ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE UN SISTEMA DE TARIFACIÓN EN BUSES INTERPROVINCIALES CON ANDROID

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

CARLOS ROBERTO CALERO ARMAS
robercal04@hotmail.com

DIRECTOR: ING. FERNANDO FLORES
fernando.flores@epn.edu.ec

Quito, agosto 2017

DECLARACIÓN

Yo Carlos Roberto Calero Armas, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Carlos Roberto Calero Armas

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Carlos Roberto Calero Armas, bajo mi supervisión.

_____

ING. FERNANDO FLORES

DIRECTOR DEL PROYECTO.

CONTENIDO

CAPÍTULO 1. 1

FUNDAMENTO TEÓRICO. 1

1.1 INTRODUCCIÓN. 1

1.2 SISTEMA DE POSICIONAMIENTO GLOBAL (GPS). 1

1.2.1. HISTORIA DEL SISTEMA DE POSICIONAMIENTO GLOBAL. 2

1.2.2. ELEMENTOS DEL SISTEMA GPS. 2

1.2.2.1. Segmento Espacial. 3

1.2.2.2. Segmento de control. 3

1.2.2.3. Segmento del Usuario. 5

1.2.3. FUNCIONAMIENTO DEL SISTEMA DE POSICIONAMIENTO GLOBAL 5

1.2.3.1. Triangulación. 5

1.2.3.2. Señales GPS. 6

1.2.3.3. Aplicaciones del Sistema de Posicionamiento Global. 7

1.3 TARIFACIÓN INTERPROVINCIAL. 9

1.3.1. INTRODUCCIÓN. 9

1.3.2. TRANSPORTE TERRESTRE. 9

1.3.3. CLASES DE SERVICIO DE TRANSPORTE TERRESTRE. 9

1.3.4. CLASIFICACIÓN DEL SERVICIO DE TRANSPORTE PÚBLICO. ..10

1.3.5. TRANSPORTE INTERPROVINCIAL. 10

1.3.6. TARIFA EN BUSES INTERPROVINCIALES. 10

1.3.6.1. Tarifa Única. 10

1.3.6.2. Tarifa Diferenciada. 10

1.3.7. ANÁLISIS DE LA TARIFA EN BUSES INTERPROVINCIALES. 10

1.3.8. PRECIO DEL PASAJE, RUTA QUITO-GUAYAQUIL. 11

1.3.9. PRECIO DEL PASAJE, RUTA QUITO-TULCÁN. 12

1.4 FUNCIONAMIENTO DE UNA TABLET. 12

1.4.1. HISTORIA. 12
1.4.2. PARTES DE UNA TABLET. 13
1.4.2.1. Procesador. 14
1.4.2.2. Memoria RAM. 14
1.4.2.3. Disco duro. 14
1.4.2.4. Pantalla. 14
1.4.2.4.1. Pantalla lcd. 15
1.4.2.4.2. Pantalla oled. 15
1.4.2.5. Batería 15
1.4.3. SISTEMAS OPERATIVOS PARA TABLETS. 15
1.5 HERRAMIENTAS DE DESARROLLO. 16
1.5.1. INTRODUCCIÓN. 16
1.5.2. ANDROID SDK 16
1.5.3. ANDROID STUDIO 16
1.5.4. ESTRUCTURA DE UN PROYECTO. 17
1.5.5. INTERFAZ DE USUARIO. 18
1.5.6. SISTEMA DE COMPILACIÓN. 18
1.5.7. HERRAMIENTAS DE DEPURACIÓN. 19
1.5.8. ECLIPSE IDE 19
1.5.9. ESTRUCTURA DE LA PLATAFORMA DE ECLIPSE 19
1.6 SISTEMA OPERATIVO ANDROID. 21
1.6.1. INTRODUCCIÓN. 22
1.6.2. API (INTERFAZ DE PROGRAMACIÓN DE APLICACIONES). 22
1.6.2.1. Nivel de API de Android. 22
1.6.3. ARQUITECTURA DE ANDROID. 23
1.6.3.1. Linux Kernel. 24
1.6.3.2. Dalvik (VM) 25
1.6.3.3. Bibliotecas Nativas. 25
1.6.3.4. Aplicación Framework 26

1.6.3.5. Aplicaciones. 26
1.6.4. CARACTERISTICAS DEL SISTEMA OPERATIVO ANDROID. 27
1.6.5. VERSIONES DEL SISTEMA OPERATIVO ANDROID [25] 28
1.6.5.1. Android 4.1: Jelly Bean. 28
1.6.5.2. Android 4.4: KitKat. 29
1.6.5.3. Android 5.0: Lollipop. 29
1.6.5.4. Android 6.0: Marshmallow. 30
1.6.5.5. Android 7.0: Nougat. 31
CAPÍTULO 2. 32
DISEÑO E IMPLEMETACIÓN DE LA APLICACIÓN. 32
2.1 LENGUAJE DE PROGRAMACIÓN JAVA. 32
2.1.1. INTRODUCCIÓN. 32
2.1.2. KIT DE DESARROLLO DE JAVA (JDK). 32
2.1.3. ENTORNO DE EJECUCIÓN DE JAVA (JRE). 32
2.1.4. MÁQUINA VIRTUAL DE JAVA (JVM). 32
2.1.5. GARBAGE COLLECTOR. 33
2.1.6. LIBRERÍAS DE JAVA. 33
2.1.7. CARACTERÍSTICAS PRINCIPALES DE JAVA. 34
2.1.7.1. Plataforma independiente. 34
2.1.7.2. Arquitectura neutral. 34
2.1.7.3. Portable. 34
2.1.7.4. Multihilo. 35
2.1.7.5. Distribuido. 35
2.1.7.6. Robusto. 35
2.1.7.7. Dinámico. 35
2.1.7.8. Seguro 35
2.1.7.9. Orientado a objetos. 35
2.1.7.10. Alto Rendimiento. 36
2.2 INSTALACIÓN Y CONFIGURACIÓN DE ANDROID STUDIO. 36

2.2.1. REQUERIMIENTOS DEL SISTEMA PARA INSTALAR ANDROID STUDIO EN WINDOWS 36
2.2.2. INSTALACÓN DE JAVA DEVELOPMENT KIT. 37
2.2.3. INSTALACIÓN DE ANDROID STUDIO. 38
2.2.4. CREACIÓN DE UN PROYECTO NUEVO EN ANDROID STUDIO. 39
2.3 FICHEROS Y CARPETAS EN ANDROID STUDIO. 41
2.4 DESARROLLO DE LA APLICACIÓN. 43
2.4.1. REQUERIMIENTOS DE LA APLICACIÓN. 44
2.4.2. DESARROLLO DE LA PROGRAMACIÓN EN ANDROID STUDIO.44
2.4.2.1. Librerías utilizadas en la Aplicación 44
2.4.2.2. Librerías de Apache HTTP Client utilizadas en la Aplicación. .. 46
2.4.2.3. Carpeta manifests 46
2.4.2.3.1. AndroidManifest.xml 46
2.4.2.4. Implementación de las rutas. 49
2.4.2.4.1. Interfaz Pantalla Principal. 49
2.4.2.4.2. Interfaz Ruta 1. 51
2.4.2.4.3. Interfaz Ruta 2. 52
2.4.2.5. Implementación del código JAVA. 52
2.4.2.5.1. MainActivity.java 53
2.4.2.5.2. Ruta1.java 54
2.5 IMPLEMENTACIÓN DEL SERVIDOR WEB. 67
2.5.1. QUE ES UN SERVIDOR WEB? 67
2.5.2. APACHE. 68
2.5.3. SERVER UBUNTU. 68
2.5.4. MySQL....69
2.5.5. INSTALACIÓN DE UBUNTU SERVER EN VITUALBOX. 69
2.5.5.1. Instalación de Apache en Ubuntu Server. 75
2.5.5.2. Instalación de MySQL en Ubuntu Server. 76
2.5.6. CREACIÓN DE LA BASE DE DATOS EN phpMyAdmin. 78
2.5.6.1. Base de datos “tarifa1” 78

2.5.6.1.1. Tabla “bus1” 78
2.6 ENVÍO Y RECEPCIÓN DE DATOS HACIA EL SERVIDOR WEB. 79
2.6.1. ENVIO DE DATOS CON EL METODO HttpGet. 79
2.6.2. RECEPCIÓN DE DATOS CON PHP. 80
2.6.2.1. Conexión a la base de datos con PHP y MySQL. 80
2.6.2.2. Inserción de datos con PHP y MySQL. 81
2.7 PÁGINA WEB. 82
2.7.1. INTERFACES WEB. 82
2.7.1.1. Interfaz index.php..... 82
2.7.1.2. Interfaz index2.php..... 83
CAPÍTULO 3. 86
PRUEBAS, ANÁLISIS Y COSTOS. 86
3.1 COMPATIBILIDAD DE LA APP CON VERSIÓN DE ANDROID. 86
3.2 SIMULACIÓN DE LA APLICACIÓN EN ANDROID STUDIO. 86
3.2.1. EMULADOR DE ANDROID STUDIO. 86
3.2.2. RESULTADO DE LAS PRUEBAS EN EL EMULADOR DE ANDROID STUDIO Y EN EL SERVIDOR WEB. 89
3.2.2.1. Análisis de los resultados del funcionamiento de la aplicación en el dispositivo virtual. 89
3.2.2.2. Análisis de los resultados en el Servidor Web. 89
3.2.2.3. Análisis de los resultados en la Página Web..... 90
3.3 INSTALACIÓN DE LA APLICACIÓN EN UN DISPOSITIVO REAL CON OS ANDROID. 90
3.3.1. INSTALACIÓN DE LA APLICACIÓN DESDE ANDROID STUDIO VIA USB.....91
3.3.2. INSTALACIÓN DE LA APLICACIÓN DESDE ANDROID STUDIO MEDIANTE EL ARCHIVO .APK. 93
3.3.3. RESULTADO DE LAS PRUEBAS EN EL DISPOSITIVO REAL.....95
3.3.3.1. Resultados del funcionamiento de la aplicación en el dispositivo real……..... 95
3.3.3.1.1. Ingreso a la aplicación. 95

3.3.3.1.2. Ingreso a la interfaz “RUTA 2” (Ruta Quito – Santo Domingo)..... 96
3.3.3.1.3. Cálculo del valor del pasaje (ida). 98
3.3.3.1.4. Cálculo del valor del pasaje (regreso). 100
3.3.3.2. Análisis de los resultados en el Servidor Web. 101
3.3.3.3. Análisis de los resultados en la Página Web..... 103
3.4 COSTO REFERENCIAL DE LA APLICACIÓN IMPLEMENTADA. 107
3.4.1. COSTO DEL DISEÑO DE LA APLICACIÓN. 108
3.4.2. COSTO DEL DESARROLLO DE LA APLICACIÓN.....108
3.4.3. COSTO DE LA DEPURACIÓN DE LA APLICACIÓN.....109
3.4.4. COSTO DE INTENET MOVIL.....109
3.4.5. COSTO TOTAL DE LA APLICACIÓN. 109
CAPÍTULO 4. 110
CONCLUSIONES Y RECOMENDACIONES 110
4.1 CONCLUSIONES..... 110
4.2 RECOMENDACIONES. 112
REFERENCIAS BIBLIOGRÁFICAS 114
ANEXOS. 118

ÍNDICE DE FIGURAS

CAPÍTULO 1.

Figura 1.1 Esquema del Sistema. 1

Figura 1.2 Constelación NAVSTAR 3

Figura 1.3 Satélite NAVSTAR GPS 3

Figura 1.4 Segmento de control 4

Figura 1.5 Área de Intersección de los dos satélites 5

Figura 1.6 Área de Intersección de los tres satélites 6

Figura 1.7 Dynabook 13

Figura 1.8 Tarjeta estado sólido (SSD) 14

Figura 1.9 Archivos de Proyecto en la vista Android 17

Figura 1.10 Ventana principal de Android Studio 18

Figura 1.11 Plataforma de eclipse 20

Figura 1.12 Arquitectura Android 24

Figura 1.13 Android 4.1 Jelly Bean 28

Figura 1.14 Android 4.4 KitKat 29

Figura 1.15 Android 5.0 Lollipop 30

Figura 1.16 Android 6.0 Marshmallow 30

Figura 1.17 Android 7.0 Nougat 31

CAPÍTULO 2.

Figura 2.1 Plataforma independiente 34

Figura 2.2 Página de descarga de Java JDK 37

Figura 2.3 Variable de entorno JAVA_HOME 38

Figura 2.4 Link de descarga de Android Studio 38

Figura 2.5 Asistente de instalación 39

Figura 2.6 Configuración de un nuevo proyecto. 39

Figura 2.7 Selección del nivel mínimo de API 40

Figura 2.8 Selección de la actividad 41

Figura 2.9 Nombre de la actividad 41

Figura 2.10 AndroidManifest.xml 42

Figura 2.11 Carpeta java. 42

Figura 2.12 Carpeta res 43

Figura 2.13 Carpeta Gradle Scripts. 43

Figura 2.14 Google APIs.. 47

Figura 2.15 Menú Credenciales.. 47

Figura 2.16 Clave API.. 48

Figura 2.17 Clave API, ubicación Android Studio. 48

Figura 2.18 Android SDK Manager. 49

Figura 2.19 Carpeta res 50

Figura 2.20 Pantalla principal... 50

Figura 2.21 Interfaz ruta 1. 52

Figura 2.22 Carpeta java aplicación. 53

Figura 2.23 Ciclo de vida de un fragmento. 55

Figura 2.24 Ubicación de paradas. 57

Figura 2.25 Parada correcta o incorrecta 59

Figura 2.26 Cálculo de la posición. 60

Figura 2.27 Cálculo de las distancias. 61

Figura 2.28 Distancia más corta al Usuario. 64

Figura 2.29 Funcionamiento de un servidor web HTTP. 67

Figura 2.30 Logo de Apache Server. 68

Figura 2.31 Máquina Virtual VirtualBox. 70

Figura 2.32 Creación de máquina virtual. 70

Figura 2.33 Tamaño de memoria. 71

Figura 2.34 Selección del disco duro 71

Figura 2.35 Tipo de archivo de disco duro 72

Figura 2.36 Almacenamiento en unidad de disco duro 72

Figura 2.37 Ubicación del archivo y tamaño 73

Figura 2.38 Selección de la Imagen. 73

Figura 2.39 Idioma para la instalación. 74

Figura 2.40 Instalar Ubuntu Server. 74

Figura 2.41 Shell de Ubuntu Server. 75

Figura 2.42 Página web de Apache Ubuntu por defecto. 76

Figura 2.43 Interfaz de inicio phpMyAdmin. 77

Figura 2.44 Interfaz de usuario phpMyAdmin. 78

Figura 2.45 Tabla “bus1”. 79

Figura 2.46 Interfaz, Inicio de sesión 83

Figura 2.47 Interfaz, Usuario 83

Figura 2.48 Reporte del bus 84

Figura 2.49 Recorrido del bus Quito – Los Bancos 84

Figura 2.50 Ingreso a la Base de Datos 85

CAPÍTULO 3.

Figura 3.1 Selección del emulador 86

Figura 3.2 Selección del dispositivo 87

Figura 3.3 Versión de Android 87

Figura 3.4 Versión de configuración 88

Figura 3.5 Aplicación en el emulador de Android Studio.... 88

Figura 3.6 Menú de la aplicación.....89

Figura 3.7 Activar opciones de desarrollador... 91

Figura 3.8 Activación Depuración de USB... 91

Figura 3.9 Botón Run app... 92

Figura 3.10 Selección del dispositivo... 92

Figura 3.11 Aplicación instalada en la Tablet... 92

Figura 3.12 Generación del archivo apk... 93

Figura 3.13 Archivo .apk.....93

Figura 3.14 Localización del archivo apk... 94

Figura 3.15 Instalación de la aplicación... 94

Figura 3.16 Ingreso a la aplicación.....95

Figura 3.17 Ruta Quito – Santo Domingo... 96

Figura 3.18 PARADA 1 (Terminal Terrestre Quitumbe). ... .96

Figura 3.19 PARADA 2 (Tambillo).....97

Figura 3.20 PARADA 3 (Tandapi) 97

Figura 3.21 PARADA 4 (Alluriquin).... 97

Figura 3.22 Terminal Terrestre Santo Domingo. 98

Figura 3.23 Parada Tambillo... 99

Figura 3.24 Ejemplo del caculo del pasaje... 99

Figura 3.25 Datos ingresados en la parada Tambillo.... 99

Figura 3.26 Datos ingresados en la parada Y de Aloag... 100

Figura 3.27 Datos ingresados en la parada Alluriquin yTandapi 101

Figura 3.28 Tabla “bus1” llena por el envió de datos desde la aplicación. 102

Figura 3.29 Interfaz de Inicio... 104

Figura 3.30 Interfaz de reporte y trayectoria del bus.... 104

Figura 3.31 Reporte del Bus, ruta Quito-Santo Domingo... 105

Figura 3.32 Visualización de la ruta Quito-Santo Domingo.... 106

Figura 3.33 Detalle de la ruta Quito-Santo Domingo..... 106

Figura 3.34 Información con marcadores de Google Maps.....107

ÍNDICE DE TABLAS

CAPÍTULO 1.

