UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

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

TEMA:
SOLUCIÓN INFORMÁTICA DE ACCESO CIUDADANO, PARA EL REGISTRO DE LA UBICACIÓN DE BACHES EN LA CIUDAD DE QUITO DM, UTILIZANDO COMANDOS DE VOZ Y GEOLOCALIZACIÓN

AUTORES:
DANIEL ANTONIO RIERA SUAREZ
JORGE PATRICIO SORIA ESPINOZA

TUTOR:
FRANKIN EDMUNDO HURTADO LARREA

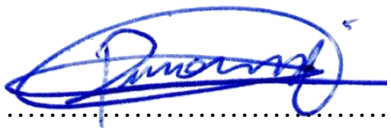
Quito, agosto del 2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros, DANIEL ANTONIO RIERA SUAREZ con documento de identificación Nº 1717376808 y JORGE PATRICIO SORIA ESPINOZA con documento de identificación Nº 1719129478, 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: "SOLUCIÓN INFORMÁTICA DE ACCESO CIUDADANO, PARA EL REGISTRO DE LA UBICACIÓN DE BACHES EN LA CIUDAD DE QUITO DM, UTILIZANDO COMANDOS DE VOZ Y GEOLOCALIZACIÓN", mismo que ha sido desarrollado para optar por el título de INGENIEROS DE SISTEMAS en la Universidad Politécnica Salesiana, quedando la misma con facultad 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 digital a la Biblioteca de la Universidad Politécnica Salesiana.

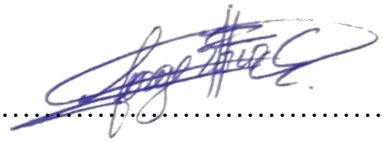


……

DANIEL ANTONIO

RIERA SUAREZ

CI: 1717376808



……

JORGE PATRICIO

SORIA ESPINOZA

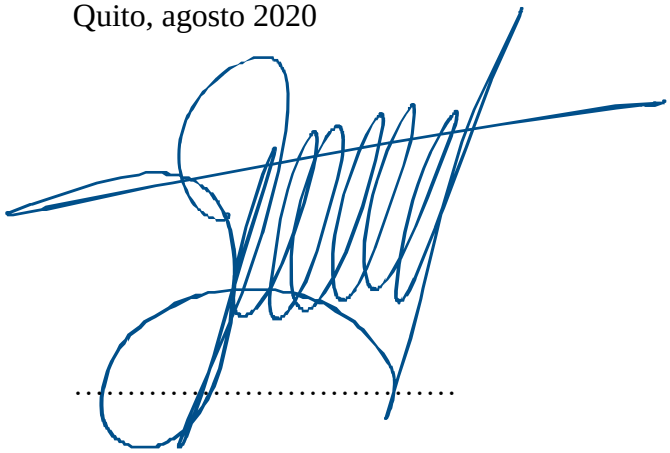
CI: 1719129478

Quito, agosto del 2020

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, con el tema: "SOLUCIÓN INFORMÁTICA DE ACCESO CIUDADANO, PARA EL REGISTRO DE LA UBICACIÓN DE BACHES EN LA CIUDAD DE QUITO DM, UTILIZANDO COMANDOS DE VOZ Y GEOLOCALIZACIÓN", realizado por DANIEL ANTONIO RIERA SUAREZ y JORGE PATRICIO SORIA ESPINOZA, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, agosto 2020



……

FRANKLIN EDMUNDO HURTADO LARREA

CI: 1713382016

DEDICATORIA

Dedico esta investigación a mi madre, Rosa Margarita Suarez Ortiz, a mis hermanos, Juan Wilberto Riera Suarez, Ángel Santiago Riera Suarez y a mi hermana, Estefanía Carolina Riera Suarez, por haberme ayudado en los momentos más difíciles, tanto de manera económica como emocional, les doy las gracias por comprenderme durante el período universitario, el cual fue unos de los más grandes retos de mi vida.

A mis tíos y primos que siempre me han apoyado.

La culminación de mi carrera profesional es la prueba de que los objetivos se pueden cumplir con esfuerzo y determinación.

Daniel Antonio Riera Suarez

Esta investigación está dedicada a mi madre, María Teresa Espinoza Bravo, quien me enseñó que la mejor herencia es la educación, a mi padre, Aníbal Lautaro Soria Ruiz y a mi hermano Julio Aníbal Soria Espinoza quienes con su amor, paciencia, trabajo y sacrificio en todos estos años me han permitido llegar a cumplir hoy uno de mis sueños. ¡Gracias! porque con sus oraciones, consejos y palabras de aliento hicieron de mí una mejor persona.

Jorge Patricio Soria Espinoza

AGRADECIMIENTO.

Con la culminación de este proyecto de titulación, damos las gracias a la Universidad Politécnica Salesiana, a su equipo docente, a nuestros familiares, amigos y compañeros; por haber formado parte de este proceso educativo.

Un agradecimiento especial a nuestro tutor, el Ing. Franklin Hurtado, por el apoyo brindado, por ser un excelente docente que nos ayudó con nuestra formación intelectual y académica, llevando a cabo una excelente tutoría, guiándonos con sus consejos y recomendaciones durante el desarrollo de nuestro proyecto de titulación.

Daniel Antonio Riera Suarez

Jorge Patricio Soria Espinoza

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

La problemática planteada es la presencia de baches en las vías del Distrito Metropolitano de Quito, siendo esta una de las causas del malestar en la ciudadanía ya que estos provocan accidentes, muertes, congestiones de tránsito, ausentismo y atrasos en los lugares de trabajo, así como también daños vehiculares, tal como menciona un artículo del Diario El Comercio. (COMERCIO, 2017)

Este proyecto se ha propuesto para brindar una solución informática mediante la cual se realizará un registro de la ubicación de baches, utilizando un comando de voz y geolocalización, esta información será generada por cada uno de los usuarios.

El documento presente se compone de la definición de la problemática, los objetivos, marco metodológico, marco teórico, la descripción del análisis y diseño, y la construcción de las dos aplicaciones con sus respectivas pruebas además de conclusiones y recomendaciones, mismo que se encuentra conformado por un aplicativo móvil que funciona con un sistema operativo Android que al detectar el comando de voz puede registrar la ubicación del bache, misma que podrá ser vista por todos los usuarios (conductores y transeúntes) en la página web.

Con este proyecto se pretende ofrecer una ayuda al usuario para que los vehículos no caigan en agujeros, prevenir accidentes y otro tipo de peligros que podrían causar los baches.

Abstract

The problem raised is the presence of potholes in the roads of the Metropolitan District of Quito, this being one of the causes of the discomfort in the citizenry since these cause accidents, deaths, traffic congestion, absenteeism and delays in the workplace, as well as well as vehicular damage, as mentioned in an article in the El Comercio newspaper. (COMERCIO, 2017)

This project has been proposed to provide a computer solution through which a record of the location of potholes will be made, using a voice command and geolocation, this information will be generated by each of the users.

The present document is made up of the definition of the problem, the objectives, the methodological framework, the theoretical framework, the description of the analysis and design, and the construction of the two applications with their respective tests, as well as conclusions and recommendations, which is made up by a mobile application that works with an Android operating system that when detecting the voice command can register the location of the pothole, which can be seen by all users (drivers and passers-by) on the website.

This project aims to offer help to the user so that vehicles do not fall into holes, prevent accidents and other types of dangers that potholes could cause.

INTRODUCCIÓN

La presencia de baches en las vías de la ciudad de Quito en estos últimos años se han convertido en uno de los mayores contratiempos para los ciudadanos quiteños lo cual a provocado desagrado en la comunidad, trayendo consigo el incremento de los costos de mantenimiento, consumo de combustible y el desgaste de neumáticos; ocasionando accidentes de tránsito y poca movilidad tanto peatonal como vehicular. Conforme a la investigación realizada, el 4 de noviembre de 2019, la EPMMOP informó que Quito posee 8.234.73km de longitud de calzada que forman parte de más 11.000 vías a lo largo de la ciudad. (EPMMOP, 2019)

PROBLEMÁTICA

En época de lluvias es más notable esta molestia, puesto que existen muchos charcos, los cuales pueden ocultar baches profundos y con bordes filosos llenos de agua. Acorde al artículo revisado sobre el sistema vial en la ciudad de Quito el 20 de marzo de 2019 el Diario El Comercio (Merizalde, 2017), menciona que las calles presentan baches de hasta 20.00 cm de profundidad siendo un peligro tanto para conductores como para transeúntes. Conforme a la revisión realizada el 16 de noviembre de 2019, la EPMMOP informó que el 80% de las vías, calles y carreteras ya cumplieron su vida útil (alrededor de 20 años), la capa de rodaje tiene aproximadamente 30 años.

El artículo del sitio web *ELENFOQUE4P* señala que en la ciudad de Quito, la administración 2014-2019 dio atención prioritaria a las obras y mantenimientos viales. (Hernández, 2017)

El aplicativo al que se le ha denominado *Bache* está diseñado para el uso cotidiano de las personas que conducen y caminan dentro de la ciudad, el mismo que funcionará en segundo plano al detectar la voz registrando la ubicación del bache que luego podrá ser vista por otros usuarios en la página web.

JUSTIFICACIÓN

Estudios realizados señalan que el sistema operativo Android, de código abierto (Linux), es el más utilizado por los diferentes fabricantes de telefonía móvil. (Inventó, 2017)

Los dispositivos móviles actuales cuentan con geolocalización (GPS) que señalan la ubicación geográfica en tiempo real ayudando así al usuario a ubicarse con exactitud.

Esta aplicación está diseñada para el sistema operativo Android que funciona en los dispositivos móviles, mediante la utilización del GPS, se registrará la ubicación del bache, en caso de que el usuario no cuente con acceso a internet, estos datos serán almacenados en una memoria temporal, para que cuando se active la conexión, éstos puedan ser incorporados a la base de datos del servidor, con lo cual estarán disponibles a través del sitio web. Debido a que el aplicativo es gratuito, accesible y amigable para usuarios de 15 años en adelante, se proyecta que sea de uso masivo, con lo que se estima se puede recolectar una cantidad suficiente de datos de libre acceso a la comunidad, que permitan visibilizar esta problemática de forma más dinámica y con ello, tener la posibilidad de plantear más alternativas de solución en menor tiempo.

Objetivo General

Desarrollar una solución informática de acceso ciudadano, para el registro de la ubicación de baches en la ciudad de Quito DM, utilizando comando de voz y geolocalización, que permita visualizar la información sobre el estado de las calles y avenidas de cada sector de la ciudad en tiempo real.

Objetivos Específicos

Hacer más eficaz y rápido el proceso de aviso de existencias de baches en las calles, avenidas y sectores de la ciudad de Quito mediante el uso de tecnologías.

Mantener informadas a las autoridades y a la ciudadanía sobre la existencia de baches dentro de la ciudad de Quito.

Poner a disposición de la ciudadanía datos relevantes de la problemática de los baches en Quito.

Marco Metodológico

La metodología Scrum fue utilizada para ejecutar el proyecto, se decidió trabajar con esta técnica después de analizar que la misma cumple con las características y requerimientos necesarios, logrando con ella, obtener resultados aceptables.

Roles

Product Owner

El dueño del producto es la persona responsable de incrementar el valor del producto y el trabajo del equipo de desarrollo ofreciendo, priorizando y validando los requerimientos. (Schwaber & Sutherland, 2016)

Tabla 1. Product Owner

Nombre	Tutor del trabajo de titulación
Rol	Product Owner

Cargo	Ing. de Sistemas
Funciones	Ofrecer, validar y priorizar los requisitos

Nota: El dueño del producto
Elaborado por: Daniel Riera y Jorge Soria (2019)

Scrum Master

El líder es el responsable de asegurar que Scrum se comprenda y sea favorable para los usuarios además da servicio al Product Owner como buscar técnicas que sirvan para ordenar y gestionar el listado de productos para incrementar el valor. (Schwaber & Sutherland, 2016)

También ayuda al Team (equipo de desarrollo) a ser multifuncional y organizado para crear productos de alto valor.

Tabla 2. Scrum Master

Nombre	Daniel Riera
Rol	Scrum Master
Cargo	Estudiante de Ingeniería de Sistemas
Funciones	Lidera, guía a entender y adoptar Scrum

Nota: Líder del proyecto
Elaborado por: Daniel Riera y Jorge Soria (2019)

Team (Equipo)

Está conformado por Programador 1 y Programador 2 de quienes dependerá el progreso del proyecto, la calidad del producto y la eficiencia del trabajo para obtener los resultados esperados.

Tabla 3. Equipo Scrum

Nombre	Daniel Riera
Rol	Programador 1
Cargo	Estudiante de Ingeniería de Sistemas

Funciones	Diseño y desarrollo de la aplicación

Nota: Programador 1 del sistema informático
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 4. Equipo Scrum

Nombre	Jorge Soria
Rol	Programador 2
Cargo	Estudiante de Ingeniería de Sistemas
Funciones	Diseño y desarrollo de la aplicación

Nota: Programador 2 del sistema informático
Elaborado por: Daniel Riera y Jorge Soria (2019)

Informe de ejecución de Sprint

Para el desarrollo de las aplicaciones web y móvil se determinaron las siguientes Sprints en los que se efectuaron las tareas para el cumplimiento de los objetivos propuestos en este proyecto.

Tabla 5. Detalle Sprint 1

Sprint 1: Levantamiento de requerimientos	
Duración: 20 días	
Actividad	**Tiempo**
Planificación de los requerimientos por parte del Product Owner.	2 días
Reunión con el Product Owner para presentar avances del proyecto.	2 días
Indagar toda la información requerida sobre detección y repavimentación de baches en el Distrito Metropolitano de Quito	5 días
Descripción de los requerimientos para el uso en la aplicación	2 días
Establecer las herramientas para la elaboración de las aplicaciones	4 días
Documentar y corregir las especificaciones de los requerimientos	5 días

Nota: Levantamiento de Requerimientos
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 6. Detalle Sprint 2

Sprint 2: Creación del web service	
Duración: 23 días	
Actividad	Tiempo
Diseño y creación de la base de datos del web service	3 días
Creación del web service	20 días

Nota: Creación web service
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 7. Detalle Sprint 3

Sprint 3: Diseño e implementación del registro y geolocalización móvil.	
Duración: 40 días	
Actividad	**Tiempo**
Diseño visual del móvil	1 día
Investigación de librerías para el registro del móvil	2 días
Creación del módulo de registro del móvil	2 días
Investigación de librerías para la geolocalización móvil	7 días
Desarrollo e implementación de la geolocalización móvil	15 días
Corrección de errores funcionales de la geolocalización móvil	13 días

Nota: Diseño e implementación del registro y geolocalización móvil
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 8. Detalle Sprint 4

Sprint 4: Diseño e implementación del comando de voz y validación de datos.	
Duración: 20 días	
Actividad	**Tiempo**
Investigación de librerías para el reconocimiento de voz	7 días
Modificación del reconocedor de voz de inglés al español	7 días
Implementación del módulo de reconocimiento de voz	1 día
Desarrollo del código de la validación de datos móviles	4 días
Implementación del módulo de validación de datos móviles	1 día

Nota: Diseño e implementación del comando de voz y validación de datos.
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 9. Detalle Sprint 5

Sprint 5: Diseño e implementación de verificación de datos y pruebas del móvil.	
Duración: 22 días	
Actividad	**Tiempo**
Desarrollo del código de verificación de datos.	10 días
Implementación del módulo de verificación de datos.	2 días
Pruebas por medio de comentarios y estadísticas de usuario, para el buen funcionamiento de la aplicación móvil.	10 días

Nota: Diseño e implementación de verificación de datos y pruebas del móvil
Elaborado por:Daniel Riera y Jorge Soria (2019)

Tabla 10. Detalle Sprint 6

Sprint 6: Diseño y desarrollo de la aplicación web	
Duración: 20 días	
Actividad	**Tiempo**
Diseño y construcción de los casos de uso, para la funcionalidad de la web.	2 días
Desarrollo del modelo conceptual y físico de la base de datos.	3 días
Investigación de las librerías para implementar los frameworks en el código, para el desarrollo de la web.	10 días
Diseño visual de la web con los respectivos frameworks implementados	5 días

Nota: Diseño y desarrollo de la aplicación web
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 11. Detalle Sprint 7

Sprint 7: Ingreso de coordenadas en la base de datos PostGIS	
Duración: 20 días	
Actividad	**Tiempo**
Investigación sobre las parroquias urbanas de Quito.	2 días
Ingreso de las coordenadas geoespaciales de cada parroquia urbana de Quito a la base de datos PostGIS.	15 días
Desarrollo de los mapas utilizando coordenadas geoespaciales.	3 días

Nota: Ingreso de coordenadas en la base de datos PostGIS
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 12. Detalle Sprint 8

Sprint 8: Pruebas finales	
Duración: 22 días	
Actividad	**Tiempo**
La web sometida a diversas pruebas para verificar el buen funcionamiento de la misma.	7 días
Pruebas por medio de comentarios y encuestas al usuario final, para mejorar el buen funcionamiento de la aplicación móvil y web.	15 días

Nota: Pruebas funcionales
Elaborado por: Daniel Riera y Jorge Soria (2019)

Seguimiento del Sprint

Se llevaron a cabo conferencias periódicas para informar los avances y posibles dificultades ya que al no existir antecedentes de un proyecto parecido, se debía desarrollar de la mejor manera para poder finalizar cada uno de los Sprint.

Canales que se utilizaron para poder realizar las reuniones:

- Videoconferencia,
- Llamadas telefónicas.

Revisión del Sprint

Cada entregable fue evaluado por el Product Owner luego de culminar tareas específicas, para recibir retroalimentación.

Tabla 13. Revisión de entregables

Entregable	Fecha
Levantamiento de requerimientos	27/05/2019
Creación del web service	25/06/2019
Diseño e implementación del registro y geolocalización móvil.	19/08/2019
Diseño e implementación del comando de voz y validación de datos.	12/09/2019
Diseño e implementación de verificación de datos y pruebas del móvil.	22/10/2019
Diseño y desarrollo de la aplicación web	16/10/2019
Ingreso de coordenadas en la base de datos PostGIS	14/11/2019
Pruebas	12/12/2019

Nota: Revisión de entregables
Elaborado por: Daniel Riera y Jorge Soria (2019)

CAPÍTULO I

1. MARCO TEÓRICO

1.1. ¿Qué son los baches?

Un bache por definición es un agujero u hoyo que son de unos 14 cm de ancho como mínimo, estos se encuentran en los caminos o carreteras.

Se forman por numerosas causas, por ejemplo, debido a filtraciones de agua que entran por las grietas que ocasionan que la capa asfáltica pierda su estructura y con el tiempo se crean agujeros y a medida que los vehículos circulan van rompiendo los bordes creando grandes baches. (Ramirez Ordaz, 2007)

Bache en la ciudad de Quito DM

Figura 1. Bache o agujero *(Merizalde, 2017)*
Elaborado por: María Belén Merizalde, EL COMERCIO

Se pueden clasificar en:

Tabla 14. Clasificación de los baches

Clasificación de los baches	
Daño menor	Menor a 4.00 cm de profundidad y unos 14.00 cm de ancho
Daño medio	De entre 4.00 a 14.00 cm de profundidad y mayor a 14.00 cm de ancho
Daño alto	Mayor a 14.00 cm de profundidad y una anchura considerable

Nota: Clasificación de los baches (Utrilla, 2007)
Elaborado por: Daniel Riera y Jorge Soria (2019)

1.2. ¿Qué problemas causan los baches?

La presencia de baches en las calles, vías y carreteras de la ciudad de Quito DM, han ocasionado muchos inconvenientes para los conductores y transeúntes lo cual ha provocado malestar en la ciudadanía, por tal razón, se han elevado los costos de mantenimiento, consumo de combustible y el desgaste de neumáticos, hecho que puede ocasionar accidentes de tránsito que puede ocasionar accidentes de tránsito.

1.3. Datos abiertos

1.3.1. ¿Qué son los datos abiertos?

Son datos alcanzables, expuestos públicamente sin ninguna confidencialidad que pueden ser utilizados, reutilizados y compartidos independientemente por cualquier individuo. (SNI, 2014)

1.3.2. ¿Por qué es importante liberar datos?

La publicación de datos abiertos responde a un derecho de toda la ciudadanía de conocer, participar y producir valor a través de la innovación que pueden llegar a ser un actor fundamental para validar la información y generar ideas, proyectos, investigación o un emprendimiento. (SNI, 2014)

1.3.3. ¿Qué es el Gobierno Abierto?

Es una estrategia innovadora que busca la colaboración de las o los ciudadanos con el Gobierno para estar mutuamente informados a través de la transparencia mejorando así el desarrollo colectivo de soluciones a los problemas de interés público y el mejoramiento de servicios públicos por medio de plataformas de gestión de la información. (DMQ, 2019)

1.4. Scrum

Scrum fue desarrollado por Jeff Sutherland y Ken Schwaber, esta herramienta ha llegado a ser conocida por gestionar proyectos de desarrollo de forma productiva dando resultados en períodos cortos de tiempo. Scrum permite realizar entregas del proceso de manera parcial, calendarizada y priorizadas para que de cumplimiento al proyecto; logrando optimizar tiempo de forma inmediata y correcta. Scrum es utilizado en proyectos que necesitan arrojar resultados a corto plazo y en los cuales el entorno sea complejo. (Pérez Arbesú, 2015).

Los equipos de Scrum son auto – organizados y multifuncionales, esto garantiza la unión de los miembros para realizar las actividades en equipo, esto da como resultado el valor y responsabilidad de cada uno de ellos.

