UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:
INGENIERO DE SISTEMAS

TEMA:
ANÁLISIS, DISEÑO Y DESARROLLO DE UNA APLICACIÓN MÓVIL CON LOS DATOS ALMACENADOS EN LA NUBE, PARA CONSULTAS DE TRANSPORTES URBANOS QUE TRANSITAN POR LA CIUDADELA TARQUI EN EL SECTOR DE LA MENA DOS

AUTOR:
DANIEL ALEJANDRO RUALES GÓMEZ

TUTOR:
FERNANDO JACINTO RODAS ORELLANA

Quito, agosto del 2017

CESIÓN DE DERECHOS DE AUTOR

Yo, Daniel Alejandro Ruales Gómez, con documento de identificación Nº.1717471955, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación con el tema "ANÁLISIS, DISEÑO Y DESARROLLO DE UNA APLICACIÓN MÓVIL CON LOS DATOS ALMACENADOS EN LA NUBE, PARA CONSULTAS DE TRANSPORTES URBANOS QUE TRANSITAN POR LA CIUDADELA TARQUI EN EL SECTOR DE LA MENA DOS", mismo que ha sido desarrollado para optar por el título de INGENIERO DE SISTEMAS en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo el derecho moral de la obra antes citada.

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

.....

DANIEL ALEJANDRO RUALES GÓMEZ

CI: 1717471955

Quito, agosto del 2017

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el proyecto técnico, con el tema: "ANÁLISIS, DISEÑO Y DESARROLLO DE UNA APLICACIÓN MÓVIL CON LOS DATOS ALMACENADOS EN LA NUBE, PARA CONSULTAS DE TRANSPORTES URBANOS QUE TRANSITAN POR LA CIUDADELA TARQUI EN EL SECTOR DE LA MENA DOS", realizado por Daniel Alejandro Ruales Gómez, 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 del 2017

.....

FERNANDO JACINTO RODAS ORELLANA

CI: 1708514821

DEDICATORIA

A lo largo de la vida siempre nos encontramos con personas especiales en el camino. A esas personas que han aportado tanto en mi vida dedico este trabajo.

A mi papi Nelson, gracias por siempre confiar en mí, por creer en que siempre puedo ser mejor, por enseñarme como verdaderamente debe ser un hombre, tu trabajo y tu cariño me han guiado siempre. Todas mis metas siempre serán dedicadas a ti.

A mi mami Paty, su preocupación y cariño siempre me han ayudado a seguir adelante, siempre será mi motivo para ser el mejor.

A mi familia, sé que donde esté siempre contaré con ustedes, gracias por tanto apoyo, a ustedes también se los dedico.

A ti Andrea, que llegaste cuando más te necesitaba, y has sido fundamental para el logro de esto, te agradeceré siempre.

A mis buenos amigos, compañeros de vida, donde quiera que estén nunca cambien.

Dedico esto a todos los que siempre creyeron en mí. Gracias por todo.

Daniel Alejandro Ruales Gómez

AGRADECIMIENTOS

A Dios que siempre ha estado a mi lado cuidándome y cuidando a los que quiero. Agradezco a la Universidad Politécnica Salesiana por la gran experiencia de ser alumno de esta institución, su enseñanza no solo en lo profesional, sino también en los valores humanos que guían ahora mi camino profesional. Gracias Ingeniero Fernando Rodas por darme la oportunidad de realizar este trabajo junto a usted, su guía y su amistad hicieron de este proceso una gran experiencia.

Daniel Alejandro Ruales Gómez

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

El presente proyecto de titulación tiene como propósito el análisis, diseño y desarrollo de una aplicación móvil con los datos almacenados en la nube, para consultas sobre el transporte urbano que transita por la ciudadela Tarqui en el sector de la Mena Dos. Los usuarios tendrán información sobre: la ubicación exacta de las unidades, en qué tiempo llegan las unidades hacia las paradas, horarios operativos de cada cooperativa, información sobre estado de la unidad y qué rutas son las que toma cada cooperativa. Esta información ayuda al usuario a tomar decisiones rápidamente acerca de cómo movilizarse en tiempo real, buscando la interacción entre los usuarios y las cooperativas mediante la cual se brinda un mejor servicio en cada una de las unidades. Para realizar estas actividades, es necesario implementar un tiempo real estricto en el envío de datos de las unidades de transporte urbano y recepción de datos en la aplicación de los usuarios, que ofrece información acerca de la distancia y el tiempo en que se encuentran los buses con respecto al usuario con un retraso imperceptible, diseñando así una aplicación móvil para los usuarios que sea intuitiva y amigable considerando las tecnologías más actuales disponibles, buscando mejorar los tiempos de llegada de los pobladores a sus destinos..

El logro de los objetivos planteados muestra la capacidad de crear nuevas ideas en el tema de movilidad lo cual podría ser motivo de estudio para su aplicación en toda la ciudad de Quito.

Abstract

This project has as objective at the analysis, design and development of a mobile application with the data stored in the cloud, for consultations of the urban transport that transit through of the sector of Mena Dos. Users will be able to have information about: which public transportation units pass through a bus stop, when the units will arrive to them, operating times of each cooperative, current capacity of each nearby unit and which routes are taken by each cooperative.

This information will help the user to make quick decisions about how to mobilize in real time, seeking the interaction among users and cooperatives to provide a better service in each of the units.

In order to carry out these activities, it is necessary to implement a strict real time in the sending of data of urban transport units and reception of data in the users' App, which will provide information about the distance and time in which the buses are With respect to the user with an imperceptible delay, thus designing a mobile application for users that is intuitive and friendly considering the most current technologies available to improve the arrival times of the settlers to their destinations, better managing the routes and times.

The achievement of the proposed objectives shows the ability to create new ideas about mobility, which could be a reason for study for its application in the city of Quito.

INTRODUCCIÓN

Antecedentes

> Con el paso del tiempo el transporte en la ciudad de Quito debe enfrentar retos de mejoramiento en cuanto a movilidad se refiere, ya que el parque automotor sigue en aumento y la geografía de la ciudad dificulta la fluidez del tráfico en horas pico (de 07h00 a 9h30 en la mañana y de 16h00 a 19h30 en la tarde) (Agencia Metropolitana de Tránsito, s.f.).

Uno de los sectores que mayor población ha adquirido es la Ciudadela Tarqui, ubicada en el sector de la Mena Dos, que en la actualidad está densamente poblado, esto ha encaminado a que tres cooperativas de buses (Vencedores de Pichincha, Quitumbe y San Francisco Corredor Sur Occidental R15) lleven sus servicios a estos sectores, pero qué, no cuentan con las unidades suficientes para abastecer en su totalidad la demanda en horas pico. Debido a esto las unidades salen llenas desde la parada principal, dificultando la movilización en el transporte público de las personas de esta zona y provocando así un retraso en el tiempo planificado, desconociendo si existe otra línea de bus que pueda tomar para embarcarse hacia su destino.

Problema de estudio

El desconocimiento de horario de las unidades, el tiempo de salida de cada una, a qué distancia se encuentra el transporte público con referencia a la ubicación del usuario y el desconocimiento de las rutas que siguen, provocan un retraso considerable en la estimación de tiempos de movilización. Este problema es cada vez más caótico en la capital ya que la movilización en el transporte público es sumamente difícil para este sector, muchas veces las personas optan por caminar hasta la avenida Mariscal Sucre en donde se puede tener mayor facilidad para acceder al transporte por la existencia

del corredor sur occidental. La falta de información sobre la ubicación de las unidades no permite que los usuarios puedan tomar alternativas para llegar hacia su destino, dificultando la movilización y colapsando las unidades que son más utilizadas, identificando así, un desconocimiento sobre la existencia de otras unidades alternativas para su movilización

Justificación

Es posible mejorar esta situación, ya que, si los usuarios de esta ciudadela pudieran acceder a información de las líneas de buses que pasan por la parada más cercana con respecto a su actual ubicación, pueden tomar decisiones rápidamente y optar por buscar otra unidad que facilite el acceso a la ruta deseada. Para esto se plantea el análisis, diseño y elaboración de una aplicación móvil que permita al usuario saber:

- Qué unidades de transporte público pasan por el lugar determinado.
- En qué tiempo llega una unidad hacia una posible parada.
- Horarios operativos de cada cooperativa.
- Qué rutas son las que toma cada cooperativa.

El correcto funcionamiento de este proyecto busca ayudar a encaminar nuevas ideas en movilidad que podría ser motivo de estudio para su aplicación en toda la ciudad de Quito, ya que es necesidad de los ciudadanos poder gestionar su tiempo entre esperas de transporte público y elegir de mejor manera el mismo, teniendo a la mano toda la información que esta aplicación puede brindar, esperando que su posible éxito pueda conllevar a la escalabilidad de la misma en un futuro.

Objetivos General - Específicos

Objetivo general

Análisis, diseñar e implementar una aplicación móvil con los datos almacenados en la nube, para consultas de transportes urbanos que transitan por la ciudadela Tarqui en el sector de la Mena Dos.

Objetivos específicos

Diseñar una aplicación móvil en Android para los usuarios en general que sea intuitiva y amigable considerando las tecnologías más actuales disponibles.

Implementar un tiempo real estricto (hard real time) en el envío de datos de las unidades de transporte urbano y recepción de datos en la App de los usuarios en general.

Promover la interacción de los pasajeros con las cooperativas de transporte público mediante el uso de formularios de sugerencias e informe de estado de las unidades.

Mostrar la ubicación en tiempo real de las unidades transporte público e información de las mismas que sea relevante para el usuario.

Evaluar el sistema en un entorno apropiado y comprobar su funcionamiento.

Marco metodológico

Para el presente proyecto se considera adecuado el uso de metodologías ágiles, que brindan un planeamiento claro, manteniendo un rumbo estable en desarrollo del proyecto en un entorno cambiante el cual comprende una serie de requisitos que son poco claros y seguirán cambiando a lo largo de la elaboración del mismo.

Para este proyecto se ha elegido el uso de Scrum, al ser una de las metodologías agiles más demandadas en la actualidad, se puede aplicar al desarrollo de cualquier proyecto

tanto pequeños como grandes. Sus principales fortalezas radican en la facilidad de auto organización que tienen los equipos y la exigencia de dividir las tareas en sub tareas, así se realizan ciclos cortos de trabajo llamados sprints, y al finalizar cada uno de los mismos se entrega un producto funcional. “es un marco que se aplica a portafolios, programas o proyectos de cualquier tamaño y complejidad; y se puede aplicar de manera efectiva en cualquier industria para crear un producto, servicio, o cualquier otro resultado” (SCRUMstudy, 2016).

Proceso de aplicación de Scrum

Roles

Una práctica recomendable para trabajar con Scrum es la división en roles que se conocen como: Product Owner, Scrum Master y el Scrum Team. Para establecer estos roles el tamaño adecuado del equipo debe ser lo suficientemente pequeño para que su desarrollo siga siendo ágil y tan grande como para que el trabajo realizado tenga un valor significativo. Teniendo en cuenta que el presente proyecto se realiza solo por una persona, la división de los roles se complica dando como resultado que el Scrum Team también tenga el rol del Scrum Master. En este caso no se encontró ninguna restricción de Scrum que impida el desarrollo con una sola persona, pero si mantiene advertencias sobre el equipo de desarrollo, el cual puede encontrar limitaciones en cuanto a conocimientos y habilidades durante la elaboración de los Sprints, riesgos que se han decidido tomar al ver los beneficios que se obtienen al trabajar con Scrum.

Historias de usuario

Son descripciones que deben ser tomadas por parte del Product Owner emitidas por parte de los clientes o stakeholders. Son las especificaciones de requisitos que describen las funcionalidades que ayudan en la elaboración del proyecto las cuales tendrán el siguiente formato:

Tabla 1. Formato de las historias de usuario

Columna	Instrucciones
Identificador (ID) de la historia	Código que identifica a cada historia de usuario
Rol	Rol que realiza el usuario cuando utiliza la funcionalidad que se especifica.
Funcionalidad	Es la función que el rol desea realizar en el sistema.
Razón / Resultado	Lo que el rol espera como resultado después de utilizar la funcionalidad
Número de escenario	Una funcionalidad puede tener varios escenarios, los cuales se enumeran en este campo.
Criterio de aceptación (título)	Describe el comportamiento de cada escenario posible de una determinada funcionalidad.
Contexto	Muestra una mayor descripción sobre las condiciones que de cada escenario.
Comportamiento esperado	Se manifiesta el resultado que se espera después de utilizar cada funcionalidad.

Nota: Contiene la descripción de cada campo que se utilizará para tomar las historias de usuario.

Sprint

Es un lapso de tiempo de un mes o menos en el que se elabora un incremento el cual es un producto utilizable, que cumple un objetivo y le de valor al producto final.

Product Backlog

Es una lista ordenada y priorizada de los requisitos que se solicitan para la elaboración del producto. Es un documento de alto nivel, que contiene descripciones que no se detallan y que es dinámica, esto quiere decir que a lo largo del desarrollo puede ser modificada cuantas veces sea necesaria.

Sprint Backlog

La pila del Sprint es una lista en donde se descompone las funcionalidades de la pila del producto en varias tareas, estas son necesarias para desarrollar un incremento la misma que debe ser operativa y tener un estado funcional para el producto. En esta fase se debe realizar una reunión llamada Sprint Planning que ayuda en la priorización de la pila del producto, buscando siempre desarrollar primero los Sprint que proveen mayor valor al cliente.

Daily Scrum

Lapso de tiempo de máximo 15 minutos al inicio de la jornada, en el que se planea las actividades diarias sobre el Sprint que se está trabajando. Para el adecuado uso del tiempo de esta reunión, debe realizarse las siguientes preguntas:

- ¿Qué hice ayer que ayudó a lograr el objetivo del Sprint?
- ¿Qué haré hoy para ayudar en el logro del objetivo del Sprint?
- ¿Veo algún impedimento que evite que se logre el objetivo del Sprint?

Sprint Review

Al terminar un Sprint se realiza una reunión en la que el Product Owner identifica las funcionalidades que han sido elaboradas, esto nos dará un feedback con que se busca mejorar el producto. Esta reunión no debe durar más de dos horas, a menos que el Sprint haya durado más de dos semanas.

Sprint Retrospective

Reunión que se realiza cuando se aprobó el Sprint anterior y antes de la reunión del Sprint siguiente, para hacer un autoanálisis de la forma de trabajar. El objetivo es la mejora continua cada vez que se inicie un Sprint. Esta reunión no se toma como revisión del resultado del Sprint, sino como revisión de cómo se está realizando el trabajo.

CAPÍTULO 1

1 Marco teórico

A lo largo de los años siempre ha existido una constante crítica hacia el transporte público. Casi 70% de los ecuatorianos frecuenta el uso del mismo. Pero cada día existen un sinnúmero de quejas por parte de estos usuarios, ya que no son bien atendidos o no logran tomar el transporte público por el exceso de pasajeros existente, el mal estado de las unidades o un pésimo servicio. A falta de soluciones comunes a estos problemas, es necesario recurrir a soluciones alternativas para poder mejorar la situación de tiempo en horas pico.

1.1 Parroquia Mena Dos – Ciudadela Tarqui

La parroquia de la Mena Dos se ubica al suroccidente de Quito en la Zona Eloy Alfaro y cuenta con área total de 869,1 hectáreas. Con el paso del tiempo, este sector va adquiriendo mayor población, la misma que busca acentuarse en las faldas de la montaña, convirtiéndose así en uno de los sectores más poblados de la ciudad, mismo que cada vez necesita de la mejora de sus servicios básicos debido a la proyección con la que contará en varios años en cuanto a población se refiere. "El Barrio está identificado principalmente por su posición geográfica, ya que se pueden encontrar importantes pendientes que van del orden del 50 hasta el 70 por ciento" (Sánchez, 2014).