Tabla 1.1 Tarifa ruta Quito-Guayaquil. 11

Tabla 1.2 Tarifa ruta Quito-Tulcán. 12

Tabla 1.3 Nivel de API en Android 23

Tabla 1.4 Características del OS Android 27

CAPÍTULO 2.

Tabla 2.1 Librerías principales de Java 33

Tabla 2.2 Atributos de la Tabla “bus1” 78

CAPÍTULO 3.

Tabla 3.1 Análisis de los resultados del funcionamiento de la aplicación. 89

Tabla 3.2 Análisis de los resultados en el Servidor Web..…..90

Tabla 3.3 Análisis de los resultados en la Página Web..…..90

Tabla 3.4 Análisis de los resultados, dispositivo real... 101

Tabla 3.5 Análisis de los resultados en el Servidor Web, dispositivo real... 103

Tabla 3.6 Análisis de los resultados en la Página Web, dispositivo real... 107

Tabla 3.7 Costo del diseño..…..108

Tabla 3.8 Costo del desarrollo.....108

Tabla 3.9 Costo de la depuración.....109

Tabla 3.10 Costo total de la aplicación....…..109

ÍNDICE DE CÓDIGOS

CAPÍTULO 2.

Código 2.1 Acceder a Internet desde la aplicación Android 46

Código 2.2 Leer datos desde un almacenamiento externo 46

Código 2.3 Permiso para Google Maps 49

Código 2.4 Acceder a una ubicación aproximada. 49

Código 2.5 Acceder a una ubicación precisa 49

Código 2.6 Código xml para botones 51

Código 2.7 Declaración de variables 53

Código 2.8 Método onCreate. 54

Código 2.9 Función showConsoleActivity 54

Código 2.10 Clase ruta 1.....56

Código 2.11 Variables. 56

Código 2.12 Creación de las paradas y subparadas 58

Código 2.13 clase LocationManager. 58

Código 2.14 Parada correcta o incorrecta 60

Código 2.15 Distancias de las paradas principales y subparadas. 62

Código 2.16 Función calcDist 63

Código 2.17 Distancia más cercana al usuario 65

Código 2.18 Segunda distancia más cercana al usuario 65

Código 2.19 Bus de ida o de regreso 66

Código 2.20 Costo del pasaje,bus de ida 66

Código 2.21 Costo del pasaje,bus de regreso 67

Código 2.22 Instalación de Apache 75

Código 2.23 Instalación de MySql 76

Código 2.24 Instalación de phpMyAdmin 76

Código 2.25 Reinicio de Apache 77
Código 2.26 Método HttpGet 79
Código 2.27 Conexión a la base de datos con PHP y MySQL 81
Código 2.28 Inserción de datos con PHP y MySQL 82

RESUMEN

El presente proyecto de titulación tiene como objetivo desarrollar e implementar una aplicación que brinde información real de donde se detienen los buses interprovinciales para recoger pasajeros en todo el trayecto hasta llegar al destino, y conocer el valor real recaudado por el cobro de pasajes de dichos usuarios, mediante el uso de coordenadas GPS que nos indicaran en que sitio se detuvo el bus, y también nos permitirá calcular la tarifa que debe pagar el usuario que abordo el bus en el recorrido. La aplicación estará instalada en una Tablet con Sistema Operativo Android, que estará a bordo del bus. Para el desarrollo de la aplicación se utilizará Android Studio, que es el entorno de programación más completo y potente para el desarrollo de aplicaciones en el Sistema Operativo Android. Los datos que se obtengan de la aplicación serán enviados a un servidor Web Apache, el envió de datos entre la aplicación y el servidor se lo realizará utilizando la red celular.

El proyecto se desarrolla en cuatro capítulos en los que se describen todas las fases del proyecto, a continuación se detalla cada uno de ellos.

En el primer Capítulo se describen los fundamentos de GPS, se realiza un breve análisis en la tarifación en buses interprovinciales. A continuación se describe el funcionamiento de una Tablet, seguido se realiza una descripción de las herramientas de desarrollo que se utilizará en la aplicación, y por último se describe el funcionamiento del Sistema Operativo Android.

En el segundo Capítulo se realizará una introducción al lenguaje de programación JAVA, se diseñará e implementará la aplicación en una Tablet con el sistema operativo Android, se describirá el desarrollo de la aplicación mediante diagramas de flujo y código implementado en la aplicación. Se realizará una breve introducción a servidores web, se describirá como se instaló y configuró el servidor web. Y se definirá el envió de la información desde la aplicación hacia el servidor utilizando la red celular.

En el tercer Capítulo se realizará las pruebas de funcionamiento de la aplicación en el bus interprovincial, se analizarán los resultados y se mostrará el costo del prototipo que se implementó.

En el cuarto Capítulo se presentarán las conclusiones y recomendaciones del prototipo desarrollado.

PRESENTACIÓN

En el transporte Interprovincial no se lleva a cabo un control real de donde se detienen los buses para recoger pasajeros en todo el trayecto hasta llegar a su destino, y tampoco se conoce el valor real recaudado por el cobro de los pasajes a dichos usuarios, generando conflictos con el personal de trabajo el dueño del bus y los usuarios.

En la actualidad el uso de dispositivos móviles como Smartphones y Tablets es común, ya que con el avance de la tecnología en la construcción de estos dispositivos y el gran número de fabricantes, existen dispositivos de todo precio en el mercado, estos dispositivos son utilizados en la vida diaria, convirtiéndose en una herramienta indispensable para las actividades diarias de una persona.

A través de estos dispositivos una persona puede acceder a Internet, ingresar a redes sociales, ingresar a páginas Web, descargar aplicaciones, saber la ubicación en cualquier momento, etc., siempre y cuando tenga un paquete de datos, o se conecte a una red Wifi. Los paquetes de datos en la actualidad están al alcance de todos, el costo es bajo para acceder a este servicio. Cabe también señalar que la mayoría de Tablets y Smartphones que hay en el mercado utilizan el Sistema Operativo Android, que es libre.

Tomando en cuenta estos antecedentes sobre los dispositivos móviles, se ha creado una Aplicación para una Tablet con Sistema Operativo Android, con esta Aplicación se pretende llevar a cabo un control de pasajeros y el valor recaudado por el cobro de pasajes por medio de coordenadas GPS, esta información será enviada por la red celular a un servidor. Accediendo a una página web que se creó para el monitoreo de la aplicación, la persona interesada en esta información podrá acceder a un reporte del bus y también podrá observar el recorrido de dicho bus en un mapa de Google Maps.

CAPÍTULO 1.

FUNDAMENTO TEÓRICO.

1.1 INTRODUCCIÓN.

En las siguientes páginas se realiza una descripción técnica breve de los elementos que intervienen en la aplicación, como el Sistema Posicionamiento Global, Tarifación Interprovincial, Funcionamiento de una Tablet, Herramientas de Desarrollo y el Sistema Operativo Android.

A continuación se muestra el esquema del sistema a implementarse (Figura 1.1):

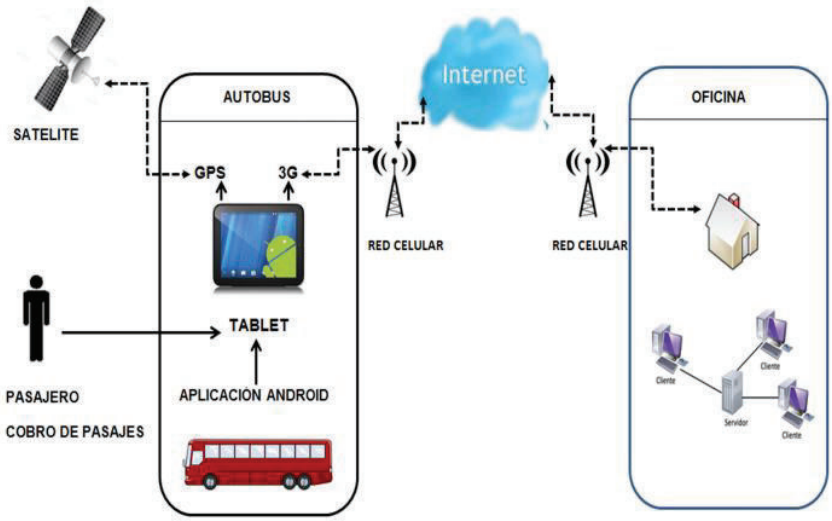


Figura 1.1 Esquema del Sistema.

1.2 SISTEMA DE POSICIONAMIENTO GLOBAL (GPS).

GPS es el acrónimo de Global Positioning System, es un sistema formado por una constelación de 24 satélites, llamados NAVSTAR, y 5 estaciones repartidas por la superficie terrestre. Se trata de un sistema que permite calcular las coordenadas de cualquier punto de la superficie terrestre a partir de la recepción de señales emitidas desde una constelación de satélites en órbita. Básicamente, su principal funcionalidad es que permite al usuario conocer, mediante un receptor, su posición en cualquier parte del planeta [1].

1.2.1. HISTORIA DEL SISTEMA DE POSICIONAMIENTO GLOBAL.

TRANSIT fue el primer sistema de navegación basado en satélites y utilizado para la localización de objetos sobre la superficie terrestre, financiado por la Fuerza Aérea y la Marina de los Estados Unidos, el cual entró en funcionamiento en 1965. Cumplía con algunos requisitos como abarcar toda la superficie del globo, tener funcionamiento continuo sin afectarle las condiciones atmosféricas y ser altamente dinámico, para posibilitar su uso en aviación y precisión.

El sistema TRANSIT estaba constituido por una constelación de seis satélites en órbita polar baja, a una altura de 1074 Km y los datos obtenidos eran bidimensionales (Latitud, Longitud). Tal configuración conseguía una cobertura mundial pero el inconveniente principal era que no se tenían los datos de la posición del objeto continuamente. Cada hora y media se podía acceder a los satélites, y para el cálculo de la posición se requería estar siguiendo al satélite durante quince minutos continuamente. Se realizaron otros experimentos como el Timation (Bidimensional pero con mejor precisión temporal y financiado por la marina de los Estados Unidos) y el sistema 621B (Tridimensional y financiado por la Fuerza Aérea de los Estados Unidos).

En 1973 el departamento de Defensa de Estados Unidos consolidó los programas Timation y 621B en un único sistema llamado NAVSTAR Global Positioning System el 17 de Agosto de 1974. El primer satélite de los 24 del sistema se lanza el 22 de febrero de 1978, y en diciembre de 1983 se declaró la fase operativa inicial del sistema GPS, en 1984 el Gobierno de los Estados Unidos permite el uso de este sistema para los civiles, ya que este sistema al principio estaba diseñado para el uso militar de este país [1].

1.2.2. ELEMENTOS DEL SISTEMA GPS.

En el sistema GPS se compone de tres elementos denominados segmentos:

1.2.2.1. Segmento Espacial.

Corresponde a los satélites que están alrededor de la Tierra, son 24 satélites y 4 de reserva que forman la constelación NAVSTAR (Navigation Satellite Timing and Ranging) que están ubicados a una altura aproximada de 26560 Km de la superficie, y realizan una vuelta alrededor de la Tierra en 12 horas (Figura 1.2).
Los satélites (Figura 1.3) se encuentran distribuidos de tal manera que un receptor GPS ubicado en cualquier parte de la superficie terrestre puede recibir las señales de al menos tres satélites para conocer su posición [2].

Figura 1.2 Constelación NAVSTAR [1]

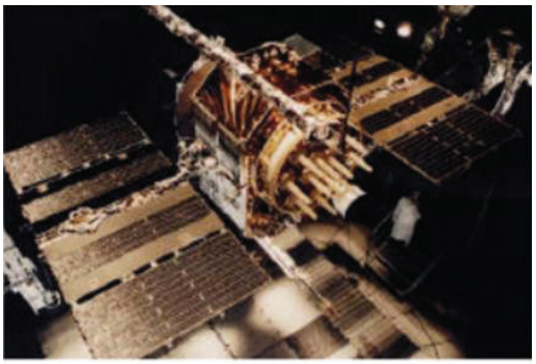


Figura 1.3 Satélite NAVSTAR GPS [1]

1.2.2.2. Segmento de control.

Está formado por cinco estaciones; de control y seguimiento ubicados en lugares estratégicos de la superficie terrestre (Figura 1.4).

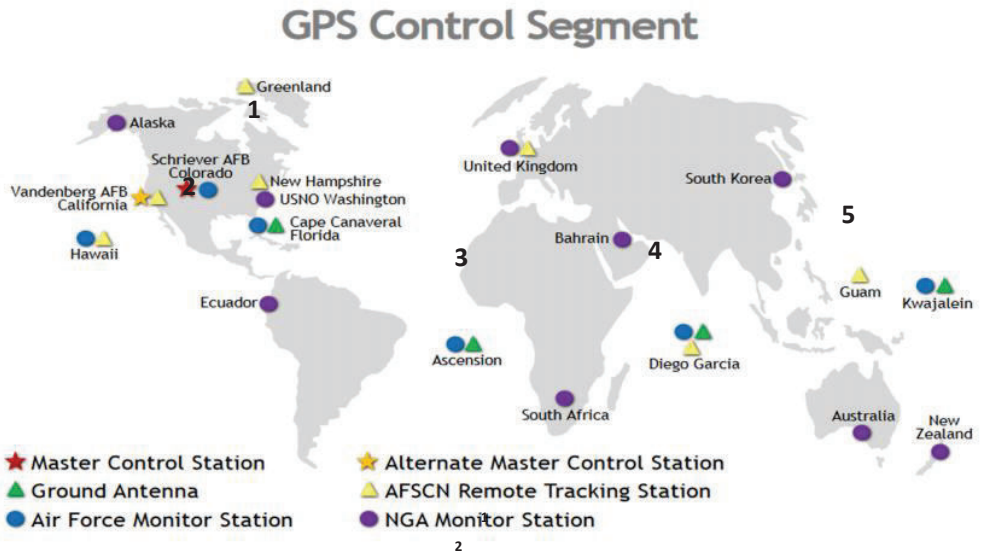


Figura 1.4 Segmento de control. 1. Estación maestra de control (MSC) [1] y estación de seguimiento localizada en Colorado Springs. 2. Hawái, estación de seguimiento. 3. Isla de Ascensión, estación de seguimiento y antena de referencia. 4. Diego García, estación de seguimiento y antena de referencia. 5. Kawajalein, estación de seguimiento y antena de referencia [3].

Dichas estaciones se encargan de mantener los satélites de la constelación NAVSTAR en órbita y calibrar los relojes satelitales, de hacer un seguimiento a los satélites y de recoger los datos enviados por los mismos, estos datos se envían de nuevo a los satélites con las correcciones de órbita para sus sistemas de navegación para calcular las orbitas exactas.

Existen tres elementos del segmento de control, la estación de control maestro es la encargada de realizar el mando y el control de la constelación de satélites GPS, las estaciones monitoras realizan un seguimiento de los satélites GPS a medida que pasan por encima y envían la información de datos atmosféricos, señales de navegación a la estación de control maestro. Las antenas terrestres se utilizan para establecer una comunicación con los satélites GPS, para realizar un control y mando de los mismos, la comunicación es transmitida en la banda S [3].