Como método, encomienda al equipo la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivos posibles, resaltando los roles, valores, prácticas de desarrollo, prácticas de gestión, implementación y demás cuestiones técnicas. (Toapanta Chancusi, 2012)

La Metodología Scrum

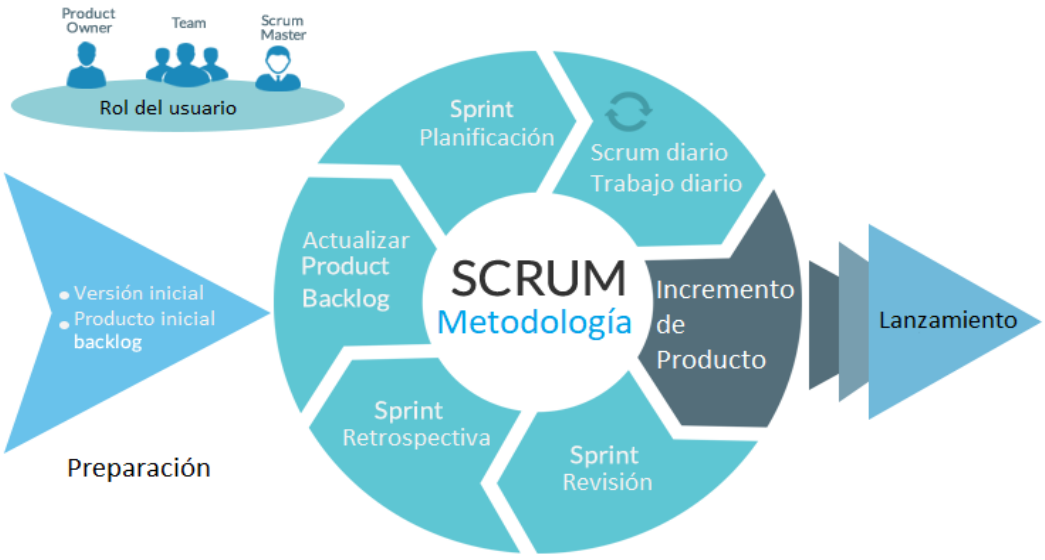


Figura 2. Metodología Scrum *(Technology, 2019)*
Elaborado por: Tech Technology (2019)

1.4.1. Elementos del Scrum

1.4.1.1. Roles

Product Owner (Dueño del producto)

Es el responsable oficial de tomar las decisiones finales sobre el proyecto, el trabajo del Equipo de desarrollo y gestiona la Lista del Producto (Product Backlog). (Toapanta Chancusi, 2012)

Las áreas de responsabilidad son:

- Requerimientos del sistema
- Lanzamiento del proyecto

Scrum Master (Líder del proyecto)

Es el responsable de coordinar los encuentros diarios con el equipo, de resolver los problemas y cumplir con las metas de acuerdo a las buenas prácticas, valores y reglas de Scrum además

de comunicarse con el cliente. Debe ser integrante del equipo y trabajar al igual. (Toapanta Chancusi, 2012)

Development Team (Equipo de desarrollo)

Es un grupo de profesionales auto-organizados y multifuncionales que están conformados de entre 3 a 9 integrantes que trabajan en el producto, independientemente de su rol interno todos son uno sólo. (Vela, 2017)

Tabla 15. Roles de Scrum

Comprometido en el proyecto	Implicados en el proyecto
Dueño del producto	Marketing
Equipo	Comercial
Scrum Master	Etc.

Nota: Los tres roles Product Owner, Scrum Master Y Development Team
Elaborado por:Daniel Riera y Jorge Soria (2019)

1.4.1.2. Poda de requerimientos

Consiste en hacer lo que realmente el cliente desea, la primera acción es armar una lista de requerimientos originales del sistema, evaluar qué requerimientos podrían ser imprescindibles, cuáles pueden postergarse y cuáles pueden eliminarse. (Toapanta Chancusi, 2012)

1.4.1.3. Product Backlog

Es una lista de requerimientos priorizados y probados con los que se arma el Product Backlog, también es considerado un documento que incorpora constantemente las necesidades del sistema, esto se mantiene durante todo el ciclo de vida y es responsabilidad del Product Owner. (Toapanta Chancusi, 2012)

1.4.1.4. Sprint

Es el período de tiempo que se desarrolla un incremento de funcionalidad.

Aspectos a tener en cuenta en un Sprint:

- La duración máxima del Sprint es de 30 días.
- Durante el Sprint no se puede modificar el trabajo que se ha acordado en el Backlog
- Puede interrumpir un Sprint sólo el Scrum Master si decide que no es viable por las siguientes razones:
    - La tecnología establecida no funciona.
    - La situación del negocio cambia.
    - El equipo ha tenido inconvenientes. (Toapanta Chancusi, 2012)

1.5. Herramientas y lenguajes

1.5.1. Android Studio

Es un sistema operativo de código abierto para dispositivos móviles se desarrolla en Java que es un lenguaje familiar y amigable para usarlo, y su núcleo está basado en Linux. (CCIA, 2014)

1.5.2. Aplicación móvil

La aplicación móvil es un software diseñado para dispositivos móviles que ayuda a solucionar problemas en la vida cotidiana. (Cuello & Vittone, 2017)

En la actualidad las aplicaciones tienen gran variedad en tipo, forma y color, las principales compañías que las distribuyen son App Store, Google Play y Windows Phone Store para las distintas plataformas existentes como Android, iOS, BlackBerry OS, Windows Phone.

Las aplicaciones móviles han ayudado a generar nuevas formas de negocios a nivel internacional haciéndoles rentables tanto para sus desarrolladores como para los que los distribuyen. (Cuello & Vittone, 2017).

Plataformas móviles

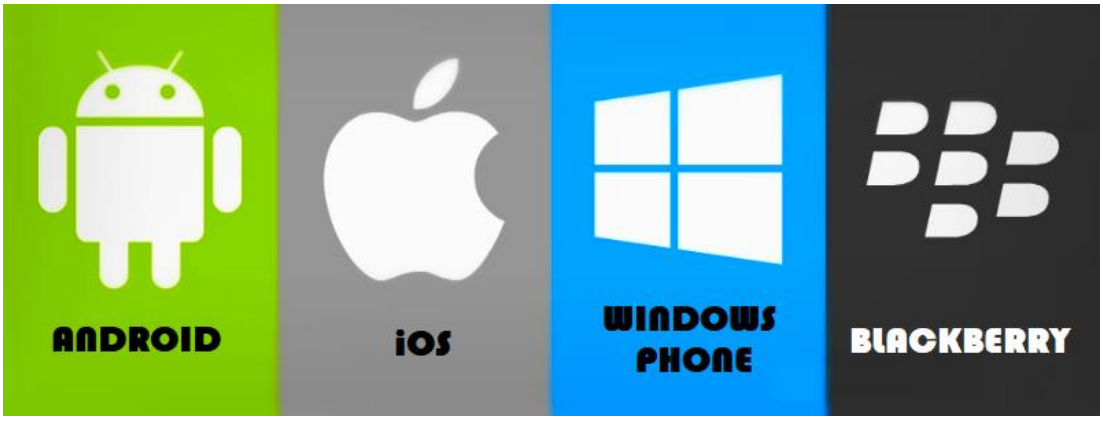


Figura 3. Plataformas existentes de móviles
Elaborado por: Daniel Riera y Jorge Soria (2019)

1.5.3. ¿Qué es la geolocalización?

"Es la capacidad de conocer la posición geográfica (coordenadas) o ubicación de un objeto. Actualmente la gran mayoría de personas poseen al menos un aparato que puede proporcionar esta información". (KZgunea, 2018)

GPS del móvil

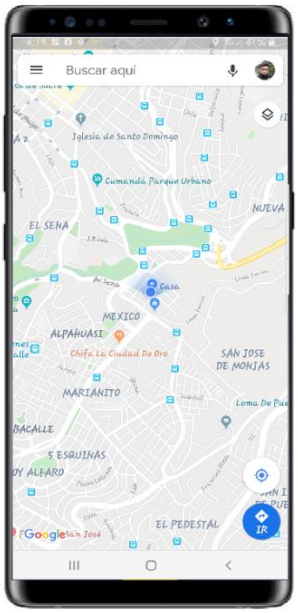


Figura 4. GPS del móvil
Elaborado por: Daniel Riera y Jorge Soria (2019)

1.5.4. Tipos de geolocalización

1.5.4.1. GPS

El GPS emite información sobre la ubicación de un individuo u objeto en cualquier horario y condiciones climáticas. (KZblog, 2017)

1.5.4.2. GSM

Es el sistema global que usa las redes telefónicas. Este sistema es el de menor exactitud y se su margen de error puede llegar hasta 200m. (KZblog, 2017)

1.5.4.3. WIFI (WPS)

Todas las redes wifi tienen un identificador llamado dirección MAC, de esta manera da a conocer el lugar en el que se encuentra el móvil u ordenador y el margen de error es menor al del sistema anterior. (KZblog, 2017)

1.5.5. Herramientas y lenguajes de programación

Las herramientas y los lenguajes de programación son una serie de instrucciones que cumplen un objetivo, un resultado o acción deseada. (Content, 2019)

1.5.5.1. Java

Java es el lenguaje de programación que está orientado a objetos, el mismo que se ejecuta en casi todas las plataformas existentes, cuenta con la Máquina Virtual Java (JVM), que ayuda a los programadores a escribir el código una sola vez. (Guevara Benites, 2016)

1.5.5.2. Prime Faces

Es una librería de componentes visuales de código abierto, que permite mejorar considerablemente el diseño de nuestras interfaces de manera clara, ligera y muy reutilizable.

1.5.5.3. Java Hibernate

Es un framework, que permite agilizar a los desarrolladores a ingresar, consultar, actualizar y eliminar datos entre la base de datos y el modelo de objetos de una aplicación, mediante

entidades que ayudan a estas relaciones, como los archivos declarativos (XML). (Guerrero, 2015)

1.5.5.4. Wildfly

Es un servidor de código abierto que ofrece una multiplataforma de alto rendimiento a todo sistema operativo que disponga de Java. (Arsys, 2017).

Cuenta con las siguientes características:

- Implementación fácil de realizar
- Escalabilidad
- Basado en estándares
- Fácilmente testable
- Basado en los proyectos de código abierto. (Arsys, 2017)

1.5.5.5. Bootstrap 4

Es un framework versión 4 de código abierto para desarrollar con HTML, CSS y JS, que facilita la estandarización de sitios web mediante estilos por defecto y haciéndoles adaptables para móviles, tabletas y ordenadores, con la biblioteca de componentes de interfaz (front-end) más popular del mundo. (Bootstrap, 2019)

1.5.5.6. PostgreSQL

PostgreSQL es una base de datos relacional con una arquitectura robusta y de alto rendimiento que brinda confiabilidad, integridad de datos y un conjunto de características extensibles como la base de datos geoespaciales PostGIS. (PostgreSQL, 2019)

1.5.5.7. PostGIS

Es un Software compatible con Open Geoespacial Consortium (OGC) utilizado como una extensión para PostgreSQL puede ser considerado como una base de datos objeto-relacional: entre sus ventajas PostGIS es libre y de código abierto, es similar al SQL y permite realizar análisis espaciales y consultas típicas sobre datos con relativa facilidad. (Morales, 2016)

1.5.5.8. HTML 5

El Lenguaje de marcado de hipertexto versión 5 (HTML5) es usado para estructurar y presentar el contenido para la web, se considera un sistema para formatear el diseño de las páginas, también realiza algunos ajustes a su aspecto. (Garro, 2014)

1.5.5.9. Lenguaje de hoja de estilo CSS

Es un lenguaje que determina el estilo de los documentos HTML y abarca opciones relativas a fuentes, colores, márgenes, líneas, altura, anchura e imágenes de fondo, entre otros y es compatible con todos los navegadores actuales. (Hernandez)

1.5.5.10. Arquitectura MVC

Es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario, se potencia para que dé facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, conocidas como: Modelos, Vistas y Controladores. (Alvarez, 2014)

Modelo, vista y controlador (MVC)

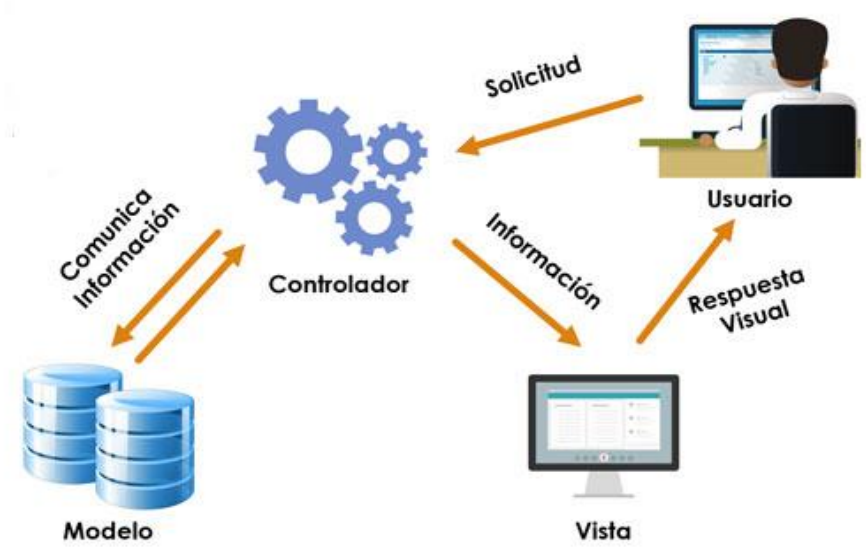


Figura 5. Modelo, vista y controlador (MVC) *(Alvarez, 2014)*
Elaborado por: Miguel Angel Alvarez, desarrolloweb.com

1.5.5.11. MVC en aplicaciones web

- Vista: la página HTML5.
- Controlador: código que obtiene datos dinámicamente y genera el contenido HTML5.
- Modelo: la información almacenada en una base de datos. (Pavón Mestras)

1.5.5.12. Componentes de la Arquitectura MVC

- **Modelo:** se encarga el desarrollador de manipular, innovar y gestionar la información utilizando una base de datos (Alvarez, 2014)
- **Vista:** se encarga de mostrar al usuario final las pantallas, ventanas, páginas y formularios además se encarga de la interfaz; la programación de la interfaz de usuario o de la visualización de las páginas web *(CSS, HTML, HTML5 y JavaScript)*. (Alvarez, 2014)
- **Controlador:** es la secuencia de eventos que captura y propaga los sucesos a la vista y al modelo. (Pavón Mestras)

1.5.6. Definición de base de datos

Una base de datos es un almacén que permite reunir grandes cantidades de información relacionada que se encuentra agrupadas o estructuradas. (Suarez Jaimes, 2015)

1.5.6.1. Manejo de bases de datos geo espaciales

La base de datos geo-espacial permite el almacenamiento de los registros geométricos dentro de la misma la cual dispone de funcionalidades para consultar y recuperar la infomación mencionada. (Llario, 2018)

PostGIS

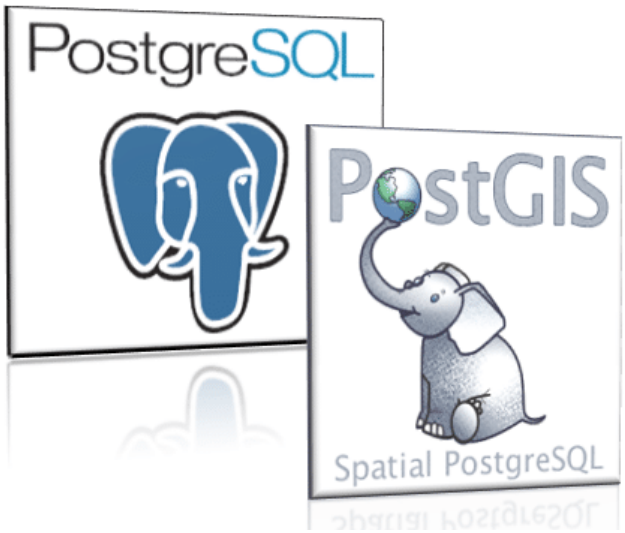


Figura 6. PostGIS

Elaborado por: Diana Alonso Aransay (Alonso Aransay, 2010)

Algunas de las principales características son:

- PostGIS es software libre y gratuito distribuido bajo licencia GNU.
- Permite importar y exportar datos.

1.5.6.2. Modelo entidad-relación

Los modelos entidad-relación es una herramienta para el modelado de datos y expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades. (Hidalgo Pérez, 2017)

1.5.6.3. Diagrama UML

El lenguaje de modelado unificado UML, permite representar visualmente la arquitectura, el diseño y la implementación de sistemas de software complejos. (Team, 2019)

- **Include:** Comparte entre varios casos de uso una funcionalidad común.
- **Extends:** Modela un subflujo que sólo se ejecuta bajo ciertas condiciones o varios flujos que se pueden insertar en un punto determinado.

1.5.6.4. Tipos de diagramas en UML

La utilización de UML permite crear varios tipos de diagramas:

- **Diagramas de casos de uso:** Intervienen en un desarrollo de software y representan a los actores.
- **Diagramas de clases:** Representan los conceptos que intervienen en un problema.
- **Otros diagramas:** Diagramas de estados, diagramas de colaboración, diagramas de secuencia, diagramas de paquetes, diagramas de arquitectura software, etc. (Krall, 2006)

1.5.7. Comando de voz Pocketsphinx

Es una librería para el reconocimiento de voz que permite realizar operaciones mediante un diccionario de palabras claves (en inglés) de forma más rápida, sencilla y constante. (JesSalvTech, 2016)

1.5.8. API REST

"Es el estándar más lógico, eficiente y habitual en la creación de APIs para servicios de Internet". (BBVAOPEN4U, 2016)

CAPÍTULO II

2. ANÁLISIS Y DISEÑO

2.1. Análisis

Se analizó toda la información obtenida de esta investigación y se logró determinar la viabilidad y funcionalidad del aplicativo móvil “Bache” mismo que servirá de ayuda para una correcta movilidad tanto vehicular y peatonal, brindando alertas anticipadas con el fin de evitar incidentes ocasionados por los baches en el sistema vial.

2.2. Casos de uso

Para mejorar los casos de uso, son necesarios los diagramas de casos de uso, los cuales permiten la descripción de los roles, actividades, acciones, componentes y particularidades de cada usuario. Estos perfiles se emplean para la aplicación móvil y web.

A continuación, se explican los casos de uso para este proyecto.

2.2.1. Caso de uso para los ciudadanos que utilizarían la aplicación web.

Caso de uso para los ciudadanos que utilizarían la aplicación web

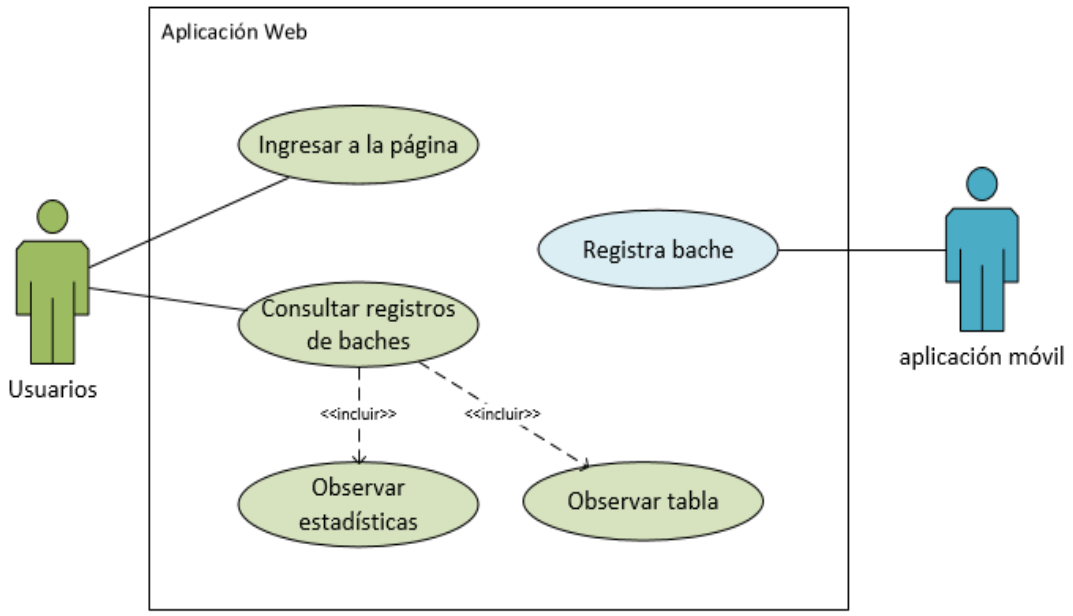


Figura 7. Caso de uso para los ciudadanos que utilizarían la aplicación web
Elaborado por: Daniel Riera y Jorge Soria (2019)

Mediante el diagrama de la Figura 7 se demuestra cómo el usuario podrá ingresar a la aplicación de manera gratuita y la información generada indica la ubicación de baches anteriormente registrados por medio de la aplicación móvil en Quito DM, la aplicación web mostrará mapas interactivos con tablas detallando la información y cuadros estadísticos que demostrarán qué parroquia tiene más problemas de baches presentes en sus calles.