Tabla 2. Proyección de población

Parroquia	Población Censo	Proyección año y tasa de crecimiento							
	2001	**2005**	**tc**	**2010**	**tc**	**2015**	**tc**	**2020**	**tc**
Mena Dos	36.825	43.510	4.3	53.106	4.1	64.137	3.8	76.655	3.6

Nota: Proyección de la población de la parroquia Mena Dos. Fuente: (Eliécer, 2010)

La gran mayoría de los pobladores del sector pertenecen a un nivel económico medio-bajo y con dicha proyección de crecimiento, hace de la misma un lugar con un volumen de comercio mediano.

La parroquia cuenta con todos los servicios básicos, lo que hace de este sector un lugar donde los pobladores pueden realizar sus actividades diarias sin inconvenientes. Uno de los indicadores que se muestra en la figura 1, es el estado de las vías, indicador que es fundamental para la planificación de movilización en el sector, siendo este un porcentaje del 92,4%, convirtiéndose en un sector de fácil acceso vehicular.

Indicadores de servicios básicos en la parroquia Mena Dos

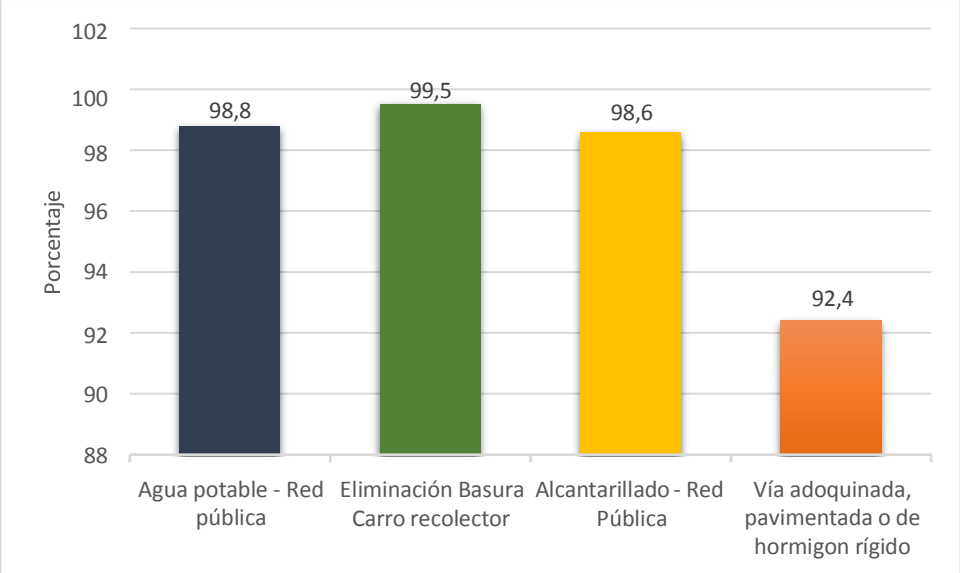

Figura 1. Indicadores de cobertura administración zonal Eloy Alfaro segun parroquias. Fuente: (Eliécer, 2010)

El municipio de Quito ha procurado cubrir todos los servicios básicos de esta parroquia, gracias a esto, los habitantes también pueden contar con servicios complementarios, los cuales son fundamentales para el desarrollo de este proyecto, ya que es necesario saber que porcentaje de la población cuenta con acceso a internet y que porcentaje cuenta con teléfonos celulares.

Tabla 3. Servicios complementarios

Descripción	Porcentaje
Viviendas con Energía por panel Solar	-
Cocinas con gas centralizado	0.03
Hogares por lo menos con un celular	88,7
Disponibilidad de computadora	49,6
Disponibilidad de servicios de internet	24,0
Disponibilidad de servicios de televisión por cable	50,3
Hogares que compran agua purificada	16,2

Nota: Porcentaje de cobertura de servicios complementarios en la parroquia de la Mena Dos. Fuente: (Eliécer, 2010)

Tomando en cuenta las cifras de la tabla 3, podemos observar que una gran parte de la población de la parroquia cuenta con acceso a un celular, mostrando así una alta posibilidad de aceptación en el uso de la aplicación que se desea realizar,

1.2 Movilidad

En el desarrollo diario de las personas, interviene un tema fundamental conocido como movilidad. Este punto es de gran interés, ya que el tratamiento de este tema conlleva el estudio y solución de los problemas de transporte desde un marco completo, pero lamentablemente el éxito de la movilidad no crece al mismo ritmo que crece el parque automotor y esto dificulta enormemente el desplazamiento de los usuarios concurrentes de este servicio. La dificultad al tratar de mejorar la movilidad se ha convertido en un tema de estudio en las grandes ciudades, que ven cada vez más difícil la movilización.

La ciudad de Quito presenta grandes dificultades en temas de movilidad, ya que el parque automotor sigue creciendo y con el pasar del tiempo no se cuenta con vías

suficientes para sostener todo el tráfico que la población genera ya que según estudios del Municipio del Distrito Metropolitano de Quito el crecimiento del parque vehicular en los últimos 10 años oscila entre el 5% y 10%. De mantenerse este crecimiento, la tendencia revela que para el 2030 se tendrá un parque automotor con aproximadamente 1'150.000 vehículos como se muestra en la figura 2, esto dará como resultado la dificultad de mantener una adecuada movilidad en la ciudad de Quito y por ende distintos problemas en las actividades diarias de la sociedad.

Crecimiento del parque automotor de Quito

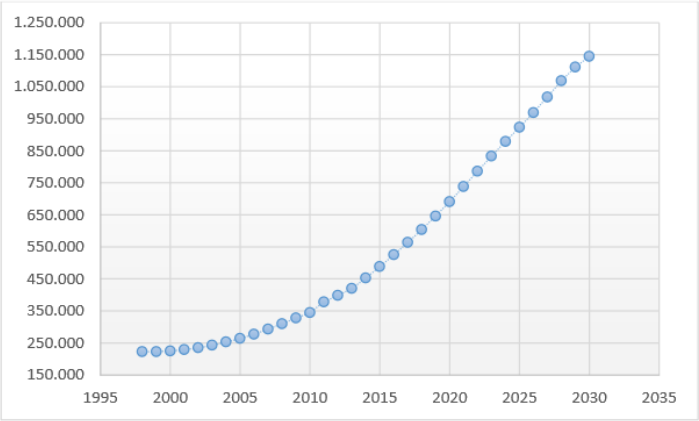

Figura 2. Evolución y proyección del parque automotor en Quito.
Fuente: (Municipio del Distrito Metropolitano de Quito, 2014).

Al tener este incremento año tras año, los pobladores deben utilizar mayores tiempos para llegar a sus destinos, pues según el Municipio se ha identificado que el tiempo promedio se han incrementado en un orden del 7% respecto a registros en el 2008, ya que la velocidad en las calles principales ha pasado de 19.9 km/h a 14,1 km/h, con excepción de las vías exclusivas de transporte público como son: trolebús, ecovia y corredor sur occidental las cuales cuentan con una velocidad de 19.8 km/h, siendo por esta razón, las rutas más demandadas por los usuarios. (Municipio del Distrito Metropolitano de Quito, 2014, pág. 8).

1.3 Modelo de operaciones de las cooperativas de transporte público.

A partir del 2 de enero del 2014 en la ciudad de Quito se implementó el sistema de caja común para el manejo del negocio de cooperativas de transporte público, modalidad que fue adoptada por 54 cooperativas de las 62 que circulan en la ciudad. El objetivo de implementar este sistema es centralizar la administración y operación de las cooperativas, organizar las rutas, horarios y mantenimiento de las unidades, pero sobre todo recaudar todos los fondos conseguidos por el desarrollo del negocio en un fondo común, este será repartido de forma equitativa en el final del mes a todos los miembros de la cooperativa. Esto se realiza con el fin de mejorar el servicio en el transporte público e incluir beneficios para los choferes y controladores para que mantengan un sueldo mensual fijo y prestaciones de ley.

1.4 Geolocalización

La geolocalización es el proceso por el cual podemos obtener la ubicación especifica de algo determinado previamente en la tierra, asignándole coordenadas las mismas que se obtienen por medio de tecnología GPS la cual puede posicionar un objeto con errores mínimos. "Quizás el sistema más conocido sea el sistema de coordenadas geográficas que utiliza dos de las tres coordenadas de un sistema de coordenadas esférico para situar cualquier punto en la superficie terrestre, la latitud y la longitud." (Observatorio Regional de la Sociedad de la Información (OSRI)., 2009).

1.5 Google Play Services

En el desarrollo del proyecto es necesaria la implementación de la API de Alphabet Google Maps, la cual facilita el manejo de mapas gracias a sus múltiples funcionalidades principalmente mostrando mapas de cualquier parte del mundo.

Para el funcionamiento adecuado de la API Google Maps es necesario tener la aplicación Google Play Services, que es una plataforma que distribuye aplicaciones y gestiona la mayoría de servicios en teléfonos basados en el sistema operativo Android, que ayuda a autentificar los servicios de Google proporcionando mapas actualizados de la ubicación deseada y nos brinda servicios de ubicación en tiempo real.

1.6 Sistema Operativo Android

Es uno de los sistemas operativos más demandados, ya que se encuentra en la mayoría de teléfonos inteligentes en la actualidad, gracias a su naturaleza de código abierto se pueden realizar un sin fin de aplicaciones para el uso en cualquier área que se necesite. "Está basado en Linux que es un núcleo de sistema operativo libre, gratuito multiplataforma. Este sistema operativo permite programar aplicaciones empleando una variación de Java llamada Dalvik" (Robledo, 2012)

1.7 Servicios web en Android

Los servicios web nos permiten desarrollar una aplicación que pueda operar entre diferentes aplicaciones sin importar el tipo de lenguaje en el que haya sido desarrollado.

"Un servicio debe poder ser accesible a través de la Web. Para ello debe utilizar protocolos de transporte estándares como HTTP, y codificar los mensajes en un lenguaje estándar que pueda conocer cualquier cliente que quiera utilizar el servicio." (Universidad de Alicante, 2013).

Para el manejo de servicios web en Android vamos a manejar los servicios web RESTful, que funcionan en base a los principios de REST manteniendo un enfoque de transferencia de recursos desde el cliente hacia los servidores y viceversa a través de HTTP como se ilustra en la figura 3.

Envío de peticiones HTTP

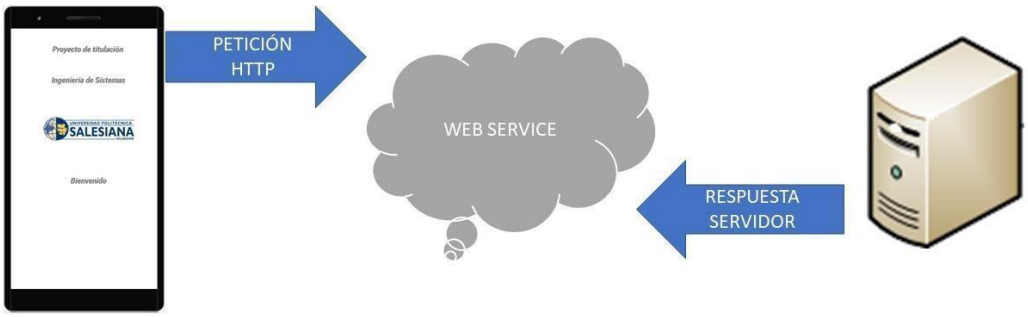

Figura 3. Representación de los servicios web por HTTP
Elaborado por: Ruales Daniel (2017).

REST utiliza formatos predecibles para quienes acceden a una API, usando direcciones similares a las que se puede observar en un navegador web, estas mismas deben contar con las condiciones necesarias para acceder a los registros y obtener los recursos deseados correctamente. Para la transferencia de estos recursos se utiliza formatos conocidos como JSON y XML, sin embargo, existen otros formatos que pueden ser aplicables.

1.8 Arquitectura de Android

El sistema operativo Android fue concebido con la idea de ser independiente de la arquitectura del hardware, pero en cuanto a software se refiere, su construcción se realizó en 4 capas relacionados entre si.

En la figura 4, podemos constatar que la estructura de Android está desarrollada sobre el Kernel de Linux. Seguidamente se encuentra la capa de Librerías la cual se relaciona con el administrador de tiempo de ejecución. Luego se encuentra la capa ubicada en el Framework que apoya en el desarrollo de las aplicaciones y finalmente capa de Aplicaciones donde correrán las aplicaciones mediante una máquina virtual.

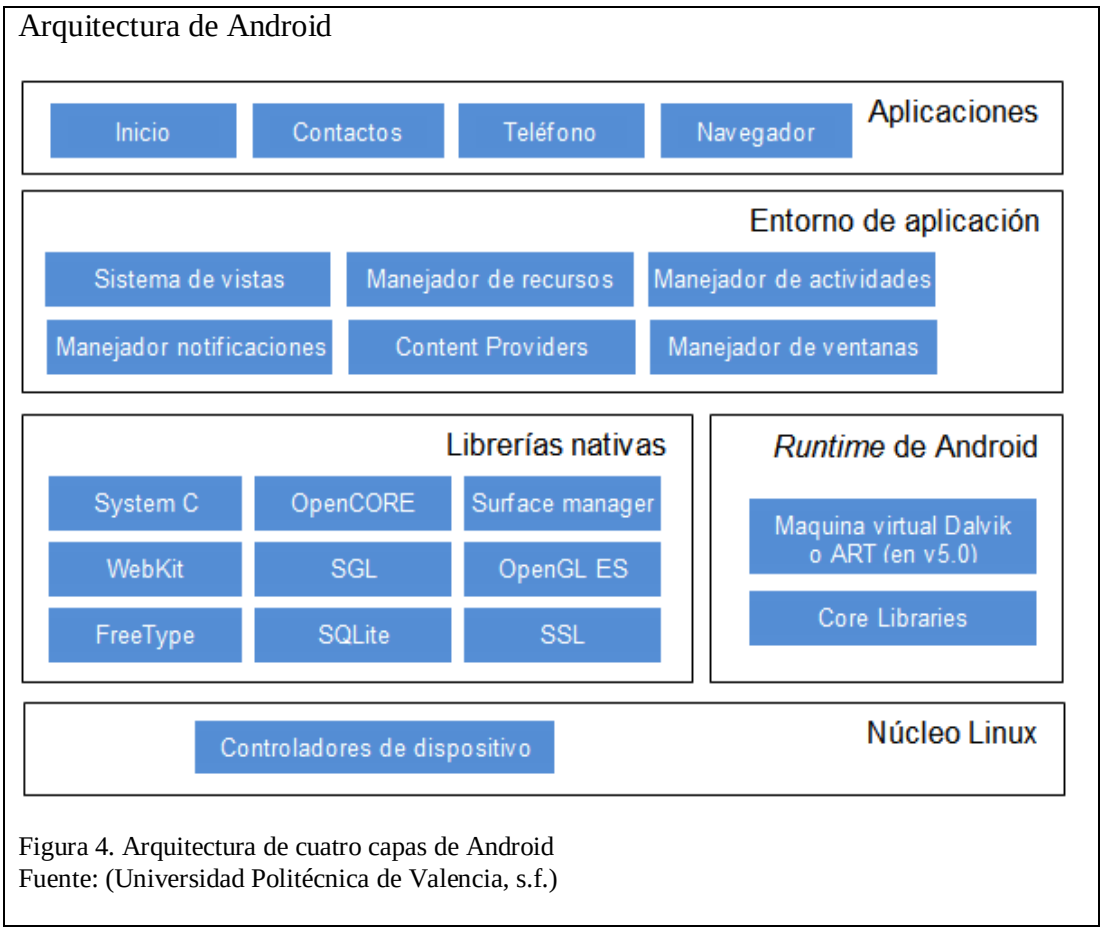

Figura 4. Arquitectura de cuatro capas de Android
Fuente: (Universidad Politécnica de Valencia, s.f.)