1 AFSN: Red de Control de satélites de la Fuerza Aérea
2 NGA: Agencia Nacional de Inteligencia Geoespacial.

1.2.2.3. Segmento del Usuario.

Es propiamente dicho el equipo receptor GPS, es el dispositivo que recibe los mensajes de los satélites GPS, y que a partir de esta recepción se calculan la distancia, posición y tiempo. Este servicio es gratuito y se han desarrollado muchas aplicaciones en beneficio de los usuarios en todo el mundo.

1.2.3. FUNCIONAMIENTO DEL SISTEMA DE POSICIONAMIENTO GLOBAL

Un satélite envía señales a los receptores GPS a intervalos precisos, luego el receptor compara el tiempo de la señal transmitida por el satélite con el tiempo que fue recibida, la diferencia de tiempos multiplicada por la velocidad de la luz se determina a que distancia está el receptor del satélite. Cuando al menos tres satélites realizan el mismo proceso se obtiene la ubicación exacta del receptor GPS, ha este proceso se lo conoce como triangulación.

1.2.3.1. Triangulación

Cuando el primer satélite envía su señal y obtiene la distancia a la que se encuentra del receptor, por sí mismo no puede determinar la ubicación de dicho receptor. Lo único que se puede determinar es la ubicación del receptor en algún lugar dentro de una circunferencia de cobertura de la señal, cuyos radios son las medidas entre el receptor y los satélites. Así mediante el uso de un segundo satélite que ha realizado el proceso para determinar la distancia desde el receptor hacia el satélite, se va a conocer también la ubicación del receptor en algún lugar de la circunferencia de cobertura de la señal, el receptor debe estar ubicado en algún lugar de la intersección de las dos circunferencias como se indica en la Figura 1.5

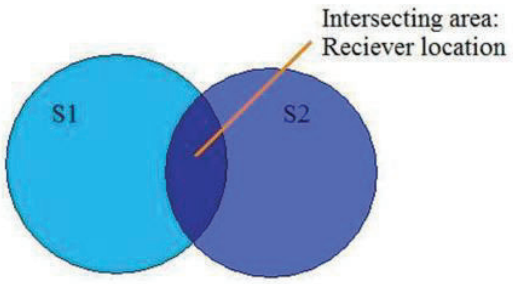


Figura. 1.5 Área de Intersección de los dos satélites [4].

Mediante el uso de un tercer satélite se va a crear una nueva circunferencia que va a cortar en las intersecciones de las tres circunferencias, en dos puntos. La posición verdadera del receptor está en un punto, el otro punto queda eliminado porque es una ubicación dudosa ya que está demasiado lejana de la superficie de la Tierra (Figura 1.6).

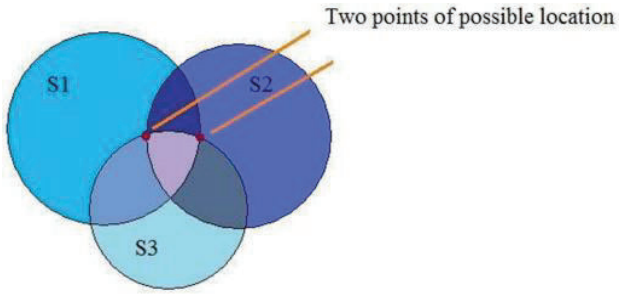


Figura. 1.6 Área de Intersección de los tres satélites [4].

Una vez que los receptores han triangulado su posición, reciben señales de baja potencia de los satélites, estas señales contienen tres bits de información un código de identificación del satélite, datos de localización del satélite, y el estado del satélite, y mediante este proceso aparecerá la ubicación del receptor GPS en el mapa.

1.2.3.2. Señales GPS.

Los satélites GPS utilizan para la transmisión dos señales de radio, estas se denominan como L1 y L2. La frecuencia de la señal portadora L1 a 1575.42 MHz, transmite los códigos C/A y P. La frecuencia de la señal portadora L2 a 1227.60 MHz, transmite información modulada en código P y están en la banda UHF.

Los dos códigos GPS que se utilizan para la transmisión de información se denominan de Adquisición Aproximada (C/A-Code) y de Precisión (P-Code). Cada código se compone de bits, los códigos son conocidos como códigos PRN[2] porque se ven como señales aleatorias(es decir son señales similares al ruido). Pero en realidad los códigos se generan utilizando un algoritmo matemático. Actualmente el C/A-Code se modula en la portadora L1, mientras que el código P se modulada en la portadora L1 y L2. Esta modulación se denomina, modulación

[2] Ruido pseudo aleatorio.

bifase, debido a que la fase de la portadora se desplaza 180 grados cuando el valor código cambia de cero a uno o de uno a cero [5].

El código C/A-Code se compone de 1023 bits que se repiten cada 1us, esto quiere decir que la tasa de velocidad es de 1023 Mbps, y una longitud de onda de 300m. A cada satélite se le asigna un C/A-Code único, que permite a los receptores GPS identificar que satélite está transmitiendo un código en particular.
El código P-Code se repite cada 266 días y es diez veces más rápido que el código C/A-Code, es decir tiene una velocidad de 10.23 Mbps y una longitud de onda de 30m. El código de 266 días de duración se divide en 38 segmentos, cada segmento tiene una semana de duración, 32 segmentos son asignados a los diversos satélites GPS, los seis segmentos restantes se reservan para otros usos. El código P está diseñado principalmente para el uso militar. Estaba disponible a todos los usuarios hasta enero 31 de 1994, en ese momento el código P se ha encriptado mediante la suma de un código desconocido W-Code, el código encriptado resultante se denomina Y-Code que tiene la misma velocidad que el P-Code. Esta encriptación es conocida como operación anti engaños.

Las señales GPS contienen tres bits de información diferente, el código pseudo aleatorio, que es simplemente un código que identifica al satélite que está transmitiendo la información. Los datos de localización del satélite, que describen los cursos orbitales del satélite, cada dispositivo GPS podrá determinar que satélite va a seguir. Los datos de efemérides, es un dato que indica al receptor GPS donde va a estar ubicado el satélite en cualquier momento, durante todo el día.

1.2.3.3. Aplicaciones del Sistema de Posicionamiento Global.

El sistema de posicionamiento global es un elemento básico en la infraestructura global de la información. Ya que es un sistema desarrollado con código abierto, libre y confiable; a conducido al desarrollo de cientos de aplicaciones que intervienen en la vida moderna. La tecnología GPS se encuentra ahora en todo, desde teléfonos celulares, relojes inteligentes hasta cajeros automáticos.

El sistema de posicionamiento global aumenta la productividad en gran parte de la economía, como en la agricultura, la construcción, la minería, topografía, en la logística para la entrega de paquetes, en sistemas bancarios, algunos servicios inalámbricos no pueden funcionar sin GPS. En el transporte aéreo GPS es vital.

A continuación se describirán una pequeña muestra de aplicaciones GPS existentes, ya que cada día se inventan nuevos usos con GPS.

- **Aviación.** Los sistemas de navegación de los aviones cuentan con un receptor GPS, que a menudo muestran un mapa en movimiento y están conectados al piloto automático para la navegación en ruta, también se usan para operaciones de aproximación final y aterrizaje [6].
- **Topografía y mapeo.** Hoy en día el GPS es vital para las actividades de topografía y mapeo ya que se obtienen datos más precisos y confiables. La recolección de datos basada en GPS es mucho más rápida que las técnicas convencionales de topografía, reduciendo la cantidad de equipo y mano de obra requerida. Las medidas que se realizan con el GPS se pueden visualizar en mapas y en sistemas de información geográfica (GIS) que almacenan, manipulan y muestran datos geográficamente referenciados [7].
- **Control y navegación del transporte.** Los vehículos pueden venir equipados con receptores GPS, los cuales pueden mostrar mapas en movimiento e información sobre la ubicación, la velocidad, la dirección y las calles y puntos de interés, también permite la localización automática del vehículo. Las aplicaciones en transporte por carreteras son las más utilizadas por los usuarios, por ejemplo para el seguimiento de mercancía, servicio de taxi, monitoreo de transporte público, navegación de servicios de socorro.
- **Medio ambiente.** Las aplicaciones científicas del GPS ayudan al monitoreo ambiental y atmosférico, al estudio sobre el comportamiento de animales, a la meteorología y la investigación climática.
- **Telecomunicaciones.** Las aplicaciones en telecomunicaciones, en particular en redes de telefonía móvil es importante la sincronización GPS. La tecnología síncronas son más eficientes que las tecnologías asíncronas,

pero requieren una fuente con exactitud estabilidad y fiabilidad, y los satélites GPS pueden proporcionar esto.

1.3 TARIFACIÓN INTERPROVINCIAL.

1.3.1. INTRODUCCIÓN.

Es conveniente conocer conceptos básicos sobre el transporte interprovincial y la tarifación que rige al mismo. Estos conceptos van ayudar a entender de mejor manera el tema de la tarifación en los buses interprovinciales, que va ser implementada en la aplicación.

1.3.2. TRANSPORTE TERRESTRE.

"El transporte terrestre automotor es un servicio público esencial y una actividad económica estratégica del Estado, que consiste en la movilización libre y segura de personas o de bienes de un lugar a otro, haciendo uso del sistema vial nacional, terminales terrestres y centros de transferencia de pasajeros y carga en el territorio ecuatoriano. Su organización es un elemento fundamental contra la informalidad, mejorar la competitividad y lograr el desarrollo productivo, económico y social del país, interconectado con la red vial internacional." [3]

1.3.3. CLASES DE SERVICIO DE TRANSPORTE TERRESTRE.

Según el Art. 51 de la Ley Orgánica de Transporte Terrestre Transito y Seguridad Vial, se establecen las siguientes clases se servicio de transporte terrestre:

a) Publico;
b) Comercial;
c) Por cuenta propia; y,
d) Particular.

[3] Art. 46 de la Ley Orgánica de Transporte Terrestre Tránsito y Seguridad Vial.

1.3.4. CLASIFICACIÓN DEL SERVICIO DE TRANSPORTE PÚBLICO.

De acuerdo al Art. 65. De la Ley Orgánica de Transporte Terrestre Transito y Seguridad Vial, el servicio de transporte publico comprende los siguientes ámbitos de operación: intracantonal, interprovincial, intrarregional, intraprovincial e internacional.

1.3.5. TRANSPORTE INTERPROVINCIAL.

"El servicio de transporte publico interprovincial es aquel que opera, bajo cualquier tipo, dentro de los límites del territorio Nacional." [4]

1.3.6. TARIFA EN BUSES INTERPROVINCIALES.

Para fijar el valor de la tarifa en el transporte interprovincial que un usuario debe cancelar por el uso de este servicio, las autoridades diseñan un sistema tarifario adecuado entre los pasajeros y las empresas que brindan este servicio.
En la actualidad el organismo que se encarga de fijar las tarifas en cada uno de los servicios de transporte terrestre es la Agencia Nacional de Transporte.

1.3.6.1. Tarifa Única.

Es el valor que debe cancelar un usuario por el servicio de transporte público, estos usuarios pagaran el valor del pasaje completo.

1.3.6.2. Tarifa Diferenciada.

Es el valor que debe cancelar un usuario con discapacidades, adultas y adultos mayores de 65 años de edad, niñas y niños, por el servicio de transporte público, estos usuarios pagaran el valor correspondiente a la mitad del pasaje completo.

1.3.7. ANÁLISIS DE LA TARIFA EN BUSES INTERPROVINCIALES.

De acuerdo al artículo 38, Capítulo V de las tarifas, del Reglamento de Transporte Público Interprovincial de Pasajeros, los valores de las tarifas corresponde a los valores que la Agencia Nacional de Tránsito, conforme las

[4] Art. 68 de la Ley Orgánica de Transporte Terrestre Tránsito y Seguridad Vial.

disposiciones contenida en la Ley Orgánica de Transporte Terrestre, Transito y Seguridad Vial, ha determinado como contraprestación económica por el servicio de transporte terrestre publico interprovincial de un sitio a otro y establecido de acuerdo a los niveles de calidad que oferte la operadora.

En el artículo 39, Capítulo V de las tarifas, del Reglamento de Transporte Público Interprovincial de Pasajeros se expresa que la Agencia Nacional de Tránsito será la encargada de la regulación de tarifas de los servicios de transporte terrestre interprovincial, para lo cual deberá considerarse e incluir análisis técnicos de los costos de operación, definición de las variables en la metodología de cálculo y levantamiento de información, los mismos que serán puestos a consideración del Directorio de la Agencia Nacional de Regulación y Control del Transporte Terrestre, Tránsito y Seguridad Vial para su aprobación, reforma o delegación, conforme lo dispone la Ley Orgánica de Transporte Terrestre, Transito y Seguridad Vial. Para realizar el análisis técnico para la regulación de las tarifas se toma en cuenta aspectos como kilómetros recorridos, tipos de vía, demanda de pasajeros, zona de servicio, mantenimiento preventivo y correctivo, entre otros.

1.3.8. PRECIO DEL PASAJE, RUTA QUITO-GUAYAQUIL.

Ejemplo de Tarifa en la ruta Quito Guayaquil (Tabla 1.1).

Provincia de Origen	Provincia Destino	Cantón de Destino	Ciudad / Sector Destino	Anexo Consolidado
PICHINCHA	PICHINCHA	MEJÍA	ALOAG	$ 0,65
PICHINCHA	PICHINCHA	MEJÍA	TANDAPI	$ 1,90
PICHINCHA	SANTO DOMINGO	SANTO DOMINGO	ALLURIQUÍN	$ 2,60
PICHINCHA	SANTO DOMINGO	SANTO DOMINGO	SANTO DOMINGO	$ 3,25
PICHINCHA	LOS RÍOS	QUEVEDO	QUEVEDO	$ 5,75
PICHINCHA	LOS RÍOS	VENTANAS	VENTANAS	$ 6,75
PICHINCHA	LOS RÍOS	BABAHOYO	BABAHOYO	$ 7,40
PICHINCHA	GUAYAS	GUAYAQUIL	GUAYAQUIL	$ 10,20

Tabla 1.1 Tarifa ruta Quito-Guayaquil [9].

1.3.9. PRECIO DEL PASAJE, RUTA QUITO-TULCÁN.

Ejemplo de Tarifa en la ruta Quito Tulcán (Tabla1.2).

Provincia de Origen	Provincia Destino	Cantón de Destino	Ciudad / Sector Destino	Anexo Consolidado
PICHINCHA	PICHINCHA	QUITO	GUAYLLABAMBA	$ 0,54
PICHINCHA	PICHINCHA	PEDRO MONCAYO	TABACUNDO	$ 0,95
PICHINCHA	PICHINCHA	CAYAMBE	CAYAMBE	$ 1,25
PICHINCHA	IMBABURA	OTAVALO	OTAVALO	$ 2,50
PICHINCHA	IMBABURA	COTACACHI	COTACACHI	$ 2,70
PICHINCHA	IMBABURA	ANTONIO ANTE	ATUNTAQUI	$ 2,70
PICHINCHA	IMBABURA	IBARRA	IBARRA	$ 3,00
PICHINCHA	CARCHI	MONTÚFAR	SAN GABRIEL	$ 5,15
PICHINCHA	CARCHI	TULCÁN	TULCÁN	$ 6,10

Tabla 1.2 Tarifa ruta Quito-Tulcán [9].

1.4 FUNCIONAMIENTO DE UNA TABLET.

Una Tablet es una computadora personal de pantalla táctil inalámbrica, que normalmente es más pequeña que una computadora portátil, pero más grande que un teléfono inteligente. Son generalmente usadas para navegar por internet, leer correos electrónicos, para navegar en redes sociales, ver videos, como lector de libros electrónicos, videojuegos, y tienen una variedad de aplicaciones de software.

1.4.1. HISTORIA.

Con el avance de la tecnología de pantallas planas, las interfaces de usuario, la miniaturización de los componentes de la computadora, trabajos experimentales relacionados con tecnología WIFI; se podía desarrollar un dispositivo que

contenga todos estos componentes en uno. Así nace el primer concepto de Tablet que es acreditado generalmente a Alan Kay de Xerox, que tuvo la idea en 1971.
En 1972 publicó un artículo sobre el dispositivo y lo llamo Dynabook (Figura 1.7), iba a ser una computadora muy similar a las tablets que tenemos hoy, con un par de excepciones. El Dynabook tenía una pantalla en la parte superior con cristal líquido, una tecnología completamente nueva en esa época, y un teclado en la parte inferior todos en el mismo plano. También podía reproducir archivos de audio, grabar notas de voz, y mucho más.