2.2.2. Caso de uso para la utilización de la aplicación móvil

Caso de uso para la utilización de la aplicación móvil

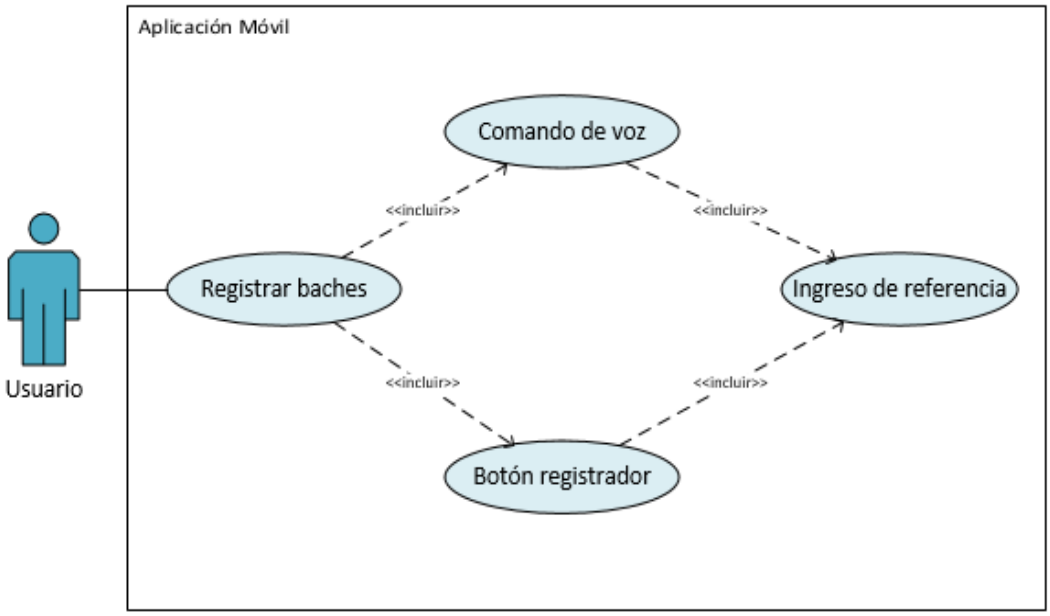


Figura 8. Caso de uso para la utilización de la aplicación móvil
Elaborado por: Daniel Riera y Jorge Soria (2019)

Mediante el diagrama de la Figura 8 el usuario podrá registrar la ubicación de los baches mediante dos opciones:

- Utilización del comando de voz con la palabra clave “Bache” para que sea fácil de registrar al momento de conducir y así evitar accidentes por distracciones.
- Presionando el botón “REGISTRAR BACHE”

2.2.3. Caso de uso para el Administrador

Caso de uso para el Administrador

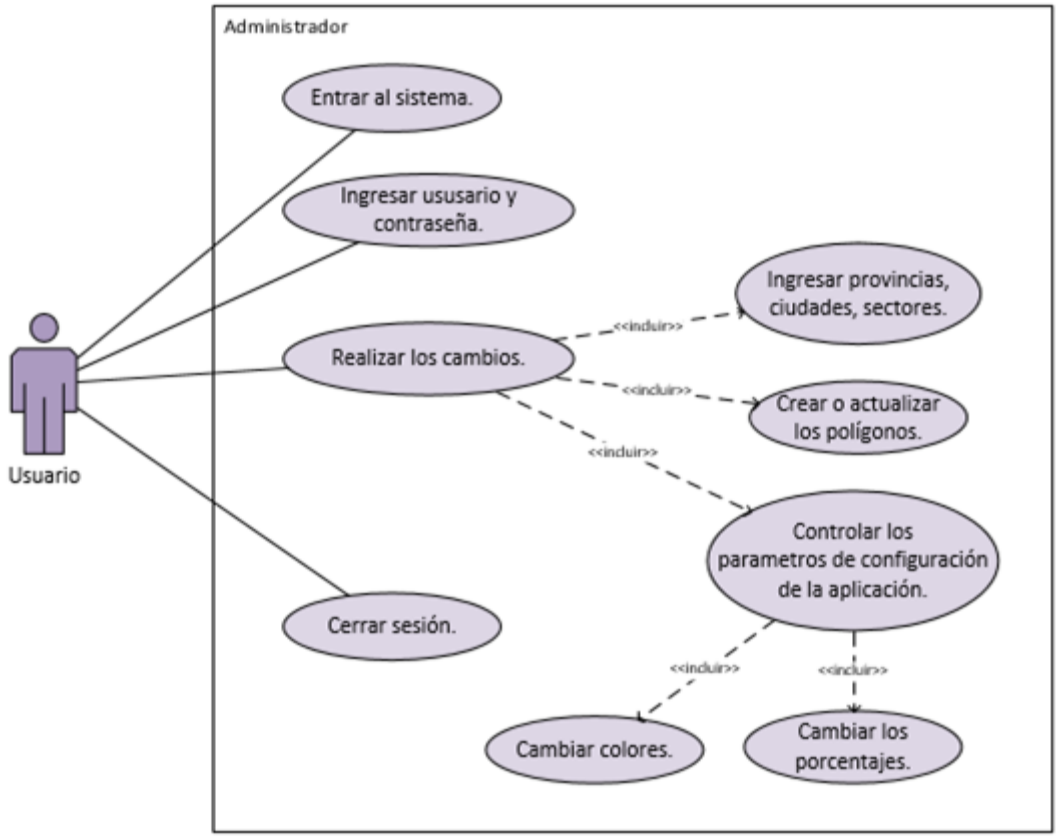


Figura 9. Caso de uso para el Administrador
Elaborado por: Daniel Riera y Jorge Soria (2019)

Por medio del diagrama de la Figura 9, el administrador tendrá acceso al sistema informático mediante el ingreso de sus credenciales: nombre de usuario y contraseña, en el cual podrá gestionar todos los cambios que deba realizar como:

- Ingresar provincias, ciudades y sectores.
- Crear o actualizar polígonos.
- Controlar los parámetros de configuración de la aplicación.

2.3. Resumen de las tablas que contienen la base de datos

2.3.1. Base de datos para la aplicación web

Se determina el diagrama para el buen funcionamiento de la estructura de la base de datos de la aplicación, el correcto manejo de la información entre la aplicación móvil y web, las mismas consumen información la una de la otra.

Diagrama de la base de datos en la aplicación web

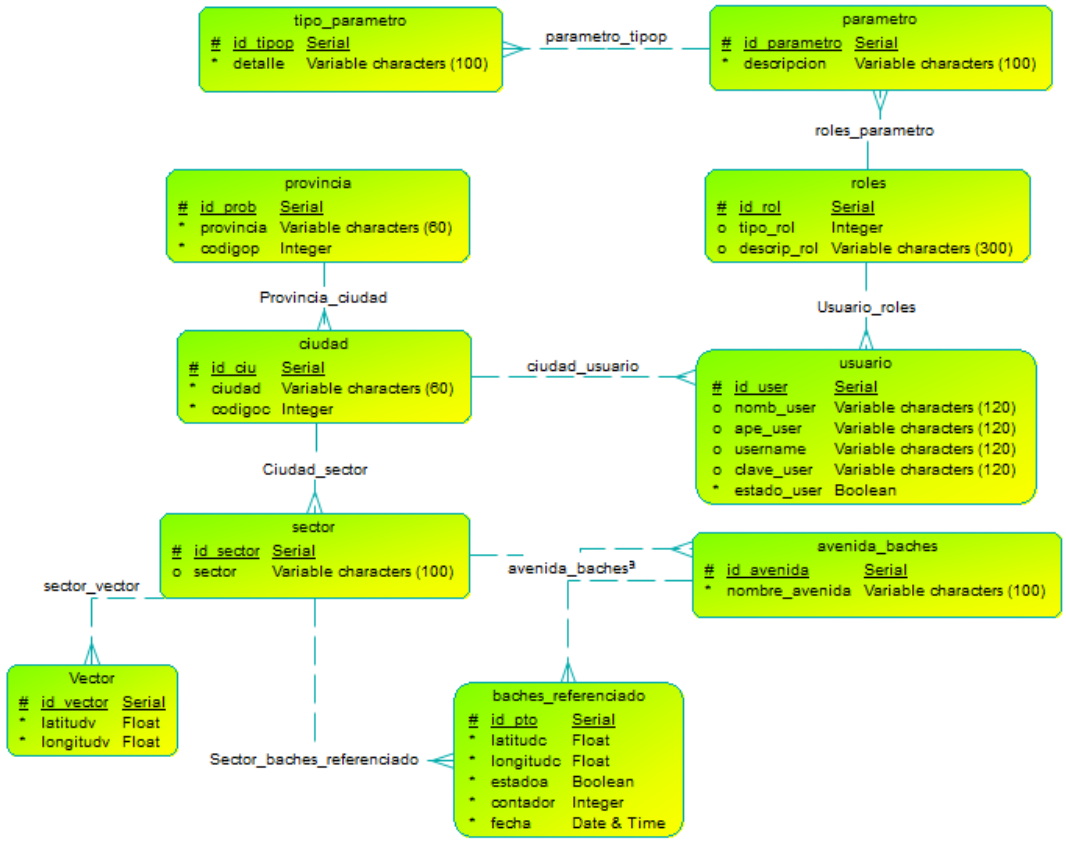


Figura 10. Diagrama de la base de datos en la aplicación web
Elaborado por: Daniel Riera y Jorge Soria (2019)

Se puede observar en el diagrama de la Figura 10 que está diseñado bajo un análisis que permite que la aplicación escale a varias ciudades del país y no sólo al Distrito Metropolitano de Quito.

Tabla 16. Diccionario de la base de datos en la aplicación web

Aplicación web		
Tablas	**Descripción**	**Campo**
Avenida_baches	Contiene la dirección dónde se registro el bache	• Id_avenida • Nombre_avenida
Ciudad	Contiene las ciudades disponibles en la aplicación	• Id_ciu • Ciudad • Codigoc
Provincia	Contiene las provincias disponibles en la aplicación	• Id_prob • Provincia • Codigop
Sector	Esta tabla contiene todas las parroquias urbanas de la ciudad de Quito DM.	• Id_sector • Sector
Vector	En esta tabla contiene las coordenadas que delimitan a cada parroquia de la ciudad de Quito DM.	• Id_vector • Latitudv • longitudv
Bache_referenciado	En esta tabla se registran todos los baches con su respectiva ubicación y la fecha en la que se registró.	• id_pto • latitudc • longitudc • estadoa • contador • fecha
Usuario	En esta tabla se registrarán los posibles administradores, ejemplo: la autoridad competente	• Id_user • Nomb_user • Ape_user • Username • Clave_user • Estado_user
Roles	Permite guardar los roles existentes, ejemplo: Administrador	• Id_rol • Descrip_rol
Parámetro	Guarda la descripción de los parámetros de administración de la página web	• Id_p • Descripción
Tipo_parametro	Es donde se almacena las configuraciones para el control de la matriz de la página web.	• Id_tipop • Detalle

Nota: Diccionario de la base de datos en la aplicación web
Elaborado por: Daniel Riera y Jorge Soria (2019)

2.3.2. Base de datos aplicación móvil

Esta diseñado de forma manejable y simple el diagrama que será utilizado en la aplicación móvil para garantizar la eficiencia de la interaccion con la aplicación web.

Diagrama de la base de datos en la aplicación móvil

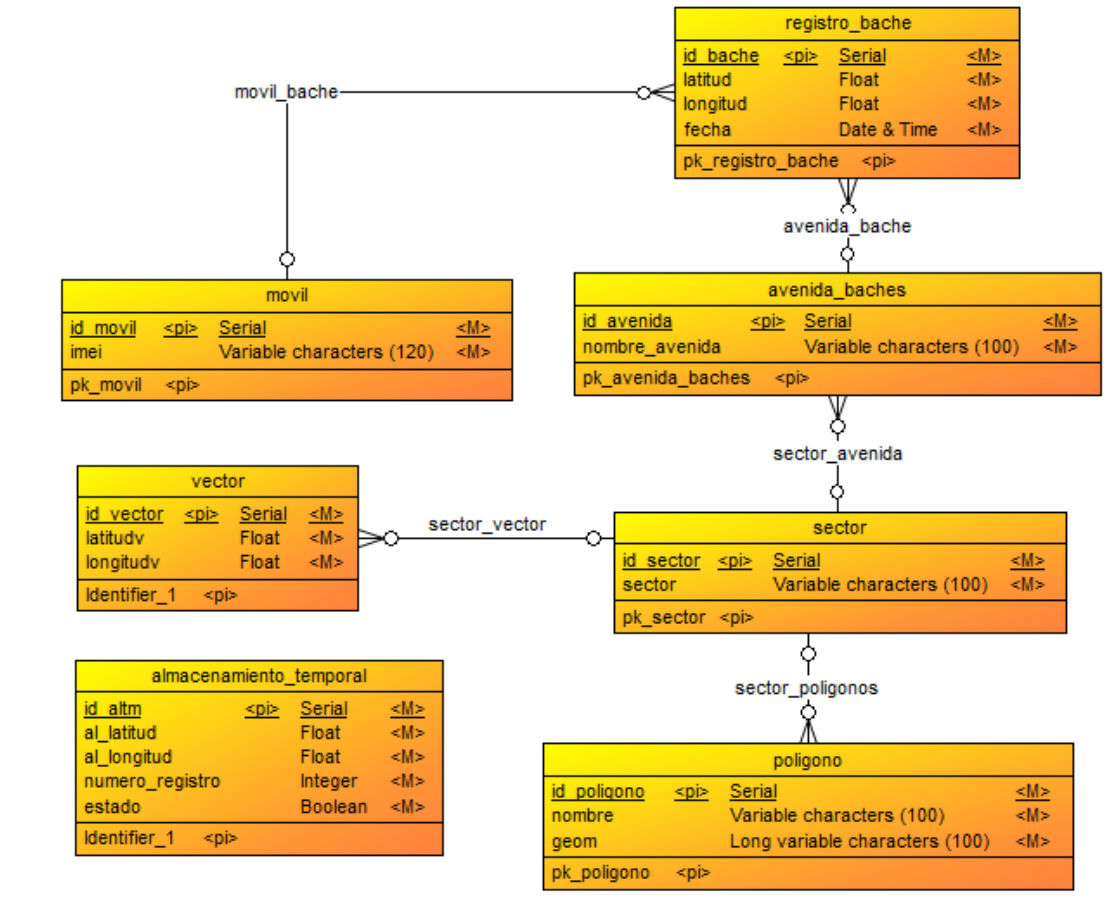


Figura 11. Diagrama de la base de datos en la aplicación móvil.
Elaborado por: Daniel Riera y Jorge Soria (2019)

Como se puede visualizar en el diagrama de la Figura 11, en el proceso de análisis y diseño se incluyó una tabla que contiene las coordenadas que son los datos geoespaciales creados por los polígonos.

Tabla 17. Tablas de la base de datos en la aplicación móvil

Aplicación móvil		
Tabla	**Descripción**	**Campo**
Móvil	Esta tabla guarda registro de los usuarios que están ocupando el teléfono con el código IMEI	• Id_movil • imei
Registro_bache	En esta tabla se registran todos los baches con su respectiva ubicación y la fecha en la que fue registrada.	• id_pto • latitudc • longitudc • fecha
Avenida_bache	Contiene la dirección donde se registró en bache	• Id_avenida • Nombre_avenida
Sector	Esta tabla contiene todas las parroquias urbanas de la ciudad de Quito DM.	• Id_sector • Sector
Vector	En esta tabla contiene las coordenadas que delimitan a cada parroquia de la ciudad de Quito DM.	• Id_vector • Latitudv • longitudv
Polígono	Esta tabla contiene los polígonos de los sectores con las coordenadas de la tabla vector.	• id_poligono • nombre • geom
Almacenamiento_temporal	Permite almacenar los registros temporalmente antes de ingresar a la aplicación web	• Id_altm • Al_latitud • Al_longitud • Numero_registro • Estado

Nota; Las tablas de la aplicación móvil y sus atributos
Elaborado por: Daniel Riera y Jorge Soria (2019)

CAPÍTULO III

3. CONSTRUCCIÓN Y PRUEBAS

3.1. Estándares de programación

La codificación por estandarización y aceptación se mantendrá en el lenguaje de programación clásico de Java, como en los casos de "*getter"* y "*setter",* en la parte funcional se utiliza la nomenclatura tradicional del desarrollador, como es en la escritura de paquetes, clases, variables, métodos, etc. Permite que la lectura pueda ser fácil y entendible para los programadores.

3.2. Paquetes, Clases y Métodos.

3.2.1. Paquetes.

Para nombrar un paquete se debe escribir con minúsculas sin utilizar caracteres especiales en el cual se almacena las clases que contienen los códigos. El nombre de los paquetes se define de la siguiente manera (com.baches), en el cual no contendrá ninguna clase Java. Otro nivel extra del paquete se determina por el nombre del proyecto o modulo. (AMAP, 2014)

Definición de los paquetes

Figura 12. Definición de los paquetes
Elaborado por: Daniel Riera y Jorge Soria (2019)

Ejemplo de paquetes

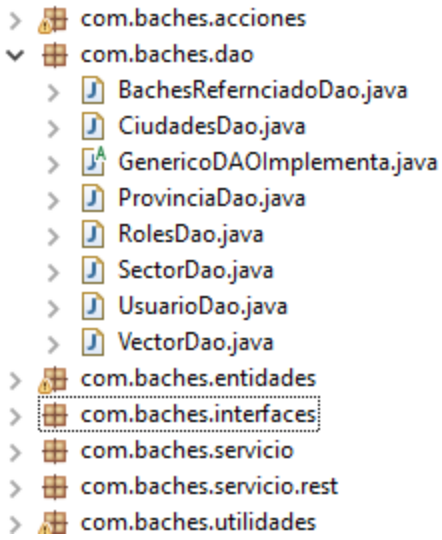


Figura 13. Ejemplo de paquetes
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.2.2. Interfaces.

Las interfaces utilizan el mismo sufijo del lenguaje de programación Java Interface y estarán compuestas por una palabra que la primera letra esta mayúscula (GenericoDAOInterface). Se debe evitar el uso de abreviaciones que dificulten la compresión del código.

Ejemplo de interfaces

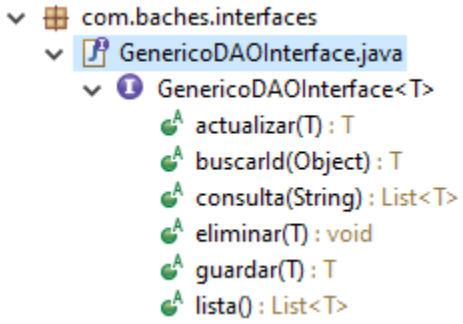


Figura 14. Ejemplo de interfaces
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.2.3. Clases.

El nombre de las clases debe estar compuesto por mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúscula (CiudadAccion).

Estas clases se deben mantener simples, descriptivas, con palabras completas y de esta forma evitar acrónimos y abreviaturas. Si la clase cumpliera con una funcionalidad específica se recomienda definirlo con el nombre.

Ejemplo de clases

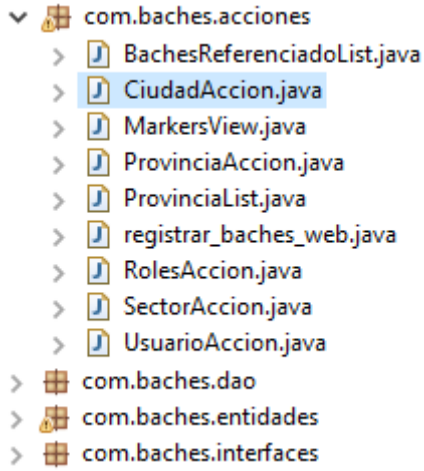


Figura 15. Ejemplo de clases
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.2.4. Métodos

El nombre de los métodos debe ser verbos en infinitivo simple y descriptivo sin importar la longitud del mismo, compuesto con la primera letra en minúscula y la primera letra de cada palabra interna en mayúscula.

Ejemplo de métodos get y set

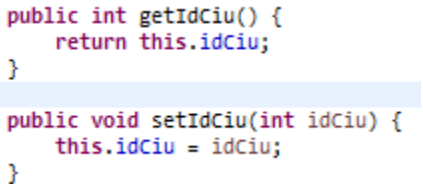


Figura 16. Ejemplo de métodos get y set
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.3. Desarrollo de Paquetes

La estructura de un árbol de paquetes que se desarrollaron para la aplicación web se define de la siguiente manera:

Tabla 18. Paquetes de la aplicación web

Paquete	Descripción
com.baches.acciones	En las acciones se encuentran los métodos que llaman los objetos con la información que se desea que se visualice en la aplicación.
com.baches.dao	Se encarga del acceso a los datos y abarca todos los métodos CRUD (create, read, update, delete), generalmente se tiene un DAO para cada tabla en la base de datos.
com.baches.entidades	Son las clases que se representan las tablas de la base de datos.
com.baches.interfaces	Se describen las operaciones que una entidad puede realizar, de igual manera establece los límites, niveles de acceso y la manera en que se desarrolla la comunicación entre dos entidades.
com.baches.servicio	Se insertan o actualizan los datos de la base de datos.
com.baches.servicio.rest	Es dónde se localizan las rutas para realizar las consultas al web service.
com.baches.utilidades	Se encuentran las clases que contienen los métodos estáticos, código que no se puede cambiar.
WEB-INF	Es este se encuentra la interfaz gráfica de la aplicación web.

Nota: Los paquetes de la aplicación web y sus características
Elaborado por: Daniel Riera y Jorge Soria (2019)

El paquete WEB-INF está definido con mayúsculas porque es un estándar del lenguaje de programación Java al desarrollar un proyecto web.

3.4. Desarrollo de Clases

La estructura de un árbol de clases que se desarrollaron para los diferentes paquetes antes mencionados de la aplicación web se define de la siguiente manera:

Tabla 19. Paquete com.baches.acciones

com.baches.acciones	
Nombre Clase	**Descripción**
BachesReferenciadoAcción.java	En esta clase están los objetos que se vinculan con la aplicación.
BachesReferenciadoList.java	Se ejecuta la consulta de la lista o listas de las tablas correspondientes a la base de datos.
ChartView.java	Representa la estadística porcentual existente de baches de cada parroquia del Distrito Metropolitano de Quito mediante barras verticales.
ChartView1.java	Figura la estadística porcentual existente de baches de cada parroquia del Distrito Metropolitano de Quito mediante un pastel gráfico.
CiudadAcción.java	En esta clase están los objetos que son llamados por el método para vincularse con la aplicación.
MarkersView.java	Se representan los baches por medio de marcadores que señalan las coordenadas en el mapa de Quito.
PolygonsView.java	Por medio de esta función se puede delimitar las parroquias de la ciudad de Quito utilizando coordenadas.
PolylinesView.java	Se puede delimitar mediante líneas las parroquias de la ciudad de Quito utilizando coordenadas.
ProvinciaList.java	Se realiza la consulta de la lista o listas de la tabla correspondiente a la base de datos.
RolesAccion.java	En esta clase están los objetos que son llamados por el método para vincularse con la aplicación.
SectorAccion.java	Los objetos son llamados por el método para vincularse con la aplicación.
UsuarioAccion.java	En esta clase están los objetos que son llamados por el método para vincularse con la aplicación.

Nota: Clases del paquete com.baches.acciones
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 20. Paquete com.baches.dao

com.baches.dao	
Nombre Clase	**Descripción**
BachesReferenciadoDao.java	Aquí se puede efectuar las diferentes consultas de acuerdo a lo que se necesite para la tabla BachesReferenciado.

CiudadesDao.java	Se puede efectuar las diferentes consultas de acuerdo a lo que se necesite para la tabla ciudades.
GenericoDAOImplementa.java	En esta clase se puede realizar las diferentes consultas de acuerdo a lo que se necesite para la tabla GenericoDAOImplementa.
ProvinciaDao.java	En este paso se puede llevar a cabo las diferentes consultas de acuerdo a lo que se necesite para la tabla provincia.
RolesDao.java	Aquí se puede realizar las diferentes consultas de acuerdo a lo que se necesite para la tabla roles.
SectorDao.java	Se realiza las diferentes consultas de acuerdo a lo que demande la tabla sector.
UsusarioDao.java	Se ejecuta las diferentes consultas para la tabla del usuario.
VectorDao.java	Se realiza las diferentes consultas para la tabla vector.

Nota: Clases del paquete com.baches.dao
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 21. Paquete com.baches.entidades

com.baches.entidades	
Nombre Clase	**Descripción**
AvenidaBaches.java	Contiene los atributos tales como constructor, getter y setter de la tabla avenida_bache.
BachesReferenciado.java	Contiene los atributos tales como constructor, getter y setter de la tabla baches_referenciado.
Ciudad.java	Contiene los atributos, constructor, getter y setter de la tabla ciudad.
Provincia.java	Contiene los atributos, constructor, getter y setter de la tabla provincia.
Roles.java	Contiene los atributos, constructor, getter y setter de la tabla roles.
Sector.java	Contiene los atributos, constructor, getter y setter de la tabla sector.
Usuario.java	Contiene los atributos, constructor, getter y setter de la tabla usuario.
Vector.java	Contiene los atributos, constructor, getter y setter de la tabla vector.

Nota: Clases del paquete com.baches.entidades
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 22. Paquete com.baches.interfaces

com.baches.interfaces	
Nombre Clase	**Descripción**

GenericoDAOInterface.java	En este paso se describen las operaciones que una entidad puede realizar, de igual manera establece los límites, niveles de acceso y la manera en que se desarrolla la comunicación entre dos entidades.

