytes), conteniendo una fotografía de dimensiones 800 x 600 px con un tamaño aproximado de 128 Kbytes, a su vez la respuesta fue de 369 bytes.

Con estas condiciones el servidor tuvo un promedio de latencia en 1.66 segundos, cuyo valor es medido justo antes de enviar la solicitud hasta justo después de que se recibió la primera respuesta. El tiempo que llevó establecer la conexión en promedio fue de 0.00095 segundos.

Análisis de resultados

Tomando en cuenta los dos escenarios evaluados el máximo flujo evaluado fue de 100 reportes concurrentes en 1 segundo, lo que resulta ser aceptable y se puede concluir que el sistema maneja adecuadamente aquella cantidad de peticiones. Cabe mencionar que en ambos casos se presentaron 0% de error es decir todas las peticiones fueron respondidas. Como cantidad máxima de peticiones en 1 segundo el sistema presenta fallas al alcanzar un número promedio de 120.

3.2.3 Pruebas de carga

Para determinar lo rápido que se realizan las tareas de reporte de baches se enviarán peticiones al API de registro de baches desde la aplicación móvil. Como instrumento de verificación se utilizó el software New Relic, una herramienta de monitoreo que ofrece datos en tiempo real sobre el rendimiento de la aplicación móvil.

- **Escenario 3**

Se simula el reporte uno tras otro de 50 baches por parte de los moderadores del Municipio de Quito.

Para este escenario se hizo una petición POST al API Web, enviando información desde la aplicación móvil, con los datos del bache reportado.

Diagrama en el que se visualizan 50 reportes de baches

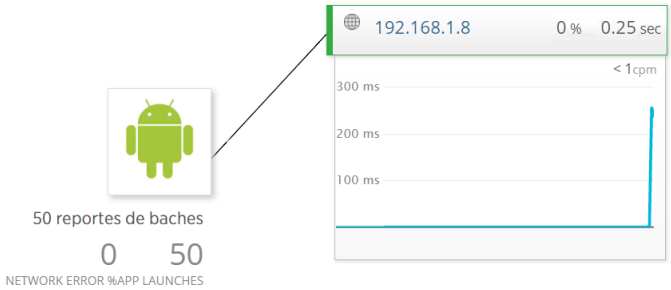

Figura 45 Prueba de carga con 50 peticiones
Elaborado por: Cristhian Castro y Paul Velasco.

Diagrama del número de peticiones HTTP

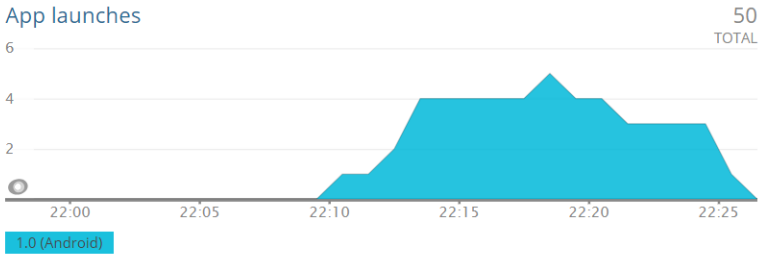

Figura 46 Número de peticiones en prueba de carga
Elaborado por: Cristhian Castro y Paul Velasco.

Tabla 80 Ejecución de prueba de carga escenario 3 (Primera Parte)

Etiqueta	# Muestras	Media	Min	Max	% Error
HTTP Request	50	249 ms	188 ms	313 ms	0.00%

- Nota: Esta tabla muestra los resultados arrojados por la simulación del escenario 3.

En la tabla No. 80 se aprecia que el tiempo máximo de respuesta a una petición fue de 0.313 segundos, y en promedio tomó un tiempo de 0.249 segundos hacer el reporte de un bache, y el valor más rápido de respuesta es de 0.188 segundos.

Diagrama de tiempo de respuesta HTTP con 50 peticiones

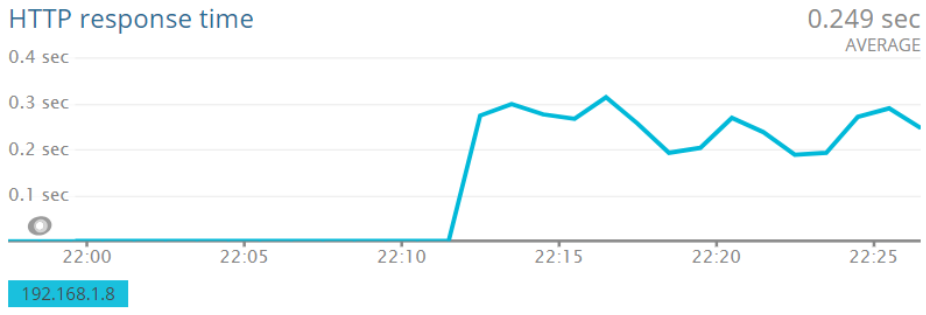

Figura 47 Tiempo promedio de respuesta en prueba de carga
Elaborado por: Cristhian Castro y Paul Velasco.