Figura. 1.7 Dynabook [10]

En 1989 se creó el GriDPad, que incluía una pantalla táctil capacitiva monocromática y un lápiz con cable. Pesaba menos de 2,3 kilogramos, en comparación con las tablets de hoy en día eran pesadas y voluminosas, la batería solo duraba tres horas. Después salieron al mercado tablets con lápiz, pero no hubo mucho interés en el público.
Realmente no fue hasta que la empresa Apple en el año 2010 presento el iPad, que las tablets se convirtieron en un éxito comercial. Hoy en día empresas como Google, Apple, Microsoft, Hp, Samsung, Sony, Toshiba, entre otras compañías, han incursionado en la producción de tablets, y siguen diseñando las nuevas generaciones de tablets, para satisfacer las necesidades de los clientes [10].

1.4.2. PARTES DE UNA TABLET.

Ya que una Tablet es un ordenador pequeño, muchas de las partes son similares a una PC. A continuación se va a describir las partes más importantes de una Tablet.

1.4.2.1. Procesador.

Es el cerebro de este dispositivo móvil, se encarga de procesar datos y de ejecutar programas. Las tablets utilizan un microprocesador de baja potencia, también conocido como unidad central de procesamiento (CPU). Un procesador de baja potencia permite que las tablets alcancen una duración de la batería de 8 a 10 horas. Los diferentes procesadores que existen para las tablets se los puede dividir en dos arquitecturas distintas, ARM o x86. ARM es la arquitectura que utilizan los sistemas operativos iOS y Android, y x86 la utilizan los sistemas operativos Linux y Windows.

1.4.2.2. Memoria RAM.

En dispositivos como Smartphones y tablets, esta memoria se encuentra integrada a la placa principal o soldada a ella, su tamaño es reducido y consume un menor voltaje. Las tablets por lo general tienen una memoria RAM de 1 a 2 gigabytes, son del tipo LPDDR que optimizan el consumo de energía. La memoria RAM ayuda a las tablets a cargar e intercambiar programas y archivos.

1.4.2.3. Disco duro.

Las tablets de gama alta trabajan normalmente con discos duros SSD (Figura 1.8), estos discos duros son un conjunto de memorias NAND, son más caros y permiten tener una tablet más delgada y ligera. También se utiliza para el almacenamiento tarjetas de memoria SD o microSD.

Figura 1.8 Tarjeta estado sólido (SSD) [11]

1.4.2.4. Pantalla.

La pantalla de una tablet va a servir como monitor y teclado a la vez, estas pantallas son de dos tipos de tecnología, las LCD (Liquid Crystal Display) y las OLED (Organic Light Emitting Diode).

1.4.2.4.1. Pantalla lcd.

Pantallas de cristal líquido, este tipo de pantallas se utilizan en relojes digitales y en varias computadoras portátiles. Las pantallas LCD utilizan dos hojas de materiales polarizados con una solución de cristal líquido entre ellas. Una corriente eléctrica que pasa a través del líquido hace que los cristales se alineen para que la luz no pueda pasar a través de ellos. Por lo tanto cada cristal permite que la luz pase o bloquee la luz [12].

1.4.2.4.2. Pantalla oled.

Es una tecnología de emisión de luz plana, que está compuesta mediante la colocación de una serie de películas delgadas orgánicas entre sus conductores. Cuando se aplica una corriente eléctrica a esta estructura, emite una luz brillante, ya que estas pantallas no necesitan retroalimentación pueden ser más delgadas y pesar menos que otras tecnologías, también tienen un ángulo de visión más amplio y solo utilizan de dos a diez voltios para funcionar [13].

1.4.2.5. Batería.

Otra parte importante para el funcionamiento de una tablet es la batería, la mayoría de dispositivos utilizan baterías de polímeros de Litio, y su capacidad se expresa en miliamperios por hora (mAh) o miliwatios por hora (mWh).

1.4.3. SISTEMAS OPERATIVOS PARA TABLETS.

El sistema operativo de las tablets debe ser simple, fácil de usar, ya que va permitir que todas las aplicaciones y programas funcionen correctamente. El sistema operativo viene preinstalado en un chip de memoria estática. Hay tres sistemas operativos principales que están disponibles para tablets.

1. **Apple iOS:** Es el Sistema operativo móvil que se ejecuta en los dispositivos móviles de Apple, es decir en iPhones y iPads. Es el principal software que le permite interactuar al usuario con el teléfono o tablet de Apple.
2. **Google Android:** El sistema operativo Android es una plataforma basada en Linux para teléfonos móviles y tablets. Las versión 3 de Android se ejecuta en tablets.

3. **Microsoft Windows:** Este sistema operativo es desarrollado por la empresa Microsoft. Windows 8 fue el primer sistema operativo diseñado para tablets.

1.5 HERRAMIENTAS DE DESARROLLO.

1.5.1. INTRODUCCIÓN.

Las herramientas de desarrollo de Android proporcionan un conjunto de aplicaciones que permite crear aplicaciones innovadoras para dispositivos móviles en un entorno de lenguaje Java.

1.5.2. ANDROID SDK

El SDK (kit de desarrollo de software) de Android es un conjunto de herramientas de desarrollo, mediante el cual podemos desarrollar aplicaciones y ejecutar un emulador de la versión de Android. Las aplicaciones se escriben utilizando el lenguaje de programación Java y se ejecutan en Dalvik, una máquina virtual personalizada. El SDK de Android incluye proyectos de ejemplo con código fuente, herramientas de desarrollo, herramientas de depuración, un emulador y bibliotecas necesarias para crear las aplicaciones.

Las plataformas de desarrollo compatibles con SDK pueden ser sistemas operativos como Windows (XP o posterior), Linux (cualquier distribución reciente) y Mac OS X (10.4.9 o posterior) [14].

1.5.3. ANDROID STUDIO.

Android Studio es un entorno de desarrollo integrado (IDE), que proporciona a los desarrolladores las herramientas necesarias para crear aplicaciones para la plataforma del sistema operativo Android, se basa en IntelliJ IDEA. Android Studio tiene funciones que mejoran la compilación de aplicaciones, algunas de ellas se muestran a continuación:

- Sistema de compilación flexible basado en Gradle[5].

[5] Gradle: Herramienta para automatizar el proceso de construcción de un proyecto.

- Un emulador rápido.
- Un entorno unificado en el cual se puede desarrollar aplicaciones para todos los dispositivos Android.
- Gran cantidad de herramientas y frameworks de prueba.
- Compatibilidad con C++ y NDK [15].

1.5.4. ESTRUCTURA DE UN PROYECTO.

En Android Studio se tiene un proyecto, y este a su vez contiene uno o más módulos que tienen archivos de código fuente y archivos de recursos. Los tipos de módulos son los siguientes:

- Módulo de aplicaciones para Android.
- Módulo de bibliotecas.
- Módulo de Google App Engine.

Los archivos de un proyecto en Android Studio se muestran en la vista de proyectos Android, como se muestra en la Figura 1.9, cada módulo de la aplicación contiene las siguientes carpetas:

- manifiestos: contiene el archivo *AndroidManifest.xml.*
- java: contiene los archivos de código fuente de Java.
- res: contiene todos los recursos, como diseños XML, cadenas IU e imágenes de mapa de bits [15].

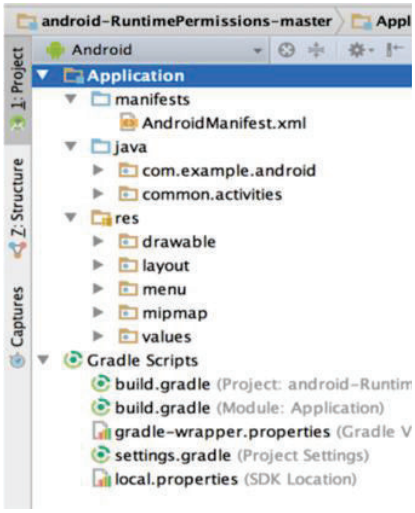


Figura 1.9 Archivos de Proyecto en la vista Android [15]

1.5.5. INTERFAZ DE USUARIO.

La interfaz de usuario en Android Studio tiene varias áreas lógicas, como se muestra en la Figura 1.10.

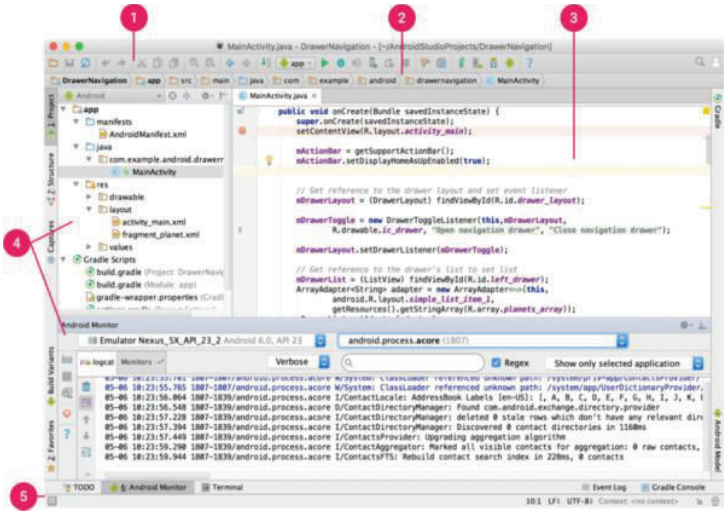


Figura 1.10 Ventana principal de Android Studio [15]

A continuación se describen cada una de las áreas lógicas de la ventana principal de Android Studio.

1. La barra de herramientas que permite realizar diferentes acciones, como la ejecución de la aplicación.
2. La barra de navegación que ayuda a explorar el proyecto y abrir archivos para editar.
3. La ventana del editor, es el área en la que se puede crear y modificar el código.
4. Las ventanas de herramientas, que permiten acceder a tareas específicas, como la búsqueda de proyectos, la administración, etc.
5. La barra de estado, muestra el estado del proyecto y el IDE, además de advertencias o mensajes [15].

1.5.6. SISTEMA DE COMPILACIÓN.

Gradle es la base en el sistema de compilación de Android Studio. Este sistema de compilación está basado en JVM (Java Virtual Machine), lo que significa que

se puede escribir un script en Java y Android Studio lo entenderá y lo usará. Los archivos de compilación que se generan en Android Studio son *19ongi.gradle*, estos archivos son de texto sin formato que tienen una sintaxis *Groovy*[6].

1.5.7. HERRAMIENTAS DE DEPURACIÓN.

Android Studio tiene integrada herramientas de depuración que sirven para las revisiones de código en la vista del depurador con verificación integrada de referencias, expresiones y valores de variables. La información de depuración integrada incluye:

- Valores de variables integradas.
- Valores de retorno de métodos.
- Valores de información sobre herramientas.
- Objetos que hacen referencia a un objeto seleccionado.

1.5.8. ECLIPSE IDE.

Eclipse es una plataforma de código abierto que ha sido diseñada para la construcción de herramientas integradas de desarrollo web y de aplicaciones basada en un modelo que emplea módulos (plugin). Eclipse proporciona un modelo común de interfaz de usuario (UI), está diseñado para ejecutarse en varios sistemas operativos. Los plugins pueden programarse en las API portátiles de Eclipse y ejecutarse en cualquier sistema operativo compatible, cada plugin puede realizar una tarea, como animar, publicar, compilar, depurar, diagramar, etc [16].

1.5.9. ESTRUCTURA DE LA PLATAFORMA DE ECLIPSE.

La plataforma Eclipse está estructurada como subsistemas que se implementan en uno o más plugins. La Figura 1.11 muestra una vista de la estructura de la plataforma.

[6] Groovy: Es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java.

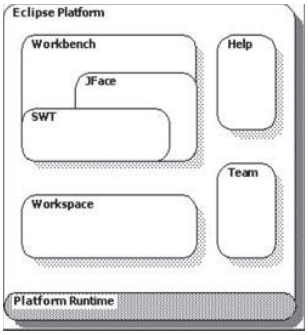


Figura 1.11 Plataforma de eclipse [17]

Los complementos que forman un subsistema definen puntos de extensión para agregar comportamiento a la plataforma. A continuación se describen los principales componentes de la plataforma.

- **Tiempo de ejecución de la plataforma**: Define el punto de extensión y el modelo de complemento. Detecta de forma dinámica los complementos y mantiene información sobre complementos y sus puntos de extensión en un registro de la plataforma, Los plugins se ponen en funcionamiento cuando el usuario de la plataforma lo requiere. El tiempo de ejecución se implementa utilizando el framework[7] OSGi[8].
- **Gestión de Recursos (workspace):** Define el API para crear y administrar recursos (proyectos, archivos y carpetas) producido por herramientas y se mantiene en el sistema de archivos.
- **Workbench UI:** Implementa la interfaz gráfica para que el usuario pueda navegar por la plataforma. Define puntos de extensión para agregar componentes de interfaz de usuario, como vistas o acciones de menú. Proporciona kits de herramientas adicionales (Jface[9] y SWT[10]) para la creación de interfaces de usuario.
- **Help:** Se define los puntos de extensión de los complementos para proporcionar ayuda.

[7] Framework: Es un entorno o ambiente de trabajo para desarrollo; dependiendo del lenguaje normalmente integra componentes que facilitan el desarrollo de aplicaciones.
[8] OSGi: (Open Services Gateway Initiative) Sistema modular para Java que establece las formas de crear módulos y la manera que estos interactúan entre sí en tiempo de ejecución.
[9] JFace: Es un conjunto de widgets para realizar interfaces de usuario construido sobre SWT.
[10] SWT: (Standard Widget Toolkit) es un conjunto de componentes para construir interfaces gráficas en Java.

- **Team:** Define un modelo de programación de equipo para administrar y gestionar recursos.
- **Otras Utilidades:** Plugins que proveen funcionalidad como buscar y comparar recursos, realizar compilaciones usando archivos de configuración XML y actualizar dinámicamente la plataforma desde de un servidor.

1.6 SISTEMA OPERATIVO ANDROID.

Un sistema operativo(OS) es el programa que, después de ser inicialmente cargado en un equipo por un programa de arranque, gestiona todos los otros programas que existen en el equipo. Los otros programas se llaman aplicaciones, las aplicaciones hacen uso del sistema operativo haciendo peticiones de servicio a través de una interfaz de programa de aplicación definida (API). Además, los usuarios pueden interactuar directamente con el sistema operativo a través de una interfaz de usuario con líneas de comandos o una interfaz gráfica de usuario (GUI). Un sistema operativo realiza estos servicios para aplicaciones:

- En un sistema operativo multitarea en el que se pueden ejecutar varios programas al mismo tiempo, él sistema operativo determina que aplicaciones deben ejecutarse en qué orden y cuánto tiempo debe permitirse para cada aplicación, antes de pasar a otra.
- Gestiona el intercambio de memoria interna entre múltiples aplicaciones.
- Maneja la entrada y salida hacia y desde dispositivos de hardware conectados como discos duros, impresoras, puertos USB.
- Envía mensajes a cada aplicación o usuario interactivo sobre el estado de la operación y cualquier error que pueda haber ocurrido.
- Todas las plataformas informáticas requieren un sistema operativo y deben ser desarrollados con diferentes características para cumplir con los requisitos [18].

1.6.1. INTRODUCCIÓN.

El sistema operativo Android es una plataforma basada en Linux, básicamente es un sistema operativo para móviles, desarrollado por Google y más tarde por la Open Handset Alliance (OHA), permite a los desarrolladores escribir código en lenguaje Java. Android está diseñado principalmente para dispositivos móviles con pantalla táctil. Los dispositivos son principalmente Smartphones y tablets, pero ahora también se los está utilizando en televisores, automóviles, relojes etc. El sistema operativo Android se compone de numerosas aplicaciones Java orientadas a objetos y bibliotecas de núcleo Java que se ejecutan en la máquina virtual Dalvik.

Dalvik es la máquina virtual (VM) propia de Android y fue diseñada específicamente para dispositivos móviles, ya que estos sistemas están limitados en términos de velocidad y procesamiento.