1.9 Android Volley

Volley es una librería de Android Studio que facilita la comunicación con Web Services siendo un cliente Http que nos brinda procesamiento concurrente de peticiones y sobre todo nos facilita el trabajo en segundo plano para una gestión automática del funcionamiento de las aplicaciones. Es una librería recomendada por Google ya que ayuda con una comunicación de alto nivel ya que ejecuta peticiones al servidor de manera clara y en tiempos cortos.

En la figura 5 se muestra el proceso de manejo de peticiones de la librería Volley, este inicia en una cola de peticiones que recepta las mismas, donde son priorizadas para luego ser enviadas.

Peticiones en Volley Android

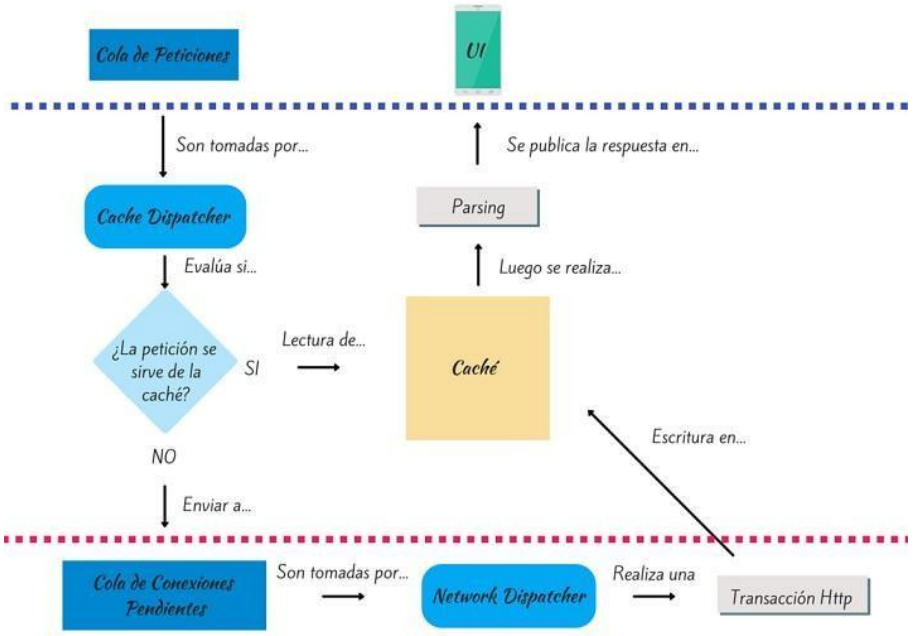

Figura 5. Diagrama de peticiones en Volley
Fuente: (Revelo, Hermosa Programación, 2015)

Esta priorización la realiza el Cache Dispatcher, su trabajo consiste en verificar si la petición actual contiene resultados previos que sirvan a esta petición y estén guardados en cache. Si obtiene una respuesta positiva, entonces se lee la respuesta almacenada en cache y luego se presenta. Si obtiene una respuesta negativa, se envía la petición a la cola de conexiones pendientes, donde esperan las peticiones para ejecutarse en el Network Dispatcher

1.10 Modelo – Vista – Controlador MVC

Como buena práctica en el desarrollo de cualquier sistema o aplicación, es necesario seguir patrones que proporcionan un marco de referencia para compartir una misma línea de trabajo con lo cual se desea conseguir productos de calidad, cumplir con los tiempos previstos y objetivos planteados.

Para el desarrollo del proyecto se utilizará la arquitectura MVC, patrón que ayuda en el diseño de sistemas cuya estructura se puede ver en la figura 6.

Fue diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales están dadas por el hecho de que, el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas. Este modelo de arquitectura se puede emplear en sistemas de representación gráfica de datos, donde se presentan partes del diseño con diferente escala de aumento, en ventanas separadas. (Fernandez & Yanette, 2012).

Ciclo de vida del MVC

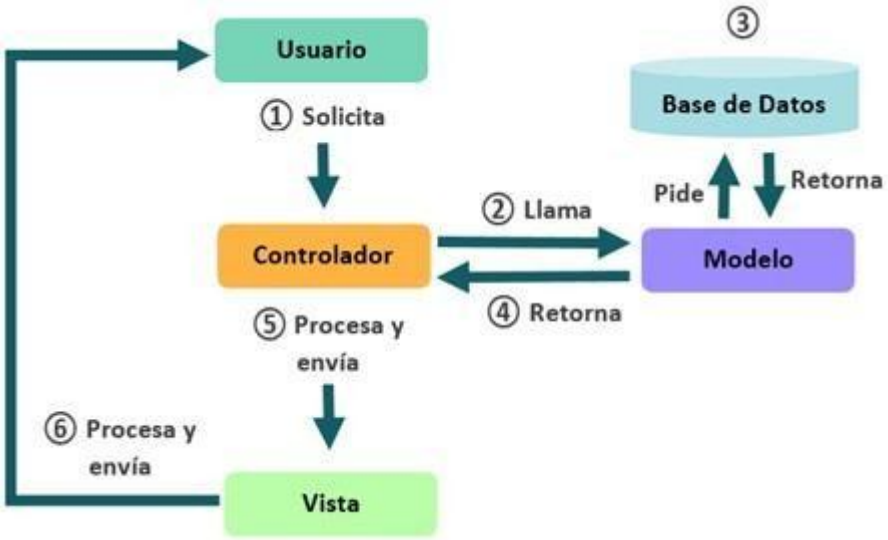

Figura 6. Ciclo de vida paso a paso del MVC
Fuente: (Goméz, 2015)

En el modelo se guarda toda la lógica del negocio con la que trabaja la aplicación, esto puede ser la forma como almacena los datos, o como utiliza los servicios de otras fuentes para cumplir con su propósito.

La vista se refiere al entorno donde encontraremos todos los elementos de programación necesarios para interactuar con el usuario, en nuestro caso serán los layouts que se presentan como interfaces graficas facilitando el diseño.

Y el controlador que es el puente o conexión entre la Vista y el Modelo, aquí el controlador recibe las peticiones del usuario, llama al modelo, el modelo retornará la información requerida y la envía al controlador, este la recibe y la envía a la vista para su publicación.

1.11 Bitbucket

"Sistema de control de versiones distribuido con el que te será muy fácil colaborar con tu equipo. La única solución Git colaborativa altamente escalable." (Atlassian, 2017).

> Bitbucket es un servicio de alojamiento basado en la web para proyectos que utilizan el sistema de control de versiones Mercurial y Git. Fue lanzado en el año 2008 por la empresa Atlassian Software y está escrito en Python mediante el framework web Django. (Hipertextual, 2014).

CAPÍTULO 2

2 Análisis y diseño

Dentro de este capítulo se muestran algunos detalles de la concepción general de la solución, empezando por el análisis de los requerimientos del sistema hasta llegar al diseño final con el que se construirá del software.

2.1 Elección de Android SDK

Para comenzar el desarrollo y realizar un análisis de factibilidad es importante tener en cuenta la versión de Android que se va a utilizar, ya que Android Studio brinda un kit de desarrollo para la creación de aplicaciones y sus distintas versiones.

Android Studio proporciona una lista de las versiones funcionales del SDK, que se deben tener en cuenta ya que muestra el porcentaje que cubre en el mercado dicha versión de SDK.

Versiones de desarrollo en Android

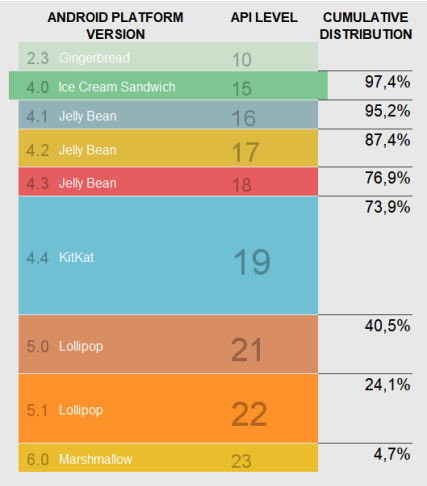

Figura 7. Versiones de Android SDK con sus porcentajes.
Fuente: (Android Plataform/API Version Distribution, 2017)

Al seleccionar una de las opciones que muestra la figura 7, obtenemos una versión de SDK que será compatible con versiones anteriores y su posible instalación en un mayor porcentaje de dispositivos.

2.2 Análisis de factibilidad

2.2.1 Factibilidad técnica

Para el desarrollo de la aplicación se ha priorizado el estudio de aspectos técnicos necesarios, como las características mínimas tanto en software como en hardware. El análisis de la factibilidad técnica es muy importante ya que el mismo afectará directamente en el análisis de factibilidad económica puesto que la elección de cada aspecto determina un costo para el proyecto.

Hardware requerido

Teléfono móvil

- Sistema operativo Android Ice Scream sándwich 4.0 o versiones superiores.
- Red GSM 850/900/1800/1900
- Compatibilidad con GPS
- RAM mínima 256MB
- 8GB de almacenamiento mínimo.

Computadora que cuente con:

- Sistema Windows 10/8/7 (32 o 64 bits).
- Procesador Intel Core I3
- Memoria RAM mínimo de 4 GB
- Espacio de almacenamiento de 350GB
- Resolución mínima de la pantalla de 1280 x 800

Servidor

- Acceso FTP
- Soporte para PHP y MySQL
- 50GB de espacio en disco

Software requerido

El software requerido debe ayudar a crear una aplicación con gastos mínimos y con facilidad de uso para el usuario, teniendo en cuenta que como restricción de diseño se debe utilizar software libre que sea estable y brinde soporte en caso de inconvenientes.

- Servidor de aplicaciones web XAMPP
- Java
- PHP
- MySql
- Emulador Genymotion
- SourceTree con conexión a Bitbucket
- Android Studio

Conclusión Factibilidad Técnica

Técnicamente es factible desarrollar este tipo de aplicación móvil ya que en el mercado existen todos los requerimientos solicitados, tendiendo como ventaja que el software a utilizar es libre y cuentan con soporte y amplia información sobre sus características, por lo tanto, se puede trabajar sin mayores inconvenientes.

En cuanto al hardware tampoco genera muchos inconvenientes puesto que todos los dispositivos requeridos se encuentran en el mercado ecuatoriano.

2.2.2 Factibilidad económica

A continuación, se toman en cuenta los gastos que se presentan en el diseño y desarrollo del presente proyecto.

Costos de software

Los costos en cuanto a licencias de software no existen ya que se optó por utilizar software libre, todos los programas mencionados en el literal de factibilidad técnica son de descarga gratuitita en internet.

Costos operativos

Los costos presentados en la tabla 4 son planteados en base al tiempo que dura el desarrollo del proyecto, tiempo que está estimado en 4 meses.

Tabla 4.Costos operativos

Costos Operativos	
Descripción	Costo
Sueldo programador Jr.	$700.00
Plan internet para dos celulares	$240.00
Arrendamiento de espacio en la nube por 4 meses	$120.96
Conexión a internet fijo 4 meses	$101.40
Alimentación	$200.00
Total	$101.40

Nota: Detalle de costos operativos para la realización del proyecto.

Costos en hardware

Tabla 5. Descripción de costos en hardware del proyecto

Costos Directos	
Descripción	Costo
2 celulares: 1 para actuar como cliente. 1 para actuar como controlador enviando coordenadas de ubicación.	$500.00
Computadora para desarrollo	$800.00
Total	$1300.00

Nota: Detalle de costos en hardware para la realización del proyecto.

En la tabla 5, se muestra un detalle de los costos en cuanto a hardware se refiere, pero hay que mencionar que no todos los costos deben ser considerados, como el uso de celulares y computadora ya que se cuenta con estos activos, dando como resultado un costo de $1362.36.

Conclusión Factibilidad Económica.

Con un costo de $1362.36 en un tiempo aproximado de dos meses de desarrollo se puede afirmar que el proyecto es factible económicamente gracias al uso de herramientas de software libre que no tienen costo alguno y considerando que el costo calculado es un costo accesible para la realización de este proyecto.

2.2.3 Factibilidad Operacional

La solución está orientada al público en general que vive en la Ciudadela Tarqui en el Sector de la Mena Dos, los cuales cuenten con un teléfono celular para recibir información de las unidades que cuenten con este sistema, grupo que está conformado por el 88.9% que se muestra en la tabla 3.

Un aspecto por tomar en cuenta es que la disponibilidad de la información está limitada al alcance de la señal de cobertura de la operadora y de la conexión a internet del equipo, además de contar con el apoyo de los operadores del transporte público, para que la aplicación funcione de manera óptima.

Conclusión factibilidad Operacional

Tomando en cuenta que gran parte de la población cuenta con acceso a un teléfono móvil y acceso a internet tanto móvil como fijo, el proyecto tiene cabida en este sector ya que tiene bastante afluencia de transporte público en el cual se puede implementar este tipo de proyectos.

2.2.4 Conclusión del análisis de factibilidad

En conclusión, la tecnología necesaria para desarrollar la solución existe y se registran proyectos similares que trabajan de manera exitosa. Económicamente la inversión necesaria para llevar a cabo el proyecto no es grande, por lo que su desarrollo resulta viable. Finalmente podemos constatar que este sector puede acceder a este tipo de aplicaciones, por lo tanto, gracias a las conclusiones antes planteadas podemos ver que el proyecto es factible sobre todos los puntos analizados.

2.3 Análisis de requerimientos

La elección de Scrum para el desarrollo de este proyecto conlleva a la creación de artefactos que son indispensables para el logro de los objetivos, dividiendo el trabajo en procesos más pequeños para facilitar su desarrollo, así también, como el uso de

historias de usuario para verificar las especificaciones que nos permitan elaborar los Sprints necesarios de trabajo.

2.3.1 Actores del sistema

Tabla 6. Actores del sistema

Actor	Tareas
Administrador	El administrador maneja un sitio web en donde puede ingresar las unidades con las que cuenta la cooperativa, ingresar nuevos controladores y leer los mensajes que vienen de la aplicación del cliente y del estado de la unidad en caso de fallo por parte del controlador
Pasajero	El pasajero mediante la aplicación puede consultar la ubicación en tiempo real de las unidades, así como también ver las rutas determinadas que tiene cada cooperativa. Recibe mensajes en caso de fallo de alguna unidad y puede enviar mensaje de sugerencias al sitio web del administrador
Controlador	El controlador mediante la aplicación puede enviar automáticamente su posición cada segundo y en caso de daño de la unidad o algún inconveniente por lo cual debe dejar de operar puede enviar mensajes de aviso.

Nota: Actores que interactúan con el sistema.

2.3.2 Requerimientos de usuario

En la tabla 7, se detalla los requerimientos por parte de los stakeholders, estos detalles se deben convertir en funcionalidades en el sistema y deben aportar valor para el cliente y se usa un lenguaje normal que sea fácil de entender, ya que estos requerimientos son tomados de conversaciones con el cliente y el Product Owner.