Tabla 81 Ejecución de prueba de carga escenario 3 (Segunda Parte)

# Muestras	Latencia	T. conexión	Envío Bytes	Resp Bytes
50	247.3 ms	0.44 ms	175087 bytes	369 bytes

- Nota: Esta tabla muestra los resultados arrojados por la simulación del escenario 3.

Cada petición al servidor envío datos con un peso de 175087 bytes (175.08 Kbytes), conteniendo la fotografía de dimensiones 800 x 600 px con un tamaño aproximado de 128 Kbytes, a su vez la respuesta fue de 369 bytes.

Con estas condiciones el servidor tuvo un promedio de latencia en 0.2473 segundos, cuyo valor es medido justo antes de enviar la solicitud hasta justo después de que se recibió la primera respuesta. El tiempo que llevó establecer la conexión en promedio fue de 0.00044 segundos.

- **Escenario 4**

Escenario en el que se simula el reporte uno tras otro de 105 baches por parte de los moderadores del Municipio de Quito.

Para este escenario se hizo una petición POST al API Web, enviando información desde la aplicación móvil, con los datos del bache reportado.

Diagrama en el que se visualizan 105 reportes de baches

Figura 48 Prueba de carga con 105 peticiones
Elaborado por: Cristhian Castro y Paul Velasco.

Diagrama del número de peticiones HTTP

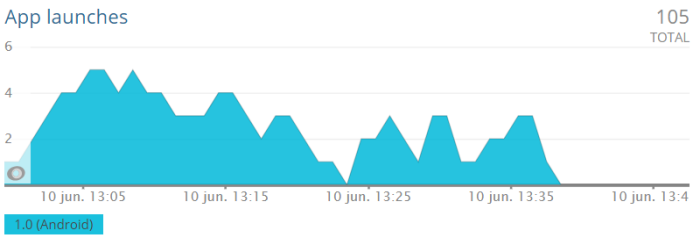

Figura 49 Número de peticiones en prueba de carga
Elaborado por: Cristhian Castro y Paul Velasco.

Tabla 82 Ejecución de prueba de carga escenario 4 (Primera Parte)

Etiqueta	# Muestras	Media	Min	Max	% Error
HTTP Request	105	328ms	146 ms	343 ms	0.00%

- Nota: Esta tabla muestra los resultados arrojados por la simulación del escenario 4.

En la tabla No. 82 se aprecia que el tiempo máximo de respuesta a una petición fue de 0.343 segundos, y en promedio tomó un tiempo de 0.328 segundos hacer el reporte de un bache, y el valor más rápido de respuesta es de 0.146 segundos.

Diagrama de tiempo de respuesta HTTP con 105 peticiones

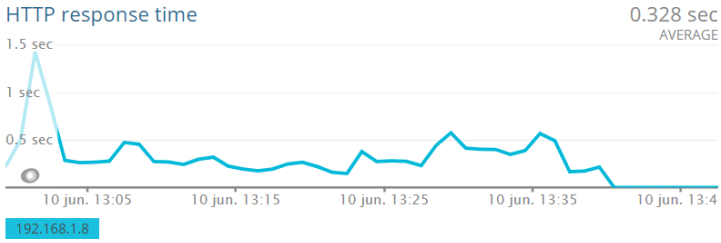

Figura 50 Tiempo promedio de respuesta en prueba de carga
Elaborado por: Cristhian Castro y Paul Velasco.

Tabla 83 Ejecución de prueba de carga escenario 4 (Segunda Parte)

# Muestras	Latencia	T. conexión	Envío Bytes	Resp Bytes
105	1458,34 ms	0.94 ms	175087 bytes	369 bytes

- Nota: Esta tabla muestra los resultados arrojados por la simulación del escenario 4.

Cada petición al servidor envío datos con un peso de 175087 bytes (175.08 Kbytes), conteniendo una fotografía de dimensiones 800 x 600 px con un tamaño aproximado de 128 Kbytes, a su vez la respuesta fue de 369 bytes. Con estas condiciones el servidor tuvo un promedio de latencia en 1.45 segundos, cuyo valor es medido justo antes de enviar la solicitud hasta justo después de que se recibió la primera respuesta. El tiempo que llevó establecer la conexión en promedio fue de 0.00094 segundos.

Análisis de resultados

Tomando en cuenta los dos escenarios evaluados el máximo flujo evaluado fue de 105 reportes uno tras otro, lo que resulta ser optimo y se puede concluir que el sistema maneja adecuadamente aquella cantidad de peticiones. Cabe mencionar que en ambos casos se presentaron 0% de error es decir todas las peticiones fueron respondidas.

3.2.4 Pruebas de consumo de datos utilizando Wi-Fi

Para determinar el uso de datos de Internet que es un dato esencial para conocer que procesos consumen más megabytes, se crearon escenarios de prueba para verificar que procesos son los que consumen más megabytes dentro de la aplicación móvil.