Nota: Clases del paquete com.baches.interfaces
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 23. Paquete com.baches.servicio

com.baches.servicio	
Nombre Clase	**Descripción**
BachesReferenciadoServicio.java	Se introduce o actualiza la información de la tabla baches_referenciado.
CiudadesServicios.java	Se incluye o actualiza la información de la tabla ciudad.
ProvinciaServicios.java	Se insertan o actualiza la información de la tabla provincia.
RolesServiocio.java	Se introduce o actualiza la información de la tabla roles.
SectorServicio.java	Se inserta o actualiza la información de la tabla sector.
UsuarioServicio.java	Se incluye o actualiza la información de la tabla usuario.
VectorServicios.java	Se inserta o actualiza la información de la tabla vector.

Nota: Clases del paquete com.baches.servicio
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 24. Paquete com.baches.servicio.rest

com.baches.servicio.rest	
Nombre Clase	**Descripción**
CiudadRest.java	Se encuentra las rutas para realizar las consultas al web service de la tabla ciudad.
ProvinciaRest.java	Se localizan las rutas para realizar las consultas al web service de la tabla provincia.

Nota: Clases del paquete com.baches.servicio.rest
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 25. Paquete com.baches.utilidades

com.baches.utilidades	
Nombre Clase	**Descripción**
Autentificador.java	Se encuentran los métodos estáticos de autenticado.

HibernateUtil.java	Se encuentran los métodos estáticos de HibernateUtil.
PlantillaEstadistica.java	Se encuentran los métodos estáticos de PlantillaEstadistica.
PlantillaVectores.java	Se encuentran los métodos estáticos de PlantillaVectores.

Nota: Clases del paquete com.baches.utilidades
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 26. Interfaz gráfica de la aplicación web

WebContent	
Nombre Clase	**Descripción**
Inicio.xhtml	Se ubica la interfaz gráfica de la aplicación web.
Login.xhtml	Se localiza la interfaz gráfica del inicio de cesión (login), para los posibles administradores de la aplicación web.
Administrador.xhtml	Se localiza la interfaz gráfica para los posibles administradores de la aplicación web.

Nota: Clases de interfaz gráfica de la aplicación
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.5. Arquitectura.

3.5.1. Arquitectura aplicación móvil y web service.

Arquitectura aplicación móvil y web service

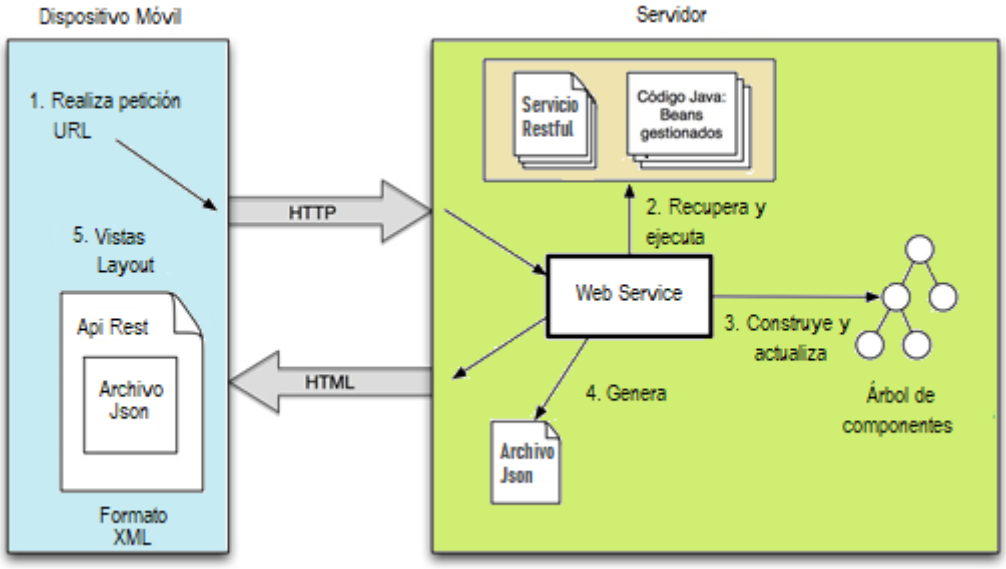


Figura 17. Arquitectura aplicación móvil y web service
Elaborado por: Daniel Riera y Jorge Soria (2019)

La comunicación es representada entre la aplicación móvil y el servidor los cuales interactúan de la siguiente manera: El usuario envía la información con una petición (Post), el cual se recibe por un web service API REST, en donde se utiliza un método para el procesamiento y almacenamiento de la información en una base de datos. Por ejemplo: El usuario envía las coordenadas geográficas de los puntos en donde existe baches en las calles de la ciudad utilizando el comando de voz “Bache” o el botón “REGISTRAR BACHE”.

3.5.2. Arquitectura aplicación web

Para el diseño del aplicativo se empleó el patrón MVC, que permite separar la lógica de la aplicación de una manera organizada y escalable.

Arquitectura aplicación web

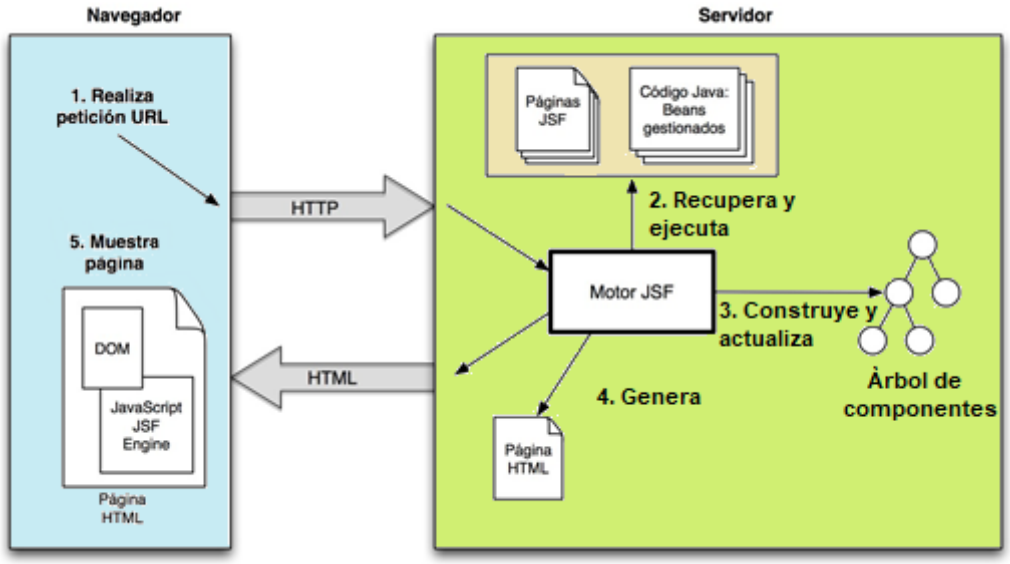


Figura 18. Arquitectura aplicación web
Elaborado por: Dpto. de Ciencia de la computaciòn e ingenieria artificial (Alicante, 2014)

Representa la comunicación entre la interfaz con el proceso interno de la aplicación web los cuales se comunican de la siguiente manera: el controlador procesa todas las peticiones que tienen afectación a la base de datos. El modelo está representado por las entidades que son las interpretaciones de las tablas de la base de datos PostgreSQL de la aplicación. El modelo realiza las inserciones, consultas, ediciones o eliminación de la información de la aplicación, para esto, se utiliza objeto/relacional (ORM) una herramienta de mapeo para entornos Java Hibernate.

3.6. Diagrama de clase

3.6.1. Diagrama de clase de la aplicación web

Se muestran las clases necesarias para las operatividades antes mencionadas que cada clase va a obtener de acuerdo a las necesidades de la aplicación web.

Diagrama de clase de la aplicación web

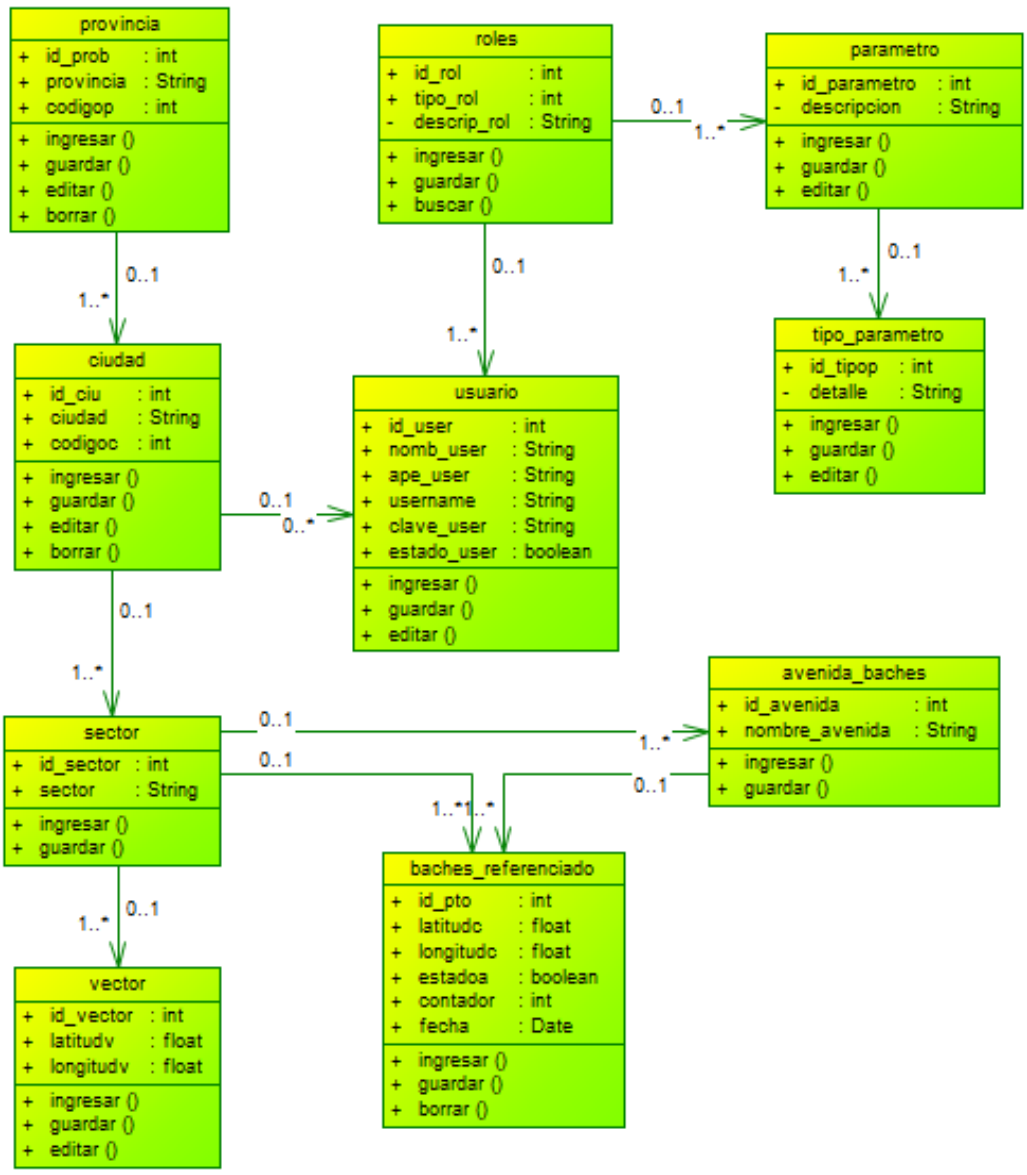


Figura 19. Diagrama de clase de la aplicación web
Elaborado por: Daniel Riera y Jorge Soria (2019)

Clase provincia: en esta se encuentra la provincia en la cual se va a realizar el procedimiento de registro de ubicación de baches en este caso Pichincha.

Clase ciudad: en esta se encuentra el cantón Quito de la provincia antes mencionada donde se va a realizar el proceso de registro de ubicación de baches utilizando comandos de voz y geolocalización.

Clase sector: en esta se encuentran todas las parroquias urbanas de la ciudad de Quito.

Clase avenida_baches: en esta se almacena las calles que el GPS del dispositivo móvil detecta.

Clase vector: en esta se localiza las coordenadas geoespaciales de cada parroquia urbana de la ciudad de Quito.

Clase baches_referenciado: en esta se registran todos los datos del bache como su ubicación por coordenadas y la fecha en la cual fue registrado.

Clase usuario: en esta se encuentran los usuarios administradores, los cuales pueden ingresar, actualizar y editar la aplicación.

Clase roles: en esta se encuentra el tipo de usuario y que acción puede realizar en la aplicación.

Clase parámetro: en esta se encuentra la descripción de la acción que el usuario administrador va a realizar en la aplicación ejemplo cambio de color de fondo.

Clase tipo_parametro: en esta se encuentra el detalle de la acción antes descrita.

3.6.2. Diagrama de clase de la aplicación móvil

Se visualiza el diagrama de clase con respecto a la aplicación móvil, la misma que expone las características de cada una de ellas: clase, atributos y relaciones de dependencia.

Diagrama de clase de la aplicación móvil

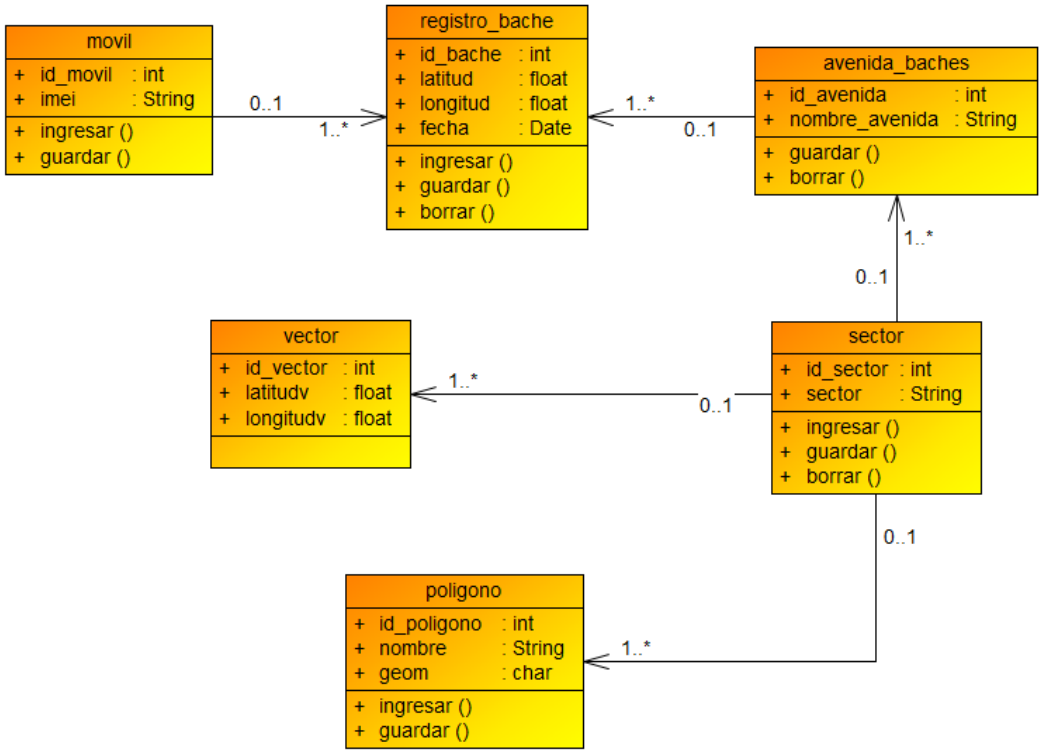


Figura 20. Diagrama de clase de la aplicación móvil
Elaborado por: Daniel Riera y Jorge Soria (2019)

Clase móvil: Contiene el código IMEI del teléfono móvil para registro de todos los dispositivos móviles que tiene instalado la aplicación.

Clase registro_bache: Contiene todos los datos del bache como su ubicación por coordenadas y la fecha en la cual fue registrado.

Clase avenida_baches: Contiene las calles que el GPS del dispositivo móvil detecta.

Clase sector: Contiene todas las parroquias urbanas de la ciudad de Quito.

Clase Vector: Contiene las coordenadas geoespaciales de cada parroquia urbana de la ciudad de Quito.

Clase polígono: Contiene los polígonos de las parroquias que se muestran en el mapa.

3.7. Código relevante de la aplicación web

3.7.1. Función para crear polígonos (función 1)

Esta función permite crear los polígonos de los diferentes sectores de la ciudad utilizando la herramienta PostGIS. Para crear los mismos se necesita que los puntos que forman la geometría del sector estén ingresados en la tabla *vector* de la base de datos, el polígono de cada sector debe de ser cerrado, el punto inicial con el punto final. Al momento de ejecutar la función hay que enviar al identificador del sector (id_sect) y el tipo que permite conocer si se va un nuevo polígono o si se realiza una actualización a lo existente (0 para crear, 1 para actualizar).

Función polígono 1

```
IF tipo = 0 THEN
    /*Contador para determinar que posición de la tabla se encuentra el registro*/
    contador = 1;
```

Figura 21. Función polígono 1
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 21 se detalla que cuando la variable de tipo *integer* es igual a cero ingresa en la condición en dónde se ejecuta el código para crear el polígono de un nuevo sector.

Función polígono 2

```
    /*busca la longitud del la consulta de los registro*/
    longitud = (select count(*) from vector where id_sector = id_sect);
```

Figura 22. Función polígono 2
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 22 se detalla como la variable *longitud* almacena el número de registros que contiene la tabla *vector* por el identificador del sector *id_sector*.

Función polígono 3

```
    /*Veidica si ya exite el poligono de ese sector */
    verificar_sector = (select id_sector from poligonos where id_sector = id_sect);
    IF verificar_sector = 0 OR verificar_sector IS NULL then
```

Figura 23. Función polígono 3
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 23 primero se verifica que no esté creado un polígono del sector seleccionado.

Función polígono 4

```
/*Recore la lista consultada*/
FOR vector_recor IN SELECT longitudv||' '||latitudv AS poligono_vector
    FROM vector WHERE id_sector = id_sect ORDER BY id_vector ASC LOOP
```

Figura 24. Función polígono 4
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 24 se detalla que si no se encuentra un polígono creado previamente, se ingresa en un *bucle* que busca las latitudes y longitudes registradas en la tabla *vector* del nuevo sector el cual tiene que formar un polígono cerrado.

Función polígono 5

```
        /*Verifica y almacena en una variable en formato que reconosca el campo geometry*/
        IF contador = 1 THEN
            poligono_geome = 'POLYGON(('||vector_recor.poligono_vector||',';
        ELSIF  contador > 1 AND contador < longitud THEN
            poligono_geome = poligono_geome||vector_recor.poligono_vector||',';
        ELSIF contador = longitud THEN
            poligono_geome = poligono_geome||vector_recor.poligono_vector||'))';
        END IF;
        contador = contador + 1;

    END LOOP;
```

Figura 25. Función polígono 5
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 25 se detalla como dentro del *bucle* se almacena en una variable de tipo *text* todos los puntos geográficos separados por comas. La variable *contador* permite conocer el registro del inicio del polígono. La variable *longitud* permite conocer el último punto del polígono que forma el sector.

Función polígono 6

```
        /* Inserta los datos en la tabla de poligonos*/
        -- RAISE NOTICE 'polygon: %',poligono_geome;
        INSERT INTO public.poligonos(
            id_poligono,id_sector, nombre, geom)
            VALUES (id_sect,id_sect, 'Polygon', poligono_geome);
        respuesta = 0;
    ELSE
        respuesta = 2;
    END IF;
```

Figura 26. Función polígono 6
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 26 muestra la finalización de esta parte del código que ingresa los datos procesados en la tabla *polígonos* que contendrá la forma geométrica del sector.

Función polígono 7

```
ELSIF tipo = 1 THEN

    contador = 1;
    longitud = (select count(*) from vector where id_sector = id_sect);

    FOR vector_recor IN SELECT longitudv||' '||latitudv AS poligono_vector
        FROM vector WHERE id_sector = id_sect ORDER BY id_vector ASC LOOP
        IF contador = 1 THEN
            poligono_geome = 'POLYGON(('||vector_recor.poligono_vector||',';
        ELSIF  contador > 1 AND contador < longitud THEN
            poligono_geome = poligono_geome||vector_recor.poligono_vector||',';
        ELSIF contador = longitud THEN
            poligono_geome = poligono_geome||vector_recor.poligono_vector||'))';
        END IF;
        contador = (contador + 1);
    END LOOP;
```

Figura 27. Función polígono 7
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 27 se especifica que la variable *tipo* es igual a uno se procesa el código para realizar una actualización al polígono de un sector específico. Este proceso es similar al crear un nuevo polígono ya que cuenta con el mismo algoritmo.

Función polígono 8

```
    /* Actualiza los datos tabla de poligonos por el sector*/
    UPDATE public.poligonos
        SET geom = poligono_geome
        WHERE id_sector =  id_sect;
        respuesta = 1;
END IF;
```

Figura 28. Función polígono 8
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 28 se detalla cómo se puede visualizar la diferencia al crear un nuevo polígono que cambia la forma geométrica de un sector previamente procesado.

3.7.2. Buscar el punto dentro de un sector (función 2)

Esta función permite encontrar el sector al que pertenece el punto de las coordenadas geográficas que fue enviado por el dispositivo móvil. Hay que enviar a la función la latitud y longitud del punto geográfico del bache para que retorne el identificador del sector antes que el servidor almacene la información enviada y si no encuentra el sector en dónde está el bache no se realizará el registro de los datos.

Función buscar el punto dentro de un sector 1

```
begin

    /* Busca el poligono del sector */
    id_sect = 0;
    respuesta = FALSE;
    FOR recor_poligono IN SELECT id_poligono, id_sector, nombre, geom
        FROM poligonos ORDER BY id_poligono ASC LOOP

        respuesta_sector = recor_poligono.geom;
```

Figura 29. Función buscar el punto dentro de un sector 1
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 29 se detalla que la variable *id_sector* de tipo *integer* almacena la clave primaria del sector que pertenecen a los puntos geográficos del bache, la variable *respuesta* permitirá saber sí el bache está dentro del sector.

El *bucle* recorre por todos los polígonos de los sectores que se registraron con anterioridad.

Función buscar el punto dentro de un sector 2

```
/*Se utiliza una funcion para seber si el poligono esta dentro del sector*/
puntos_obtenidos = 'POINT ('||longitud||' '||latitud||')';
```

Figura 30. Función buscar el punto dentro de un sector 2
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 30 se especifica cómo las coordenadas geográficas fueron enviadas desde el web service y son trasformadas a texto para que PostGIS lo pueda procesar.

Función buscar el punto dentro de un sector 3

```
/*Función que busca el punto dentro del sector*/
respuesta = (SELECT ST_Contains(respuesta_sector, puntos_obtenidos));
```

Figura 31. Función buscar el punto dentro de un sector 3
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 31 se muestra cómo la variable *respuesta* almacena la contestación de la función *st_contains*, en donde realiza los cálculos para saber si las coordenadas enviadas pertenecen al sector.