Tabla 7. Historias de usuario

ID	Rol	Funcionalidad	Resultado	Escenario	Título	Comportamiento esperado
HUPAS1	Como un pasajero	Necesito un login	Con la finalidad de ingresar a la App	1	Login	El usuario ingresa al sistema y verá sus opciones dependiendo del tipo de usuario que sea
HUPAS2	Como un pasajero	Necesito un mapa con marcador en movimiento	Con la finalidad de ver la ubicación en tiempo real	1	Horario laboral	A continuación, se mostrará un mensaje de aviso: "Escoja una cooperativa"
				2	Horario no laboral	Se mostrará un mensaje de aviso: "No existen unidades disponibles"

HUPAS3	Como un pasajero	Necesito escoger una cooperativa	Con la finalidad de consultar a que distancia está la unidad más cercana de la cooperativa con respecto a la parada en la que se encuentra en ese momento el pasajero	1	Escoger al menos una cooperativa	El sistema muestra en el mapa un marcador con un color que identifica a la cooperativa
				2	Escoger varias cooperativas	El sistema muestra en el mapa marcadores de distintos colores para cada cooperativa
HUPAS4	Como un pasajero	Necesito consultar las rutas que realiza cada cooperativa	Con la finalidad de verificar que cooperativa me conviene tomar	1	Escoger al menos una cooperativa	El sistema muestra en el mapa la ruta con un color que identifica a la cooperativa
				2	Escoger varias cooperativas	El sistema muestra en el mapa las rutas con distintos colores para cada cooperativa
HUPAS5	Como un pasajero	Necesito recibir una advertencia en la App	Con la finalidad de saber que el bus se está acercando	1	Alarma para advertencia	El sistema advertirá la aproximación de la unidad con una antelación de 10 minutos.
HUPAS6	Como un pasajero	Necesito enviar una sugerencia	Con la finalidad de enviar sugerencias u observaciones	1	Envío de mensaje	El sistema recibirá quejas y sugerencias
HUCON 7	Como un controlador	Necesito un login	Con la finalidad de ingresar a la App	1	Login	El usuario ingresa al sistema y verá sus opciones dependiendo del tipo de usuario que sea
HUCON 8	Como un controlador	Necesito enviar mi ubicación	Con la finalidad de mostrar en el mapa del pasajero mi ubicación en tiempo real	1	Envío de coordenadas	El sistema muestra en el mapa un marcador con un color que identifica a la cooperativa
HUCON 9	Como un controlador	Necesito informar del estado del bus	Con la finalidad de informar sobre problemas con la unidad	1	Problemas con la unidad	El sistema informará de las novedades al usuario y al administrador
				2	Modificación de ruta	El sistema informará de las novedades al usuario y al administrador
HUADM 10	Como un administrador	Necesito saber las rutas que siguió dicha unidad	Con la finalidad de dar seguimiento a las unidades	1	Consulta de rutas	El sistema mostrará en un mapa la ruta hecha por la unidad
HUADM 11	Como un administrador	Necesito recibir las sugerencias	Con la finalidad de mejorar la atención a los pasajeros	1	Consultar sugerencias	El sistema Mostrará un listado de sugerencias enviadas
HUADM 12	Como un administrador	Necesito registrar datos	Con la finalidad de registrar nuevos controladores	1	Registrar controladores	El sistema registrará a los nuevos controladores que emitirán su ubicación

Nota: Detalle de las historias de usuario donde se exponen los requerimientos necesarios.

Las historias de usuario pueden ir cambiando con el paso del tiempo en el desarrollo del producto, pero hay que tener en cuenta que solo el Product Owner puede solicitar estos cambios.

2.3.3 Product backlog

La Lista de Producto es una lista ordenada de todo lo que podría ser necesario en el producto y es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto (Product Owner) es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación. (Schwaber & Sutherland, 2016)

Tabla 8. Producto backlog

ID Historia	Historia de usuario	Tiempo estimado (horas)	Dependencia
HUPAS1	Módulo Login de ingreso para Pasajero	20	-
HUPAS2	Módulo Inicio de App	14	HUPAS1
HUPAS3	Módulo Escoger cooperativas para consulta	20	HUPAS1-HUCON8
HUPAS4	Módulo Consultar rutas de las cooperativas	18	HUPAS1 - HUPAS3
HUPAS5	Módulo Recibir advertencia de cercanía de bus	50	HUPAS1 - HUPAS3
HUPAS6	Módulo Enviar sugerencias de servicio	12	HUPAS1
HUCON7	Módulo Login de ingreso para Controlador	10	-
HUCON8	Módulo Enviar ubicación del bus	24	HUCON7
HUCON9	Módulo Informar sobre estado del bus	18	HUCON7
HUADM10	Módulo Consultar rutas realizadas por el bus	18	-
HUADM11	Módulo Recibir sugerencias de los pasajeros	14	-
HUADM12	Módulo Registrar controladores	8	-

Nota: Detalle del product backlog en base a las historias de usuario

El product backlog está formado por la lista de funcionalidades de la historia de usuario, las mismas que deben ser priorizadas y pre estimadas.

2.3.4 Sprint backlog

“La pila del sprint (sprint Backlog) es la lista que descompone las funcionalidades de la pila del producto (historias de usuario) en las tareas necesarias para construir un incremento: una parte completa y operativa del producto” (Palacio, 2015, pág. 26).

Tabla 9. Sprint backlog

Tarea ID	Historia ID	Tareas	Estimación (horas)	Prioridad	Sprint
1	HUPAS1 HUCON7	Modelo Conceptual base MySQL	2	Alta	Sprint 1
2	HUPAS1 HUCON7	Modelo lógico base MySQL	2	Alta	Sprint 1
3	HUPAS1 HUCON7	Generar Script base de datos	2	Alta	Sprint 1
4	TODAS	Diseño de la aplicación pasajero	2	Media	Sprint 2
5	HUPAS1 HUCON7	Implementación del lógin	2	Alta	Sprint 2
6	HUPAS1 HUCON7	Conexión del lógin por web services	4	Alta	Sprint 2
7	HUPAS1 HUCON7	Validación del lógin con los datos en la nube con fecha y hora del último ingreso	4	Alta	Sprint 2
8	HUPAS1 HUCON7	Gestionar los perfiles distintos	4	Alta	Sprint 2
9	HUPAS1	Implementar formulario de registro de nuevos usuarios	4	Media	Sprint 3
10	HUPAS1	Envío de datos de los nuevos usuarios por web services	4	Media	Sprint 3
11	HUPAS2	Validar mensajes de existencia de unidades en horarios laborales y no laborales	14	Baja	Sprint 13
12	HUCON8	Diseño de la aplicación controlador	6	Alta	Sprint 4
13	HUCON8	Implementación de la aplicación del controlador	6	Alta	Sprint 4
14	HUCON8	Envío de coordenadas por web services	6	Alta	Sprint 4
15	HUCON8	Envío de coordenadas cada segundo por web services	6	Alta	Sprint 4
16	HUPAS3	Mapa para la aplicación	4	Alta	Sprint 5
17	HUPAS3	Consulta de coordenadas de ubicación del bus	4	Alta	Sprint 5
18	HUPAS3	Marcador para mostrar ubicación del bus	4	Alta	Sprint 5
19	HUPAS3	Actualización cada medio segundo de la ubicación del bus	4	Alta	Sprint 5
20	HUPAS3	Mostrar en el mapa uno o varios marcadores de distinto color	4	Alta	Sprint 5
21	HUPAS4	Tomar coordenadas y dibujar una ruta en el mapa	6	Alta	Sprint 7
22	HUPAS4	Select con los datos de las cooperativas existentes	6	Alta	Sprint 7
23	HUPAS4	Mostrar en el mapa una o varias rutas de distinto color	6	Alta	Sprint 7
24	HUPAS5	Dibujar posibles paradas	50	Alta	Sprint 6
25	HUPAS5	Calcular distancias posibles		Alta	Sprint 6

26	HUPAS5	Alarma de prueba desde una parada hasta una unidad		Alta	Sprint 6
27	HUPAS5	Escoger una parada		Alta	Sprint 6
28	HUPAS5	Programar alarma		Alta	Sprint 6
29	HUPAS6	Implementar formulario de envío de mensaje para sugerencias	6	Media	Sprint 8
30	HUPAS6	Enviar datos a base de datos mediante web services	6	Media	Sprint 8
31	HUCON9	Implementar formulario de envío de mensaje de estado de la unidad	6	Media	Sprint 9
32	HUCON9	Enviar datos a base de datos mediante web services	6	Media	Sprint 9
33	HUCON9	Mensaje de aviso en aplicación de pasajero	6	Media	Sprint 9
34	HUCON9	Mensaje de aviso en la página web del administrador	6	Media	Sprint 9
35	HUADM10 HUADM11	Implementar aplicación web para administración	10	Alta	Sprint 10
36	HUADM10	Portal de recepción de mensajes	4	Alta	Sprint 10
37	HUADM11	Consultar sugerencias por fecha	6	Alta	Sprint 12
38	HUADM12	Registrar nuevos controladores en la base de datos	8	Alta	Sprint 12

Nota: Detalle del sprint backlog.

2.4 Casos de Uso

Para el desarrollo del proyecto, es necesario mejorar el panorama descrito en la lista de requerimientos, para esto vamos a recurrir a los casos de uso que, mediante los gráficos respectivos nos ayuda a comprender de que manera se maneja el sistema y la secuencia de eventos con que opera cada uno de los usuarios que intervienen en el sistema.

Adicionalmente se ve la necesidad de aumentar un actor como es el administrador, quien trabaja con un sitio web sencillo, para realizar la administración de controladores y unidades. La razón de esta actividad adicional se basa en que es pertinente que se involucre un actor para la administración ya que el sistema solo cuenta con registro de pasajeros, pero no de controladores, esta actividad debe ser realizada por un sitio web para su respectivo control.

Caso de uso administrador

Figura 8. Caso de Uso sobre el comportamiento del administrador.
Elaborado por: Ruales Daniel (2017).

Tabla 10 Descripción caso de Uso administrador.

Actores:	Administrador
Objetivo:	Manejar el sistema desde la perspectiva del administrador

Descripción:	• Administrar controladores
	  • Crear controlador
	  • Modificar controlador
	  • Eliminar controlador
	• Administrar unidades
	  • Crear unidades
	  • Modificar unidades
	  • Eliminar unidades
	• Administrar mensajes
	  • Leer mensaje
	  • Eliminar mensaje
	• Iniciar sesión
	  • Ingresar Usuario
	  • Ingresar contraseña

Nota: Descripción del caso de uso desde el perfil del administrador descrito en la figura 8.

Caso de uso pasajero

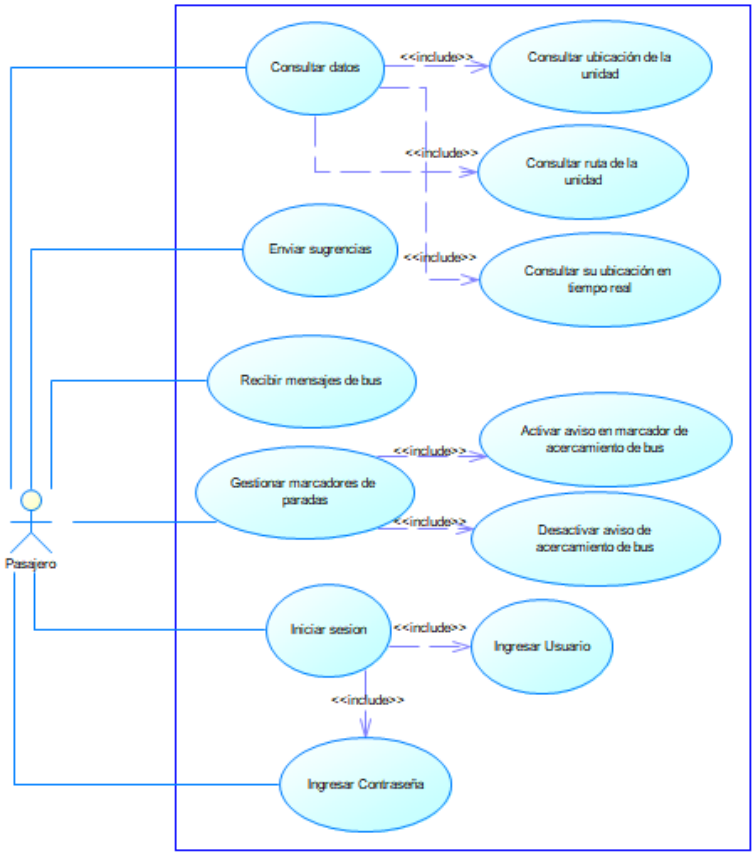

Figura 9. Caso de Uso sobre el comportamiento del pasajero
Elaborado por. Ruales Daniel (2017)

Tabla 11. Descripción caso de Uso pasajero

Actores:	Pasajero
Objetivo:	Manejar el sistema desde la perspectiva del pasajero

Descripción:	• Consultar datos
	  • Consultar ubicación de unidades en tiempo real
	  • Consultar rutas de las unidades
	• Recibir mensajes de la unidad
	• Gestionar marcadores de parada
	  • Activar notificación de cercanía de la unidad
	  • Activar notificación de cercanía de la unidad
	• Iniciar sesión
	  • Ingresar Usuario
	  • Ingresar contraseña

Nota: Descripción del caso de uso desde el perfil del pasajero descrito en la figura 9.

Caso de uso controlador

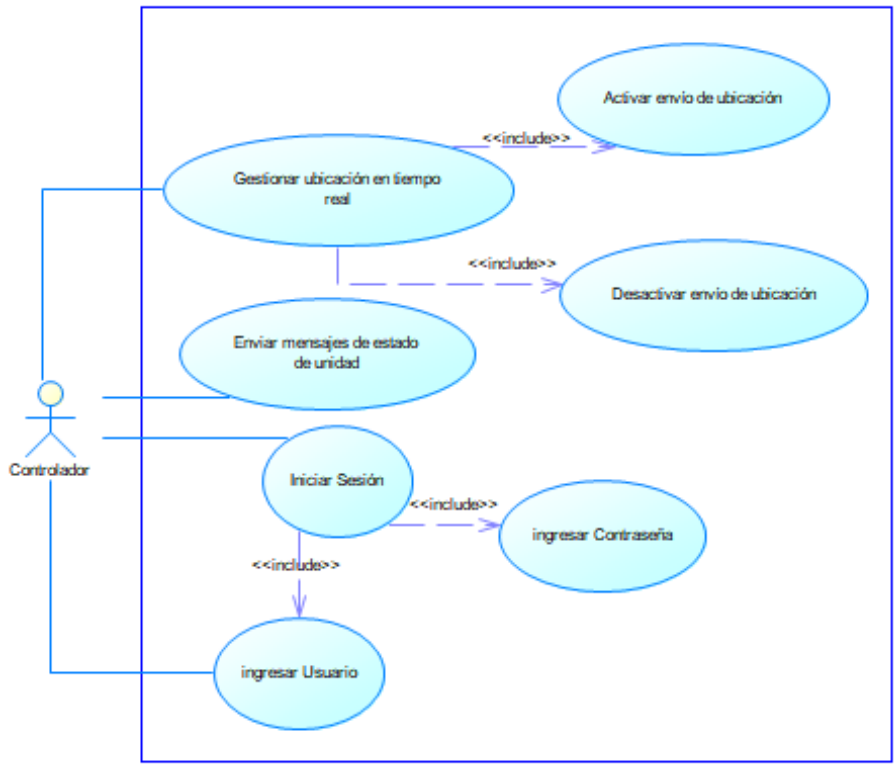

Figura 10. Caso de Uso sobre el comportamiento del controlador
Elaborado por: Ruales Daniel (2017).