1.6.2. API (INTERFAZ DE PROGRAMACIÓN DE APLICACIONES).

La interfaz de programación de aplicaciones (API) es un conjunto de rutinas, protocolos y herramientas para crear aplicaciones de software. API es una especificación precisa escrita por los proveedores de un servicio que los desarrolladores deben seguir al utilizar este servicio. Por ejemplo una API popular es la API de Google Maps, la API de Google Maps permiten a los desarrolladores integrar Google Maps en la páginas Web mediante una interfaz JavaScript o Flash. La API de Google Maps está diseñada para trabajar con dispositivos móviles.

1.6.2.1. Nivel de API de Android.

Nivel de API es el número dado a la revisión de la Framework API para su identificación única.
Android proporciona una Framework API que las aplicaciones pueden usar para interactuar con el sistema subyacente de Android. La Framework API consiste en lo siguiente:

- Un conjunto básico de clases y paquetes.

- Un conjunto de atributos y elementos XML para declarar un archivo de manifiesto.
- Un conjunto de atributos y elementos XML para declarar recursos y acceder a estos.
- Un conjunto de ítems.
- Un conjunto de permisos que las aplicaciones pueden solicitar, al igual que cumplimientos de permisos incluidos en el sistema.

Las actualizaciones a la API están diseñadas de tal manera que la nueva API siga siendo compatible con las versiones anteriores de API.

El nivel de API en Android es identificado por un valor entero. En la siguiente tabla se muestra el nivel de API compatible con las ultimas versiones de la plataforma Android [20].

Versión de la plataforma	Nivel de API	VERSION_CODE
Android 7.0	24	N
Android 6.0	23	M
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP
Android 4.4W	20	KITKAT_WATCH
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2 y 4.2.2	17	JELLY_BEAN_MR1
Android 4.1 y 4.1.1	16	JELLY_BEAN

Tabla 1.3 Nivel de API en Android [20]

1.6.3. ARQUITECTURA DE ANDROID.

La arquitectura Android está compuesta por cinco capas:

- Aplicación.
- Aplicación Framework
- Bibliotecas
- Android Runtime

- Kernel[11] de Linux [21].

A continuación se observa en la figura 1.12 la arquitectura Android.

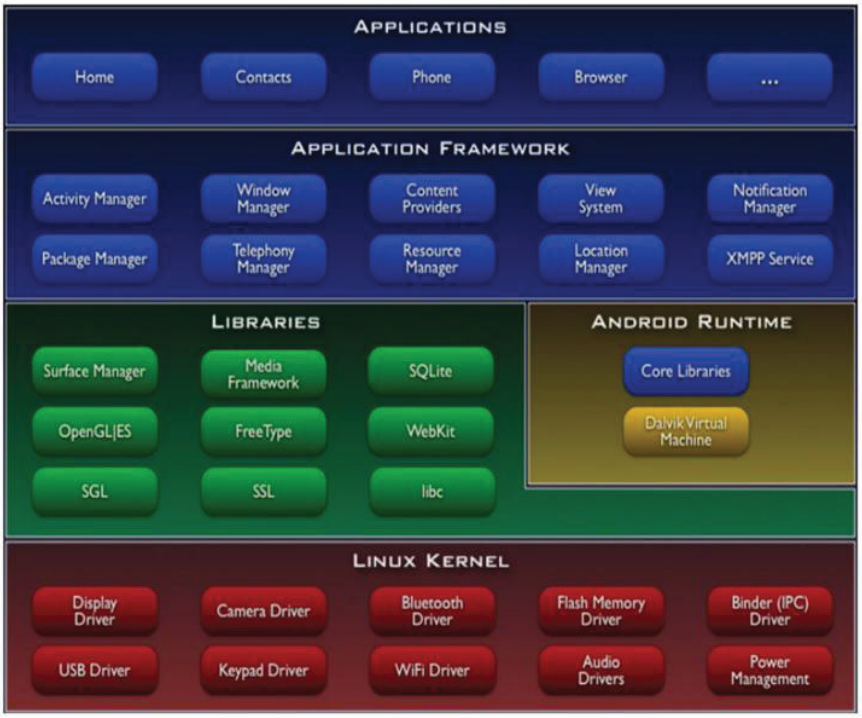


Figura 1.12 Arquitectura Android [22]

1.6.3.1. Linux Kernel.

Como se muestra en la Figura 1.12 Android se construye sobre el Kernel de Linux, como en cualquier sistema Unix[12], el Kernel proporciona controladores para el hardware, la red, el acceso al sistema de archivos y la gestión de procesos.
Cuando el software necesita del hardware para realizar algo, envía una petición al núcleo, y cuando decimos algo, nos referimos a cualquier cosa, desde el brillo de la pantalla, hasta el nivel del volumen, hasta el inicio de una llamada a través de la radio es controlado en última instancia por el Kernel, esto se realiza ya que existen controladores en el Kernel que son utilizado por el software para realizar alguna de las tareas mencionadas anteriormente. Tanto el hardware como el software se comunican en ambos sentidos con el Kernel.

[11] Kernel: Nucleo de Linux, el encargado de que el software y el hardware puedan trabajar juntos.
[12] Unix: Sistema operativo que admite múltiples usuarios, así como múltiples tareas.

1.6.3.2. Dalvik (VM)

La mayor parte de Android se implementa en Java, y como tal se ejecuta en una máquina virtual Java (JVM). Dalvik fue diseñado para dispositivos móviles y no se puede ejecutar bytecode[13] de Java (archivos .class[14]) directamente, el formato de entrada nativo se llama Dalvik Executable (DEX) y está en formato .dex. A su vez los archivos .dex se envían dentro de las bibliotecas Java del sistema (archivos JAR) o dentro de las aplicaciones de Android (archivos APK).

En esencia los archivos .class se convierten en archivos .dex (muy parecido a un archivo JAR si se usaba la VM de Java) y luego se leen y se ejecutan en la máquina virtual Dalvik. También se reemplazó la VM de Java con la VM de Dalvik por la licencia, mientras que el lenguaje Java, las herramientas, las bibliotecas son gratuitas, la máquina virtual Java no lo es.

1.6.3.3. Bibliotecas Nativas.

Las bibliotecas nativas son bibliotecas escritas en C / C++, a menudo tomadas de la comunidad de código abierto, para proporcionar los servicios necesarios a la capa Aplicación de Android.

Entre las principales librerías se incluyen las siguientes:

Webkit: Un motor web rápido utilizado por Safari, Chrome y otros navegadores.

SQLite: Una base de datos SQL completa.

OpenGL: Bibliotecas de gráficos 3D,

OpenSSL: Capa de comunicaciones seguras.

FreeType: Trabaja con diferentes tipos de fuente.

Media Libraries: Maneja los códec para el contenido multimedia que soporta Android.

LIBC: Incluye todas las cabeceras y funciones según el estándar del lenguaje C.

Surface Mananger: es la encargada de componer los diferentes superficies de dibujo en la pantalla [19].

---

[13] Bytecode: código binario de Java.

[14] .class: archivo binario que se obtiene luego de compilar un archivo .java

1.6.3.4. Aplicación Framework

La capa Aplicación Framework proporciona numerosos servicios para ayudar al desarrollador de aplicaciones a realizar su trabajo. En la capa Aplicación framework se encontrara numerosas bibliotecas Java específicamente diseñadas para Android.

Se puede encontrar servicios para el acceso a base de datos y archivos, también incluye servicios que permiten interactuar con el hardware del dispositivo, así como servicios que aprovechan los servicios de nivel superior que ofrece el sistema.

El framework de Android incluye los siguientes servicios:

- **Administrador de actividad:** Controla todos los aspectos del ciclo de vida de la aplicación y de la pila de actividades.
- **Proveedores de contenido:** Permite que las aplicaciones publiquen y compartan datos con otras aplicaciones.
- **Administrador de recursos:** Proporciona acceso a recursos no codificados como cadenas de texto, configuraciones de color y diseños de interfaz de usuario.
- **Administrador de notificaciones:** Permite que las aplicaciones muestren alertas y notificaciones al usuario.
- **Sistema de vista:** Este sistema es utilizado para crear interfaces de usuario de aplicaciones [23].

1.6.3.5. Aplicaciones.

En el nivel más alto de la arquitectura de Android están las aplicaciones, que son los programas con los que los usuarios interactúan directamente. Un usuario interactúa con esta capa para funciones básicas como hacer llamadas telefónicas, acceder al navegador Web, etc. A las capas inferiores acceden principalmente los desarrolladores y programadores. Varias aplicaciones estándar vienen preinstaladas en todos los dispositivos, como las siguientes:

- Dialer
- Web Browser
- SMS client app

1.6.4. CARACTERISTICAS DEL SISTEMA OPERATIVO ANDROID.

Android es un sistema operativo potente y flexible que compite con el iOS de Apple o Windows 8.1. Algunas de sus características se muestran a continuación:

CARACTERISTICAS	DESCRIPCION
Interfaz de usuario.	El sistema operativo Android proporciona una interfaz intuitiva y amigable con el usuario.
Conectividad	CDMA, UMTS, GSM / EDGE, LTE, EV-DO, Wi-Fi, NFC, Bluetooth, IDEN y WiMAX.
Almacenamiento	Android utiliza una base de datos relacional ligera llamada SQLite para el almacenamiento de datos.
Soporte multimedia	H.263, H.264, AMR, AMR-WB, MPEG-4 SP, AAC 5.1, AAC, MIDI, GIF, WAV, , PNG, MP3, BMP y JPEG.
Mensajería	MMS y SMS
Navegador Web	El navegador Web de Android se basa en el motor de búsqueda Webkit y el motor de JavaScript V8 de Chrome, que son compatibles con CSS3 y HTML5.
Multi-touch	Multi-touch es soportado nativamente por Android para pantallas capacitivas.
Multitarea	Los usuarios pueden navegar de una aplicación a otra, y se puede ejecutar varias aplicaciones al mismo tiempo.
Widgets redimensionables	Los widgets predeterminados y los que se descarguen se pueden cambiar de tamaño. Pueden hacerse más pequeños para ahorrar espacio.
Multi-Lenguaje	Android soporta diferentes idiomas.
Android Beam	Se basa en la tecnología de comunicación de campo cercano, los usuarios pueden compartir datos instantáneamente acercando los dispositivos que tienen habilitado NFC en ellos.
Google Play	Es una tienda virtual de aplicaciones para Android, algunas son gratis y otras tienen un costo.

Tabla 1.4 Características del OS Android [24]

1.6.5. VERSIONES DEL SISTEMA OPERATIVO ANDROID [25]

Android se actualiza día a día desde su lanzamiento. Estas actualizaciones del sistema operativo se centran principalmente en el control de errores, así como la adición de nuevas características para proporcionar un entorno más cómodo.

Generalmente cada nueva versión del sistema operativo Android se desarrolla bajo un nombre de código basado en un elemento de postres.

A continuación se describen las diferentes versiones del sistema operativo Android, a partir de la versión 4.1 y los postres asociados a ellos.

1.6.5.1. Android 4.1: Jelly Bean.

Jelly Bean (Figura 1.13) salió en el 2012, a continuación los cambios más importantes en la versión Android 4.1:

- Búsqueda por voz
- Mejoras en la aplicación de cámara.
- Gesture Typing, para escribir más rápido.
- Multiusuario para tablets.
- Google Now.
- Notificaciones accionables.
- Mejoras de seguridad y rendimiento.
- Resolución de 4K.
- Capacidad para crear perfiles [32].

Figura 1.13 Android 4.1 **Jelly Bean** [32]

1.6.5.2.Android 4.4: KitKat.

Google anuncio que Android 4.4 se llamaría KitKat (Figura 1.14) el 3 de septiembre del 2013. A continuación los cambios más importantes:

- Grabación de pantalla.
- Nueva interfaz de usuario del sistema translucido.
- Acceso mejorado a las notificaciones. Sensorr
- Configuración de todo el sistema para subtítulos.
- Mejoras en el rendimiento.
- Corrección de errores
- Mejora en la cámara en el Nexus 5 [33].

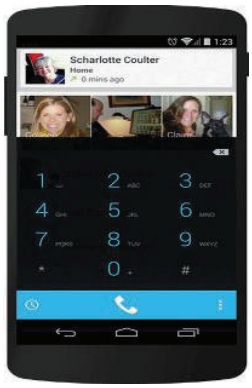


Figura 1.14 Android 4.4 KitKat [33]

1.6.5.3.Android 5.0: Lollipop.

En de 2014 se presenta Android 5.0 y se llama Lollipop (Figura 1.15), y presenta las siguientes mejoras:

- Multipantalla, se puede cambiar fácilmente de un teléfono u una tablet, de un reloj Android a una Tv Android.
- Nuevo diseño (diseño material).
- Mejora de velocidad.
- Mejora del consumo de batería.
- Notificaciones, las notificaciones aparecen en la pantalla de bloqueo [34].

Figura 1.15 Android 5.0 Lollipop [34]

1.6.5.4. Android 6.0: Marshmallow.

Desde octubre del 2015 Android 6.0 ya está disponible para los dispositivos Nexus (Figura 1.16).

Se detalla las siguientes mejoras:

- Soporta USB tipo C.
- Soporte de autenticación de huellas dactilares.
- Mejor duración de batería con “sueño profundo”.
- Panel de control de permisos.
- Android Pay.
- Soporte MIDI[15].
- Mejoras en Google Now [35].

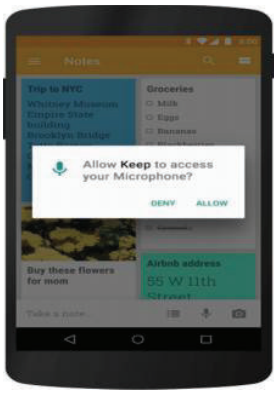


Figura 1.16 Android 6.0 Marshmallow [35]

[15] MIDI: es un estándar tecnológico que describe un protocolo, una interfaz digital y conectores que permiten que varios instrumentos musicales electrónicos, computadoras y otros dispositivos relacionados se conecten y comuniquen entre sí.

1.6.5.5. Android 7.0: Nougat.

Android 7.0 Nougat (Figura 1.17) se lanzó el 22 de agosto de 2016, y tiene las siguientes características:

- Unicode 9.0 emoji.
- Mejor multitarea.
- Modo multi-ventana.
- Mejor rendimiento.
- Modo realidad virtual.
- Mejoras en el gestor de almacenamiento.
- Función multiventana nativa para ejecutar dos apps a la vez en la misma pantalla
- Mayor personalización y configuraciones para los atajos rápidos.
- Forma más inteligente de bloquear llamadas entrantes [36].

Figura 1.17 Android 7.0 Nougat [36]

CAPÍTULO 2.

DISEÑO E IMPLEMETACIÓN DE LA APLICACIÓN.

2.1 LENGUAJE DE PROGRAMACIÓN JAVA.

2.1.1. INTRODUCCIÓN.

Java es un lenguaje de programación orientado a objetos, Java tiene su propia estructura, reglas de sintaxis, y en muchos aspectos es similar a C++ [37].

2.1.2. KIT DE DESARROLLO DE JAVA (JDK).

El JDK es un entorno de desarrollo de software utilizado para desarrollar aplicaciones y applets Java. Incluye el entorno de ejecución de Java (JRE), un intérprete (java), un compilador (javac), un archivo (jar), un generador de documentos (javadoc) y otras herramientas necesarias en el desarrollo de Java.

2.1.3. ENTORNO DE EJECUCIÓN DE JAVA (JRE).

El entorno de ejecución de Java (JRE) es parte del kit de desarrollo de Java (JDK). Contiene un conjunto de librerías y herramientas para desarrollar aplicaciones en Java. El JRE proporciona los requisitos mínimos para ejecutar una aplicación.

2.1.4. MÁQUINA VIRTUAL DE JAVA (JVM).

La máquina virtual de Java (JVM) es el entorno en el que se ejecutan los programas Java. Es un software que se implementa en la parte superior del hardware del sistema operativo. Cuando se compila el código fuente (archivos .java), se traduce en bytecode y luego se coloca en archivos (.class) la JVM ejecuta estos bytecode. Por lo tanto los bytecode de Java pueden ser considerados como el lenguaje de máquina de la JVM. La JVM se debe implementar en una plataforma determinada antes de que los programas compilados se ejecuten en esa plataforma.