Función buscar el punto dentro de un sector 4

```
        /*Si la respuesta es positiva toma el id del sector*/
        IF respuesta = TRUE THEN
            id_sect = recor_poligono.id_sector;
            RETURN id_sect;
        END IF;

    END LOOP;

    RETURN id_sect;
end;
```

Figura 32. Función buscar el punto dentro de un sector 4
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 32 se especifica que si el retorno de la función es positivo se devuelve el identificador o clave primaria del sector dónde pertenece el bache, en caso de obtener una respuesta negativa durante toda la ejecución del *bucle* la función se convierte en cero.

3.7.3. Ingresar los registros a la web (función 3)

Esta función permite buscar todos los puntos geográficos dentro de un radio de cinco metros, tanto de los registros antiguos como de los registros nuevos, esto suma la concurrencia dentro del rango establecido; además busca la vía al que pertenece los baches sí se encontrarón registrados. En el caso de no estarlo, se procederá a insertar en la base de datos la información de la vía.

Parte de la función para buscar las vías 1

```
consulta = 'SELECT COUNT(*) as num_registro FROM avenida_baches';
num_registros_aba = (SELECT * FROM dblink(condbr,consulta) AS t1(num_registro integer));
conta_ingreso = 1;
IF num_registros_aba = 0 OR num_registros_aba IS NULL THEN
    num_registros_abs = (SELECT COUNT(*) as num_registro FROM avenida_baches);
    IF num_registros_abs > 1 THEN
```

Figura 33. Parte de la función para buscar las vías 1
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 33 se muestra que la variable *consulta* de tipo *text* que guarda el *Query*, el cual busca en la base de datos la tabla *avenida_baches* en la aplicación web.

La variable *num_registros_aba* recibe el número de registros que están ingresados en la tabla *avenidas_baches* de la base de datos en la aplicación web que utiliza el componente *Dblink* para realizar la consulta.

La variable *conta_ingreso* sirve para determinar la posición en la que se encuentra mientras es realizada la ejecución del *bucle* el cual se visualizará su utilidad más adelante.

La primera condición sirve para determinar si hay avenidas registradas en la base de datos de la aplicación web, en caso de no existir datos se ejecuta otra parte del código, la segunda condición sirve para saber si el número de registros es mayor a uno, estas condiciones ayudan a controlar el procesamiento de datos.

Parte de la función para buscar las vías 2

```
FOR av_bache IN SELECT id_avenida, id_sector, nombre_avenida
    FROM avenida_baches ORDER BY id_avenida ASC LOOP
    IF conta_ingreso = 1 THEN
        insertar =  'INSERT INTO avenida_baches(id_avenida, id_sector, nombre_avenida)
                        VALUES ('||av_bache.id_avenida||', '||av_bache.id_sector||', '''
                        ||av_bache.nombre_avenida||'''),';
    ELSIF conta_ingreso > 1 AND conta_ingreso < num_registros_abs THEN
        insertar = insertar||'('||av_bache.id_avenida||', '||av_bache.id_sector||', '''
                        ||av_bache.nombre_avenida||'''),';
    ELSIF conta_ingreso = num_registros_abs THEN
        insertar = insertar||'('||av_bache.id_avenida||', '||av_bache.id_sector||', '''
                        ||av_bache.nombre_avenida||''');';
    END IF;
    conta_ingreso = conta_ingreso + 1;
END LOOP;
```

Figura 34. Parte de la función para buscar las vías 2
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 34 se especifica que durante la ejecución del *bucle* se toma todas las avenidas recién ingresadas a la base de datos del web service para almacenarlos en la variable *insertar* que es de tipo *text*. La variable *conta_ingreso* ayuda a controlar el orden de almacenamiento en la variable *insertar*.

Parte de la función para buscar las vías 3

```
ver_insertar = (SELECT dblink_exec(condbr, insertar));
insertar = '';
```

Figura 35. Parte de la función para buscar las vías 3
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 35 se detalla que finalizado el proceso del *bucle*, los datos almacenados en la variable *insertar* serán utilizados para registrar las avenidas en la base datos en la aplicación web con el componente *dblink_exec*. La variable *ver_insertar* de tipo *text* almacena la confirmación de las avenidas almacenadas.

Parte de la función para buscar las vías 4

```
    ELSE
        FOR av_bache IN SELECT id_avenida, id_sector, nombre_avenida
            FROM avenida_baches ORDER BY id_avenida ASC LOOP
            insertar =  'INSERT INTO avenida_baches(id_avenida, id_sector, nombre_avenida)
                            VALUES ('||av_bache.id_avenida||', '||av_bache.id_sector||', '''
                            ||av_bache.nombre_avenida||''');';
        END LOOP;
        ver_insertar = (SELECT dblink_exec(condbr, insertar)) ;
    END IF;

END IF;
```

Figura 36. Parte de la función para buscar las vías 4
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 36 se detalla la ejecución y finalización de esta condición del código cuando existe un sólo registro en la tabla *avenida_baches* para que se suba la información en la aplicación web.

Dentro del *bucle* se busca los baches por sector y avenida que se registraron hasta la fecha actual; El código de la Figura 37 es el encargado de realizar este proceso.

Dentro de un bucle se busca todos los baches por sector y avenida

```
FOR recor_avs IN SELECT rb.id_avenida AS avenida, av.id_sector AS sector FROM registro_bache rb
    INNER JOIN avenida_baches av ON av.id_avenida = rb.id_avenida WHERE fecha = fecha_act
    GROUP BY av.id_sector, rb.id_avenida ORDER BY rb.id_avenida ASC LOOP
    conta_ingreso = 1;
    conta_bache = 1;

        -- CÓNDIGO QUE SE MOSTRARÁ MÁS ADELANTE

END LOOP;
```

Figura 37. Dentro de un bucle se busca todos los baches por sector y avenida
Elaborado por:Daniel Riera y Jorge Soria (2019)

3.7.4. Función que suma la concurrencia

Esta parte de la función es la encargada de sumar la concurrencia de los baches registrados hasta la fecha actual, previamente ingresando la información a la base de datos en la aplicación web.

Suma la concurrencia 1

```
DELETE FROM public.almacenamiento_temporal;
ALTER SEQUENCE almacenamiento_temporal_id_altm_seq RESTART WITH 1;
```

Figura 38. Suma la concurrencia 1
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 38 se detalla el ingreso de los nuevos baches y se eliminan todos los registros de la tabla *almacenamiento_temporal* la cual muestra la secuencia del identificador o clave primaria retorna a uno.

Suma la concurrencia 2

```
num_registro_rbm = (SELECT COUNT(*) FROM registro_bache rb INNER JOIN avenida_baches av ON av.id_avenida = rb.id_avenida
                    WHERE rb.id_avenida = recor_avs.avenida and  av.id_sector = recor_avs.sector);
IF num_registro_rbm > 1 THEN
    FOR regis_baches IN SELECT rb.latitud AS latit, rb.longitud AS longi FROM registro_bache rb
        INNER JOIN avenida_baches av ON av.id_avenida = rb.id_avenida
        WHERE rb.id_avenida = recor_avs.avenida and  av.id_sector = recor_avs.sector ORDER BY rb.id_bache ASC LOOP
    RAISE NOTICE 'INGRESO VECES FOR: %', conta_ingreso;
```

Figura 39. Suma la concurrencia 2
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 39 se explica cómo la variable *num_registros_rbm* de tipo *integer* se almacena el número de datos de los baches por avenida y sector para conocer cuántos registros van a ser procesados. Para ingresar al *bucle* se necesita la variable *num_registros_rbm* sea mayor a uno. El *bucle* será el encargado de llamar a todos los datos por avenida y sector para procesarlos.

Suma la concurrencia 3

```
        IF conta_ingreso = 1 THEN
            alma_latitud = regis_baches.latit;
            alma_longitud = regis_baches.longi;
            punto_uno = 'POINT('||alma_longitud||' '||alma_latitud||')';
```

Figura 40. Suma la concurrencia 3
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 40 se explica cómo la variable *conta_ingreso* de tipo *integer* ayuda a conocer cuál es el primer registro de dónde se toma la longitud y latitud que transforma el primer punto de

referencia y el cual se almacenará en la variable *punto_uno* de tipo *text*. El componente POINT sirve a PostGIS para identificar un punto de datos geográficos.

Suma la concurrencia 4

```
ELSIF conta_ingreso > 1 THEN
    punto_dos = 'POINT('||regis_baches.longi||' '||regis_baches.latit||')';

    distancia = st_distance(ST_GeomFromText(punto_uno), ST_GeomFromText(punto_dos), FALSE);
RAISE NOTICE 'distancia: %'. distancia;
```

Figura 41. Suma la concurrencia 4
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 41 se detalla como en esta condición se procesa todos los datos superiores al primer registro cuando la variable *conta_ingreso* sea mayor a uno. En la variable *punto_dos* de tipo *text* se almacena el punto (latitud y longitud) superiores al primer registro. La función *st_distance* realizará el cálculo para saber la distancia entre dos puntos, la función *ST_GeomFromText* transformará los puntos geográficos de texto a un tipo de datos *geometry* para que PostGIS pueda procesarlos. En la variable *distancia* de tipo *double* da la precisión de almacenar la respuesta de la función *st_distance* y se utiliza el RAISE NOTICE para imprimir la respuesta en la consola.

Suma la concurrencia 5

```
IF distancia <= 5 THEN
    conta_bache = conta_bache + 1;
IF conta_ingreso = num_registro_rbm THEN
    num_registro_atm = (SELECT COUNT(*) FROM almacenamiento_temporal);
    RAISE NOTICE 'numero registro alm: %', num_registro_atm;
```

Figura 42. Suma la concurrencia 5
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 42 se detalla cuando la distancia es menor a cinco metros entra en la condición de la suma de uno con otro para calcular la concurrencia de baches registrados dentro de ese sector, se ocupa la variable *conta_baches* de tipo *integer*. Si la variable *conta_ingreso* es similar a la variable *num_registro_atm* ingresa en la condición dónde se empieza a registrar los datos

procesados en la tabla *almacenamiento_temporal*. En la variable *num_registro_atm* de tipo *integer* se almacena el número de registros encontrados en la tabla *almacenamiento_temporal*.

Suma la concurrencia 6

```
IF num_registro_atm = 0 THEN
    INSERT INTO almacenamiento_temporal(al_latutud, al_longitud, numero_registros)
        VALUES (alma_latitud, alma_longitud, conta_bache);
    alma_latitud = regis_baches.latit;
    alma_longitud = regis_baches.longi;
    punto_uno = 'POINT('||alma_longitud||' '||alma_latitud||')';
ELSIF num_registro_atm = 1 THEN
    INSERT INTO almacenamiento_temporal(al_latutud, al_longitud, numero_registros)
        VALUES (alma_latitud, alma_longitud, conta_bache);
    alma_latitud = regis_baches.latit;
    alma_longitud = regis_baches.longi;
    punto_uno = 'POINT('||alma_longitud||' '||alma_latitud||')';
ELSIF num_registro_atm > 1 THEN
    punto_alm = FALSE;
    FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
        almacenamiento_temporal ORDER BY id_altm ASC LOOP
        punto_dos_tm = 'POINT('||recor_atm.al_longitud||' '||recor_atm.al_latutud||')';
        distancia = st_distance(ST_GeomFromText(punto_uno), ST_GeomFromText(punto_dos), FALSE);
        IF distancia < 5 AND punto_alm = FALSE THEN
            conta_bache = conta_bache + recor_atm.numero_registros;
            UPDATE almacenamiento_temporal SET numero_registros=conta_bache
                WHERE id_altm = recor_atm.id_altm;
            punto_alm = TRUE;
        END IF;
    END LOOP;

    IF punto_alm = FALSE THEN

        INSERT INTO almacenamiento_temporal(al_latutud, al_longitud, numero_registros)
            VALUES (alma_latitud, alma_longitud, conta_bache);
        alma_latitud = regis_baches.latit;
        alma_longitud = regis_baches.longi;
        punto_uno = 'POINT('||alma_longitud||' '||alma_latitud||')';

    END IF;

END IF;
END IF;
```

Figura 43. Suma la concurrencia 6
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 43 se detalla cuando la variable *num_registro_atm* es igual a cero o uno, se ingresa los datos en la tabla *almacenamiento_temporal* y los registros que está en la variable *punto_dos* pasan a la variable *punto_uno*, cuando la variable *num_registro_atm* es mayor a uno, ingresa en un proceso que se encuentra en un *bucle,* el mismo que permite saber si se va a realizar una actualización de datos utilizando la variable *distancia* que da a conocer si se está en un radio menor a cinco metros y la variable *punto_alm* es de tipo *boolean* permite saber sí el ingreso está dentro de la condición o no.

Sí no se ingresó en la condición que está dentro del *bucle* se procede con el almacenamiento de los datos y los registros que están en la variable *punto_dos* y pasan a la variable *punto_uno*.

Suma la concurrencia 7

```
            ELSIF distancia > 5 THEN

                num_registro_atm = (SELECT COUNT(*) FROM almacenamiento_temporal);
                IF num_registro_atm = 0 THEN
                    INSERT INTO almacenamiento_temporal(al_latutud, al_longitud, numero_registros)
                        VALUES (alma_latitud, alma_longitud, conta_bache);
                    alma_latitud = regis_baches.latit;
                    alma_longitud = regis_baches.longi;
                    punto_uno = 'POINT('||alma_longitud||' '||alma_latitud||')';
                ELSIF num_registro_atm = 1 THEN
                    INSERT INTO almacenamiento_temporal(al_latutud, al_longitud, numero_registros)
                        VALUES (alma_latitud, alma_longitud, conta_bache);
                    alma_latitud = regis_baches.latit;
                    alma_longitud = regis_baches.longi;
                    punto_uno = 'POINT('||alma_longitud||' '||alma_latitud||')';
                ELSIF num_registro_atm > 1 THEN
                    punto_alm = FALSE;
                    FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
                        almacenamiento_temporal ORDER BY id_altm ASC LOOP
                        punto_dos_tm = 'POINT('||recor_atm.al_longitud||' '||recor_atm.al_latutud||')';
                        distancia = st_distance(ST_GeomFromText(punto_uno), ST_GeomFromText(punto_dos), FALSE);
                        IF distancia < 5 AND punto_alm = FALSE THEN
                            conta_bache = conta_bache + recor_atm.numero_registros;
                            UPDATE almacenamiento_temporal SET numero_registros=conta_bache
                                WHERE id_altm = recor_atm.id_altm;
                            punto_alm = TRUE;
                        END IF;
                    END LOOP;

                    IF punto_alm = FALSE THEN

                        INSERT INTO almacenamiento_temporal(al_latutud, al_longitud, numero_registros)
                            VALUES (alma_latitud, alma_longitud, conta_bache);
                        alma_latitud = regis_baches.latit;
                        alma_longitud = regis_baches.longi;
                        punto_uno = 'POINT('||alma_longitud||' '||alma_latitud||')';

                    END IF;
                END IF;
                conta_bache = 1;
            END IF;
        END IF;
        conta_ingreso = conta_ingreso + 1;
    END LOOP;
```

Figura 44. Suma la concurrencia 7
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 44 se detalla que la variable *distancia* está en un radio mayor a cinco metros, ingresa a un algoritmo similar de cuando la variable *distancia* tenga un valor menor a un radio de cinco metros, lo que cambia es que a la variable *conta_bache* almacena un valor igual a uno.

Se va sumando de uno en uno en la variable *conta_ingreso* cada vez que termina la condición de está, es decir mayor a uno.

3.7.5. Actualizar la concurrencia

Una vez procesada toda la información actual se carga a la base de datos en la aplicación web y se actualiza cada vez que se registran los baches existentes en un rango de cinco metros, dando como resultado que la información no sea acumulada y redundada.

Rango de los 5.00 metros 1

```
consulta = 'SELECT COUNT(*) as num_registro FROM baches_referenciado
            WHERE id_avenida = '||recor_avs.avenida||'and id_sector = '||recor_avs.sector||';';

num_registros_ing = (SELECT * FROM dblink(condbr,consulta) as tl(num_registro integer));
RAISE NOTICE 'numero registro ing: %', num_registros_ing;

IF num_registros_ing = 0 OR num_registros_ing IS NULL THEN
    num registro atm = (SELECT COUNT(*) FROM almacenamiento temporal);
```

Figura 45. Rango de los 5.00 metros 1
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 45 se detalla que la variable *consulta* almacena el *Query* que se ejecuta con el componente *dblink* a la tabla *baches_referenciado* de la base de datos así obteniendo el número de baches por sector y avenida almacenados en la variable *num_registros_ing* de tipo *integer*. Con el *RAICE NOTICE* se imprime los datos en la consola de PostgreSQL.

Rango de los 5.00 metros 2

```
IF num_registro_atm = 1 THEN
    FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
            almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP
        insertar = 'INSERT INTO baches_referenciado(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                    VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                    ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||''');';
    END LOOP;
    ver_insertar = (SELECT dblink_exec(condbr, insertar));
```

Figura 46. Rango de los 5.00 metros 2
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 46 se detalla que la variable *num_registro_atm* de tipo *integer* es igual a uno indica que en la tabla *almacenamiento_temporal* existe un sólo registro y se utiliza la variable *insertar* para almacenar el *Query* que es el encargado de registrar la información que va a ser ingresada en la tabla *baches_referenciado* de la base de datos del componente *dblink_exec* en la aplicación web.

En la variable *ver_insertar* almacena la respuesta del ingreso de la información.

Rango de los 5.00 metros 3

```
ELSIF num_registro_atm > 1 THEN
    conta_atm = 1;
    FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
            almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP
        IF conta_atm = 1 THEN
            insertar = 'INSERT INTO baches_referenciado(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||'''),';
        ELSIF conta_atm > 1 AND conta_atm <  num_registro_atm THEN
            insertar = insertar||'(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||'''),';
        ELSIF conta_atm = num_registro_atm THEN
            insertar = insertar||'(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||''');';
        END IF;
    END LOOP;
    ver_insertar = (SELECT dblink_exec(condbr, insertar));
END IF;
```

Figura 47. Rango de los 5.00 metros 3
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 47 se explica cuando la variable *num_registro_atm* es mayor a uno se ingresa a un *bucle* que se almacena una lista de registros en la variable *insertar* que posteriormente van a ser ingresados a la base de datos en la tabla *baches_referenciado* de la aplicación web.

Rango de los 5.00 metros 4

```
ELSIF num_registros_ing = 1 THEN
    consulta = 'SELECT id_pto, latitudc, longitudc, contador
            FROM baches_referenciado WHERE id_avenida = '||recor_avs.avenida||' and id_sector = '||recor_avs.sector||';';
```

Figura 48. Rango de los 5.00 metros 4
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 48 se explica que cuando la variable *num_registros_ing* es igual a uno se ingresa en la condición *bucle*, está realiza la consulta a la base de datos en la tabla *baches_referenciado* de la aplicación web.

Rango de los 5.00 metros 5

```
FOR recor_brba IN SELECT id_pto, latitudc, longitudc, contador FROM dblink(condbr,consulta) as tl(id_pto integer, latitudc DOUBLE PRECISION,
            longitudc DOUBLE PRECISION , contador INTEGER) LOOP
    num_registro_atm = (SELECT COUNT(*) FROM almacenamiento_temporal);
```

Figura 49. Rango de los 5.00 metros 5
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 49 se detalla que dentro del *bucle* se busca nuevamente el número de registros ingresados en la tabla *almacenamiento_temporal*, la información es guardada en la variable *num_registros_atm*.

Rango de los 5.00 metros 6

```
IF num_registro_atm = 1 THEN
    FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
            almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP

        punto_uno = 'POINT('||recor_atm.al_longitud||' '||recor_atm.al_latutud||')';
        punto_dos = 'POINT('||recor_brba.longitudc||' '||recor_brba.latitudc||')';
        distancia = st_distance(ST_GeomFromText(punto_uno), ST_GeomFromText(punto_dos), FALSE);
        IF distancia <= 5 THEN
            total_baches = recor_brba.contador + recor_atm.numero_registros;
            actualizar = 'UPDATE public.baches_referenciado SET estadoa='||true||', contador='||total_baches||', fecha='''||fecha_act||
                    '''WHERE id_pto='||recor_brba.id_pto||';';
            ver_insertar = (SELECT dblink_exec(condbr, actualizar));
            UPDATE public.almacenamiento_temporal SET estado = 1 WHERE id_altm = recor_atm.id_altm;
        END IF;
    END LOOP;
```

Figura 50. Rango de los 5.00 metros 6
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 50 se explica que la variable *num_registros_atm* es igual a uno se debe ingresar en la condición *bucle* dónde va a procesar el único registro que se encuentra en la tabla *almacenamiento_temporal* pero antes se debe comprobar sí la distancia del radio es menor a cinco metros. Cuando la distancia es mayor a cinco metros no se realiza ninguna actualización y se almacena la variable para ser procesado después.

Rango de los 5.00 metros 7

```
ELSIF num_registro_atm > 1 THEN

    conta_atm = 1;
    FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
            almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP
        punto_uno = 'POINT('||recor_atm.al_longitud||' '||recor_atm.al_latutud||')';
        punto_dos = 'POINT('||recor_brba.longitudc||' '||recor_brba.latitudc||')';
        distancia = st_distance(ST_GeomFromText(punto_uno), ST_GeomFromText(punto_dos), FALSE);
        IF distancia <= 5 THEN
            total_baches = recor_brba.contador + recor_atm.numero_registros;
            actualizar = 'UPDATE public.baches_referenciado SET estadoa='||true||', contador='||total_baches||', fecha='''||fecha_act||
                    '''WHERE id_pto='||recor_brba.id_pto||';';
            ver_insertar = (SELECT dblink_exec(condbr, actualizar));
            UPDATE public.almacenamiento_temporal SET estado=1 WHERE id_altm = recor_atm.id_altm;
        END IF;
    END LOOP;
END IF;
END LOOP;
```

Figura 51. Rango de los 5.00 metros 7

Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 51 se detalla que la variable *num_registros_atm* es mayor a uno se ingresa en un *bucle* que procesa todos los datos ingresados en la tabla *almacenamiento_temporal*. Durante el proceso el *bucle* busca los puntos que se encuentran dentro de un radio de cinco metros. Cuando la distancia del radio es mayor a cinco metros se almacena la información para procesarse después.

Rango de los 5.00 metros 8

```
num_registro_atm = (SELECT COUNT(*) FROM almacenamiento_temporal WHERE estado = 0);
```

Figura 52. Rango de los 5.00 metros 8
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 52 se detalla la variable *num_registro_atm* siempre se actualizará al momento de ingresar en el *bucle* inicial para conocer cuántos datos hay que procesar.