- **Escenario 5**

En este escenario se realizaron de 1 a 100 inicios de sesión desde la aplicación móvil la cual se conectó con el API de Login de la aplicación Web, todo esto con el fin de verificar el consumo de datos utilizando la tecnología inalámbrica de Wi-Fi.

Tabla 84 Consumo de datos en el módulo de Inicio de Sesión

Etiqueta	Número de Inicios de Sesión	Medida
Uso de datos con Wi-Fi	1	10.08 KB
	50	0.80 MB
	100	1.18 MB

- Nota: Esta tabla muestra el consumo de datos del módulo Inicio de Sesión

- **Escenario 6**

En este escenario se realizaron de 1 a 100 reportes de baches desde la aplicación móvil la cual se conectó con el API de Reportar de la aplicación Web, todo esto con el fin de verificar el consumo de datos utilizando la tecnología inalámbrica de Wi-Fi.

Tabla 85 Consumo de datos en el módulo de Reportar

Etiqueta	Número de Reportes de Baches	Medida
Uso de datos con Wi-Fi	1	0.90 MB
	50	3.27 MB
	100	5.10 MB

- Nota: Esta tabla muestra el consumo de datos del módulo Reportar

Análisis de resultados

Se pudo observar que el consumo de datos al enviar un reporte es mayor al de iniciar sesión, esto debido a que los reportes deben enviar imágenes transformadas a formato BASE64 lo cual representa mayor uso del Internet y por ende el consumo de datos es mayor. Cabe destacar que el uso de datos a mayor número de peticiones realizadas no aumenta de forma exponencial esto debido a que el dispositivo móvil almacena

preferencias de los sitios visitados, o conexiones realizadas, tal es el caso que la primera petición consumió mayores datos porque tardaba en direccionar la petición a un sitio no conocido.

3.2.5 Pruebas contra requerimientos

Con la aplicación de estas pruebas se estableció el nivel de cumplimiento de los requerimientos planteados por el cliente.

3.2.5.1 Aplicación Web

En la tabla No. 86 se organiza los requerimientos funcionales planteados por módulo, y en contraste se especifica el nivel de cumplimiento de cada uno de ellos.

Tabla 86 Evaluación de cumplimiento de requerimientos (aplicación Web)

#	Prueba	RF	Acciones		Observaciones
1	Creación de usuario.	1.1	- Acceder al sistema Web con credenciales de administrador. - Abrir la pestaña "*Usuarios*". - Pulsar el botón "*Añadir usuario*". - Completar el formulario con los datos personales del nuevo usuario, administración zonal en la que trabaja, perfil asignado. - Pulsar el botón "*Agregar*".	Completo	- La contraseña por defecto asignada al usuario es su cédula de identidad. - Tal como manifiesta el RF1.1 el nombre de usuario es generado automáticamente
2	Modificación de usuario.	1.2	- Acceder al sistema Web con credenciales de administrador. - Abrir la pestaña "*Usuarios*". - En el listado ubicar al usuario a modificar para esto puede usar los filtros. - Entre las acciones disponibles para cada usuario registrado, se encuentra la de "*Editar*" - Pulsar el botón con el icono de un lápiz. - Cambiar los datos que requiera en el formulario presentado. - Pulsar el botón "*Guardar*".	Completo	- El campo de contraseña no es posible ser cambiada mediante el formulario provisto para la edición de usuarios. - Para el restablecimiento de contraseña a un usuario, dentro de las acciones individuales se implementó un botón que cumple dicha función.

3	Búsqueda filtrada de usuario.	1.3	- Acceder al sistema Web con credenciales de administrador. - Abrir la pestaña "*Usuarios*".	Completo	- El listado de usuarios visible puede ser filtrado, por perfil, por estado, y cuenta un campo multicriterio.
4	Iniciar sesión en la aplicación Web.	2.1	- Acceder a la dirección principal de la aplicación. - Proveer el nombre de usuario asignado junto a la contraseña. - Pulsar el botón "*Iniciar sesión*"	Completo	
5	Listar registros de baches.	3.2	- Acceder al sistema Web con credenciales de administrador. - Abrir la pestaña "*Registros*".	Completo	
6	Visualizar baches en mapa dinámico.	3.2	- Acceder al sistema Web con credenciales de administrador. - Abrir la pestaña "*Mapa*".	Completo	
7	Cambiar de estado a registros.	3.1	- Acceder al sistema Web con credenciales de administrador. - Abrir la pestaña "*Registros*". - Emplear el filtro para la clasificación por estado y seleccionar "*Reportados*". - Seleccionar el grupo de baches a cambiar. - Pulsar el botón "*Ejecutar*".	Completo	- El único cambio de estado permitido es el de reportado a en-curso.
8	Asignar un bache para su validación.	3.3	- Acceder al sistema Web con credenciales de administrador. - Abrir la pestaña "*Registros*". - Emplear el filtro para la clasificación por estado y seleccionar "*En curso*". - Elegir que muestre solo los reportes que aún no han sido asignados para su validación. **INDIVIDUAL** - En las acciones disponibles para cada reporte se encuentra "*Asignar*", con el ícono de una persona. - Pulsar "Si" en el mensaje de confirmación. **ACCIÓN GRUPAL** - Elegir los reportes a asignar y apretar en el botón "*Ejecutar*" que se encuentra en la parte inferior del listado. - Pulsar el botón con el ícono de una persona.	Completo	- La asignación de un moderador para validar un bache, se realiza de manera automática, en función del número de validaciones asignadas pendientes.
9	Generar reportes estadísticos.	3.4	- Acceder al sistema Web con credenciales de administrador. - Abrir la pestaña "*Dashboard*". - Elegir los multiparámetros disponibles como rango de fechas, parroquia, administración zonal, estado, tipo de gráfico. - Visualizar o exportar el gráfico estadístico generado.	Completo	