2.1.5. GARBAGE COLLECTOR.

Garbage Collector es el sistema del programa Java que se ejecuta en segundo plano junto con un programa regular de Java para recopilar un espacio de memoria no utilizado para mejor el rendimiento de nuestras aplicaciones.

2.1.6. LIBRERÍAS DE JAVA.

Hay dos tipos de librerías en Java, la librería principal que forma parte del kit de desarrollo de Java (JDK) y las librerías adicionales, que son opcionales. Las librerías principales deben estar presentes en cada programa de Java, mientras que las opcionales pueden estar presentes o no. La tabla 2.1 se enumera las librerías principales de Java [38].

NOMBRE	DESCRIPCIÓN
java.lang	Soporte para el lenguaje de Java.
java. Applet	Soporte de applets[16].
Java.awt	Soporte de GUI.
Javax.swing	Suplemento de java.awt y mejora el soporte de GUI.
Java.io	Soporte de entradas y salidas.
Java.util	Soporte estructura de datos.
Java.rmi	Soporte de llamadas de métodos.
Java.sql	Soporte base de datos.
Java.security	Soporte codificación y decodificación de datos.
Java.net	Soporte TCP/IP, UDP, IP y otras redes.
Java.beans	Soporte de software de componentes.
Java.text	Soporte para elementos de texto
java.math	Soporte para decimales y tipos de números
java.accessibility	Soporte para tamaños de texto grande.

Tabla 2.1 Librerías principales de Java [38].

[16] Applet: Es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo en un navegador Web.

2.1.7. CARACTERÍSTICAS PRINCIPALES DE JAVA.

2.1.7.1. Plataforma independiente.

Una plataforma es el entorno de hardware o software en el que se ejecuta un programa. La plataforma de Java se diferencia de la mayoría de las otras plataformas en el sentido de que es una plataforma basada en software que se ejecuta en la parte superior de otras plataformas basadas en hardware. Java se puede ejecutar en todos los sistemas operativos con respecto a su desarrollo y compilación (Figura 2.1).

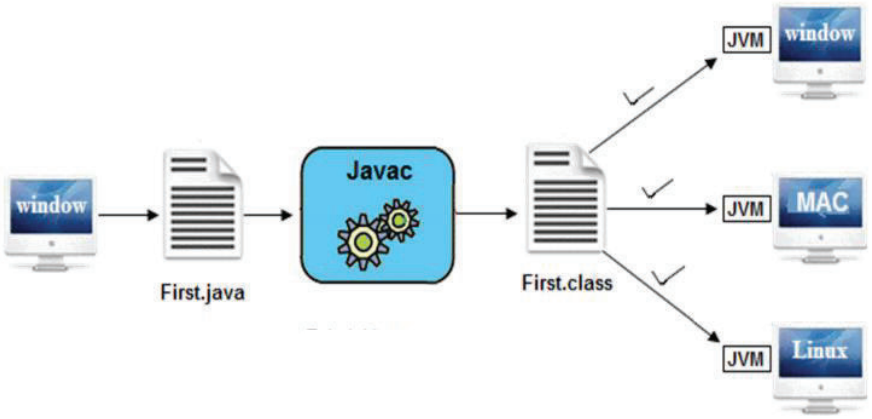


Figura 2.1 Plataforma independiente [39].

2.1.7.2. Arquitectura neutral.

Un lenguaje o tecnología se dice que tiene arquitectura neutral cuando se puede ejecutar en cualquier procesador disponible en el mercado, sin tener en cuenta la marca, la arquitectura del proveedor. Java cumple con estos requisitos y se puede ejecutar en cualquier procesador.

2.1.7.3. Portable.

Si un lenguaje tiene una plataforma independiente y arquitectura neutral, se puede decir que estas características son de un lenguaje portátil. Java es un lenguaje portátil según SUN microsystem.

2.1.7.4. Multihilo.

Un hilo es como un programa aparte, ejecutándose simultáneamente. Podemos escribir programas en Java que se ocupen de muchas tareas a la vez mediante la definición de múltiples subprocesos. La ventaja principal de multi-threading es que no ocupa memoria para cada hilo, comparte un área de memoria común. Los hilos son importantes para aplicaciones multimedia, Web, etc.

2.1.7.5. Distribuido.

Usando el lenguaje Java se puede crear aplicaciones distribuidas. Se puede acceder a los archivos a través de la red llamando a los métodos de cualquier máquina de internet.

2.1.7.6. Robusto.

Es un lenguaje de programación robusto o fuerte debido a su capacidad para manejar errores en tiempo de ejecución, recolección automática de basura, manejo de excepciones. Todos estos puntos hacen robusto a Java.

2.1.7.7. Dinámico.

Es compatible con la asignación de memoria dinámica y se reduce el desperdicio de memoria, lo que mejora el rendimiento de la aplicación. El proceso de asignar el espacio de memoria a la entrada del programa en un tiempo de ejecución se conoce como asignación de memoria dinámica.

2.1.7.8. Seguro.

Es un lenguaje más seguro comparado con otros lenguajes. En este lenguaje todo el código está cubierto en código de bytes después de la compilación, lo que no es legible para el ser humano.

2.1.7.9. Orientado a objetos.

Orientado a objetos significa que organizamos nuestro software como una combinación de diferentes tipos de objetos que incorporan datos y comportamiento. La programación orientada a objetos es una metodología que

simplifica el desarrollo y mantenimiento de software proporcionando algunas reglas. Los conceptos básicos de POO son:

- Objeto
- Clase
- Herencia
- Polimorfismo
- Abstracción
- Encapsulamiento

2.1.7.10. Alto Rendimiento.

Tiene alto rendimiento debido a las siguientes razones:

- Este lenguaje utiliza Bytecode que es más rápido que el código de puntero normal, por lo que el rendimiento de este lenguaje es alto.
- El recolector de basura, recopila el espacio de memoria no utilizado y mejora el rendimiento de la aplicación.
- No tiene punteros lo que conlleva a que se pueda desarrollar una aplicación muy fácilmente.

2.2 INSTALACIÓN Y CONFIGURACIÓN DE ANDROID STUDIO.

2.2.1. REQUERIMIENTOS DEL SISTEMA PARA INSTALAR ANDROID STUDIO EN WINDOWS [40].

A continuación se describen los requerimientos para instalar Android Studio en el sistema operativo Windows, ya que en este sistema operativo se desarrollara la aplicación desde el entorno de Android Studio.

- Microsoft® Windows® 7/8/10 (32 o 64 bits).
- 3 GB de memoria RAM como mínimo; se recomiendan 8.

- 2 GB de espacio mínimo disponible en el disco; se recomiendan 4 (500 MB para IDE + 1,5 GB para el Android SDK y la imagen de sistema del emulador).
- Resolución de pantalla mínima de 1280 x 800.
- Para el emulador acelerado: Sistema operativo de 64 bits y procesador Intel® compatible con Intel® VT-x, Intel® EM64T (Intel® 64) y la funcionalidad Execute Disable (XD) Bit.

2.2.2. INSTALACÓN DE JAVA DEVELOPMENT KIT.

Android Studio utiliza una cadena de herramientas Java para su funcionamiento, primero hay que instalar el compilador de Java y la máquina virtual, en la Figura 2.2 se muestra la página de descarga. Se puede descargar el JDK del siguiente link: http://www.oracle.com/technetwork/es/java/javase/downloads/index.html.

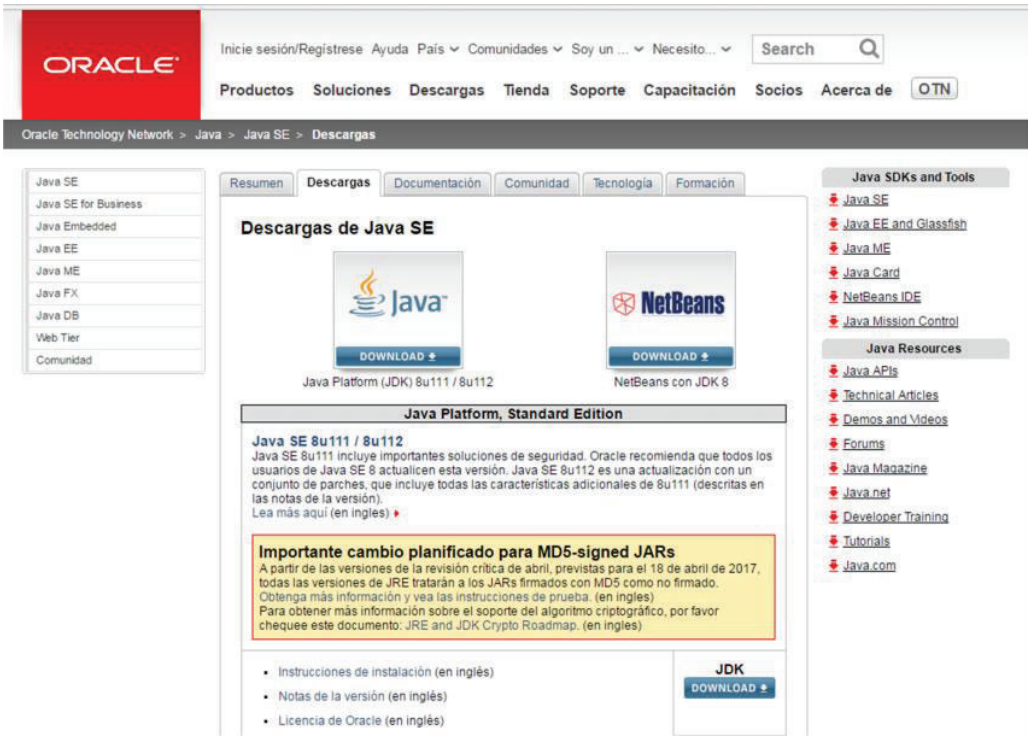


Figura 2.2 Página de descarga de Java JDK.

Hay que tomar en cuenta que hay que configurar la variable de entorno JAVA_HOME de Windows para que el JDK sea encontrado por Android Studio, como se muestra en la Figura 2.3.

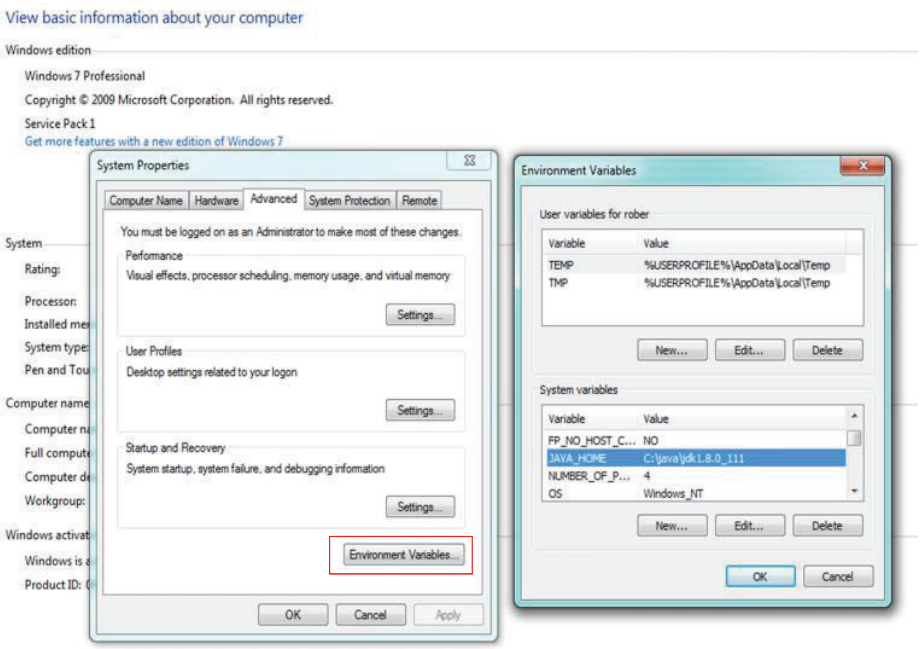


Figura 2.3 Variable de entorno JAVA_HOME.

2.2.3. INSTALACIÓN DE ANDROID STUDIO.

Procedemos a descargar Android Studio del siguiente link: https://developer.android.com/studio/index.html?hl=es-419

A continuación en la Figura 2.4 se observa la página de descarga de Android Studio.

Figura 2.4 Link de descarga de Android Studio

Una vez que hemos realizado la descarga del programa, procedemos a la instalación, seguimos todos los pasos que el asistente de instalación por defecto nos proporciona (Figura 2.5).

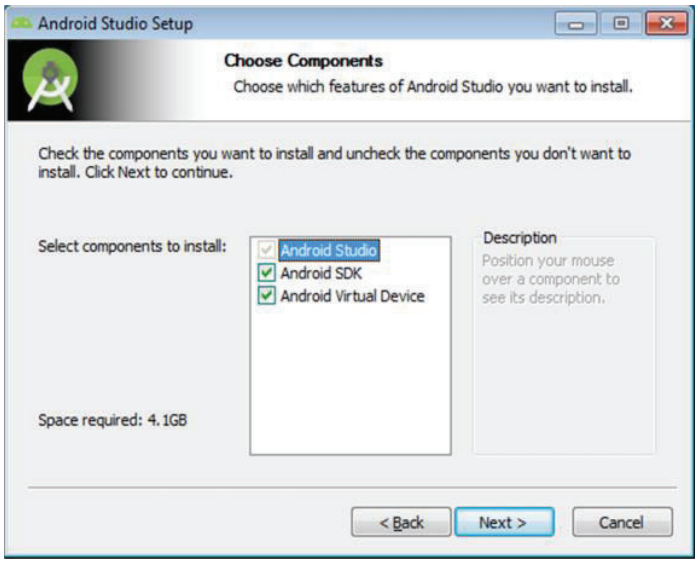


Figura 2.5 Asistente de instalación

2.2.4. CREACIÓN DE UN PROYECTO NUEVO EN ANDROID STUDIO.

En el asistente para crear un nuevo proyecto en el primer cuadro de dialogo, nos va a pedir que ingresemos el nombre de la aplicación, el nombre de dominio de la empresa, y la ubicación de nuestro proyecto en el disco (Figura 2.6).

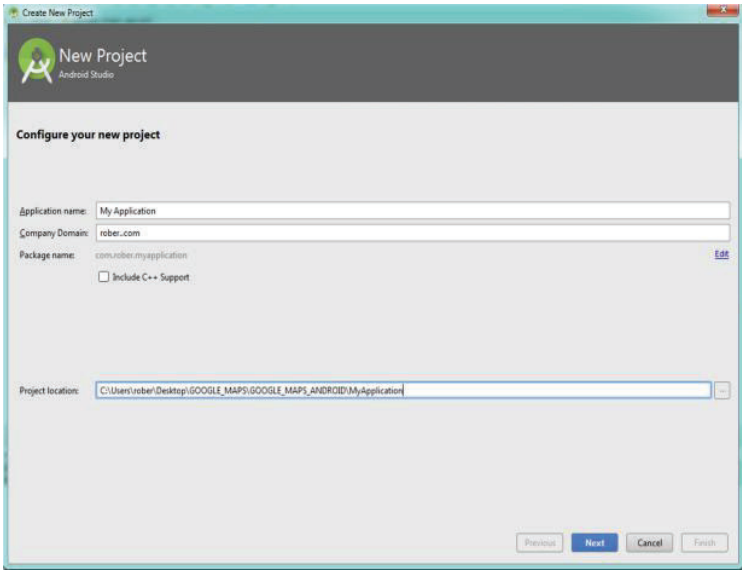


Figura 2.6 Configuración de un nuevo proyecto.

El sistema operativo Android puede ejecutarse en muchas plataformas incluyendo televisores, relojes, gafas, Smartphones, tablets. Para nuestra aplicación utilizaremos la plataforma de teléfono y tablet, como se muestra en la Figura 2.7; se seleccionara la versión mínima de Android en donde se desarrollara nuestra aplicación.