Rango de los 5.00 metros 9

```
IF num_registro_atm = 1 THEN
    FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
                     almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP

        insertar = 'INSERT INTO baches_referenciado(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                    VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                    ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||''');';

    END LOOP;
    ver_insertar = (SELECT dblink_exec(condbr, insertar));

ELSIF num_registro_atm >=1 THEN
    conta_atm = 1;
    FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
                     almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP
        IF conta_atm = 1 THEN
            insertar = 'INSERT INTO baches_referenciado(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                        VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                        ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||'''),';
        ELSIF conta_atm > 1 AND conta_atm <  num_registro_atm THEN
            insertar = insertar||'(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                        VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                        ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||'''),';

        ELSIF conta_atm = num_registro_atm THEN
            insertar = insertar||'(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                        VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                        ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||''');';
        END IF;
    END LOOP;
    ver_insertar = (SELECT dblink_exec(condbr, insertar));
END IF;
```

```
ELSIF num_registros_ing > 1 THEN
    consulta = 'SELECT id_pto, latitudc, longitudc, contador
                FROM baches_referenciado WHERE id_avenida = '||recor_avs.avenida||' and id_sector = '||recor_avs.sector||';';
    RAISE NOTICE 'numero registro ing: %', num_registros_ing;

    FOR recor_brba IN SELECT id_pto, latitudc, longitudc, contador FROM dblink(condbr,consulta) as tl(id_pto integer, latitudc DOUBLE PRECISION,
                        longitudc DOUBLE PRECISION , contador INTEGER) LOOP
        num_registro_atm = (SELECT COUNT(*) FROM almacenamiento_temporal);

        RAISE NOTICE 'for atm: %', num_registro_atm;
        RAISE NOTICE 'id brba: %', recor_brba.id_pto;



        IF num_registro_atm = 1 THEN
            RAISE NOTICE 'atm =1: %', num_registro_atm;
            FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
                    almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP
                RAISE NOTICE 'num registro_1: %', recor_atm.numero_registros;
                RAISE NOTICE 'num estado_1: %', recor_atm.numero_registros;
                punto_uno = 'POINT('||recor_atm.al_longitud||' '||recor_atm.al_latutud||')';
                punto_dos = 'POINT('||recor_brba.longitudc||' '||recor_brba.latitudc||')';
                distancia = st_distance(ST_GeomFromText(punto_uno), ST_GeomFromText(punto_dos), FALSE);
                IF distancia <= 5 THEN
                    total_baches = recor_brba.contador + recor_atm.numero_registros;
                    actualizar = 'UPDATE public.baches_referenciado SET estadoa='||true||', contador='||total_baches||', fecha='''||fecha_act||
                                 '''WHERE id_pto='||recor_brba.id_pto||';';
                    ver_insertar = (SELECT dblink_exec(condbr, actualizar));
                    UPDATE public.almacenamiento_temporal SET estado = 1 WHERE id_altm = recor_atm.id_altm;
                END IF;

            END LOOP;

        ELSIF num_registro_atm > 1 THEN
            RAISE NOTICE 'atm > 1: %', num_registro_atm;
            conta_atm = 1;
            FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
                    almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP
                RAISE NOTICE 'num registro_2: %', recor_atm.numero_registros;
                RAISE NOTICE 'num estado_2: %', recor_atm.numero_registros;
                punto_uno = 'POINT('||recor_atm.al_longitud||' '||recor_atm.al_latutud||')';
                punto_dos = 'POINT('||recor_brba.longitudc||' '||recor_brba.latitudc||')';
                RAISE NOTICE 'punto 1_2: %', punto_uno;
                RAISE NOTICE 'punto 2_2: %', punto_dos;
                distancia = st_distance(ST_GeomFromText(punto_uno), ST_GeomFromText(punto_dos), FALSE);
                RAISE NOTICE 'distacia: %', distancia;
                RAISE NOTICE 'fecha: %', fecha_act;
                IF distancia <= 5 THEN
                    total_baches = recor_brba.contador + recor_atm.numero_registros;
                    actualizar = 'UPDATE public.baches_referenciado SET estadoa='||true||', contador='||total_baches||', fecha='''||fecha_act||
                                 '''WHERE id_pto='||recor_brba.id_pto||';';
                    ver_insertar = (SELECT dblink_exec(condbr, actualizar));
                    UPDATE public.almacenamiento_temporal SET estado = 1 WHERE id_altm = recor_atm.id_altm;
                END IF;
            END LOOP;
        END IF;
    END LOOP;
```

Figura 53. Rango de los 5.00 metros 9
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 53 se detalla cómo una vez que los datos de la concurrencia de los baches serán ingresados y procesados para ser enviados por los usuarios en la aplicación móvil o se realizará las actualizaciones que falten con algoritmos similares a los visualizados con anterioridad.

Rango de los 5.00 metros 10

```
    num_registro_atm = (SELECT COUNT(*) FROM almacenamiento_temporal WHERE estado = 0);

    IF num_registro_atm = 1 THEN
        FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
                         almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP
        insertar = 'INSERT INTO baches_referenciado(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                    VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                    ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||''');';

        END LOOP;
        ver_insertar = (SELECT dblink_exec(condbr, insertar));
    ELSIF num_registro_atm >=1 THEN
        conta_atm = 1;
        FOR recor_atm IN SELECT id_altm, al_latutud, al_longitud, numero_registros FROM
                         almacenamiento_temporal WHERE estado = 0 ORDER BY id_altm ASC LOOP
        IF conta_atm = 1 THEN
            insertar = 'INSERT INTO baches_referenciado(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                        VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                        ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||'''),';
        ELSIF conta_atm > 1 AND conta_atm <  num_registro_atm THEN
            insertar = insertar||'(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                        VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                        ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||'''),';
        ELSIF conta_atm = num_registro_atm THEN
            insertar = insertar||'(id_avenida, id_sector, latitudc, longitudc, estadoa, contador, fecha)
                        VALUES ('|| recor_avs.avenida||','|| recor_avs.sector||','||recor_atm.al_latutud||
                        ','||recor_atm.al_longitud||','||true||','||recor_atm.numero_registros||', '''||fecha_act||''');';
        END IF;
        END LOOP;
        ver_insertar = (SELECT dblink_exec(condbr, insertar));
    END IF;

END IF;
```

Figura 54. Rango de los 5.00 metros 10
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 54 se detalla cómo se realiza una búsqueda de los registros que faltan por procesar con la ayuda de la variable *num_registro_atm* ingresando con dos condiciones de algoritmos similares a los que ya se han visto.

3.7.6. Función para suprimir baches.

Esta función permite visualizar la concurrencia de baches en un tiempo de tres meses; si no existe una actualización se bloquea la información en la base de datos para que no aparezca en la aplicación web.

Si no existe una actualización se bloquea

```
begin

    fecha_bache = (select (current_date -interval '3 month'));
    UPDATE public.baches_referenciado SET estadoa = false WHERE fecha= fecha_bache;
    respuesta = 1;
    RETURN respuesta;

end;
```

Figura 55. Si no existe una actualización se bloquea
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 55 se detalla la variable *fecha_bache* de tipo *date*, es dónde se resta tres meses de la fecha para saber cuales son los baches que se van a bloquear y no aparezcan en la aplicación web. Para realizar la actualización se utiliza el *Query Update*. La función siempre responderá con el número uno cada vez que se ejecute.

3.7.7. Código relevante de la aplicación móvil

3.7.7.1. Registro inicial

Se toma el código IMEI del teléfono móvil para el registro de todos los dispositivos móviles que tienen instalado la aplicación.

Toma el código IMEI del teléfono móvil

```
@SuppressLint("MissingPermission")
private String obtenerIMEI() {
        final TelephonyManager telephonyManager= (TelephonyManager)
this.getSystemService(Context.TELEPHONY_SERVICE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
//Hacemos la validación de métodos, ya que el método getDeviceId()
//ya no se admite para android Oreo en adelante, debemos usar el método getImei()
            return telephonyManager.getImei();
        }
        else {
            return telephonyManager.getDeviceId();
        }
}
```

Figura 56. Toma el código IMEI del teléfono móvil
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 56 se detalla el envío del código IMEI del dispositivo móvil al servidor encargado de recibir los datos, mediante el API REST del web service en formato JSON. La información es enviada para verificar la conexión con el URL del API, en caso de no existir el enlace no se registrara el dispositivo, esto impide que se envié la información de las coordenadas de los baches.

Envía el IMEI del dispositivo móvil al servidor

```
private class RegistroUsuarioAnonimo extends AsyncTask<String, Integer, Boolean> {
    private int id = 0;
    protected Boolean doInBackground(String... params){
        boolean resultado = true;
```

Figura 57. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 57, se crea la clase en donde se encuentra el API REST del web service, que permite subir el registro del código IMEI. En la variable *identificador* de tipo *int* se almacena la respuesta del web service al momento de registrar la información. El método de tipo *boolean doInBackground* contiene el código que se observara en las siguientes Figuras. La variable *resultado* de tipo *boolean*, permite confirmar la conexión con el web service.

Envía el IMEI del dispositivo móvil al servidor

```
try{
    consultarPermiso(Manifest.permission.READ_PHONE_STATE, PHONESTATS);
    ThreadImei esperarImei = new ThreadImei(imei);
    esperarImei.start();
```

Figura 58. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En el código de la Figura 58, se consulta sí la aplicación tiene el permiso para obtener el código IMEI del dispositivo móvil, en caso de no tener el permiso, la aplicación móvil lo solicitará. La aplicación móvil se demora un tiempo hasta obtener el código IMEI, para lo cual se utiliza un hilo de espera hasta lograr obtenerlo.

Envía el IMEI del dispositivo móvil al servidor

```
URL url = new URL("http://[redacted]:8180/BachesSevice/rest/MovilRest/registro");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
urlConnection.setRequestMethod("POST");
urlConnection.setRequestProperty("Content-Type", "application/json; charset=utf-8");
```

Figura 59. Envía el IMEI del dispositivo móvil al servidor
Elaborado por:Daniel Riera y Jorge Soria (2019)

En Figura 59, se utiliza la variable *url* de tipo *objeto* URL, en donde se escribe el API REST de conexión con el web service, la variable *urlConnection* de tipo *objeto HttpURLConnection*, el tipo de método que se ocupará con el API REST, empieza a establecer la conexión con el web service, el formato de archivo que se enviará y el idioma que el web service tendría que reconocer.

Envía el IMEI del dispositivo móvil al servidor

```
JSONObject jsonObject = new JSONObject();
jsonObject.accumulate("imei", imei);
```

Figura 60. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 60, la variable *jsonObject* de tipo *objeto* crea el archivo *json* con los datos que se enviará al web service.

Envía el IMEI del dispositivo móvil al servidor

```
OutputStream os = urlConnection.getOutputStream();
BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, "UTF-8"));
writer.write(jsonObject.toString());
Log.i(MainActivity.class.toString(), jsonObject.toString());
writer.flush();
writer.close();
os.close();
```

Figura 61. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 61, se transforma el archivo *json* que se creó en la Figura 60, en código binario para ser enviado a través de la red.

Envía el IMEI del dispositivo móvil al servidor

```
urlConnection.connect();
String mensaje = urlConnection.getResponseMessage()+"";
Log.i("Mensaje:",mensaje );
//int code = Integer.parseInt(urlConnection.getResponseMessage());
if (!mensaje.equals("OK")) {
    throw new IOException("Invalid response from server: " + mensaje);
}
```

Figura 62. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 62, se completa la conexión con el web service y se envía el archivo *json*, y a su vez el web service retorna un mensaje en un archivo *json*, confirmando si la información fue procesada correctamente.

Envía el IMEI del dispositivo móvil al servidor

```
BufferedReader rd = new BufferedReader(new InputStreamReader(urlConnection.getInputStream()));
String resp = rd.readLine();
JSONObject respJSON = null;
if (resp != null)
    respJSON = new JSONObject(resp);
id = respJSON.getInt("id");
```

Figura 63. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 63 se detalla como el código transforma la información del archivo *json* de binario a texto, la cual una vez procesada se almacena en la variable *identificador*, que permite identificar que el dispositivo móvil está enviando la información.

Envía el IMEI del dispositivo móvil al servidor

```
}catch (Exception ex){
    Log.e("ServicioRest","Error!", ex);
    resultado = false;
}
return  resultado;
```

Figura 64. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 64 se detalla como se almacena el boolean *false* en la variable *resultado*, en el caso de encontrar un error de conexión con el web service.

Envía el IMEI del dispositivo móvil al servidor

```
    protected void onPostExecute(Boolean result) {
        if (result){
            userId = id;
            if (datosIni.ingresarDatos(userId)) {
                ThreadGps esperaGps = new ThreadGps(longitudeGPS, latitudeGPS);
                esperaGps.start();
                Intent intent = new Intent(MainActivity.this, MapaBachesActivity.class);
                Bundle bundle = new Bundle();
                bundle.putDouble("latitud", latitudeGPS);
                bundle.putDouble("longitud", longitudeGPS);
                intent.putExtras(bundle);
                startActivity(intent);
            }else{
                Toast.makeText(MainActivity.this, "Error en el registro", Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Figura 65. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 65 se muestra como el método *onPostExecute* procesa los datos que el web service envió previamente y el cual se almacenará en el móvil. Una vez guardada la información ingresa directo a la pantalla que permite enviar el registro del bache.

3.7.7.2. Envío de información de baches

Una vez registrado el dispositivo móvil en el servidor, el usuario podrá enviar la información de los baches con el comando de voz que utiliza la palabra *Bache* como clave.

Envía el IMEI del dispositivo móvil al servidor

```
private static class SetupTask extends AsyncTask<Void, Void, Exception> {
        WeakReference<MapaBachesActivity> activityReference;
        SetupTask(MapaBachesActivity activity) {
            this.activityReference = new WeakReference<>(activity);
        }
        @Override
        protected Exception doInBackground(Void... params) {
            try {
                Assets assets = new Assets(activityReference.get());
                File assetDir = assets.syncAssets();
                activityReference.get().setupRecognizer(assetDir);
            } catch (IOException e) {
                return e;
            }
            return null;
        }
        @Override
        protected void onPostExecute(Exception result) {
            if (result != null) {
                result.printStackTrace();
            } else {
                activityReference.get().switchSearch(KWS_SEARCH);
            }
        }
    }
```

Figura 66. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 66 se detalla la clase *SetupTask* que es la encargada de ejecutar el proceso en la aplicación móvil que utiliza para ser escuchado el comando de voz. En el método *SetupTask* se inicia el objeto *WeakReference*. En el método *doInBackground* se inicia las librerías, métodos y objetos que la aplicación utilizará para ejecutar el comando de voz. En el método *onPostExecute* al no existir ningún error previamente, iniciará el proceso para escuchar el comando de voz que se está utilizando para enviar la información.

Envía el IMEI del dispositivo móvil al servidor

```
   private void setupRecognizer(File assetsDir) throws IOException {
        recognizer = SpeechRecognizerSetup.defaultSetup()
                .setAcousticModel(new File(assetsDir, "es_es_ptm"))
                .setDictionary(new File(assetsDir, "es.dict"))
                .setKeywordThreshold(Float.valueOf("1.e-"+2*sensibility))
                //    .setBoolean("-allphone_ci", true)
                .setRawLogDir(assetsDir)
                .getRecognizer();
        recognizer.addListener(this);
//Crear búsqueda de activación de palabras clave.
        recognizer.addKeyphraseSearch(KWS_SEARCH, trigger);
 //Cree una búsqueda gramatical para la selección entre demostraciones
        File menuGramamar = new File(assetsDir, "menu.gram");
        recognizer.addGrammarSearch(MENU_SEARCH, menuGramamar);
    }
```

Figura 67. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 67 se detalla el método *setupRecognizer* que busca el idioma y diccionario utilizado en el comando de voz.

Envía el IMEI del dispositivo móvil al servidor

```
    private void switchSearch(String searchName) {
        recognizer.stop();
// If we are not spotting, start listening with timeout (10000 ms or 10 seconds).
        Log.i("Frase bache: ", searchName);
        if (searchName.equals(KWS_SEARCH)) {
            recognizer.startListening(searchName);
        }else {
            recognizer.startListening(searchName, 10000);
        }
    }
```

Figura 68. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 68 se detalla el método *switchSearch* verifica sí el comando de voz es el correcto, procesando cada palabra que pronuncia el usuario, sí el comando de voz es correcto se toma 10 segundos para volver a iniciar el reconocimiento de voz.

Envía el IMEI del dispositivo móvil al servidor

```
@Override
public void onPartialResult(Hypothesis hypothesis) {
    if (hypothesis == null){
        return;
    }
    String text = hypothesis.getHypstr();
    Log.i("Frase ", text);
    if (text.equals(trigger)) {
        switchSearch(MENU_SEARCH);
    }
}
```

Figura 69. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 69 se detalla el método *onPartialResult* verifica sí el sonido escuchado es una palabra será transformado a texto. Sí se confirma que el sonido escuchado es texto se ejecuta el método *switchSearch.*

Envía el IMEI del dispositivo móvil al servidor

```
@Override
public void onResult(Hypothesis hypothesis) {
    if (hypothesis != null) {
        String text = hypothesis.getHypstr();
        if (text.equals(trigger)){
            ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();
            if (networkInfo != null && networkInfo.isConnected()) {
                toggleLocationUpdates();
            } else {
                almacenamientoLocal ();
            }
        }
    }
}
```

Figura 70. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 70 se detalla el método *onResult* que procesa todas las palabras que son escuchadas por la aplicación, sí la palabra es correcta será ejecutada por el método que se encarga de enviar el registro de bache.

Envía el IMEI del dispositivo móvil al servidor

```
private void buscarCoordenadasLocal(){
        Cursor res = coordenadasBaches.getCoordenadas();
        if (res.getCount() != 0){
            ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();
```

Figura 71. Envía el IMEI del dispositivo móvil al servidor

Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 71 se detalla el método *buscarCoodenadasLocal* que ejecuta el código para actualizar la posición del GPS en la que se encuentra el usuario. El cursor va tomando las coordenadas actuales de donde esté ubicado el usuario. Cada vez que se ejecuta este método se verifica la conexión a internet para poner el marcador en la ubicación del bache.

Envía el IMEI del dispositivo móvil al servidor

```
if (networkInfo != null && networkInfo.isConnected()) {
    almacenarCoordenadasList = new ArrayList<AlmacenarCoordenadas>();
```

Figura 72. Envía el IMEI del dispositivo móvil al servidor
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 72 se detalla la confirmación de la conexión a internet ejecutandose el proceso de actualización de las coordenadas.

Envía el IMEI del dispositivo móvil al servidor

```
while (res.moveToNext()){
    AlmacenarCoordenadas alc = new AlmacenarCoordenadas();
    double latit = res.getDouble(0);
    double longit = res.getDouble(1);
    alc.setLatitud(latit);
    alc.setLongitud(longit);
```

Figura 73. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 73 se detalla la ejecución del *bucle* para actualizar las coordenadas geográficas cada vez que el usuario registre un bache.

Envía el IMEI del dispositivo móvil al servidor

```
            if (latit != 0.0 && longit !=0.0){
                try{
                    Geocoder geocoder = new Geocoder(getBaseContext(), Locale.getDefault());
                    List<Address> list = geocoder.getFromLocation(latitudeBest, longitudeBest, 1);
                    if (!list.isEmpty()){
                        Address dirAvenida = list.get(0);
                        String direccion = dirAvenida.getAddressLine(0);
                        String [] cadena = direccion.split(",");
                        avenida = cadena [0];
                    }
                }catch (IOException e){
                    e.printStackTrace();
                }
            }
            alc.setAvenida(avenida);
            almacenarCoordenadasList.add(alc);
        }
```

Figura 74. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 74 se detalla cómo se busca el nombre de la avenida que se está registrando el bache dentro del *bucle* antes mencionado.

Envía el IMEI del dispositivo móvil al servidor

```
            RegistoListaBaches registoListaBaches = new RegistoListaBaches();
            registoListaBaches.execute();
            coordenadasBaches.eliminarDatos();
        }else {
            makeText(getApplicationContext(),"Datos no enviados",Toast.LENGTH_SHORT).show();
        }
    }
}
```

Figura 75. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 75 se detalla cómo se almacena en una base temporal las coordenadas geográficas y el nombre de la avenida que van a ser enviadas al servidor.

Envía el IMEI del dispositivo móvil al servidor

```
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
//locationManager = (LocationManager) getSystemService (Context.LOCATION_SERVICE);
        float zoom = 17;
        Log.i("Longitud ", ""+longitudeGPS);
        Log.i("Latitud ", ""+latitudeGPS);
// Add a marker in Sydney and move the camera
        LatLng latLng = new LatLng(latitudeBest, longitudeBest);
        mMap.addMarker(new MarkerOptions().position(latLng).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, zoom));
    }
```

Figura 76. Envía el IMEI del dispositivo móvil al servidor
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 76 se detalla el método *onMapReady* que pone el marcador en el mapa de las coordenadas geográficas que se registraron.

Envio de las coordenadas geográficas

```
private void toggleLocationUpdates() {
    enableLocationUpdates();
    if (latitudeBest != 0.0 && longitudeBest != 0.0){
        actualizarMapa();
        RegistoBaches registoBaches = new RegistoBaches();
        registoBaches.execute();
    }
}
```

Figura 77. Envio de las coordenadas geográficas
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 77 se detalla el método *toggleLocationUpdates* que es ejecutado para actualizar las coordenadas geográficas y envía la ejecución del objeto para remitir la información al servidor en dónde se está almacenando.

Envio de las coordenadas geográficas

```
private class RegistoBaches extends AsyncTask<String, Integer, Boolean> {
    private String mensaje = "";
    protected Boolean doInBackground(String... params) {
        boolean resultado = true;
```

Figura 78. Envio de las coordenadas geográficas
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 78 se detalla cómo se crea la clase en donde se encuentra el API REST del web service que permite subir el registro del bache. En la variable *mensaje* de tipo *string* se almacena la respuesta del web service al momento de registrar la información. El método de tipo *boolean doInBackground* contiene el código que se observará en las siguientes figuras. La variable *resultado* de tipo *boolean* permite confirmar la conexión con el web service.

Envió de las coordenadas geográficas

```
try {
    Cursor res = datosIni.getDato();
    int tomarId = 0;
    if (res.getCount() != 0) {
        if (res.moveToNext()) {
            tomarId = res.getInt(0);
        }
    }
    Log.e("Id usuario ", "" + tomarId);
```

Figura 79. Envió de las coordenadas geográficas
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 79 se detalla cómo se busca el identificador del dispositivo móvil al enviar previamente la información para el registro del bache.

Envió de las coordenadas geográficas

```
URL url = new URL("http://186.71.19.82:8180/BachesSevice/rest/GeolocalizacionRest/GeoPosicion/" + tomarId);
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
urlConnection.setRequestMethod("POST");
urlConnection.setRequestProperty("Content-Type", "application/json; charset=utf-8");
```

Figura 80. Envió de las coordenadas geográficas
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 80 se observa la utilización de la variable *url*, donde se escribe el API REST de conexión con el web service, la variable *urlConnection* de tipo *objeto HttpURLConnection* empieza a establecer la conexión con el web service.

Envió de las coordenadas geográficas