- Nota: Esta tabla muestra el análisis de las pruebas contra requerimientos de la aplicación Web.

3.2.5.2 Aplicación móvil

En la tabla No. 87 se organiza los requerimientos funcionales planteados por módulo, y en contraste se especifica el nivel de cumplimiento de cada uno de ellos.

Tabla 87 Evaluación de cumplimiento de requerimientos (aplicación móvil)

#	Prueba	RF	Acciones	Estado	Observaciones
1	Iniciar sesión en la aplicación móvil.	2.1	- Acceder a la aplicación. - Proveer el nombre de usuario asignado junto a la contraseña. Pulsar el botón "*Iniciar sesión*"	Completo	
2	Registrar bache.	4.1	- Acceder a la aplicación y dirigirse al menú "*Reportar*". - Completar el formulario de reporte. - Seleccionar adjuntar fotografía por medio de la cámara. - Llenar el campo referencia. - Pulsar el botón "*Enviar*".	Completo	
3	Validar bache.	4.2	- Acceder a la aplicación y dirigirse al menú "*Validar*". - Seleccionar la tarea de validación a realizar. - Adjuntar fotografía por medio de la cámara. Pulsar el botón "*Enviar*".	Completo	

- Nota: Esta tabla muestra el análisis de las pruebas contra requerimientos de la aplicación móvil.

3.2.6 Pruebas al proceso

Mediante estas pruebas se compara el tiempo que toma efectuar el proceso manual de registro y validación de baches contra el tiempo efectuando el proceso automatizado mediante el sistema informático.

Para la definición de escenarios se consideró que el proceso actual se lo realiza los días sábados y domingos. Dando como resultado que se registran en promedio 32 baches a la semana.

Cabe mencionar que el tiempo de reparación de los baches es independiente al proceso ejecutado por las administraciones zonales por estar sujeto a estudios técnicos y factibilidad, sin embargo, por datos obtenidos de los actores esta acción se suele demorar 15 a 20 días.

- **Escenario 7**

Se llevó a cabo el proceso manual bajo condiciones óptimas con lo cual se obtuvieron los siguientes resultados.

Registro y validación de bache

Tabla 88 Prueba de registro y validación de baches (manual).

Prueba	Tiempo unitario	Acciones
Registrar un bache.	25 min/por registro	-Acudir al lugar donde se encontró el bache. -Capturar una fotografía. -Registrar en la matriz de mantenimiento rutinario los datos de: Fecha, calle principal, calle secundaria, parroquia, referencia, nombre de la fotografía asociada.
Validar un bache.	25 min/por registro	-Acudir al lugar donde se efectuó la reparación del bache. -Capturar una fotografía. -Registrar en la bitácora la correcta validación del bache.

- Nota: Esta tabla muestra los resultados obtenidos de la prueba de reporte y validación de baches.

Digitalización de información

Tabla 89 Prueba de digitalización de información de reportes (manual).

Prueba	Tiempo	Acciones
Validar de archivos fotográficos y digitalizar documento matriz de mantenimiento rutinario.	25 minutos	-Constatar la correcta asociación de archivos fotográficos a cada uno de los registros de la matriz. -Traspasar la información de la matriz a un archivo de Excel.
Generar gráficos de reporte.		-Contrastar los datos de reportes y validaciones, se hace uso de la herramienta Excel. Generar reportes en base a la información generada en el paso anterior.

- Nota: Esta tabla muestra los resultados obtenidos de la prueba de digitalización de información.

Asignar tareas de validación

Tabla 90 Prueba de asignar tareas de validación (manual).

Prueba	Tiempo	Acciones
Asignar tareas de validación	20 minutos	- Organizar al personal. - Establecer un mapa de ruta al que debe dirigirse el personal.

- Nota: Esta tabla muestra los resultados obtenidos de la prueba de asignación de tareas de validación.