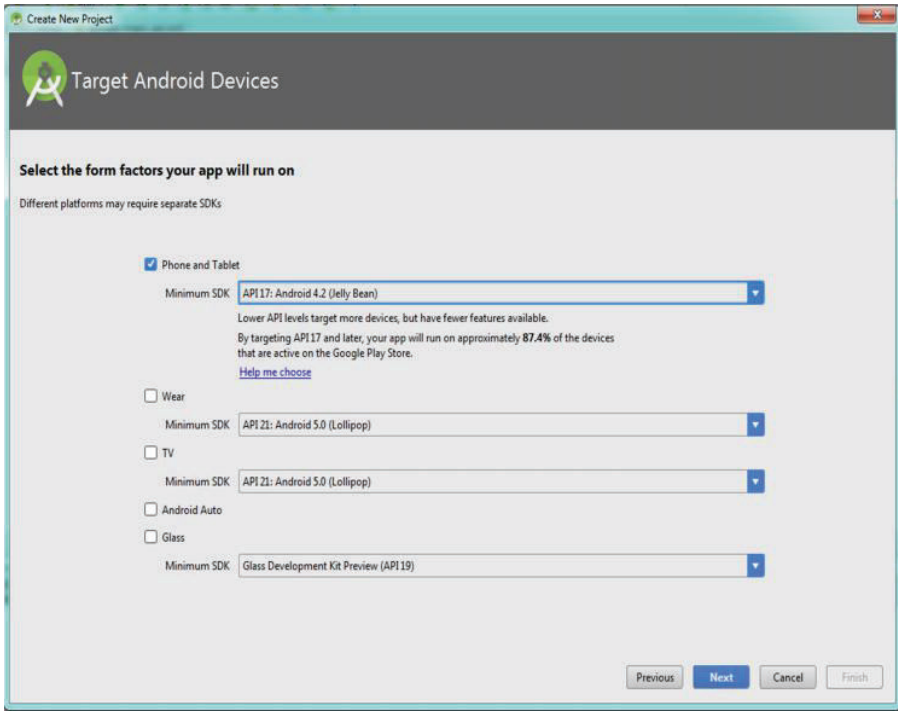


Figura 2.7 Selección del nivel mínimo de API.

Nota: Hay que tomar en cuenta que para el desarrollo de nuestra aplicación en Android Studio, se tiene que trabajar con un nivel mínimo API 17 y con la versión Android 4.2 o versiones superiores.

En la ventana que se muestra en la Figura 2.8 se selecciona la actividad que vamos a realizar en nuestra aplicación, cuando se selecciona la actividad se va a generar un código básico para la aplicación, en nuestro caso se seleccionó la actividad Google Maps Activity, ya que la aplicación que se desarrollara se basa en coordenadas GPS.

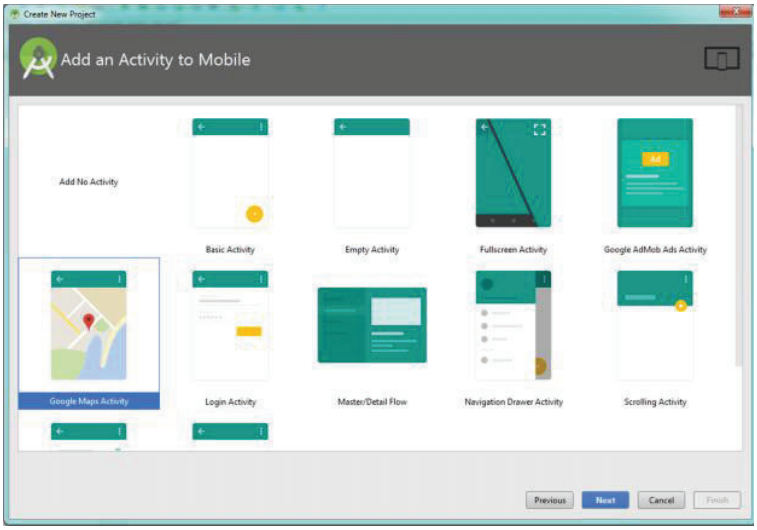


Figura 2.8 Selección de la actividad.

Finalmente la pantalla siguiente en el asistente para crear un nuevo proyecto se tiene que indicar el nombre de la actividad, se dejara los nombres predeterminados como se muestra en la Figura 2.9

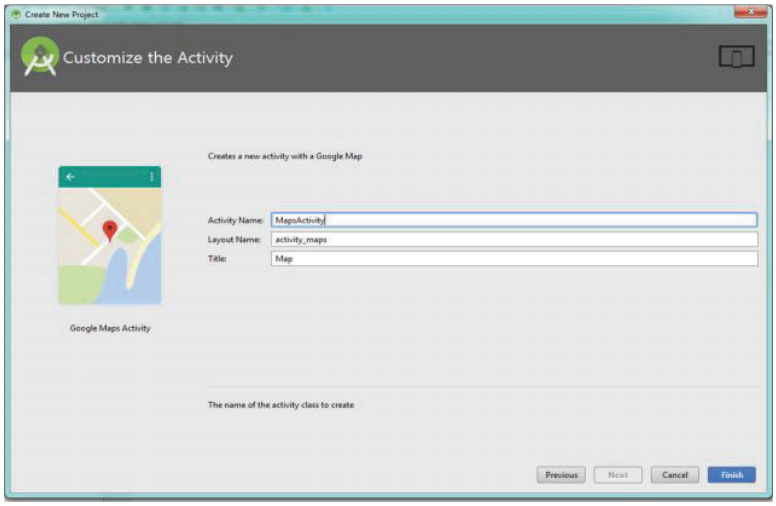


Figura 2.9 Nombre de la actividad.

2.3 FICHEROS Y CARPETAS EN ANDROID STUDIO [41].

En Android Studio se tiene varios módulos, el módulo llamado app es el que se va a utilizar para desarrollar la aplicación. Este módulo app está formado por el manifiesto de la app (manifests), el código fuente en Java, ficheros con recursos (res) y ficheros para la construcción del módulo (Gradle Scripts).

AndroidManifest.xml: Es un archivo XML que proporciona información sobre la aplicación en Android. Describe los componentes de la aplicación como los servicios, las actividades. También se declaran los permisos que deben tener la aplicación para acceder a las partes protegidas de una API e interactuar con otras aplicaciones (Figura 2.10).

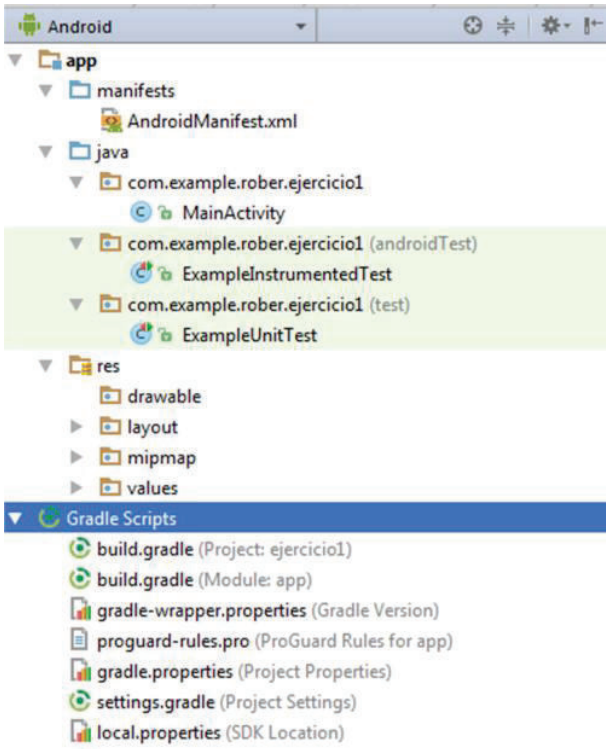


Figura 2.10 AndroidManifest.xml [42]

Java: Esta carpeta contiene el código fuente de la aplicación, los ficheros Java se almacenan en carpetas según el nombre del paquete:

MainActivity: código de la actividad principal, código Java [42].

AplicationTest: clase Java para insertar código de testeo de la aplicación utilizando el API Junit (Figura 2.11).

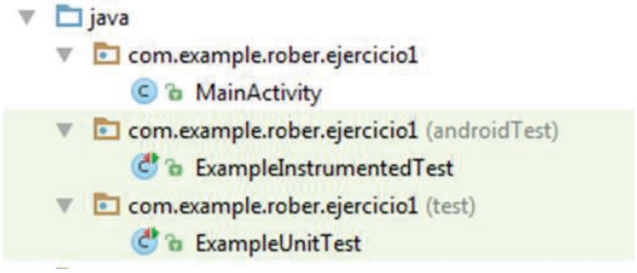


Figura 2.11 Carpeta java.[42]

res: Carpeta que contiene los recursos usados por la aplicación [42].

drawable: en esta carpeta se almacena los ficheros de imágenes(JPG o PNG) y descriptores de imágenes en XML.

Layout: contiene ficheros XML con vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que comprenderán la interfaz de usuario de la aplicación.

Mipmap: Es una carpeta con la misma finlidad de res/drawable.

Values: es usado para indicar valores en la aplicación, como strings, color, styles y dimens (Figura 2.12).

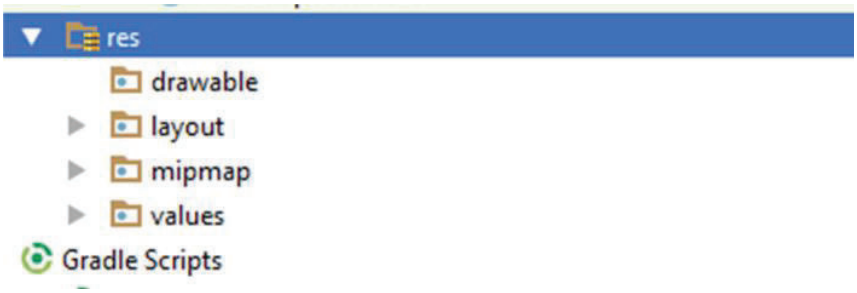


Figura 2.12 Carpeta res [42].

Gradle Scripts: en esta carpeta se almacenan una serie de ficheros Gradle que permiten compilar y construir la aplicación (Figura 2.13).

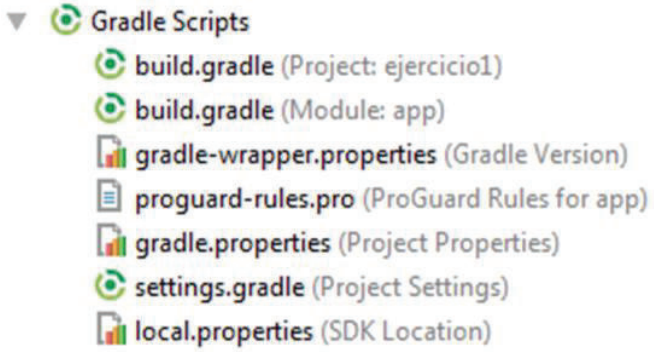


Figura 2.13 Carpeta Gradle Scripts [42].

2.4 DESARROLLO DE LA APLICACIÓN.

Esta aplicación permitirá realizar el cobro de pasajes en buses interprovinciales mediante el uso de coordenadas GPS.

2.4.1. REQUERIMIENTOS DE LA APLICACIÓN.

Los requisitos que debe cumplir la aplicación son los siguientes:

- Transmisión de los datos por medio de la red celular o Wifi.
- Verificación de la posición por medio de las coordenadas GPS.
- Verificación de las paradas principales a lo largo de la ruta.
- Ingreso del número de parada, para realizar el respectivo cobro del pasaje.
- Verificación del cobro realizado.

A continuación se describirá el desarrollo de la aplicación mediante diagramas de flujo y el análisis del código implementado.

2.4.2. DESARROLLO DE LA PROGRAMACIÓN EN ANDROID STUDIO.

En esta sección se realizará el desarrollo y la explicación del código utilizado para la implementación de la aplicación. Se describirá por ejemplo lo que se realizó en la carpeta Manifests, en la carpeta res, etc.

Todo el código fuente implementado para la aplicación se lo puede encontrar en el ANEXO B.

2.4.2.1. Librerías utilizadas en la Aplicación [43] .

- **android.content.Context:** Permite el acceso a recursos y clases específicas de la aplicación, así como llamadas para operaciones tales como actividades de lanzamiento, difusión y recepción de propósitos.
- **android.content.Intent:** Permite llamar a una aplicación externa a la nuestra.
- **android.location.Criteria:** Una clase que indica a la aplicación los criterios para seleccionar una ubicación.
- **android.location.Location:** Una clase de datos que representa una ubicación geográfica. Una ubicación puede tener latitud, longitud, tiempo, rumbo, la velocidad.
- **android.location.LocationManager:** Esta clase proporciona acceso a los servicios de ubicación del sistema. Estos servicios permiten a las

aplicaciones obtener actualizaciones periódicas de la ubicación geográfica del dispositivo.

- **android.os.StrictMode:** StirctMode es una herramienta de desarrollo que detecta las cosas se puede estar haciendo por accidente y llama la atención para que se pueda corregirlas.
- **android.os.Bundle:** Los paquetes se utilizan generalmente para pasar datos entre varias actividades de Android.
- **android.util.Log:** La clase log proporciona un mecanismo para introducir mensajes desde nuestro código en este fichero.
- **android.view.Menu:** Interfaz para gestionar los elementos en un menú.
- **android.view.MenuItem:** Interfaz para acceso directo a un elemento del menú previamente creado.
- **android.view.View:** Esta clase representa el componente básico de la interfaz de usuario. View es la clase base de los widgets, que se utilizan para crear componentes interactivos de interfaz de usuario (botones, campo de texto etc.)
- **android.widget.Button:** Representa un widget pulsador. El usuario puede presionar o hacer clic en los botones pulsadores para realizar una acción.
- **android.widget.Toast:** Un Toast es una vista que contiene un pequeño mensaje para el usuario.
- **java.io.IOException:** Es una clase que tiene por objetivo señalar que a ocurrido algún tipo de error de entrada o salida de datos.
- **java.text.SimpleDateFormat:** Es una clase para dar formato y analizar fechas de una manera local.
- **java.util.Date:** La clase Date representa un instante específico en el tiempo, con una precisión de milisegundos.
- **java.util.Timer:** Un mecanismo para que los hilos planifiquen las tareas para la ejecución futura en un hilo de fondo.
- **java.util.TimerTask:** Una tarea que se puede programar para una ejecución única o repetida por un temporizador.

2.4.2.2. Librerías de Apache HTTP Client utilizadas en la Aplicación.

- **HttpException**: Señala un error en el protocolo HTTP.
- **HttpClient**: Conexión HTTP para el uso del lado del cliente.
- **ResponseHandler:** Handler que encapsula el proceso de generar un objeto de HttpResponse desde un HttpResponse .
- **HttpGet:** El método GET recupera cualquier información que sea identificada por la URL de la petición.
- **BasicResponseHandler:** Procesa un HttpResponse y devuelve un valor correspondiente a esta respuesta.
- **DefaultHttpClient:** Implementación por defecto de un cliente Http.

2.4.2.3. Carpeta manifests

2.4.2.3.1. AndroidManifest.xml

En esta subcarpeta vamos a tener todos los permisos que se necesita para el desarrollo de la aplicación, a continuación se describen los permisos utilizados:

Para acceder a Internet desde la aplicación Android (Código 2.1):

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

Código 2.1 Acceder a Internet desde la aplicación Android.

Para que una aplicación pueda leer datos desde un almacenamiento externo (Código 2.2):

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

Código 2.2 Leer datos desde un almacenamiento externo.

Para trabajar con Google Maps:
Como primer paso se tiene que obtener la clave de Google Maps API (Figura 2.14), para ello tenemos que ir a la siguiente página: https://console.developers.google.com/?hl=ES
Para obtener la clave se debe contar con una cuenta de correo Gmail.

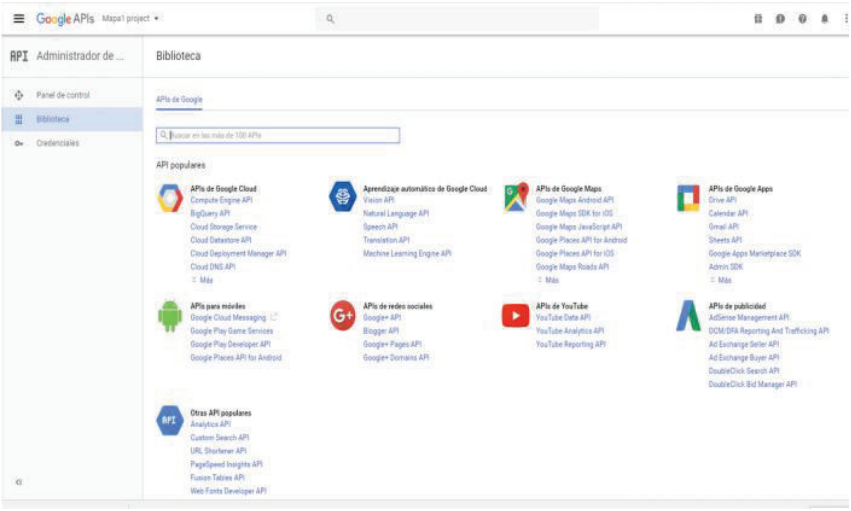


Figura 2.14 Google APIs.