Tabla 12.Descripción caso de Uso controlador

Actores:	Controlador
Objetivo:	Manejar el sistema desde la perspectiva del controlador

Descripción:	• Gestionar ubicación en tiempo real
	  • Activar envío de ubicación
	  • Desactivar envío de ubicación
	• Enviar mensajes de estado de la unidad en caso de que haya algún inconveniente por el cual la unidad deba salir de servicio.
	• Gestionar marcadores de parada
	  • Activar notificación de cercanía de la unidad
	  • Desactivar notificación de cercanía de la unidad
	• Iniciar sesión
	  • Ingresar Usuario
	  • Ingresar contraseña

Nota: Descripción del caso de uso desde el perfil del controlador descrito en la figura 10.

2.5 Diagrama de base de datos

Durante el proceso se usó una base de datos MySQL que ha sido diseñada en base a los requerimientos y posee información acerca de las rutas, direcciones, vehículos, conductores y dispositivos, relacionada de forma tal que nos ayude a la obtención de los datos requeridos, enmarcándonos en un modelo de Entidad-Relación. La figura 11 representa el esquema de la base de datos.

Diagrama de físco de la base de datos

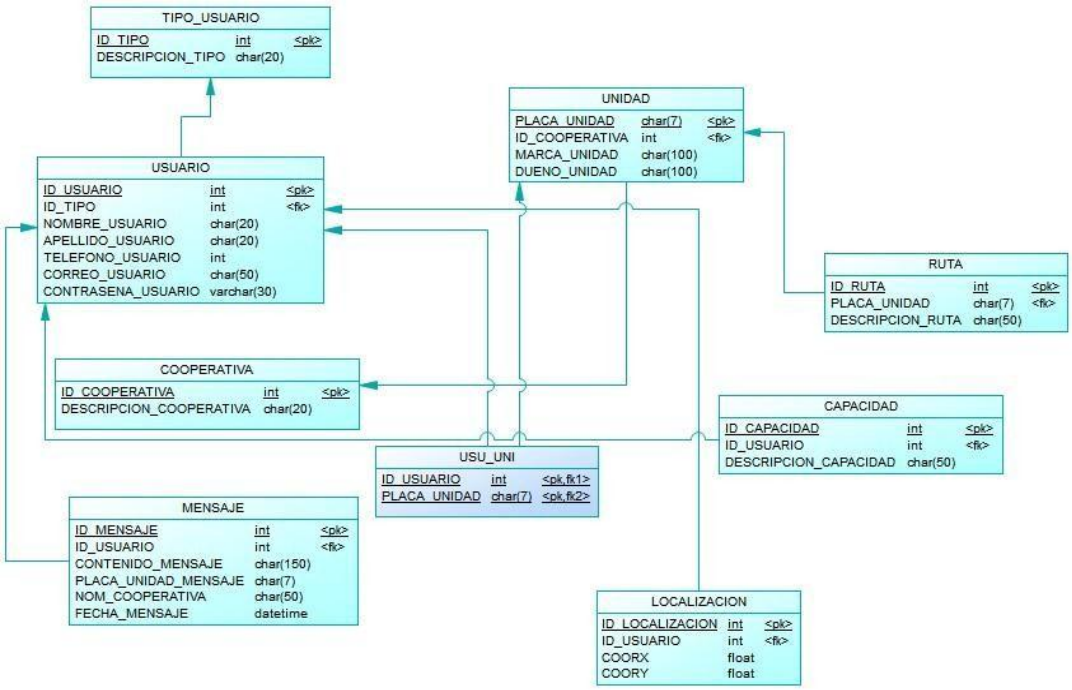

Figura 11.Diagrama físico de base de datos.
Elaborado por: Ruales Daniel (2017).

2.5.1 Diccionario de datos

En la tabla 13, se muestra el diccionario de datos de la base propuesta en la figura 11, a continuación, se detalla cada una de las tablas y sus distintos atributos.

Tabla 13. Detalle del diccionario de datos de la base

TIPO_USUARIO		
Campo	Tipo	Descripción
ID_TIPO	int	Identificador de la tabla TIPO_USUARIO
DESCRIPCION_TIPO	Char(20)	Descripción de cada tipo de usuario

USUARIO		
Campo	Tipo	Descripción
ID_USUARIO	Int	Identificador de la tabla ID_USUARIO
ID_TIPO	Int	Clave foránea de la relación con la tabla TIPO_USUARIO
NOMBRE_USUARIO	Char(20)	Nombre del usuario
APELLIDO_USUARIO	Char(20)	Apellido del usuario
TELEFONO_USUARIO	Int	Teléfono del usuario
CORREO_USUARIO	Char(50)	Correo del usuario
CONTRASENA_USUARIO	Varchar(30)	Contraseña del usuario
MENSAJE		
Campo	Tipo	Descripción
ID_MENSAJE	Int	Identificador de la tabla ID_MENSAJE
ID_USUARIO	Int	Clave foránea de la relación con la tabla USUARIO
CONTENIDO_MENSAJE	Char(150)	Contenido del mensaje
PLACA_UNIDAD_MENSAJE	Char(7)	Placa de la unidad
NOM_COOPERATIVA	Char(100)	Nombre de la cooperativa
FECHA_MENSAJE	datetime	Fecha de envío de mensaje
UNIDAD		
Campo	Tipo	Descripción
PLACA_UNIDAD	Char(7)	Identificador de la tabla PLACA_UNIDAD
ID_COOPERATIVA	Int	Clave foránea de la relación con la tabla COOPERATIVA
MARCA_UNIDAD	Char(100)	Marca de la unidad
DUENO_UNIDAD	Char(100)	Dueño de la unidad
LOCALIZACION		
Campo	Tipo	Descripción
ID_LOCALIZACION	Int	Identificador de la tabla LOCALIZACION
ID_USUARIO	Int	Clave foránea de la relación con la tabla USUARIO
COORX	Float	Latitud
COORY	float	Longitud
CAPACIDAD		
Campo	Tipo	Descripción
ID_CAPACIDAD	Int	Identificador de la tabla ID_CAPACIDAD
ID_USUARIO	Int	Clave foránea de la relación con la tabla USUARIO
DESCRIPCION_CAPACIDAD	Char(50)	Descripción de la capacidad
RUTA		
Campo	Tipo	Descripción
ID_RUTA	Int	Identificador de la tabla ID_RUTA
PLACA_UNIDAD	Char(7)	Clave foránea de la relación con la tabla PLACA_UNIDAD
DESCRIPCION_RUTA	Char(200)	Descripción de la ruta

COOPERATIVA		
Campo	Tipo	Descripción
ID_COOPERATIVA	Int	Identificador de la tabla ID_COOPERATIVA
DESCRIPCION_COOPERATIVA	Char(200)	Descripción de la cooperativa
USU_UNI		
Campo	Tipo	Descripción
ID_USUARIO	Int	Clave foránea de la relación con la tabla ID_USUARIO
PLACA_UNIDAD	Char(7)	Clave foránea de la relación con la tabla PLACA_UNIDAD

Nota: Detalle de los campos que componen la base de datos.

CAPÍTULO 3

3 Construcción y pruebas

3.1 Construcción

Para la construcción de la aplicación móvil se presenta una serie de requisitos, diagramas, así como también se muestra el código más relevante con su respectiva explicación para su mejor entendimiento.

3.2 Arquitectura

La arquitectura escogida es Modelo Vista Controlador que nos permite estructurar de mejor manera el diseño del software, ya que, al separar nuestro código en estas tres capas, podremos manejar cada capa independientemente sin alterar el diseño o la lógica de negocio.

Diagrama de componentes del proyecto

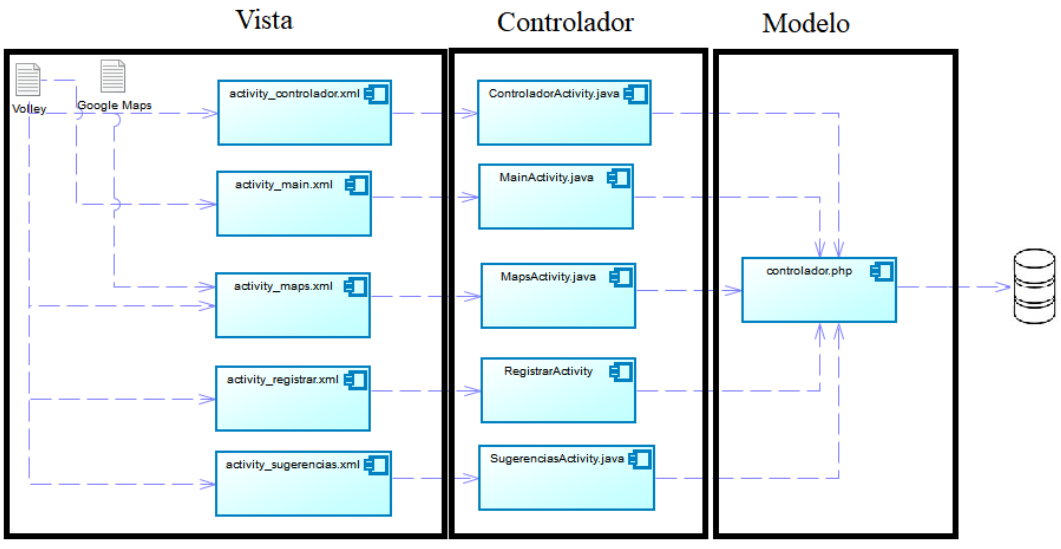

Figura 12. Diagrama de componentes MVC.
Elaborado por: Ruales Daniel (2017).

Android Studio facilita la estructura del proyecto con el manejo de la arquitectura MVC ya que divide la vista en este caso los layouts, y los controladores conocidos como activities como se muestra en la figura 12,

3.2.1 Modelo

En esta capa se desarrolla el trabajo de transacciones con la base de datos, contiene todas las funciones necesarias para obtener y enviar información. Este controlador tiene la capacidad de devolver respuestas en formato JSON, lo que facilita el manejo de datos para procesar en los controladores o mostrar en las vistas.

A continuación, se detallan todas las funciones y sentencias que tiene el modelo controlador.php.

Para cada transacción que hagamos, cada controlador cuenta con la variable $opcion como se muestra en la figura 13, que permite diferenciar la función que se usa en el manejo de los datos, esta variable es enviada al modelo mediante el método GET.

Código de opciones de las funciones existentes en controlador.php

```
<?php
$opcion = $_REQUEST['opcion'];
switch ($opcion) {
    case "1":
        return get_datos();
        break;
    case "2":
        $ide = $_GET['op1'];
       borrar_datos($ide);
        break;
    case "3":
        $ide = $_GET['op1'];
        return consultaedita_datos($ide);
        break;
    case "4":
        $ide = $_GET['op1'];
        $name = $_GET['nam'];
        $email = $_GET['ema'];
        $mobile = $_GET['mob'];
        update_datos($name,$email,$mobile,$ide);
        break;
    case "5":
        $placa = $_GET['placa'];
        $coop = $_GET['coop'];
        $marca = $_GET['marca'];
        $dueno = $_GET['dueno'];
        insertar_datosUnidad($placa,$coop,$marca,$dueno);
        break;
    case "6":
        $cadena = $_GET['cedula'];
        $cedula = strtoupper($cadena);
        $nombreCon = $_GET['nombreCon'];
        $apellido = $_GET['apellido'];
        $telefono = $_GET['telefono'];
        $correo = $_GET['correo'];
        $contrasena = $_GET['contrasena'];
        $unidad = $_GET['unidad'];
        insertar_datosControlador($cedula,$nombreCon,$apell
        insertar_datosRelacion($cedula,$unidad);
        break;
```

Figura 13. Código de opción para elección de las distintas funciones existentes en controlador.php. Elaborado por Ruales Daniel (2017).

Mediante la variable $opcion manejamos el llamado a cada función y el envío de sus respectivas variables según el caso deseado.

Tabla 14. Funciones existentes en controlador.php

Funciones	Consulta	Descripción
function get_datosSelectUnidades	$sql = 'SELECT * FROM cooperativa';	Consulta los datos de toda la tabla cooperativa.
function get_datosMensaje()	$sql = 'SELECT * FROM mensaje ORDER BY ID_MENSAJE DESC';	Consulta los datos de la tabla mensaje y los muestra de manera descendente.
function get_datos()	$sql = 'SELECT cooperativa.ID_COOPERATIVA, cooperativa.DESCRIPCION_COOPERATIVA, unidad.ID_COOPERATIVA, unidad.PLACA_UNIDAD, unidad.MARCA_UNIDAD, unidad.DUENO_UNIDAD FROM unidad, cooperativa WHERE unidad.ID_COOPERATIVA = cooperativa.ID_COOPERATIVA ORDER BY unidad.PLACA_UNIDAD';	Consulta compuesta por datos de las tablas unidad y cooperativa para mostrar los datos de las unidades existentes.
get_datosControlador()	$sql = 'SELECT usuario.ID_USUARIO, usu_uni.PLACA_UNIDAD, usuario.NOMBRE_USUARIO,usuario.APELLIDO_USUARIO, usuario.TELEFONO_USUARIO FROM usuario, usu_uni WHERE usuario.ID_USUARIO = usu_uni.ID_USUARIO ORDER BY usuario.APELLIDO_USUARIO';	Consulta compuesta por datos de las tablas usuario y usu_uni para mostrar los datos de los controladores existentes.
insertar_datosUnidad($placa,$copo,$marca,$dueno)	$sql = " INSERT INTO unidad (`PLACA_UNIDAD`, `ID_COOPERATIVA`, `MARCA_UNIDAD`, `DUENO_UNIDAD`) VALUES(?,?,?,?) ";	Inserta los datos de las unidades en la tabla unidad, recibiendo como parámetros la placa, cooperativa, marca y dueño.
function insertar_datosControlador ($cedula,$nombreCon,$apellido,$telefono,$correo, $contrasena)	$sql = " INSERT INTO usuario (`ID_USUARIO`, `ID_TIPO`, `NOMBRE_USUARIO`, `APELLIDO_USUARIO`, `TELEFONO_USUARIO`, `CORREO_USUARIO`, `CONTRASENA_USUARIO`) VALUES(?,'2',?,?,?,?,?) ";	Inserta los datos de los controladores en la tabla usuario, recibiendo como parámetros la cedula, nombre, apellido, teléfono, correo y contraseña. Esta función se utiliza también para el ingreso de nuevos usuarios desde la aplicación móvil.
function get_datosLogin($usuario)	$sql = "SELECT ID_USUARIO, CONTRASENA_USUARIO, ID_TIPO, NOMBRE_USUARIO FROM `USUARIO` WHERE CORREO_USUARIO=?";	Consulta los datos ingresados desde el login de la aplicación móvil.
function insertar_datosUbicacion($idus,$latitud,$longitud,$dato)	$sql = " INSERT INTO localizacion (`ID_LOCALIZACION`, `ID_USUARIO`, `LATITUD`, `LONGITUD`,`FECHA_LOCALIZACION`) VALUES(NULL,?,?,?,?) ";	Inserta los datos de las coordenadas, función que será llamada como petición cada dos segundos.
function insertar_Mensaje($cedula,$mensaje,$placa,$cooperativa,$dato)	$sql = " INSERT INTO mensaje (`ID_MENSAJE`, `ID_USUARIO`, `CONTENIDO_MENSAJE`, `PLACA_UNIDAD_MENSAJE`,`NOM_COOPERATIVA`,`FECHA_MENSAJE`) VALUES(NULL,?,?,?,?,?) ";	Inserta los mensajes enviados desde el perfil de pasajero o desde el perfil de controlador de la aplicación móvil.

Nota: Tabla elaborada para mostrar las distintas funciones que conforman a controlador.php.
Elaborado por: Ruales Daniel (2017).

En cuanto a funciones se refiere, el resultado siempre será el envío de un JSON con el formato que se muestra en la figura 14.