- **Escenario 8**

Se llevó a cabo el proceso automatizado bajo condiciones óptimas con lo cual se obtuvieron los siguientes resultados.

Registro y validación de bache

Tabla 91 Prueba de registro y validación de baches (automatizado).

Prueba	Tiempo unitario	Acciones
Registrar un bache.	6 minutos por registro	-Acudir al lugar donde se encontró el bache. -Abrir la aplicación -Capturar una fotografía. -Llenar el campo de referencia (opcional).
Validar un bache.	7 minutos por registro	-Acudir al lugar donde se efectuó la reparación del bache. -Abrir la aplicación -Capturar una fotografía.

- Nota: Esta tabla muestra los resultados obtenidos de la prueba de reporte y validación de baches.

Digitalización de información

No Aplica.

Asignar tareas de validación

Tabla 92 Prueba de asignar tareas de validación (manual).

Prueba	Tiempo	Acciones
Asignar tareas de validación	4 minutos	- Acceder a la aplicación. - Seleccionar los baches a validar. - Asignar personal responsable.

- Nota: Esta tabla muestra los resultados obtenidos de la prueba de asignación de tareas de validación.

Análisis de resultados

- Los resultados obtenidos en el proceso manual con el registro de un sólo bache, se evidencia un tiempo de 25 minutos en condiciones óptimas, y con la ejecución del proceso automatizado se emplea un tiempo de 6 minutos, obteniendo una disminución del 76% del tiempo utilizado.

- En cuanto a los resultados obtenidos en el proceso manual con la validación de un sólo bache, se evidencia un tiempo de 25 minutos en condiciones óptimas, y con la

ejecución del proceso automatizado se emplea un tiempo de 7 minutos, obteniendo una disminución del 72% del tiempo en relación al proceso manual.

- Referente a la digitalización de la información el porcentaje de optimización es 100%, debido a que la información se almacena directamente en el sistema informático
- Con relación a la asignación de tareas de validación, se demuestra un tiempo de 20 minutos, y con el uso de la aplicación se emplea un tiempo de 4 minutos, obteniendo una disminución del 80% del tiempo en relación al proceso manual en la asignación de tareas de validación.
- En total el proceso manual controlado por la administración zonal toma un tiempo de 95 minutos y el tiempo necesario para llevar a cabo el proceso automatizado es 17 minutos lo que da como resultado una mejora al tiempo en un 82.11%, lo que demuestra que el proceso automatizado es aproximadamente 6 veces más rápido.

Capítulo 4

Implementación

Una vez efectuado el cierre formal al proyecto, por la entrega de todos los productos de conformidad con los requerimientos planteados por el cliente, tal como se refleja en el acta de cierre y entrega del proyecto (Ver anexo 3), se procedió a ejecutar la implementación del sistema detallada en el presente capítulo, en la que se enfatiza en las configuraciones necesarias para el despliegue del proyecto en ambiente de producción.

4.1 Descripción hardware y software

4.1.1 Hardware

Tabla 93 Características de hardware

Nombre	Descripción
Marca	Hp
Modelo	Proliant MI110
Disco duro	4 Tb
Procesador	Xeon-b 3104 6core
Memoria RAM	16 Gb

- Nota: Esta tabla muestra las características del hardware donde fue implementado el sistema.

4.1.2 Software

Tabla 94 Características de software

Nombre	Descripción
Sistema Operativo	Windows Server
Versión	2012
Arquitectura	64 bit

- Nota: Esta tabla muestra las características del software donde fue implementado el sistema.

4.2 Descripción técnica de componentes

Tabla 95 Descripción técnica componentes

Nombre	Versión
Python	3.6
PostgreSQL	9.3
Mod_wgsi	4.6.4
Apache Server	2.4

- Nota: Esta tabla muestra las características del software donde fue implementado el sistema.

4.3 Configuración del servidor wsgi-server

WSGI (Web Server Gateway Interface) es una interface simple y universal entre los servidores Web y las aplicaciones basadas en Python.

Entorno virtual Python

Para iniciar, el primer paso es instalarlo en las dependencias del entorno virtual de Python usado en el proyecto.

- Abrir la consola de comando en Windows.
- Establecer la ruta raíz de la instalación de Apache
  set "MOD_WSGI_APACHE_ROOTDIR=C:\Apache24"
- Ejecutar el comando *"pip install mod_wsgi"*

Como siguiente paso es registrar la instalación de *"mod_wsgi"* en el archivo settings.py en el apartado INSTALLED_APPS del proyecto.

- INSTALLED_APPS = […, 'mod_wsgi.server']

Una vez instalado y registrado *"mod_wsgi"* se genera el archivo *"wsgi.py"* el que debe ser modificado con los siguientes valores.