```
            JSONObject jsonObject = new JSONObject();
            jsonObject.accumulate("latitud", latitudeBest);
            jsonObject.accumulate("longitud", longitudeBest);
            jsonObject.accumulate("avenida", avenida);
```

Figura 81. Envió de las coordenadas geográficas
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 81, se detalla cómo la variable *jsonObject* crea el archivo *json* con los datos que se enviará al web service.

Envió de las coordenadas geográficas

```
            OutputStream os = urlConnection.getOutputStream();
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, "UTF-8"));

            writer.write(jsonObject.toString());
            Log.i(MainActivity.class.toString(), jsonObject.toString());
            writer.flush();
            writer.close();
            os.close();
```

Figura 82. Envió de las coordenadas geográficas
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 82 se detalla cómo se transforma el archivo *json* que se creó en la Figura 81, en código binario para ser enviado a través de la red.

Envió de las coordenadas geográficas

```
            urlConnection.connect();
            String mensaje = urlConnection.getResponseMessage() + "";
            Log.i("Mensaje:", mensaje);
//int code=Integer.parseInt(urlConnection.getResponseMessage());
            if (!mensaje.equals("OK")) {
                throw new IOException("Invalid response from server: " + mensaje);
            }
```

Figura 83. Envió de las coordenadas geográficas
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la Figura 83, se muestra la conexión completa con el web service y envía el archivo *json*, a su vez el web service retorna un mensaje en el mismo archivo confirmando sí la información fue procesada correctamente.

Envió de las coordenadas geográficas

```
BufferedReader rd = new BufferedReader(new InputStreamReader(urlConnection.getInputStream()));
String resp = rd.readLine();
JSONObject respJSON = null;
if (resp != null)
    respJSON = new JSONObject(resp);
this.mensaje = respJSON.getString("mensaje");
```

Figura 84. Envió de las coordenadas geográficas
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 84 se detalla cómo el código transforma la información del archivo *json* de binario a texto, la cual una vez procesada se almacena en la variable *resp*, la cual permite confirmar si se registró correctamente la información.

Envió de las coordenadas geográficas