Ejemplo de JSON que se envía

Figura 14. Ejemplo de JSON enviado por una de las funciones de controlador.php.
Elaborado por: Ruales Daniel (2017).

3.2.2 Vista

La facilidad que Android Studio nos brinda es que la vista está separada del modelo de negocio, con lo cual se puede diseñar sin afectar a la programación, estas vistas se muestran en un formato XML como se muestra en la figura 15.

Ejemplo código XML

```
LinearLayout
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FDCA2E"
    android:orientation="vertical"
    tools:context="innovarer.app.ruales.daniel.com.proyectotitulacion.ControladorActivity">


    <LinearLayout
        android:layout_width="368dp"
        android:layout_height="166dp"
        android:orientation="vertical"
        tools:layout_editor_absoluteX="8dp"
        tools:layout_editor_absoluteY="5dp">

        <TextView
            android:id="@+id/textBienvenida"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:paddingBottom="5dp"
            android:paddingTop="5dp"
            android:text="Bienvenido"
            android:textColor="#000000"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/textNombreusuario"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
```

Figura 15. Código XML de la Vista activity_controlador.xml.
Elaborador por Ruales Daniel (2017).

Cada layout cuenta con distintas propiedades y características propias del sistema que brindan la facilidad del desarrollo en cuanto a diseño. Además, no solo se cuenta con una división en cuanto a layouts sino también en cuanto a sus atributos, ya que, si existen atributos generales para su reutilización, se puede crear un archivo XML con cada uno de ellos y utilizarlos en cualquier layout.

A continuación, veremos cada una de las vistas que conforma el proyecto.

Layout activity_main.xml para login

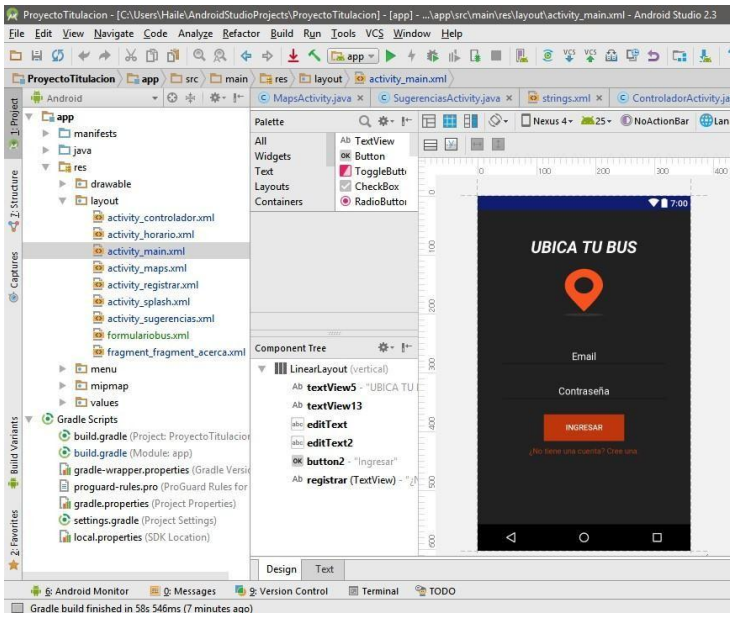

Figura 16. Layout activity_main.xml.
Elaborado por: Ruales Daniel (2017).

En la figura 16 se muestra la vista del layout activity_main.xml que se utiliza para el ingreso del controlador o el pasajero.

Layout activity_controlador.xml principal del controlador

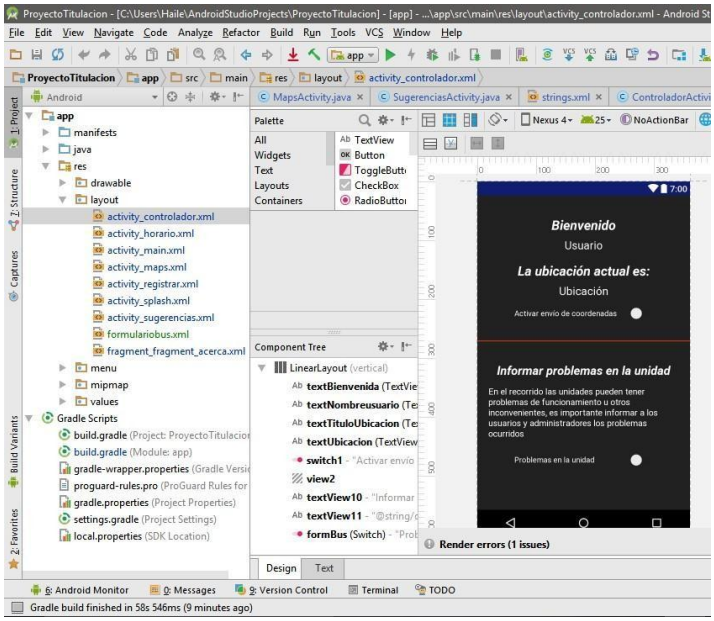

Figura 17.Layout activity_controlador.xml.
Elaborado por: Ruales Daniel (2017).

Luego de validar los datos en la vista layout activity_main.xml cada actor tiene una vista especifica. El controlador ingresa al Layout activity_controlador.xml que se muestra en la figura 17, y el pasajero se dirige hacia la vista Layout activity_maps.xml que se muestra en la figura 18.

Layout activity_maps.xml principal del usuario

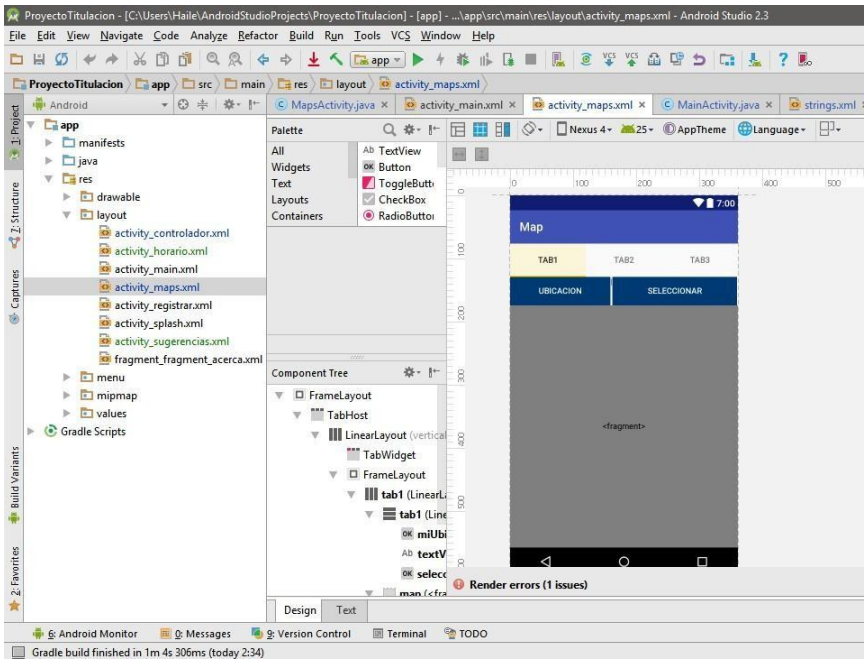

Figura 18. Layout activity_maps.xml.
Elaborado por: Ruales Daniel (2017).

En la pantalla de diseño de layout no se aprecia el mapa debido a que este debe ponerse como fragment en la vista, lo cual no permite apreciar una imagen del mapa en el que se trabaja. Hay que tener en cuenta que en el emulador se debe tener instalado Google play services que ayuda en la visualización de los mapas, para este proyecto se utiliza Genymotion que cuenta con emuladores de una gran variedad de teléfonos móviles.

La aplicación móvil cuenta con un registro para nuevos pasajeros, esta vista se muestra en la figura 19.

Layout activity_registrar.xml registrar usuarios

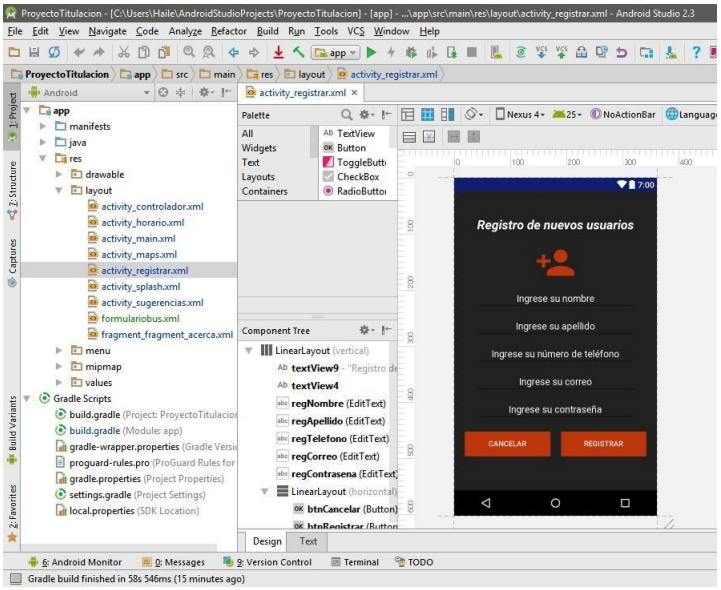

Figura 19. Layout activity_registrar.xml
Elaborado por: Ruales Daniel (2017)

La activity splash en la figura 20 es un diseño que se incorpora como pantalla de bienvenida en las aplicaciones, esta permanece un determinado tiempo y luego avanza hacia la pantalla principal

Layout activity_spalsh.xml

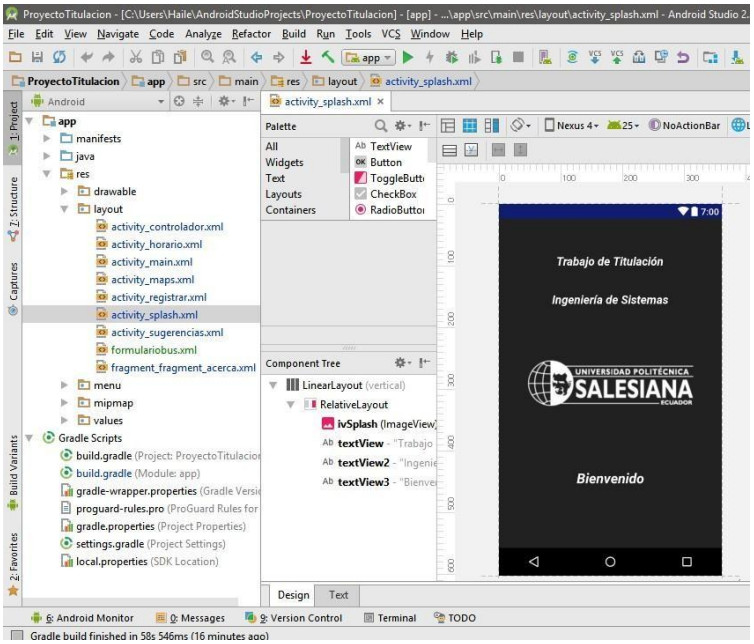

Figura 20. Layout activity_spalsh.xml.
Elaborado por: Ruales Daniel (2017).

Layout activity_sugerencia.xml sugerencia de usuario

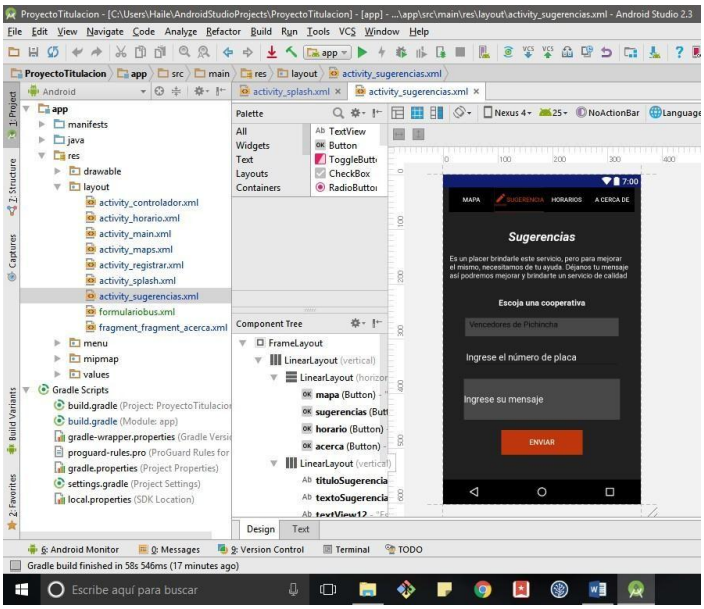

Figura 21.Layout activity_sugerencia.xml.
Elaborado por: Ruales Daniel (2017).

Uno de los objetivos fundamentales es promover una interacción entre el los pasajeros y las cooperativas de transporte, para esto se crea una vista de sugerencias como se muestra en la figura 21 que envía mensajes a la plataforma web del administrador.

3.2.3 Controlador

En la capa controlador se desarrolla todo lo necesario para responder y satisfacer las peticiones que se realizan por parte del usuario. Esta capa sirve como comunicación entre la capa Vista y capa Modelo por esa razón, la capa Controlador no muestra ningún dato y tampoco extrae datos de la base, pero si gestiona el flujo de información entre ellos.

3.2.3.1 Código relevante

Se ha programado un número determinado de controladores, a continuación, se presenta el código más relevante para el desarrollo de la aplicación móvil.

3.2.3.1.1 Controlador MainActivity.java

Controlador MainActivity.java

```
    btnLogin.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v){
            String servidor = "http://192.168.0.11:8080/proyectotitulacion/php/control...";
        ConsultaPass(servidor+"?user="+editEmail.getText().toString()+"&opcion=9");
         }
    });
}

private void ConsultaPass(String URL) {
    RequestQueue queue = Volley.newRequestQueue(this);
    StringRequest stringRequest =  new StringRequest(Request.Method.GET, URL, (response) → {

            try {
                ja = new JSONArray(response);
                JSONObject objeto = ja.getJSONObject(0);
                String idUser = objeto.getString("ID_USUARIO");
                String contra = objeto.getString("CONTRASENA_USUARIO");
                int tipoUsuario = objeto.getInt("ID_TIPO");
                String nombreUsuario = objeto.getString("NOMBRE_USUARIO");
                if(contra.equals(editContrasena.getText().toString())){

                    Toast.makeText(getApplicationContext(),"Bienvenido",Toast.LENGTH_SHORT).show();
                    if (tipoUsuario == 2) {
                        Intent intent = new Intent(MainActivity.this, MapsActivity.class);
                        startActivity(intent);
                    }
                    else {
                        Intent intent = new Intent(MainActivity.this, ControladorActivity.class);
                        intent.putExtra("nombreUsuario", nombreUsuario);
                        intent.putExtra("idUser",idUser);
```

Figura 22.Controlador MainActivity.java.
Elaborado por: Daniel Ruales (2017).

En el controlador MainActivity.java de la figura 22, se encuentra código para realizar el login de la aplicación, que mediante la clase setOnClickListener() envía la petición por el método GET hacia el modelo. La librería volley facilita la recepción de la respuesta JSON con el método JSONArray que recibe una cadena y mediante el uso JSONbject podemos transformar este JSON a una lista de objetos que resulta fácil de manejar en java, de esta manera podemos extraer los datos y comparar si existe o no el usuario ingresado.