Código fuente en archivo wsgi.py

```
import os, sys, site
site.addsitedir('C:\ProyectosDjango\ambientes\geobache\Lib\site-
packages')
sys.path.append('C:\ProyectosDjango\ProyTesis\geobache')
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "AppWeb.settings")
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
```

Figura 51. Archivo wsgi.py
Elaborado por: Cristhian Castro y Paul Velasco.

La configuración anterior permite definir las rutas de ambiente virtual Python, proyecto, y archivo *"settings.py"* de la aplicación.

Servidor Apache

Para que Apache reconozca el módulo *"mod_wsgi"* es necesario configurar el archivo *"httpd.conf"* en el que se agrega las siguientes líneas.

Código fuente en archivo httpd.conf

```
LoadFile "c:/program files (x86)/python36-32/python36.dll"
LoadModule wsgi_module "c:/proyectosdjango/ambientes/geobache/lib/site-
packages/mod_wsgi/server/mod_wsgi.cp36-win32.pyd"
WSGIPythonHome "c:/proyectosdjango/ambientes/geobache"
WSGIPassAuthorization On
```

Figura 52. Archivo httpd.conf
Elaborado por: Cristhian Castro y Paul Velasco.

En donde se define las rutas de:

- Instalación de Python
- Módulo *"mod_wsgi"* anteriormente instalado
- Proyecto Django

Adicional se puede observar la variable *"WSGIPassAuthorization"* con el valor On, esta es usada con el fin de que los encabezados de autorización enviados en las peticiones sean manejados por el servidor wsgi y no por Apache.

Ahora queda por crear un host virtual dentro de Apache para ello debemos acceder al archivo *"httpd-vhosts.conf"* en él se escribe.

Código fuente en archivo httpd-vhosts.conf

```
<VirtualHost 0.0.0.0:80>
    ServerName localhost
    ErrorLog "logs/supervisiontool.error.log"
    CustomLog "logs/supervisiontool.access.log" combined
  WSGIScriptAlias /
     "C:/ProyectosDjango/ProyTesis/geobache/AppWeb/AppWeb/wsgi.py"
  <Directory "C:/ProyectosDjango/ProyTesis/geobache/AppWeb/AppWeb">
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    Alias /static "C:\ProyectosDjango\ProyTesis\geobache\static"
    <Directory "C:\ProyectosDjango\ProyTesis\geobache\static">
        Require all granted
  </Directory>

    Alias /fotografias
     "C:\ProyectosDjango\ProyTesis\geobache\AppWeb\fotografias"
    <Directory
     "C:\ProyectosDjango\ProyTesis\geobache\AppWeb\fotografias">
   Require all granted
    </Directory>
</VirtualHost>
```

Figura 53. Archivo httpd-vhost.conf
Elaborado por: Cristhian Castro y Paul Velasco.

Primero se establece que los hosts permitidos funcionarán en el puerto 80, se definen la ruta donde se encontrarán los archivos logs, se enlaza a la ruta del archivo "*wgsi.py*" que es el vínculo a la aplicación Django, y finalmente se registran los directorios de archivos estáticos y multimedia a los que accede la aplicación Web.

4.4 Servidor de base de datos

Por la naturaleza de Django es necesario proveer de una configuración para la conexión a la base de datos en el archivo *"settings.py"*.

Configuración base de datos archivo settings.py

```
DATABASES = {
  'default': {
    'ENGINE': 'django.contrib.gis.db.backends.postgis',
    'NAME': 'GeoBache',
    'USER': 'postgres',
    'PASSWORD':'****',
    'HOST': 'localhost',
    'PORT': 5432,
  }
}
```

Figura 54. Configuración base de datos archivo settings.py
Elaborado por: Cristhian Castro y Paul Velasco.

4.5 Diagrama de despliegue

Por medio de la figura No. 55, se expone la distribución de los elementos necesarios para que el sistema funcione en ambiente de producción.