```
        } catch (Exception ex) {
            Log.e("ServicioRest", "Error!", ex);
            resultado = false;
        }
        return resultado;
    }
}
```

Figura 85. Envió de las coordenadas geográficas
Elaborado por:Daniel Riera y Jorge Soria (2019)

En la Figura 85 se detalla cómo se almacena el *boolean false* en la variable *resultado* cuando existe error de conexión con el web service.

3.8. Pruebas

Para garantizar los resultados del funcionamiento de la aplicación web y móvil, se llevaron a cabo las pruebas individuales y en conjunto para encontrar deficiencias y de esta manera mejorar la calidad.

3.8.1. Pruebas funcionales

Por medio de estas pruebas se logró exponer el trabajo realizado de la aplicación móvil y web, de acuerdo a las funcionalidades presentadas, se evaluó la operatividad y respuesta de los componentes que manejan las aplicaciones garantizando la debida atención a cada uno de los requerimientos.

En las tablas se encuentran registrados los casos de prueba y a continuación se detallan los siguientes campos:

- **Número de caso de prueba:** Es el orden secuencial del caso de prueba.
- **Ciudadano:** Actor ejecutor.
- **Referencia al caso de uso:** Es la descripción del caso de uso que está sometido a una prueba.
- **Nombre:** Es la explicación del caso de prueba.
- **Entrada:** Iniciar la ejecución de la prueba.
- **Salida:** Es el resultado exitoso en el cumplimiento de la prueba.
- **Descripción:** Información detallada acerca del caso de uso aplicado.
- **Seguimiento de prueba:** Es la secuencia de pasos para el desarrollo de la prueba.
- **Resultado:** Explicación de la ejecución del procedimiento en el aplicativo móvil y web por parte del ciudadano.
- **Código:** Identificador del caso de prueba.
    - PR-FU-00 para las pruebas de funcionalidad del ciudadano.
    - PR-FA-00 para las pruebas de funcionalidad del administrador.

Tabla 27. Caso de prueba: Indicar la ubicación en tiempo real al ciudadano

PR-FU-01	
Número de caso de prueba	01
Ciudadano	Conductor y transeúnte.
Referencia al caso de uso	Ingresar al aplicativo móvil *Bache*.
Nombre	Indicar al ciudadano su ubicación en tiempo real.
Entrada	Por medio de la función *touch* abrir la aplicación móvil.
Salida	Mostrar la geolocalización en el mapa al ciudadano en tiempo real.
Descripción	Visualizar el correcto funcionamiento del GPS en la aplicación móvil con y sin servicio de internet.
Seguimiento de prueba	1) Abrir la aplicación móvil. 2) Visibilizar la ubicación del ciudadano en el mapa de la ciudad de Quito en tiempo real.
Resultado	El ciudadano gracias a la geolocalización puede registrar los baches mediante coordenadas.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 28. Caso de prueba: Mostrar al ciudadano el funcionamiento del comando de voz

PR-FU-02	
Número de caso de prueba	02
Ciudadano	Conductor y transeúnte.
Referencia al caso de uso	Ingresar al aplicativo móvil *Bache*.
Nombre	Informar al ciudadano el correcto funcionamiento del comando de voz.
Entrada	Por medio de la función *touch* abrir la aplicación móvil.
Salida	Registrar el bache por medio del comando de voz con la palabra clave *bache*.
Descripción	Comprobar el buen funcionamiento del comando de voz.
Seguimiento de prueba	1) Abrir la aplicación móvil. 2) Usar el comando de voz *Bache* para registrar al bache.
Resultado	El ciudadano registra la localización del bache en el mapa de Quito.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 29. Caso de prueba Indicar las funciones del menú

PR-FU-03	
Número de caso de prueba	03
Ciudadano	Conductor y transeúnte.
Referencia al caso de uso	Observar el menú de la aplicación móvil *Bache*.
Nombre	Informar al ciudadano de las funciones del menú
Entrada	Abrir el menú.

Salida	Indicar las características y el funcionamiento de cada ítem.
Descripción	Seleccionar la opción deseada.
Seguimiento de prueba	1) Abrir la aplicación móvil. 2) Abrir el menú que está en la parte superior izquierda de la pantalla, mediante la función *touch.* 3) Elegir la función deseada. • *Visitar página web:* accede a la página web. • *Registrar Bache:* permite registrar el bache. • *Acerca de:* explica el funcionamiento de la aplicación • *Cerrar Aplicación:* finaliza la sesión en la aplicación
Resultado	El ciudadano puede visualizar varias funciones en este menú como por ejemplo un acceso directo a la página web.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 30. Caso de prueba Visualizar los baches detalladamente en la aplicación web

PR-FU-04	
Número de caso de prueba	04
Ciudadano	Conductor y transeúnte.
Referencia al caso de uso	Ingresar a la aplicación web *Bache*.
Nombre	Informar al ciudadano detalladamente de la existencia de los baches en la ciudad de Quito.
Entrada	Abrir la aplicación web a través del buscador

Salida	Mostrar la información detallada mediante tablas y estadísticas de la presencia de los baches en la ciudad de Quito.
Descripción	Seleccionar la ubicación donde se encuentra el bache.
Seguimiento de prueba	1) Ingresar al buscador y digitar la dirección web. 2) Luego de digitar la dirección web se abre la aplicación. 3) El menú aparece e indica los mapas, la tabla con la ubicación detallada de los baches y las estadísticas a nivel regional.
Resultado	El ciudadano puede visualizar la ubicación del bache detalladamente mediante mapas, tablas y estadísticas.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 31. Caso de prueba: Ingresar nuevos datos

PR-FA-05	
Número de caso de prueba	05
Ciudadano	Administrador.
Referencia al caso de uso	Retroalimentar la base de datos en el sistema.
Nombre	Ingresar nuevos datos.
Entrada	Ingreso al sistema Detalle de nuevos datos: • provincia • ciudad

	• usuario
Salida	Mostrar de forma ordenada y exitosa el ingreso de los datos.
Descripción	En la base de datos se almacena la información.
Seguimiento de prueba	1) Iniciar sesión en la página web. 2) Llenar los datos en los campos: provincia, ciudad, usuario. 3) Seleccionar la opción Grabar. 4) Se revisa y se acepta la información guardándose exitosamente.
Resultado	El sistema valida e ingresa la información en la base de datos.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 32. Caso de prueba: Modificar y actualizar la información

PR-FA-06	
Número de caso de prueba	06
Ciudadano	Administrador.
Referencia al caso de uso	La base de datos se actualiza y modifica en el sistema.
Nombre	Modificación de la información.
Entrada	Ingreso al sistema. Detalle de los nuevos datos: • parámetros • rangos

Salida	Visualización amigable, sencilla y vistosa para el ciudadano.
Descripción	Actualización de colores de fondo, tamaño de fuente y rango de conteo de baches.
Seguimiento de prueba	1) Iniciar sesión en la página web. 2) Llenar los datos en los campos de los parámetros que se van actualizar. 3) Seleccionar la opción Grabar. 4) Se revisa y se acepta la información guardándose exitosamente.
Resultado	En la base de datos el sistema valida y modifica la información.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.8.2. Prueba de usabilidad

El propósito de llevar a cabo estas pruebas de usabilidad es examinar el desempeño que el ciudadano muestra al utilizar las aplicaciones móvil y web.

En las tablas se encuentran registrados los casos de prueba y a continuación se detallan los siguientes campos:

- **Nombre:** Es la descripción del caso de prueba.
- **Seguimiento de prueba:** Secuencia de pasos para el desarrollo de la prueba.
- **Número de caso de prueba:** Es el orden secuencial del caso de prueba.
- **Descripción:** Información detallada acerca del caso de uso aplicado.
- **Resultado:** Descripción de la ejecución del procedimiento en el aplicativo móvil y web por parte del ciudadano.

- **Código:** Identificador del caso de prueba.
    - PR-UU-00 para las pruebas de usabilidad del ciudadano.

Tabla 33. Caso de prueba: Usabilidad de aplicación móvil.

PR-UU-07	
Número de caso de prueba	07
Nombre	Usabilidad
Descripción	Determina la sencillez con la que los ciudadanos pueden usar el aplicativo móvil.
Seguimiento de prueba	Se procedió a verificar la efectividad y la eficiencia de los elementos *touch* y el tamaño de la fuente sea legible.
Resultado	En conclusión, la aplicación móvil genera una buena experiencia para los ciudadanos.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tabla 34. Caso de prueba: Usabilidad de página web.

PR-UU-08	
Número de caso de prueba	08
Nombre	Usabilidad
Descripción	Es el grado de la facilidad de uso con la que los ciudadanos pueden navegar por la página web.
Seguimiento de prueba	Se inició a verificar que la interacción sea sencilla, intuitiva, agradable y segura.
Resultado	En conclusión, la aplicación web genera un ambiente amigable e interactivo con los ciudadanos.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.8.3. Prueba de geolocalización

El propósito de las pruebas de geolocalización es examinar, la precisión con la que trabaja la aplicación; y para esto se elaboró un cuadro comparativo que se visualiza en la Tabla36 donde indica el margen de error del sistema GPS.

En las tablas se encuentran registrados los casos de prueba y a continuación se detallan los siguientes campos:

- **Número de caso de prueba:** Orden secuencial del caso de prueba.
- **Descripción:** Información detallada acerca del funcionamiento del GPS.
- **Resultado:** Descripción de la ejecución del procedimiento en el aplicativo móvil por parte de los ciudadanos.
- **Código**: Identificador del caso de prueba.
  - PR-GM-00 para las pruebas de geolocalización del móvil.
- **Nombre:** Es la descripción del caso de prueba.
- **Procedimiento de prueba:** Secuencia de pasos para el desarrollo de la prueba.

Tabla 35. Caso de prueba: Geolocalización de dispositivo móvil.

PR-GM-09	
Número de caso de prueba	09
Nombre	Geolocalización
Descripción	Determina la precisión y el margen de error del GPS con y sin datos (internet), de los diferentes dispositivos móviles.
Seguimiento de prueba	Se procedió con dos procesos 1) Se ejecutó la aplicación con el uso de datos móviles y se verifico posteriormente la exactitud del GPS.

	2) Se ejecutó la aplicación sin el uso de datos móviles y se verifico posteriormente la exactitud del GPS.
Resultado	El primer resultado con uso de datos fue satisfactorio ya que su margen de error alcanzó menos de un metro en los distintos dispositivos móviles. El segundo resultado sin uso de datos fue insatisfactorio ya que el GPS no muestra los datos reales. Concluyendo que los dispositivos necesitan estar conectados obligatoriamente a los datos móviles y/o red wifi.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

En esta tabla se visualizó la precisión y el margen de error que arrojó las pruebas realizadas al sistema GPS en la ciudad de Quito con diferentes dispositivos móviles.

Tabla 36. Valores obtenidos en la prueba de calibración del GPS

		Real		Celular		Margen de error
	Repeticiones	**Latitud**	**Longitud**	**Latitud**	**Longitud**	**Distancia en metros**
Ubicación 1	1	-0.240675	-78.51167	-0.2406901	-78.5116577	2.16
	2	-0.241032	-78.511938	-0.2410412	-78.5119275	1.55
Ubicación 2	1	-0.255716	-78.523279	-0.255711	-78.5232884	1.18

	2	-0.256488	-78.524061	-0.2565018	-78.5240459	2.27
Ubicación 3	1	-0.290517	-78.561165	-0.2905065	-78.5611581	1.4
	2	-0.291284	-78.559467	-0.2912783	-78.5594659	0.65
Ubicación 4	1	-0.124479	-78.489818	-0.1244799	-78.489813	0.57
	2	-0.124474	-78.489817	-0.1244761	-78.4898115	0.66
Ubicación 5	1	-0.10427	-78.490108	-0.1042739	-78.490102	0.79
	2	-0.104224	-78.490096	-0.104229	-78.4900881	1.04
					Promedio	1.23

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.8.4. Resultados de pruebas

Las pruebas validan a este proyecto de titulación como resultado a la problemática de la existencia de los hoyos en la ciudad de Quito.

A continuación, se detallan los resultados obtenidos de las pruebas funcionales y de usabilidad; con todos los casos de prueba y sus respectivos procedimientos, los cuales se ejecutaron con la participación de un administrador y varios ciudadanos (conductores y transeúntes).

Las pruebas realizadas sobre las aplicaciones móvil y web Bache consistieron en evaluaciones y actualizaciones periódicas por distintos ciudadanos, los cuales fueron evaluando su funcionamiento y usabilidad, proporcionando soluciones constructivas acerca de cómo hacer más ágil a las mismas.

Tabla 37. Resultados enfocados en la funcionalidad y usabilidad de las aplicaciones web y móvil.

Código de prueba	Satisfactoria		Observaciones
	Si	No	
Funcionalidad			
PR-FU-01	X		Aprobado
PR-FU-02	X		Aprobado
PR-FU-03	X		Aprobado
PR-FU-04	X		Aprobado
PR-FA-05	X		Aprobado
PR-FA-06	X		Aprobado
Usabilidad			
PR-UU-07	X		Aprobado
PR-UU-08	X		Aprobado
Geolocalización			
PR-GM-09	X		El uso de datos (internet) es obligatorio ya que sin ellos el GPS no señala la ubicación real del conductor o transeúnte.

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

3.8.5. Pruebas de rendimiento página web

Estas pruebas permiten medir el rendimiento, la capacidad y las condiciones en las cuales responde una aplicación web al ser expuesto a una posible condición extrema como el ingreso de un volumen elevado de ciudadanos al mismo tiempo.

Para las pruebas de rendimiento se utilizó la aplicación web “GTmetrix”, la cual fue diseñada para probar el desempeño de una página web.

Se tomó en consideración 3 parámetros: tiempo completamente cargado, tamaño total de página y número de peticiones.

3.8.5.1. Resultados

A continuación, se detalla la tabla de resultados aproximados que se obtuvieron en las pruebas de rendimiento mediante la aplicación web anteriormente mencionado.

Tabla 38. Resultados de rendimiento de la página web

Tiempo completamente cargado	Tamaño total de página	Número de Peticiones
12.1 segundos	12.1 MegaBite	94

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tiempos de carga de la página web

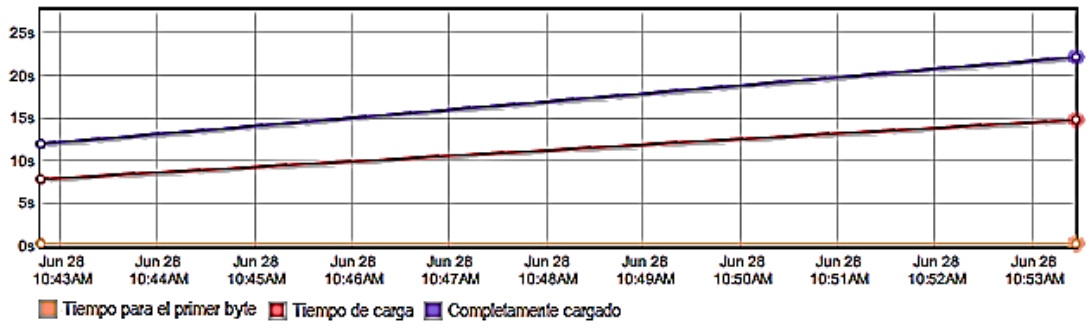


Figura 86. Tiempos de carga de la página web
Elaborado por: Daniel Riera y Jorge Soria (2019)

Tamaños de página y recuentos de peticiones y solicitudes

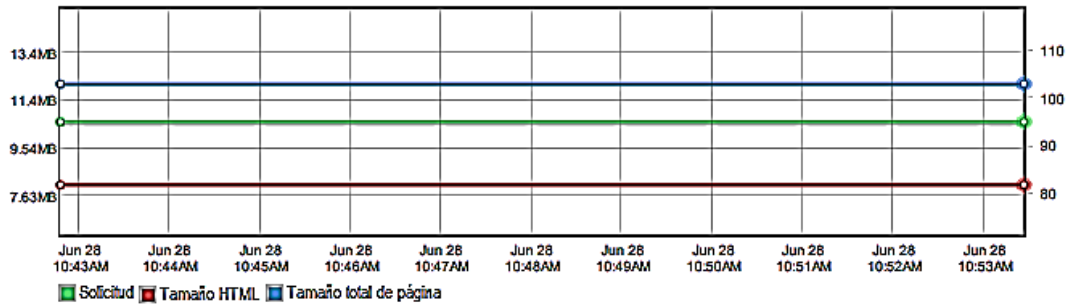


Figura 87. Tamaños de página y recuentos de solicitudes
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la tabla se puede observar que la aplicación soporta satisfactoriamente hasta 94 peticiones con un tiempo de respuesta mínimo de 12.1 segundos por lo cual se considera que el software está dentro del rango aceptable.

3.8.6. Pruebas de rendimiento aplicación web service.

La aplicación *SoapUI* es una herramienta diseñada para realizar pruebas a las aplicaciones y soporta *protocolos SOAP, REST, HTTP, JMS, AMF y JDBC.*

Para las pruebas de rendimiento se tomó en consideración 5 parámetros:

- Tiempo mínimo de peticiones
- Tiempo máximo de peticiones
- Consumo de megas
- Conteo de número de peticiones
- Conteo de número de errores.

3.8.6.1. Resultados

A continuación, se detalla la tabla de resultados aproximados que se obtuvieron en las pruebas de rendimiento realizadas con la aplicación web anteriormente mencionados.

Tabla 39. Resultados de rendimiento del web service

Tiempo Mínimo	Tiempo Máximo	Consumo de Megas	Número de Peticiones	Número de errores
284 milisegundos	5.836 segundo	0.001692 MB	60	0

Nota: Descripción general
Elaborado por: Daniel Riera y Jorge Soria (2019)

En la tabla anterior se puede visualizar que el servicio web soporta satisfactoriamente la concurrencia de 60 peticiones con un consumo de 0.001692 megabytes, con un tiempo de respuesta de 284 milisegundos por cada una. El total del tiempo fue 5.836 segundos por todas las solicitudes, durante las pruebas no hubieron errores por tal razón se determina que el servicio web funciona satisfactoriamente.

3.9. Óptica visual del software

3.9.1. Óptica visual de la aplicación web

Pantalla principal

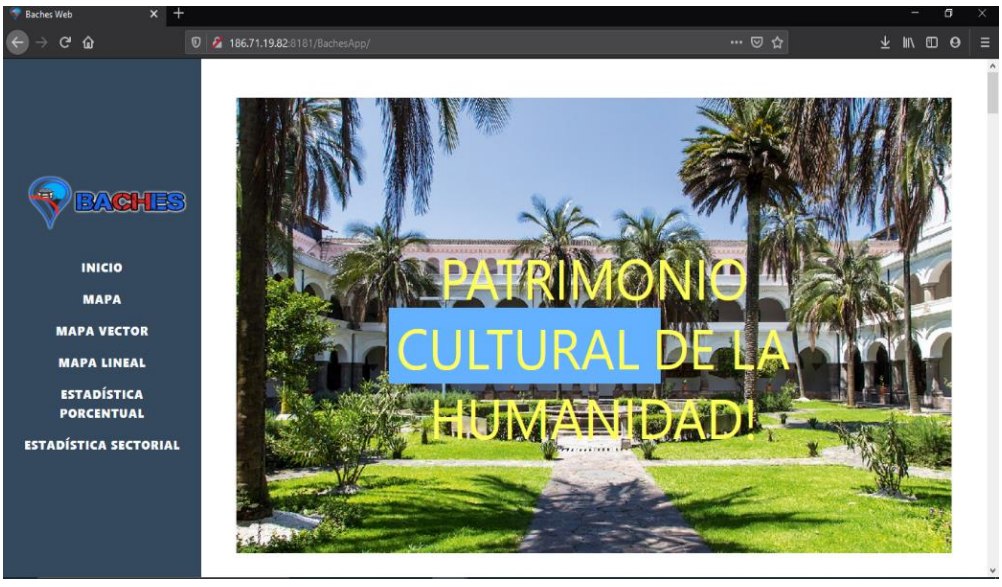


Figura 88. Pantalla principal de la aplicación web
Elaborado por: Daniel Riera y Jorge Soria (2019)

Mapa

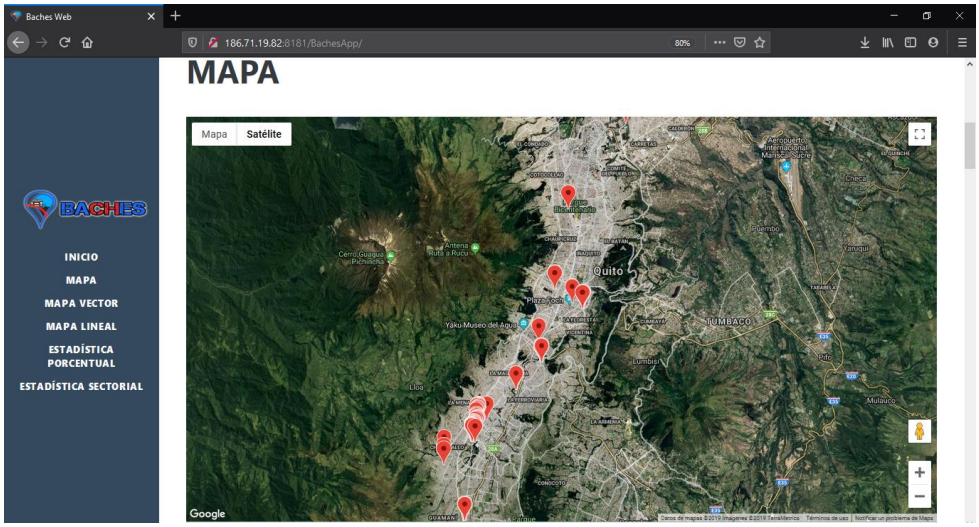


Figura 89. Muestra la ubicación de los baches en el mapa DQM
Elaborado por: Daniel Riera y Jorge Soria (2019)

Muestra la ubicación de los baches mediante coordenadas (latitud y longitud) en el mapa de Quito, registrados anteriormente por la aplicación móvil.

Tabla

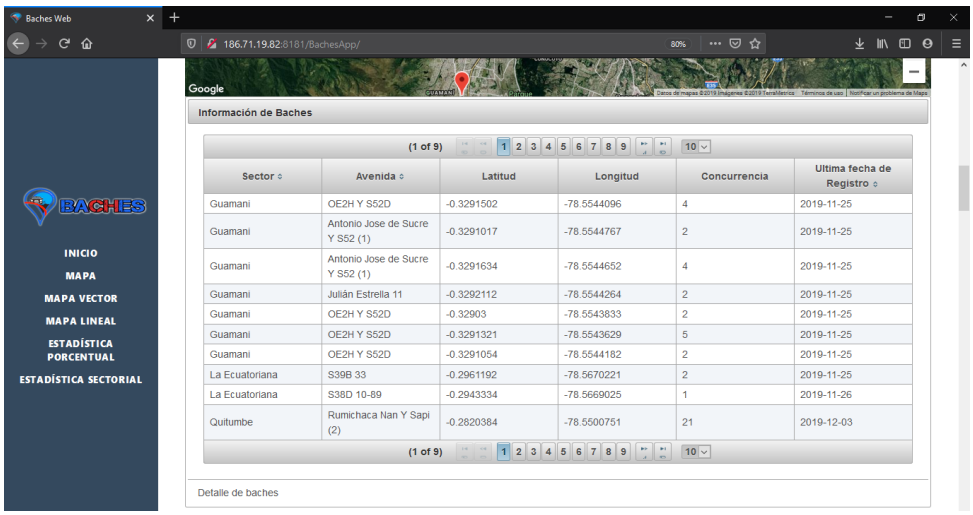


Figura 90. Tabla de registros de baches
Elaborado por: Daniel Riera y Jorge Soria (2019)

En esta tabla se presentan todos los registros realizados por la aplicación, detallados de la siguiente manera:

- **Sector:** la parroquia en donde se registró el bache.
- **Avenida:** la calle en la cual está ubicada el bache.
- **Latitud y longitud:** Las coordenadas para precisar en qué posición del mapa de Quito se encuentra el bache.
- **Concurrencia:** número de veces que ha sido registrado el mismo bache.
- **Última fecha de registro:** señala la fecha en la que fue registrado por última vez.

La tabla ayuda al ciudadano a buscar de forma ascendente y descendente los registros.

Mapa Vector y Lineal

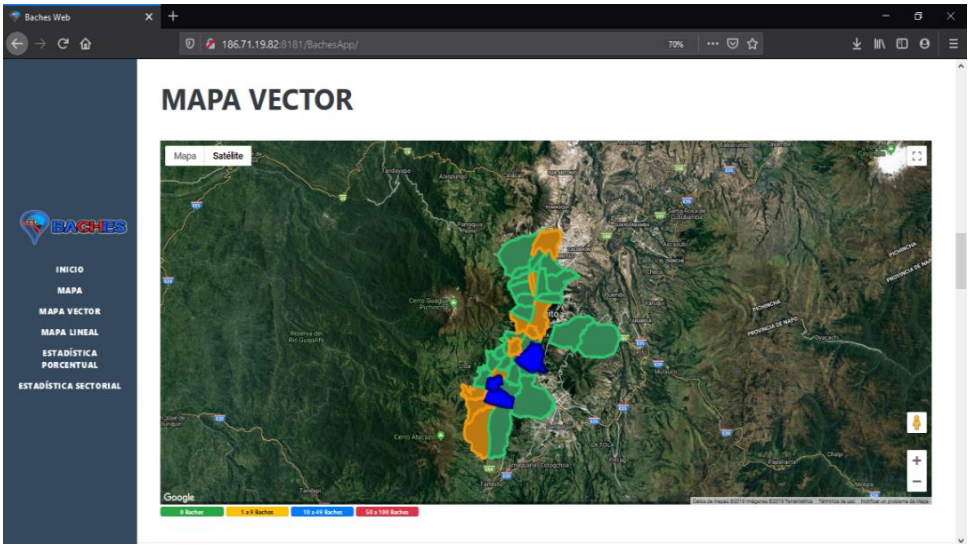


Figura 91. Mapa vector y lineal
Elaborado por: Daniel Riera y Jorge Soria (2019)

El mapa vector o lineal señala la cantidad de baches existentes en cada parroquia urbana de Quito, que tiene delimitaciones por un conjunto de coordenadas espaciales, las cuales ayudan a marcar dónde está ubicado el bache que ha sido registrado.

Estadística Porcentual y Estadística Sectorial

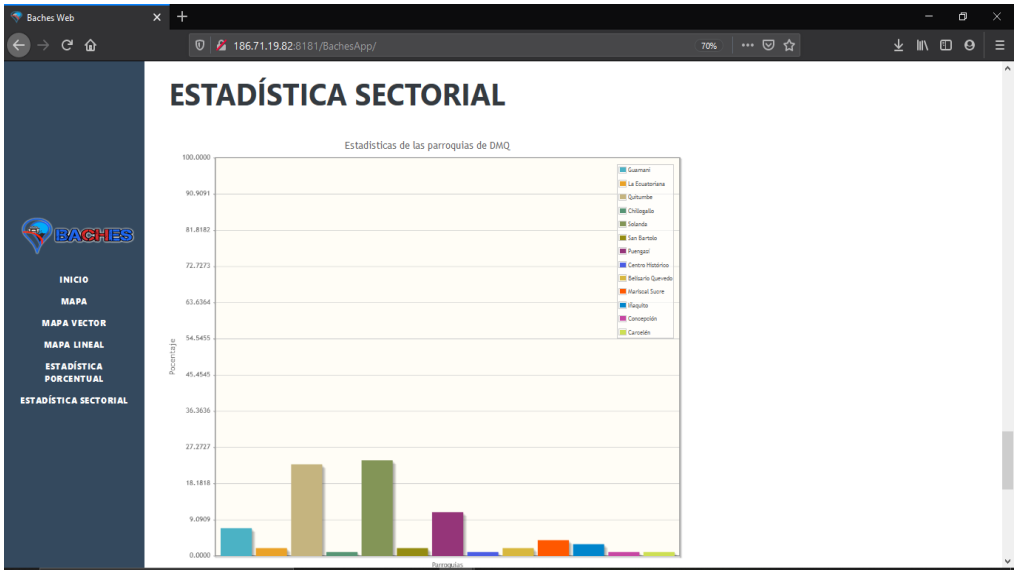


Figura 92. Estadística Porcentual y Estadística Sectorial
Elaborado por: Daniel Riera y Jorge Soria (2019)

Mediante estas estadísticas podemos demostrar cuales son las parroquias más atendidas y las menos atendidas dentro de la ciudad de Quito.

3.9.2. Perspectiva visual de la aplicación móvil

Aplicación móvil

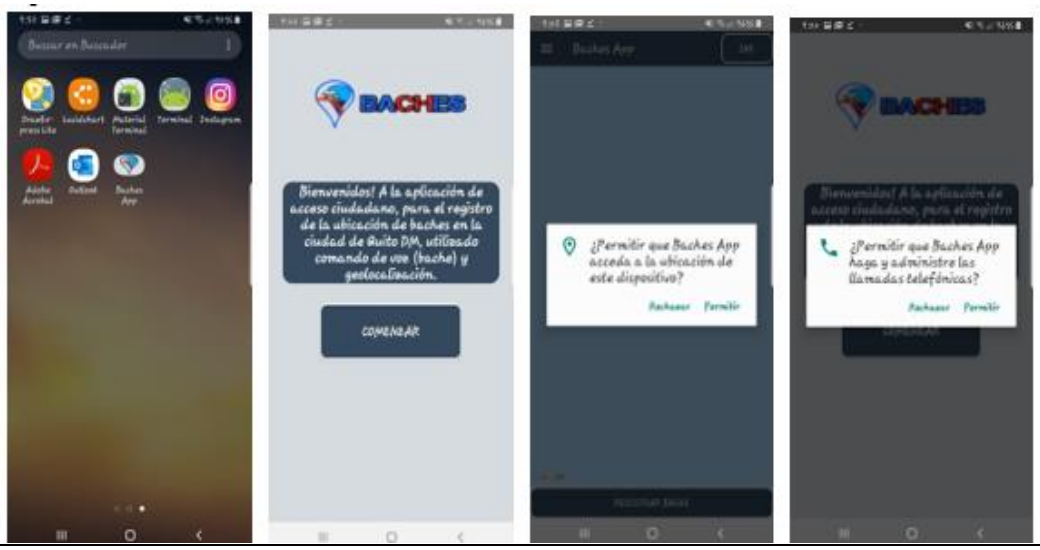


Figura 93. Aplicación móvil
Elaborado por: Daniel Riera y Jorge Soria (2019)

Aplicación móvil

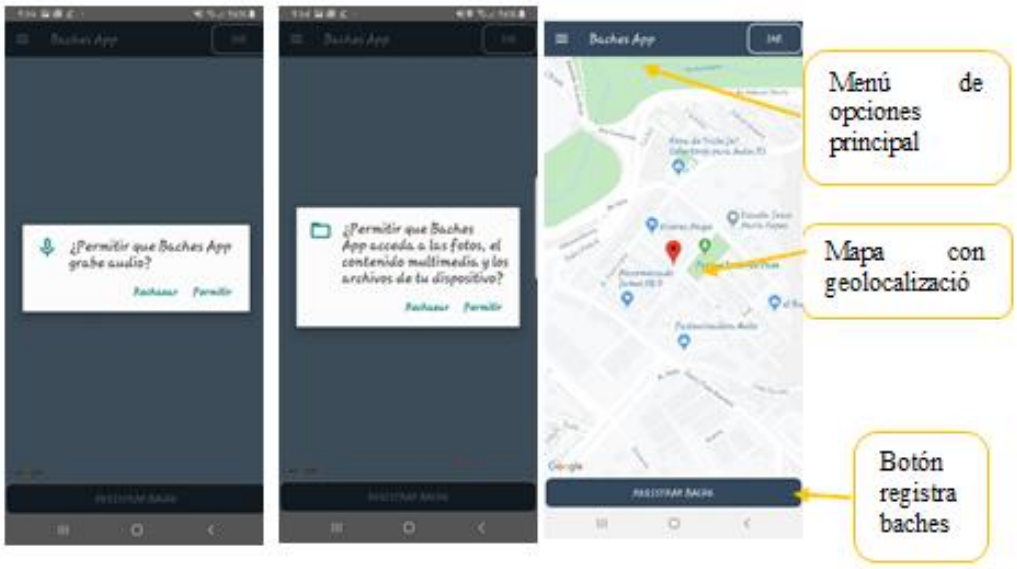


Figura 94. Pantalla principal y sus características
Elaborado por: Daniel Riera y Jorge Soria (2019)

Aplicación móvil

Figura 95. Menú y sus características
Elaborado por: Daniel Riera y Jorge Soria (2019)

CONCLUSIONES

- Esta aplicación ayudará a mantener la información actualizada de la existencia de baches evidenciando en tiempo real la existencia de este problema en las calles de la ciudad de Quito.
- Una vez que esta aplicación salga a producción se espera que se recolecte información que mantenga prevenida a las autoridades, conductores y transeúntes de la problemática que genera los baches en la ciudad de Quito.
- La forma en la que se interactúa con la aplicación es mediante el *comando de voz* que permite brindar a los ciudadanos la facilidad de registrar un bache, con el propósito de brindar un fácil manejo a los conductores.
- Al realizar el proyecto de titulación se comprobó cuál es el estado de las calles y avenidas de la ciudad de Quito constatando la existencia de baches en la mayor parte de las vías de la ciudad siendo este un problema que causa accidentes de tránsito y de transeúntes lo cual podría desencadenar en un final fatal por tal razón se considera la importancia de anticiparse a la caída de un bache.

RECOMENDACIONES

- En las versiones futuras se podría recomendar el desarrollo e implementación de la aplicación para el sistema operativo móvil iOS.
- Se recomienda incluir nuevas funciones como un reporte anticipado de las vías y calles en mal estado y de esta forma evitar la caída en un bache no visible para prevenir daños o accidentes. .
- Se sugiere realizar la actualización a la aplicación para que funcione de forma correcta en las versiones futuras de los sistemas operativos móviles.
- Se recomienda que el dispositivo móvil cuente con servicio de internet para que la ubicación con el GPS sea más precisa.

GLOSARIO DE TÉRMINOS

API. – Interfaz de Programación de Aplicaciones.

CSS. – Hojas de Estilo en Cascada.

CCIA. – Asociación de la Industria de Computación y Comunicaciones.

DNI. – Documento Nacional de Identidad

ERD. – Diagrama Entidad - Relación

GA. – Gobierno Abierto.

GNU. – Sistema Operativo de Linux

GPL. – Licencia Pública General

HTML. – Lenguaje de Marcado de Hipertexto.

IMEI. – Identidad Internacional de Equipo Móvil

JVM. – Máquina Virtual Java

JS. – JavaScript

JSF. – Java Server Faces

JSON. – Notación de Objeto de JavaScript

MVC. – Modelo Vista Controlador.

MAC. – Control de Acceso a Medios

OGC. – Consorcio Geoespacial Abierto.

SNI. – Sistema de Información Nacional.

URL. – Localizador Uniforme de Recursos

UML. – Lenguaje de Modelo Unificado.

API REST. – Transferencia de Estado Representacional

POINT. – Identifica un punto de los datos geográficos de PostGIS.

RAISE NOTICE. – Imprime la respuesta en la consola.

LISTA DE REFERENCIAS

Alicante, U. d. (26 de Junio de 2014). *Dpto. de Ciencia de la computaciòn e ingenieria artificial*. Recuperado el 6 de Febrero de 2020, de Tìtulo de experto universitario en desarrollo de aplicaciones y servicios con java EE: http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion01-apuntes.html

Alonso Aransay, D. (23 de Abril de 2010). *MappingGIS*. Obtenido de MappingGIS formación que impulsa tu perfil: https://mappinggis.com/2020/01/como-integrar-postgresql-postgis-en-r/

Alvarez, M. A. (2 de Enero de 2014). *Qué es MVC?* Obtenido de desarrolloweb.com: https://desarrolloweb.com/articulos/que-es-mvc.html

AMAP. (5 de Diciembre de 2014). *Estándar de codificación Java*. Obtenido de AMAP: https://amap.cantabria.es/amap/bin/view/AMAP/CodificacionJava

Arsys. (08 de 03 de 2017). *WildFly*. Obtenido de Arsys: https://www.arsys.es/blog/programacion/wildfly-cloud/

BBVAOPEN4U. (23 de Marzo de 2016). *API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos*. Obtenido de BBVA API_Market: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

Bootstrap, E. (2019). *Bootstrap*. Obtenido de Bootstrap: https://getbootstrap.com/

CCIA. (2014). *Android.* Obtenido de Desarrollo de Aplicaciones para Android: http://www.jtech.ua.es/cursos/apuntes/moviles/daa2013/wholesite.pdf

COMERCIO, E. (4 de Marzo de 2017). *Más baches en Quito*. Obtenido de EL COMERCIO: https://www.elcomercio.com/opinion/editorial/masbachesenquito-vias-editorial-opinion-columna.html

Content, R. R. (20 de Abril de 2019). *¿Qué es un lenguaje de programación y qué tipos existen?* Obtenido de Rockcontent: https://rockcontent.com/es/blog/que-es-un-lenguaje-de-programacion/

Cuello, J., & Vittone, J. (2017). *Las aplicaciones*. Obtenido de appdesignbook: http://appdesignbook.com/es/contenidos/las-aplicaciones/

DMQ. (2019). *¿Qué es el Gobierno Abierto?* Obtenido de Gobierno Abierto: http://gobiernoabierto.quito.gob.ec/?page_id=988

EPMMOP. (9 de Septiembre de 2019). *Rehabilitación y Mantenimiento Vial*. Obtenido de Repavimentación Km a Km: http://www.epmmop.gob.ec/epmmop/gestion-2019/obras-publicas/rehabilitacion-y-mantenimiento-vial.html

Garro, A. (28 de Enero de 2014). *HTML5*. Obtenido de HTML5 ARKAITZ GARRO: https://www.arkaitzgarro.com/html5/index.html

Guerrero, N. (8 de Junio de 2015). *¿Que es Java Hibernate?* Obtenido de Programa en linea: http://programaenlinea.net/que-es-java-hibernate/

Guevara Benites, A. (2016). *¿Que és Java y por qué aprenderlo?* Obtenido de DevCode: https://devcode.la/blog/que-es-java/

Hernández, J. (10 de Mayo de 2017). *¿Por qué Mauricio Rodas vuela tan bajo?* Obtenido de ELENFOQUE: https://4pelagatos.com/2017/05/10/por-que-mauricio-rodas-vuela-tan-bajo/

Hernandez, O. (s.f.). *Características de la CSS.* Obtenido de https://www.academia.edu/10934492/CARACTER%C3%8DSTICAS_DE_LAS_CSS?auto=download

Hidalgo Pérez, L. (3 de Julio de 2017). *Modelo entidad relación: descripción y aplicaciones.* Obtenido de ICEMD : https://www.icemd.com/digital-knowledge/articulos/modelo-entidad-relacion-descripcion-aplicaciones/

Inventó, Q. (20 de Junio de 2017). *¿Quién creó Android?* Obtenido de Quién inventó: https://www.quieninvento.org/quien-creo-android/

JesSalvTech. (4 de Mayo de 2016). *Aplicación con control por voz con PocketSphinx*. Obtenido de Tecnología referente a dispositivos móviles: https://jestechblog.wordpress.com/2016/05/04/tutorial-aplicacion-con-control-por-voz-con-pocketsphinx/

Krall, C. (2006). *¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML.* Obtenido de aprendeaprogramar.com: https://www.aprenderaprogramar.com/attachments/article/688/DV00205D%20que%20es%20uml%20versiones%20uml%20para%20que%20sirve%20lenguaje%20unificado%20modelado.pdf

KZblog. (31 de Marzo de 2017). *Geolocalización, qué es y cómo funciona*. Obtenido de Ikt- El Konektaturik conectados a las tic: http://kzgunea.blog.euskadi.eus/blog/2017/03/31/geolocalizacion-que-es/

KZgunea. (26 de Septiembre de 2018). *Geolocalización, Qué es y cómo funciona* . Obtenido de KZblog: http://kzgunea.blog.euskadi.eus/blog/2017/03/31/geolocalizacion-que-es/

Llario, J. C. (2018). *PostGIS 2 Anális Espacial Avanzado.* España: Universidad Politècnica de València.

Merizalde, M. B. (31 de Marzo de 2017). *Los baches se multiplican en Quito*. Obtenido de EL COMERCIO: https://www.elcomercio.com/actualidad/baches-lluvias-quito-municipio-epmmop.html

Morales, A. (18 de Marzo de 2016). *10 motivos para utilizar PostGIS.* Obtenido de MappingGIS: https://mappinggis.com/2012/09/por-que-utilizar-postgis/

Pavón Mestras, J. (s.f.). El patrón Modelo-Vista-Controlador (MVC). *Estructura de las Aplicaciones Orientadas a Objetos*, 6-7.

Pérez Arbesú, L. B. (05 de Agosto de 2015). *¿Cómo ayuda la metodología Scrum a la gestión de proyectos de TI?* Obtenido de TechTarget: https://searchdatacenter.techtarget.com/es/cronica/Como-ayuda-la-metodologia-Scrum-a-la-gestion-de-proyectos-de-TI

PostgreSQL, G. d. (14 de Noviembre de 2019). *PostgreSQL*. Obtenido de PostgreSQL: https://www.postgresql.org/about/

Ramirez Ordaz, J. M. (2007). Los Baches. *Asfáltica Revista Técnica*, 8-9.

Schwaber, K., & Sutherland, J. (Julio de 2016). *La Guía Definitiva de Scrum: Las Reglas del Juego.* Obtenido de La Guía de Scrum TM: https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf#zoom=100

SNI. (2014). *Catálogos de Datos de Abiertos Gubernamentales*. Obtenido de Sistema Nacional de Información: https://sni.gob.ec/datosabiertos

Suarez Jaimes, C. (2015). Base de Datos. *Diseño de Base de Datos*. Santander, Colombia: Universidad Pamplona. Obtenido de https://www.academia.edu/18160008/HISTORIA_DE_LAS_BASES_DE_DATOS

Team, L. C. (5 de Abril de 2019). *Introducción de tipos de diagramas UML*. Obtenido de Lucidchart: https://www.lucidchart.com/blog/types-of-UML-diagrams

Technology, T. (18 de Diciembre de 2019). *Tech: Difference between Agile Methodology and Scrum Methodology*. Obtenido de Tech Technology: https://theinscribermag.com/tech-difference-between-agile-methodology-and-scrum-methodology/

Toapanta Chancusi, K. M. (noviembre de 2012). *Método ágil Scrum, aplicado a la implantación de un sistema informático para el proceso de recolección masiva de información con tecnología móvil.* Obtenido de Escuela Politécnica del Ejército: https://repositorio.espe.edu.ec/bitstream/21000/5893/1/T-ESPE-034427.pdf

Utrilla, A. d. (18 de Mayo de 2007). *Capítulo2. Los baches.* Recuperado el 26 de Noviembre de 2019, de Universidad de las Américas Puebla : http://catarina.udlap.mx/u_dl_a/tales/documentos/lic/de_l_a/

Vela, J. P. (2017). *SCRUM: roles y responsabilidades*. Obtenido de Deloitte: https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html#