Por medio de la clase setLocation() podemos obtener la ubicación exacta gracias a los servicios de Google, ya que geocoder(), puede acceder a estos servicios a través de peticiones HTTP, este servicio nos proporciona la latitud y longitud

3.2.3.1.2 Controlador ControladorActivity.java

Código ControladorActivity.java para envió de ubicación

```
//Inseertar coordenadas en la base
ConnectivityManager connMgr = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

if (networkInfo != null && networkInfo.isConnected()) {
    String latitude = String.valueOf(loc.getLatitude());
    String logintude = String.valueOf(loc.getLongitude());
    String operacion = "c";
    cedula=(getIntent().getExtras().getString("idUser"));

    if (latitude.isEmpty() || logintude.isEmpty() || operacion.isEmpty()) {
        Toast.makeText(getApplicationContext(), "Campos vacios", Toast.LENGTH_LONG).show();
    } else {
        url = "http://192.168.0.11:8080/envioCoordenadas/proceso.php";
        parametros = "operacion=" + operacion + "&clatitud=" + latitude + "&clongitud=" + logintude + "&id=" + cedula;
        new Solicitud().execute(url);
    }

    new Solicitud().execute(url);
} else {
    Toast.makeText(getApplicationContext(), "La conexión fue detectada", Toast.LENGTH_LONG).show();
}
}
```

Figura 23.Código ControladorActivity.java para envió de ubicación.
Elaborado por: Daniel Ruales (2017).

Pero antes debemos realizar una comprobación de red, esta información nos da ConnectivityManager. Si se obtiene el mensaje de conexión, se podrá realizar él envió de la petición por medio de GET, utilizando los parámetros necesarios que se mencionan en la figura 23. Este envió de petición se realiza cada 2 segundos teniendo en cuenta que debemos mostrar nuestra posición en tiempo real evitando perder la ubicación por espacios largos de tiempo. Para el envío de cada cierto tiempo de peticiones se ha utilizado tareas asíncronas (AsyncTask) que funcionan como hilos y se lanzaran cada cierto tiempo para el que fueron programados.

Es requisito que, si la unidad tiene algún problema en el trayecto de su ruta, informe sobre su estado, para esto volvemos a utilizar la librería Volley, esta cuenta con RequestQueue que mejora la velocidad de consultas ya que Volley tiene la capacidad de guardar datos en cache, esto soluciona problemas en cuanto a consultas al servidor esta estructura de código que se puede visualizar en la figura 24, pero presenta problemas en cuanto al número de peticiones http realizadas al mismo.

Código ControladorActivity.java para envió de mensajes.

```
private void crear() {
    dir = "http://192.168.0.11:8080/proyectotitulacion/php/control...";
    strq = new StringRequest(Request.Method.POST, dir,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    Log.d("rta_servidor", response);
                    Toast.makeText(ctx, response, Toast.LENGTH_SHORT).show();
                }
            },  new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.d("error_servidor", error.toString());
        }
    }) {
        @Override
        protected Map<String, String> getParams()  {
            Map<String, String> parametros = new HashMap<>();
            parametros.put("cooperativa", cooperativa);
            parametros.put("cedula", cedula);
            parametros.put("placa", "pbp3315");
            parametros.put("mensaje", mensaje.getText().toString());
            parametros.put("opcion", "11");

            return parametros;
        }
    };

    rqt.add(strq);
```

Figura 24. ControladorActivity.java para envió de mensajes.
Elaborado por: Daniel Ruales (2017).

Utilizaremos el método POST para el envío de los datos mediante HashMap que proporciona la estructura del url para realizar las peticiones. Este mismo principio se utiliza para los controladores RegistrarActivity.java y SugerenciasActivity.java ya que los dos envían un mismo tipo de formulario con diferentes parámetros.

3.2.3.1.3 Controlador MapsActivity.java

Como podemos ver en el proyecto, las peticiones son las mismas para varios controladores, esto gracias a que Volley simplifica mucho las consultas al controlador.php. El controlador MapsActivity.java muestra una estructura similar a las anteriores, ya que debe pedir la ubicación actual de la unidad.

En la figura 25 podemos observar la codificación para mostrar los marcadores en el mapa. Para obtener esta ubicación se ha creado un hilo que llama al controlador.php cada segundo y actualiza la posición por medio de la consulta que está programada para dar el último dato guardado en la tabla localización.

Hilo controlador MapsActivity.java

```
public void hilo(){
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

public void ejecutar(){
    time time = new time();
    time.execute();
    if (time instanceof time) {
        acum=acum+1;
        Toast.makeText(MapsActivity.this, "Consulta"+acum, Toast.LENGTH_SHORT).show();
        ConsultarCorredor("http://192.168.0.11:8080/envioCoordenadas/login.php?id=1&prueba=b&coor="+acum);
       // ConsultarVencedores("http://192.168.0.11:8080/envioCoordenadas/login.php?id=2&prueba=b&coor="+(acum+1000));
        ConsultarQuitumbe("http://192.168.0.11:8080/envioCoordenadas/login.php?id=3&prueba=a&coor=1");
    }
}

public class time extends AsyncTask<Void,Integer,Boolean>{
    @Override
    protected Boolean doInBackground(Void... params) {
        for (int i=1; i<=2;i++){
            hilo();
        }
        return true;
```

Figura 25. Hilo controlador MapsActivity.java.
Elaborado por: Ruales Daniel (2017).

A continuación, se crea la clase agregarMarcador(doublé lat, doublé lng), que tiene como parámetros de entrada los datos tomados en la consulta de la figura 26.

Con esta clase se crea los marcadores, que son elementos que provienen de la clase onMapReady, siendo esta una API de Google que ayuda en la presentación de cualquier lugar y tipo de mapa.

Seguimiento de la unidad

```
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageMa
        return;
    }
    mMap.getUiSettings().setZoomControlsEnabled(true);

}

private void agregarMarcador(double lat, double lng) {
    LatLng coordenadas = new LatLng(lat, lng);
    CameraUpdate miUbicacion = CameraUpdateFactory.newLatLngZoom(coordenadas, 16);
    if (marcador != null) marcador.remove();
    marcador = mMap.addMarker(new MarkerOptions().position(coordenadas).title("Mi posición actual")
            .icon(BitmapDescriptorFactory.fromResource(R.drawable.pactual)));
    mMap.animateCamera(miUbicacion);
}

private void actualizarUbicación(Location location) {
    if (location != null) {
        lat = location.getLatitude();
        lng = location.getLongitude();
        agregarMarcador(lat, lng);
    }
}
```

Figura 26. Seguimiento de la unidad en tiempo real.
Elaborado por: Ruales Daniel (2017).

Como parte final de la construcción se ha programado la clase sendNotofications() que se muestra en a figura 27, la cual tiene el trabajo de notificarnos cuando un bus está por llegar a la parada, esta notificación se da, cuando el bus se encuentre a una distancia de 1,5 km, distancia prudente poder tomar el bus en la parada más cercana

Notificación de aproximación de unidad

```
private void sendNotification(){

    Uri soundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    NotificationCompat.Builder builder = new NotificationCompat.Builder(MapsActivity.this);
    builder.setSmallIcon(R.drawable.notificacion)
            .setColor(Color.RED)
            .setContentTitle("BUS")
            .setContentText("Su bus esta muy cerca")
            .setSound(soundUri)
            .setPriority(NotificationCompat.PRIORITY_MAX)
            .setLights(Color.CYAN,1,0)
            .setOngoing(false)
            .setLights(Color.RED,1000,1000)
            .setVibrate(new long[]{250,250,250,250});

    Intent resultIntent = new Intent(this, MapsActivity.class);
    resultIntent.putExtra(NOTIFICATION_SERVICE,("Prueba"));
    TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
    stackBuilder.addParentStack(MapsActivity.class);
    stackBuilder.addNextIntent(resultIntent);
    PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);
    builder.setContentIntent(resultPendingIntent);

    NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    notificationManager.notify(0,builder.build());

}
```

Figura 27.Notificación de aproximación de unidad.
Elaborador por: Ruales Daniel (2017).

3.2.3.2 Aplicación web para el ingreso de rutas en la base de datos.

Opción de rutas

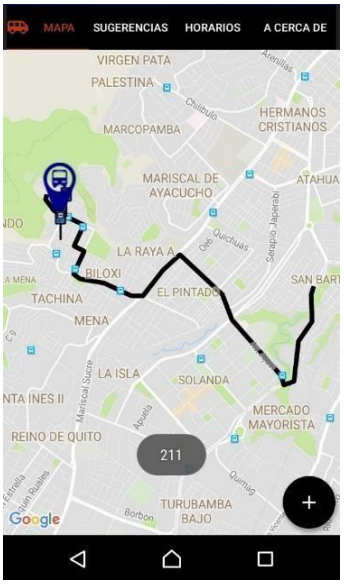

Figura 28. Opción de rutas que muestra la aplicación móvil
Elaborador por: Ruales Daniel (2017).

La aplicación móvil brinda información importante para el usuario, información necesaria como las rutas que realiza cada cooperativa como se muestra en la figura 28. Las rutas que se dibujan en esta sección utilizan la función polyline() de Google maps, esta toma como referencia dos puntos y traza una línea diseñada en el código, los puntos son tomados desde la base de datos con los cuales se puede trazar las líneas que se requieran para dibujar una ruta especifica.

Como requerimiento importante en el desarrollo de la aplicación se toman las coordenadas registradas en la base para elaborar las rutas, estas deben ser introducidas de la manera más sencilla posible, pensando en que las coordenadas deben ser ingresadas por un usuario y no por un profesional técnico que ingresa directamente los datos a la base, se desarrolló una aplicación web que mediante el uso de la función polyline() trazamos rutas y guardamos puntos de referencia con sus respectivas coordenadas como se muestra en la figura 29.

Aplicación web para ingreso de ruta

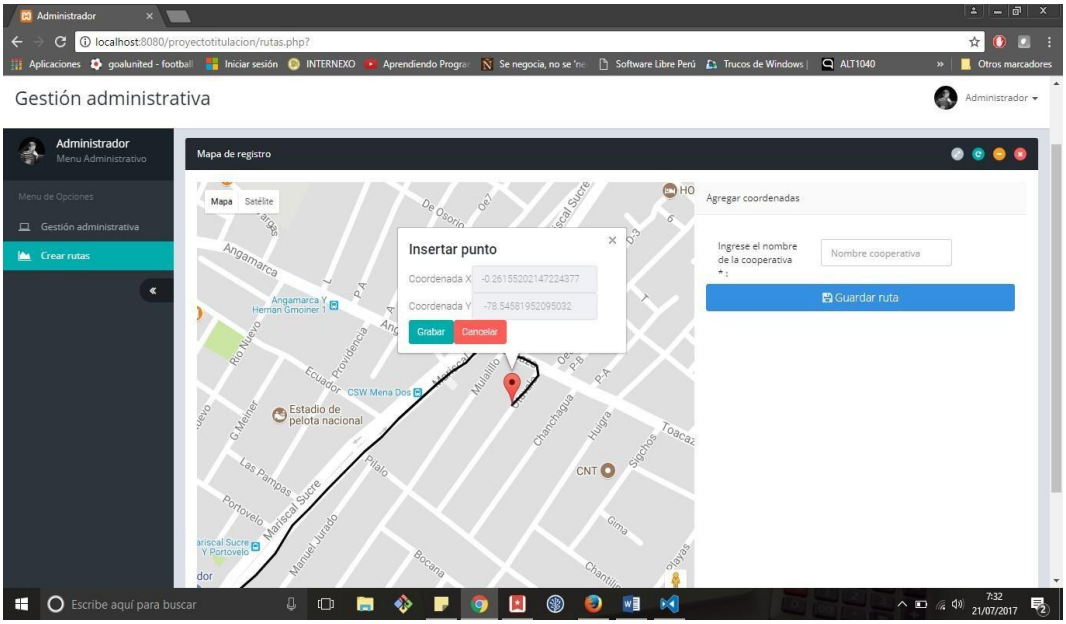

Figura 29. Aplicación web para ingreso de ruta.
Elaborador por: Ruales Daniel (2017).

Google maps cuenta con Javascript API para el diseño de mapas personalizados en aplicaciones web, Android, etc. Para el uso de esta API es necesario estar familiarizado con el lenguaje Javascript y HTML5 aspectos necesarios para una implementación adecuada.

Para obtener el resultado esperado que se muestra en la figura 29, se desarrolló distintas funciones. A continuación, se muestra el código relevante que se utiliza para el desarrollo de la aplicación web.

Código de la función initMap()

```
219
220     function initMap() {
221       var formulario = $("#formulario");
222       map = new google.maps.Map(document.getElementById('map'), {
223         center: {lat: -0.2604523271862326, lng: -78.55035781860352},
224         zoom: 15,
225       });
226
227       var destinations = new google.maps.MVCArray();
228       var polylineOptions = {path: destinations};
229       var polyline = new google.maps.Polyline(polylineOptions);
230       polyline.setMap(map);
231       google.maps.event.addListener(map,'click',function (e){
232           var currentPath = polyline.getPath();
233           currentPath.push(e.latLng);
234
235       });
236
237            google.maps.event.addListener(map, "click", function(event){
238       var coordenadas = event.latLng.toString();
239       coordenadas = coordenadas.replace("(","");
240       coordenadas = coordenadas.replace(")","");
241       var lista = coordenadas.split(",");
242       //Mostrar las coordenadas por separado
243       //alert("La coordenadas en X es " + lista[0]);
244       //alert("La coordenadas en Y es " + lista[1]);
245
```

Figura 30. Código de la función initMap() para la creación de mapas.
Elaborador por: Ruales Daniel (2017).

Para la creación del mapa se utiliza la función initMap() que toma el valor de la variable map para mantener el mapa en una sección de la página que tiene como identificador (id = “map”) dicho valor.

En la figura 30 podemos observar el código de trazado de rutas a partir de la línea 227 que corresponde al uso de google.maps.Polyline, este requiere del argumento polylineOptions que contiene el path de los distintos puntos que se utiliza en el trazado. En este caso es necesario utilizar como puntos google.maps.MVCArray(), que toma los puntos ubicados en el mapa en ese instante sin necesidad de recargar el mapa.

Las coordenadas se guardan en el MVCArray() una por una, al hacer clic en el mapa muestra un marcador, este tiene programado un infoWindow()que contiene el botón de guardar el cual almacena dicho punto en la base de datos.

Código para mostrar ventanas de información

```
            var infowindow = new google.maps.InfoWindow({
          content: contentString
        });

        var marcador = new google.maps.Marker({
            position:direccion,
            map:map,
            animation:google.maps.Animation.DROP,
            draggable:false
        });
        marcador.addListener('click', function() {
          infowindow.open(map, marcador);
           formulario.find("input[name='cx']").val(lista[0]);
        formulario.find("input[name='cy']").val(lista[1]);
        });


        marcadores_nuevos.push(marcador);
        google.maps.event.addListener(marcador, "click", function(){
        });
        quitar_marcadores(marcadores_nuevos);
        marcador.setMap(map);
      });
      }
```

Figura 31. Código para crear ventanas de información sobre marcadores.
Elaborador por: Ruales Daniel (2017).

La función Google.maps.InfoWindow() que se muestra en la figura 31, contiene los datos que son mostrados cuando se da clic sobre el marcador que deseamos guardar, en este caso contentString es una variable que diseña un formulario para el almacenamiento de las coordenadas en la base que se muestra en la figura 32.

Código para desarrollo de formulario

```
            var contentString = '<div id="content">'+
        '<h4>Insertar punto</h4>'+
        '<div id="bodyContent">'+
        '<form name="formulario" id="formulario" method="post" onsubmit="return enviar();">' +
        '<table>'+
        '<tr>'+
        '<td>Coordenada X</td>'+
        '<td><input type="text" class="form-control" value="'+lista[0]+'" disabled="disabled" id="cx" autocomple
        '</tr>'+
        '<td>Coordenada Y</td>'+
        '<td><input type="text" class="form-control" value="'+lista[1]+'" disabled="disabled" id="cy" autocomple
        '</tr>'+
        '<tr>'+
        '</table>'+
        '<td><button type="submit" class="btn btn-success btn-sm">Grabar</button></td>'+
        '<td><button type="button" class="btn btn-danger btn-sm">Cancelar</button></td>'+
        '</form>'+
        '</div>'+
        '</div>';
```

Figura 32. Código para desarrollo de formulario en la venta de información.
Elaborador por: Ruales Daniel (2017).