Diagrama de despliegue ambiente de producción

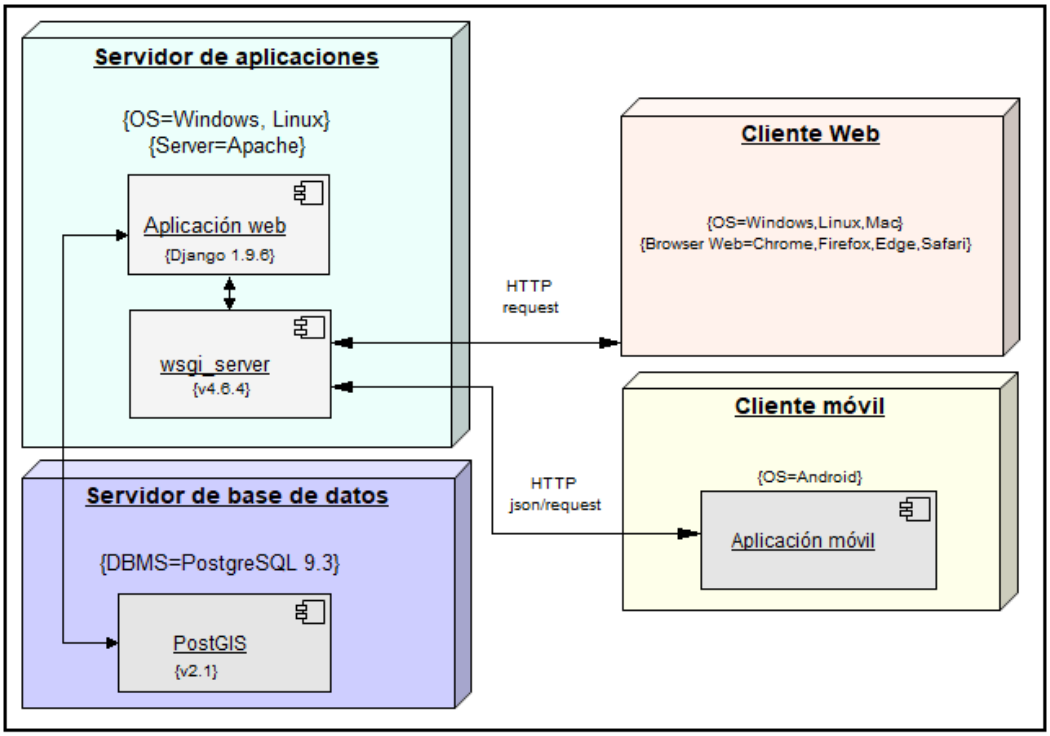

Figura 55 Diagrama de despliegue ambiente de producción.
Elaborado por: Cristhian Castro y Paul Velasco.

CONCLUSIONES

- Con la implementación del sistema informático se evidencia que el proceso es un 82.11% más rápido en la ejecución de tareas, resultado de las pruebas realizadas al proceso de reporte y validación de baches tanto de forma manual como de forma automatizada, además de brindar una mejor gestión y control en el uso de recursos y asignación de tareas al personal en las Administraciones Zonales del Municipio de Quito.
- El uso del sistema informático minimiza los errores humanos que pudieran surgir en el anotar y digitalizar los datos de baches o por perdida de los archivos fotográficos.
- El involucramiento dentro del proceso manual que se lleva a cabo en las administraciones zonales permitió conocer las tareas operativas y administrativas ejecutadas a diario por los respectivos actores, lo que facilitó la definición y construcción de un sistema informático flexible que asiste en la solución de la problemática generada en la ciudad a causa de la presencia de baches en las vías.
- La inclusión del cliente dentro del equipo de trabajo, cualidad propia de la metodología aplicada en la programación del sistema, demuestra que la construcción de los requerimientos detallados en las historias de usuario es más eficiente debido a la retro alimentación aportada en cada una de las iteraciones logradas, lo que derivó en un producto adaptable a la necesidad del usuario.
- Durante el desarrollo del sistema informático específicamente en la programación de la aplicación móvil, se utilizó Android Studio debido a que la codificación es más ágil y la gestión de cada error es más factible, dando como resultado interfaces agradables, sólidas y convergentes con diferentes

tecnologías como Retrofit que funciona como cliente Rest para hacer peticiones a la aplicación Web, y encargarse automáticamente de las respuestas.

- La utilización de Django en la aplicación Web promovió un desarrollo rápido por incorporar nativamente funciones comunes en este tipo de sistemas tales como login, implementación de seguridades, y funciones CRUD, además de facilitar la integración con múltiples herramientas para su potenciación y adaptación de nuevas funciones como Rest Framework en la construcción de API's, o GeoDjango para la manipulación de datos geográficos.
- La adecuada planeación y edificación concebida en la fase de diseño dio como resultado una arquitectura del sistema correctamente apegada a la lógica del proceso, lo que se evidenció en las pruebas ejecutadas donde surgieron cambios en detalles que no veían comprometido el desarrollo estructural del sistema.
- New Relic un software de monitoreo y verificación de rendimiento de aplicaciones en tiempo real, fue utilizado para realizar pruebas sobre la aplicación móvil, en la que 105 usuarios no concurrentes hicieron peticiones hacia el servidor obteniendo un tiempo de respuesta promedio de 328 ms, resultando ser óptimo en él envió de información de todo el sistema informático.

RECOMENDACIONES

- En futuras versiones de la aplicación móvil se recomienda el desarrollo para diferentes sistemas operativos móviles como iOS, o Windows Phone, con el objetivo de que el grupo de usuarios no tenga limitante del dispositivo en el que se ejecutará el aplicativo.
- En una nueva versión del sistema informático se recomienda añadir la funcionalidad de reportar trayectos en mal estado, donde se consideraría un punto de inicio y un punto final.
- Se recomienda realizar la ampliación del grupo objetivo para que la aplicación móvil se encuentre disponible para la ciudadanía en general.

LISTA DE REFERENCIAS

Android Studio. (25 de mayo de 2018). *Android Studio Developers*. Obtenido de https://developer.android.com/studio/

Bache-24. (06 de junio de 2018). *Bache-24 CDMX*. Obtenido de https://bache24.agucdmx.gob.mx

Bahit, E. (2011-2012). *Scrum y eXtreme Programming.* Buenos Aires.

Bootstrap. (01 de mayo de 2018). *Bootstrap Documentación*. Obtenido de https://getbootstrap.com/docs/3.3/

Borja Yolanda. (06 de junio de 2018). *Metodología Ágil de Desarrollo de Software – XP.* Obtenido de Proyecto EtnoMatemática - RunaYupay: http://www.runayupay.org/publicaciones/2244_555_COD_18_290814203015.pdf

Buenos Aires Ciudad. (06 de junio de 2018). *Buenos Aires Ciudad (BA)*. Obtenido de http://www.buenosaires.gob.ar/

Calabria, L., Píriz, P., & Mousques, G. (2003). *Metodología XP.* Obtenido de Universidad ORT: https://fi.ort.edu.uy/innovaportal/file/2021/1/metodologia_xp.pdf

Cifuentes, J. (15 de julio de 2017). Problemas en el registro y control de baches en la ciudad de Quito. Quito.

Django. (01 de mayo de 2018). *Django Framework*. Obtenido de https://www.djangoproject.com/

Django. (01 de mayo de 2018). *GIS*. Obtenido de https://docs.djangoproject.com/en/2.0/ref/contrib/gis/

DjangoRestFramework. (01 de mayo de 2018). *Django Rest Framework-Documentación*. Obtenido de http://www.django-rest-framework.org/

EPMMOP. (2016). *Rehabilitación y Mantenimiento vial.* Recuperado el 13 de noviembre de 2017, de Procesos técnicos de mejoramiento y mantenimiento de las vías de Quito: http://www.epmmop.gob.ec/epmmop/index.php/proyectos/obras-publicas/rehabilitacion-y-mantenimiento-vial

EPMMOP. (03 de abril de 2017). *Plan de obras viales 2017 - 2018.* Recuperado el 15 de Noviembre de 2017, de Rehabilitación y mantenimiento vial EPMMOP: http://www.epmmop.gob.ec/epmmop/index.php?option=com_content&view=article&id=64

Gallego, M. (18 de junio de 2012). *Desarrollo detallado de la fase de aprobación de un proyecto informático mediante el uso de metodologías ágiles (Metodología Scrum).* Obtenido de Universitat Oberta Catalunya: http://openaccess.uoc.edu/Webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

Git. (25 de mayo de 2018). *Git*. Obtenido de https://git-scm.com/

Java. (01 de mayo de 2018). *Java*. Obtenido de https://www.java.com/es/about/

JustinMind. (25 de Mayo de 2018). *JustinMind*. Obtenido de https://www.justinmind.com/

Kniberg, H. (2007). *Scrum y Xp desde las trincheras.* Estados Unidos de América: InfoQ.

Letelier, P., & Penadés, M. (s.f.). *Métodologías ágiles para el desarrollo de software:*. Obtenido de Universidad Politécnica de Valencia: http://users.dsic.upv.es/asignaturas/eui/lds/doc/masyxp.pdf

Montero, S. I. (30 de abril de 2012). *Maestros del Web by Platzi*. Obtenido de http://www.maestrosdelWeb.com/curso-django-entendiendo-como-trabaja-django/

Organización para la Cooperación y el Desarrollo Económicos "OCDE". (2013). *Propuesta de Norma Práctica para encuestas de Investigación y Desarrollo Experimental - Manual de Frascati.* F-Iniciativas Ltda.

Palacio, J. (2008). *Flexibilidad con Scrum.*

PostGIS. (25 de mayo de 2018). *PostgreSQL-PostGIS*. Obtenido de https://postgis.net/

PostgreSQL. (25 de Mayo de 2018). *PostgreSQL-Documentación*. Obtenido de https://www.postgresql.org/docs/

Python. (01 de mayo de 2018). *Python*. Obtenido de https://www.python.org/

Real Academia Española. (2017). Diccionario de la lengua española. Madrid, España.

SAP. (25 de mayo de 2018). *SAP-Power Designer*. Obtenido de https://www.sap.com/products/powerdesigner-data-modeling-tools.html

Star UML. (25 de mayo de 2018). *Star UML*. Obtenido de http://staruml.io/

Takeuchi, H., & Nonaka, I. (enero de 1986). *The New New Product Development Game.* Obtenido de Harvard Business Review: https://hbr.org/1986/01/the-new-new-product-development-game

Visual Studio Code. (01 de mayo de 2018). *Visual Studio Code*. Obtenido de https://code.visualstudio.com/docs

Xtreme Programming. (8 de octubre de 2013). *XP-org*. Obtenido de http://www.extremeprogramming.org/

ANEXOS

Para revisar los anexos de este trabajo, por favor diríjase al CD.