Para el uso de esta ventana de información es necesario lanzar la función infoWindow.open() en cada marcador, está función será llamada cada que se llame al evento marcador.addListener() que crea un solo marcador por cada clic realizado en el mapa.

Código para enviar datos de coordenadas

```
188     function enviar(){
189
190         var cx = document.getElementById('cx').value;
191         var cy = document.getElementById('cy').value;
192         var nombre_coop = document.getElementById('nombre_coop').value;
193         if(nombre_coop.length == 0){
194             alert("Escriba el nombre de la cooperativa");
195         }
196         else{
197         var datacoor = 'cx=' + cx + '&cy=' + cy + '&id=13' + '&nombre_coop='+ nombre_coop + '&opcion=13';
198         numConsultas = numConsultas + 1;
199
200         $.ajax({
201             url: 'php/controlador.php',
202             data: datacoor,
203             type: 'GET',
204             success: function (resp) {
205                 $("#bodyContent").html(resp);
206             }
207         });
208         return false;
209         }
210     }
```

Figura 33. Código para enviar datos de coordenadas
Elaborador por: Ruales Daniel (2017).

El botón de guardar en la ventana de información de los marcadores llama a la función enviar(), en esta función se almacenan las coordenadas y el nombre de la cooperativa tomadas de los campos de texto que serán enviados mediante url por método GET hacia el modelo para realizar la consulta planteada en php/controlador.php que se codifica en la figura 33. Para esto se utiliza Ajax que evita la recarga de la página cada vez que enviemos las coordenadas a la base.

3.3. Pruebas

Las pruebas de caja negra se realizan sin tomar en cuenta la estructura del código, se basa en probar el funcionamiento general de la aplicación móvil esto nos ayuda a verificar si hay funciones incorrectas o faltantes y comprobar el rendimiento.

Tabla 15.Pruebas de caja negra. Prueba 1

Prueba	Prueba 1
Objetivo	Apertura y navegación correcta en la aplicación
Descripción	Se navega por todas las ventanas de la aplicación. Se cambia de orientación de la pantalla para ver si no hay cambios bruscos o fallos.
Resultado	Superada

Nota: Descripción prueba 1 de caja negra.

Tabla 16.Pruebas de caja negra. Prueba 2

Prueba	Prueba 2
Objetivo	Visualización de mi ubicación
Descripción	Pulsamos el botón mi ubicación y debe dirigirnos en el mapa hacia nuestra posición actual
Resultado	Superada

Nota: Descripción prueba 2 de caja negra.

Tabla 17. Pruebas de caja negra. Prueba 3

Prueba	Prueba 3
Objetivo	Comprobar los movimientos en el mapa
Descripción	Se constata que los movimientos en el mapa coincidan con la llegada de unidades.
Resultado	Superada

Nota: Descripción prueba 3 de caja negra.

Tabla 18. Pruebas de caja negra. Prueba 4

Prueba	Prueba 4
Objetivo	Comprobar el correcto envío y recepción de los mensajes
Descripción	Se constata que los mensajes se envíen, tanto el de estado de la unidad en caso de daño o el de sugerencia del usuario, esto se constata en la plataforma del administrador.
Resultado	Superada

Nota: Descripción prueba 4 de caja negra.

Tabla 19. Pruebas de caja negra. Prueba 5

Prueba	Prueba 5
Objetivo	Señalización de la ruta adecuada
Descripción	Se muestra mediante una selección de la cooperativa, cuál es su ruta de trabajo
Resultado	Superada

Nota: Descripción prueba 5 de caja negra.

Tabla 20. Pruebas de caja negra. Prueba 6

Prueba	Prueba 6
Objetivo	Comprobar la correcta visualización de la unidad localizada en el mapa.
Descripción	Se visualiza un patrón constante y correcto por la vía en el mapa sobre el recorrido que realiza la unidad.
Resultado	Superada

Nota: Descripción prueba 6 de caja negra.

Tabla 21. Pruebas de caja negra. Prueba 7

Prueba	Prueba 7
Objetivo	Comprobar la localización de más de una unidad simultáneamente
Descripción	La aplicación es capaz de dibujar varios marcadores, visualizando en el mapa la existencia de varias unidades
Resultado	Superada

Nota: Descripción prueba 8 de caja negra.

Tabla 22. Pruebas de caja negra. Prueba 8

Prueba	Prueba 8
Objetivo	Envío de coordenadas con una periodicidad dada (1 segundo)
Descripción	El perfil del controlador envía su posición cada dos segundos automáticamente desde el momento en que se abre la aplicación
Resultado	Superada

Nota: Descripción prueba 8 de caja negra automatizada.

Tabla 23. Pruebas de caja negro. Prueba 9

Prueba	Prueba 9
Objetivo	Ingreso de perfiles Usuario y Controlador
Descripción	La aplicación permite loguearse como perfil controlador y usuario en una misma aplicación
Resultado	Superada

Nota: Descripción prueba 8 de caja negra automatizada.

Pruebas unitarias

Las pruebas unitarias se ocuparon para verificar el funcionamiento adecuado de las clases que envían y reciben datos desde el servidor a través de peticiones http. Con

estas pruebas se corrigió errores hasta conseguir su funcionamiento adecuado. Android Studio facilita este proceso ya que cuenta con la librería AplicationTest la cual detecta errores mediante el compilador Gradle de este entorno.

Prueba unitaria para la clase MainActivity.java

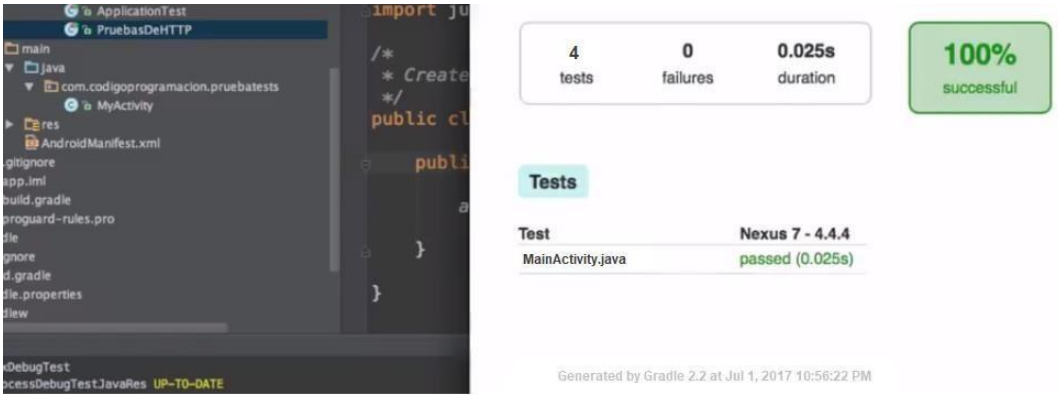

Figura 34. Prueba unitaria para la clase MainActivity.java.
Elaborado por: Ruales Daniel (2017).

En la figura 34 se puede verificar la prueba unitaria realizada a la clase que envía los datos de verificación de los actores en la aplicación.

Prueba unitaria para la clase MapsActivity.java

Figura 35. Prueba unitaria para la clase MapsActivity.java.
Elaborado por: Ruales Daniel (2017).

La prueba realizada en la figura 35 comprueba la petición de datos por parte de MapsActivity.java para verificar el movimiento en tiempo real de las unidades.

Prueba unitaria para la clase Controlador.java

Figura 36. Prueba unitaria para la clase Controlador.java.
Elaborado por: Ruales Daniel (2017).

Para el envío constante de la ubicación de la unidad se realizó la prueba que se muestra en la figura 36, que analiza la confirmación de envío de coordenadas en un tiempo determinado.

CONCLUSIONES

- La librería volley optimiza el envío de peticiones Http hacia el servidor, pero es necesario analizar en que momento nos será útil, ya que por el uso de memoria cache es difícil realizar peticiones constantes, como el envío o consulta de coordenadas cada dos segundos.
- La ubicación de las unidades en el mapa se puede percibir en tiempo real con un retraso imperceptible el cual depende de varios factores, como la señal de internet del teléfono móvil tanto del usuario como del controlador.
- Mediante la aplicación fue posible obtener datos importantes como la distancia, el tiempo, la ubicación y las rutas de trabajo de las unidades, datos que ayudan a los pasajeros para que puedan tomar mejores decisiones en cuanto a su movilización se refiere.
- La interacción entre el pasajero y el controlador es importante ya que sin la información que provee el controlador es difícil que la aplicación móvil tenga aceptación, es importante que, si en un futuro se llega a implantar una aplicación de este tipo, se socialice a las cooperativas de la importancia de promover el uso de tecnología en su negocio.
- El uso de la aplicación demuestra que no solo puede ser utilizada en la localización de buses de transporte público, este principio de localización puede ser usado en distintas áreas en la que se desee monitorear movimientos desde Google maps.

RECOMENDACIONES

- Es necesario saber que Google maps es una plataforma de pago, las consultas en los mapas están limitados a 25000 consultas diarias, por lo que debemos tomar en cuenta que para el presente proyecto funciona sin problemas, pero para proyectos más profesionales que harán de esto su constante uso. El precio de este uso dependerá de las APIs que se desee contratar mediante Google Maps API.
- La implementación de este servicio debe realizarse en zonas que cuenten con buena conexión para telefonía móvil debido a la señal de los teléfonos móviles, este tema será parte fundamental para el éxito de este tipo de aplicaciones.
- Se recomienda tener en cuenta la información que brinda Android Studio en cuanto a la elección de la API para el desarrollo, ya que, para mantener una aplicación en funcionamiento durante largo tiempo, es necesario desarrollar la misma para el mayor número de dispositivos Android posibles.

GLOSARIO DE TÉRMINOS

XML: Especificación para diseñar lenguajes de marcado, que permite definir etiquetas personalizadas para descripción y organización de datos. Representar información estructurada en la web (todos documentos), de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos. (Universidad a distancia de Madrid, s.f.)

JSON: "(JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. (JSON.ORG, s.f.)"

Scrum: "Es un proceso de la Metodología Ágil que se usa para minimizar los riesgos durante la realización de un proyecto, pero de manera colaborativa. (Gutiérrez, 2014)"

Sprint: Iteración tiene que proporcionar un resultado completo, un incremento de producto que sea potencialmente entregable, de manera que cuando el cliente (Product Owner) lo solicite sólo sea muy necesario un esfuerzo mínimo para que el producto esté disponible para ser utilizado. (Proyectos ágiles.org, s.f.)

App: El término app es una abreviatura de la palabra en inglés application. Es decir, una app es un programa. Pero con unas características especiales. Se refieren sobre todo a aplicaciones destinadas a tablets (como el iPad o equipos Android) o a teléfonos del tipo smartphone (como el iPhone o el Samsung Galaxy). También las hay en Windows 8. (Gutiérrez A., 2015)

Migración: "Se denomina migración, al proceso que tiene por objeto tanto la importación como la exportación de una determinada información almacenada en un sistema de bases de datos, para llevar a cabo su traspaso. (Blázquez, 2014)".

Backlog: (Bitácora) Documento donde se debe incluir todos los elementos que debe tener el producto que se va a desarrollar. Es la guía sobre la cual se empezará a trabajar.

Sprint: Es un ciclo de desarrollo del producto, cuya culminación significará haber desarrollado una parte del producto que será funcionable. No pueden prolongarse demasiado tiempo (hasta 4 semanas).

GPS: Global Positioning System (Sistema de posicionamiento global).

Versionamiento: Diversos cambios que se realizan a algún producto.

API: Application Programming Interface (Interfaz de programación de aplicaciones).

SDK: Software development kit (Kit de desarrollo de software).

IDE: Integrated Development Environment (Entorno de desarrollo gráfico).

LISTA DE REFERENCIAS

Agencia Metropolitana de Tránsito. (s.f.). *Agencia metropolitana de Tránsito*. Obtenido de Pico y placa: http://www.amt.gob.ec/index.php/pico-placa-homepage.html

Atlassian. (25 de Mayo de 2017). *Bitbucket*. Obtenido de Programa, gestiona, colabora: https://es.atlassian.com/software/bitbucket

Eliécer, E. (s.f.). *Alcaldía de Quito*. Obtenido de Indicadores de cobertura de la administración zonal Eloy Alfaro segun parroquias: http://sthv.quito.gob.ec/images/html/Serbas10.htm

Fernandez, Y., & Yanette, D. (1 de Enero-Abril de 2012). *Revista digital de las tecnologías de la información y comunicación.* Obtenido de Patrón Modelo-Vista-Controlador: file:///C:/Users/Haile/Downloads/15-1-42-2-10-20140122.pdf

Goméz, R. (11 de noviembre de 2015). *Rodrigo Gómez.* Obtenido de Modelo Vista Controlador: http://rodrigogr.com/blog/modelo-vista-controlador/

Hipertextual. (23 de mayo de 2014). *Hipertextual*. Obtenido de Github y Bitbucket: servicios de Git en la nube: https://hipertextual.com/archivo/2014/05/github-y-bitbucket/

Municipio del Distrito Metropolitano de Quito. (30 de Octubre de 2014). *Diagnóstico de la movilidad en el Distrito Metropolitano de Quito para el plan metropolitano de desarrollo territorial*. Obtenido de Gobierno Abierto, Alcaldia de Quito: http://gobiernoabierto.quito.gob.ec/wp-content/uploads/documentos/pdf/diagnosticomovilidad.pdf

Observatorio Regional de la Sociedad de la Información (OSRI). (2009). *Sistemas de localización e información geograficá.* Salamanca, España: Consejería de Fomento.

Palacio, J. (2015). *Scrum Manager I. Las reglas de scrum.* Scrum Manager.

Revelo, J. (22 de Febrero de 2015). *Hermosa Programación*. Obtenido de Realizar Peticiones Http Con La Librería Volley En Android: http://www.hermosaprogramacion.com/2015/02/android-volley-peticiones-http/

Robledo, D. (2012). *Desarrollo de aplicaciones para Android II.* Madrid: Hugo Alvarez.

Sánchez, J. L. (Enero de 2014). *Repositorio digital Universidad internacional SEK.* Obtenido de Asentamiento de vivienda emergente para la ciudadela Tarqu (La Mena II): http://repositorio.uisek.edu.ec/handle/123456789/696

Schwaber, K., & Sutherland, J. (Julio de 2016). *Scrum Guides*. Obtenido de La guía de Scrum: Las Reglas del Juego:

http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf

SCRUMstudy. (2016). *Una guía para el conocimiento de Scrum (Guía SBOK)*. Phoenix, Arizona: VMEdu, Inc.

Unidad de Estudios; DMPT-MDMQ. (s.f.). *Alcaldia de Quito*. Obtenido de Proyección de la población de la administración zonal Eloy Alfaro por quiteños según parroquias: http://sthv.quito.gob.ec/images/indicadores/proyeccion_zonal.htm

Universidad de Alicante. (2013). *Experto en Java*. Obtenido de Servicios Web y SOA: http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/wholesite.pdf

Universidad Politécnica de Valencia. (s.f.). *Universidad Politécnica de Valencia*. Obtenido de Master en Desarrollo de Aplicaciones Android: http://www.androidcurso.com/index.php/99