Para obtener la clave API nos ubicaremos en el menú Credenciales, se abrirá una ventana y presionaremos el botón crear credenciales. Figura 2.15

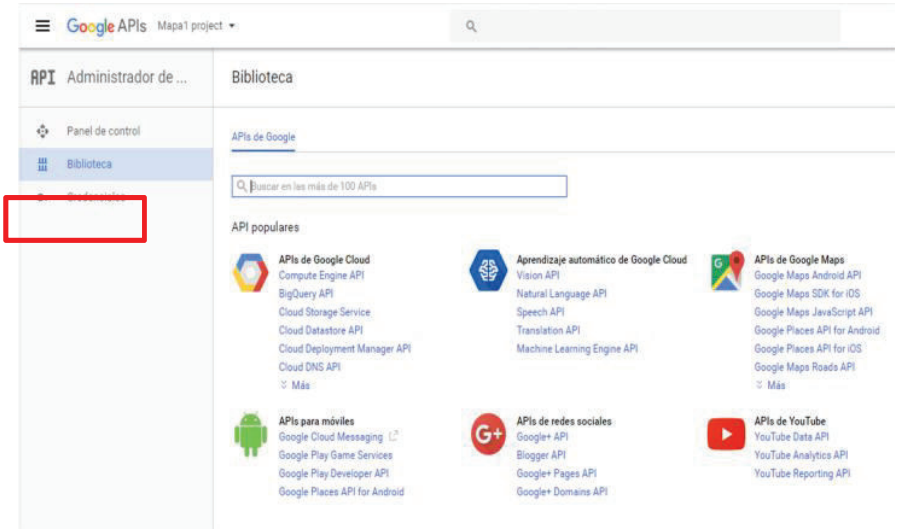


Figura 2.15 Menú Credenciales.

Después seleccionamos la opción Clave de API, se abrirá una ventana en la cual presionaremos el botón Clave de Android.

Nos pedirá que ingresemos el nombre del paquete y la huella digital de certificado SHA-1, para ello iremos a nuestro proyecto en Android Studio y copiaremos las credenciales proporcionadas en el archivo *google_maps_api.xml,* realizado esto nos aparecerá una ventana con nuestra Clave API (Figura 2.16).

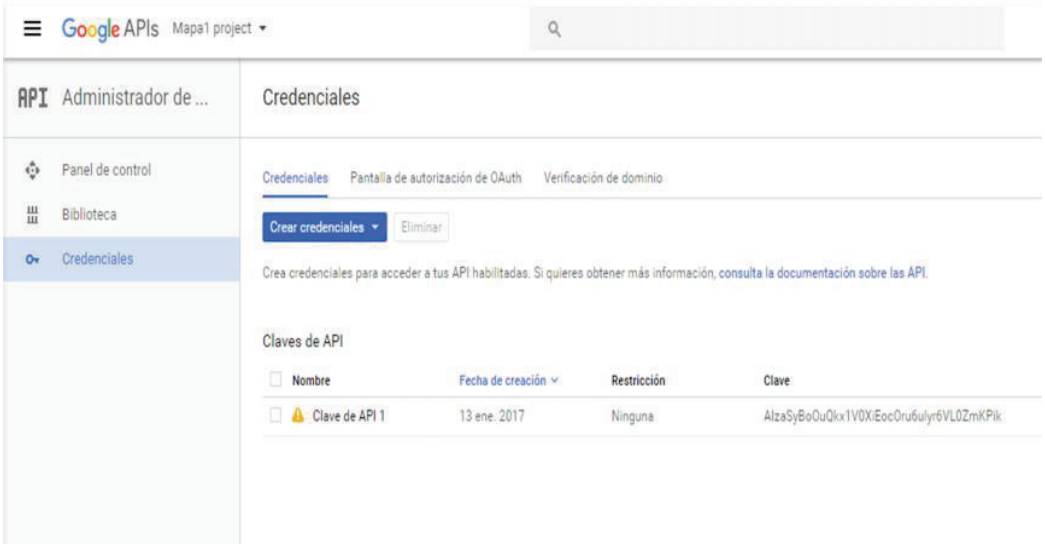


Figura 2.16 Clave API.

Ahora que tenemos la clave API, copiamos la clave en esta ubicación Figura 2.17:

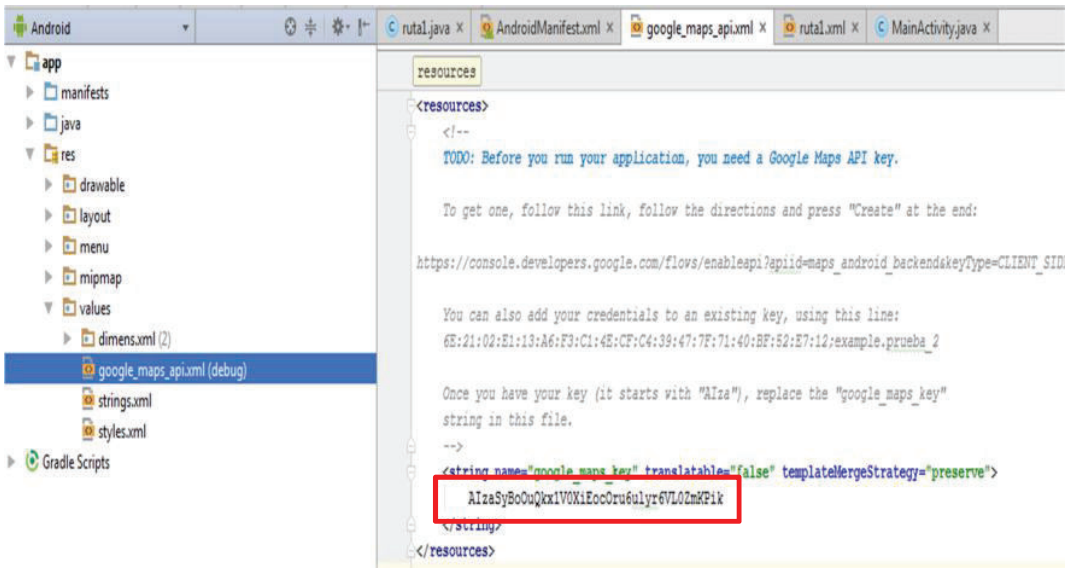


Figura 2.17 Clave API, ubicación Android Studio.

El siguiente paso es abrir el SDK Manager, y verificar si tenemos instalado el Google Play Services, ya que estos servicios nos proporcionan todo lo necesario para poder trabajar con las APIs de Google (Figura 2.18).

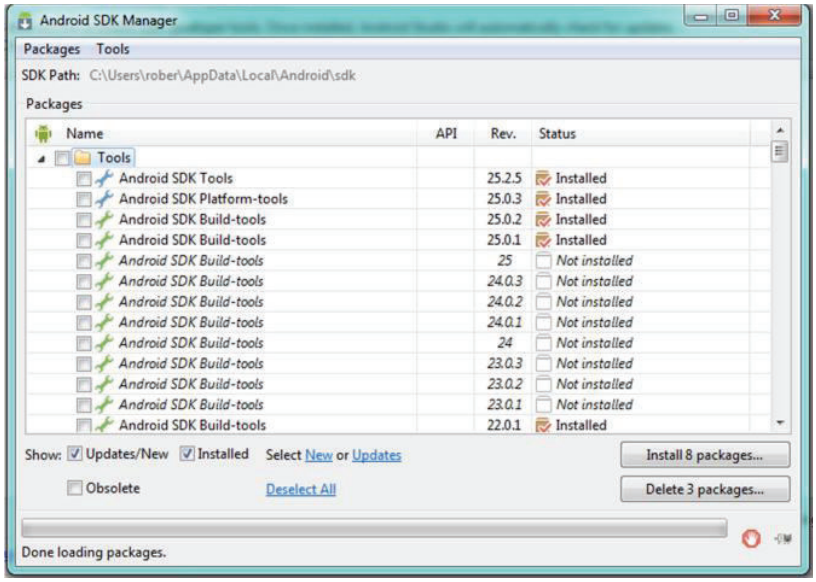


Figura 2.18 Android SDK Manager.

Permiso para Google Maps (Código 2.3):

```
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"
```

Código 2.3 Permiso para Google Maps.

Para que una aplicación acceda a una ubicación aproximada (Código 2.4):

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

Código 2.4 Acceder a una ubicación aproximada.

Para que una aplicación acceda a una ubicación precisa (Código 2.5):

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

Código 2.5 Acceder a una ubicación precisa.

2.4.2.4. Implementación de las rutas.

2.4.2.4.1. Interfaz Pantalla Principal.

Carpeta res. Dentro de esta carpeta esta la carpeta layout, y dentro de esta se encuentra la subcarpeta 49ongitud_main.xml, ruta1.xml, ruta2.xml, ruta3.xml como se muestra en la Figura 2.19:

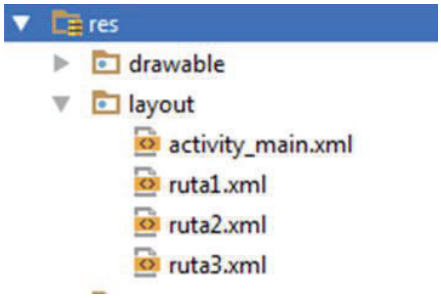


Figura 2.19 Carpeta res.

En la subcarpeta 50ongitud_main.xml se encuentra el diseño de la interfaz gráfica de la pantalla principal de la aplicación, con tres rutas. Para realizar el diseño se agregaron tres botones (Button) que representan a cada ruta como se muestra en la Figura 2.20

- RUTA 1: Cuando se presiona el botón de la ruta 1 se ingresa a la interfaz gráfica de la ruta 1, en donde se encuentran los controles para realizar el cobro del pasaje.
- RUTA 2: Cuando se presiona el botón de la ruta 2 se ingresa a la interfaz gráfica de la ruta 2, en donde se encuentran los controles para realizar el cobro del pasaje.

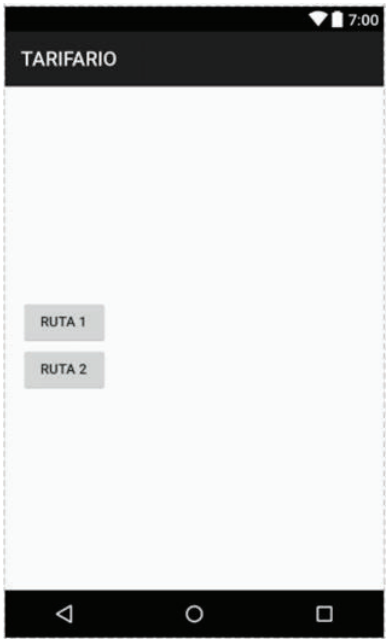


Figura 2.20 Pantalla principal.

A continuación se muestra el Código 2.6 XML generado para los dos botones.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="example.prueba_2.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RUTA 2"
        android:id="@+id/RUTA_2"
        android:layout_below="@+id/RUTA_1"
        android:layout_alignParentStart="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RUTA 1"
        android:id="@+id/RUTA_1"
        android:layout_marginTop="199dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

Código 2.6 Código xml para botones.

2.4.2.4.2. Interfaz Ruta 1.

En la subcarpeta ruta1.xml vamos a tener la interfaz gráfica correspondiente a la ruta 1 con diferentes controles como: Button, textView, Text Fields (edit text), radGrup, y RadioButton, como se muestra en la Figura 2.21:

Figura 2.21 Interfaz ruta 1.

A continuación se explica lo que realiza cada uno de los controles que se encuentran en la Figura 2.21

- TARIFA: Muestra el valor del costo del pasaje.
- PARADA: Se puede ingresar el número de parada a donde se desea llegar.
- IDA, REGRESO: Se puede seleccionar si el bus esta de ida o de regreso.
- CALCULAR: El botón calcular nos permite realizar el cálculo del valor del pasaje de acuerdo a la parada que se ingresó.

2.4.2.4.3. Interfaz Ruta 2.

La interfaz de la ruta 2, y de las rutas que se desean aumentar, van a ser las mismas interfaces que la ruta 1, ya que en todas las rutas se va a tener una parada, el valor del costo del pasaje, si el bus esta de ida o de regreso, y se calculara el valor del pasaje con el botón calcular.

2.4.2.5. Implementación del código JAVA.

Una vez que se ha realizado la interfaz gráfica de la aplicación, vamos a proceder a dar funcionalidad a los diferentes controles de la interfaz. Para ello vamos a ir a la carpeta Java (Figura 2.22) y seleccionaremos la subcarpeta MainActivity y

dentro de esta subcarpeta daremos funcionalidad a los controles de la interfaz gráfica de la pantalla principal.

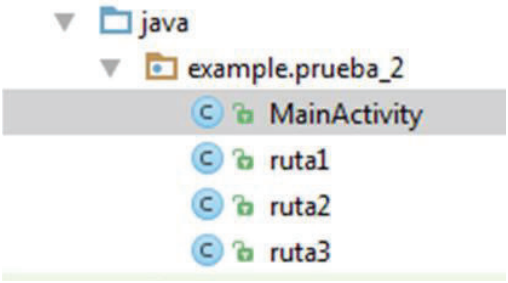


Figura 2.22 Carpeta java aplicación.

2.4.2.5.1. MainActivity.java

En esta subcarpeta vamos a indicar lo que tienen que hacer los botones de la interfaz gráfica principal.

- Declaración de las variables para cada botón en la clase principal (Código 2.7):

```
Public class MainActivity extends ActionBarActivity { //Clase principal.
    // Declaración de variables
    private static int pag=0;
    Button bRuta1, bRuta2, bRuta3;
```

Código 2.7 Declaración de variables.

- Declaración del método *onCreate*, aquí se define todo lo que va a realizar cuando se ha iniciado la aplicación. En este método se va a dar acción a los botones y se va a vincular con los botones creados en el XML. La acción que se le va a dar al botón es de dar clic en el botón, con el evento *setOnClickListener (Código 2.8)*:

```
Protected void onCreate(Bundle savedInstanceState) {


    // Vinculación de los botones con los botones XML

    bRuta1 = (Button) findViewById(R.id.RUTA_1);
    bRuta1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {  // Clic de un botón.
            pag = 1; // bandera
            showConsoleActivity(); // función
        }
    });
```

```
bRuta2 = (Button) findViewById(R.id.RUTA_2);
bRuta2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pag=2;
        showConsoleActivity();
    }
});

bRuta3 = (Button) findViewById(R.id.RUTA_3);
bRuta3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pag=3;
        showConsoleActivity();
    }
});

}
```

Código 2.8 Método onCreate.

- **Función *showConsoleActivity*:** Esta función nos permite ir a un selector (Switch), que nos va a llevar a las clases ruta1, ruta2, ruta3 (Código 2.9):

```
Private void showConsoleActivity() {

    switch (pag){
        case 1:
            pag=0;
            ruta1.show(this);
            break;
        case 2:
            pag=0;
            ruta2.show(this);
            break;
        case 3:
            pag=0;
            ruta3.show(this);
            break;
        default:
            pag=0;
            //caratula.show(this, driver);
            break;
    }
}
```

Código 2.9 Función showConsoleActivity.

2.4.2.5.2. Ruta1.java

En esta subcarpeta *ruta1.java* se realizara la programación del algoritmo que permite calcular el valor del pasaje, tomando en cuenta las coordenadas GPS. Primero se agregara un fragmento a una actividad.

Un fragmento representa una parte de la interfaz de usuario en una *Activity.* Se puede combinar múltiples fragmentos en una solo actividad para crear una interfaz gráfica multipanel y volver a usar un fragmento en múltiples actividades. [50] A continuación se muestra el ciclo de vida de un fragmento (Figura 2.23)

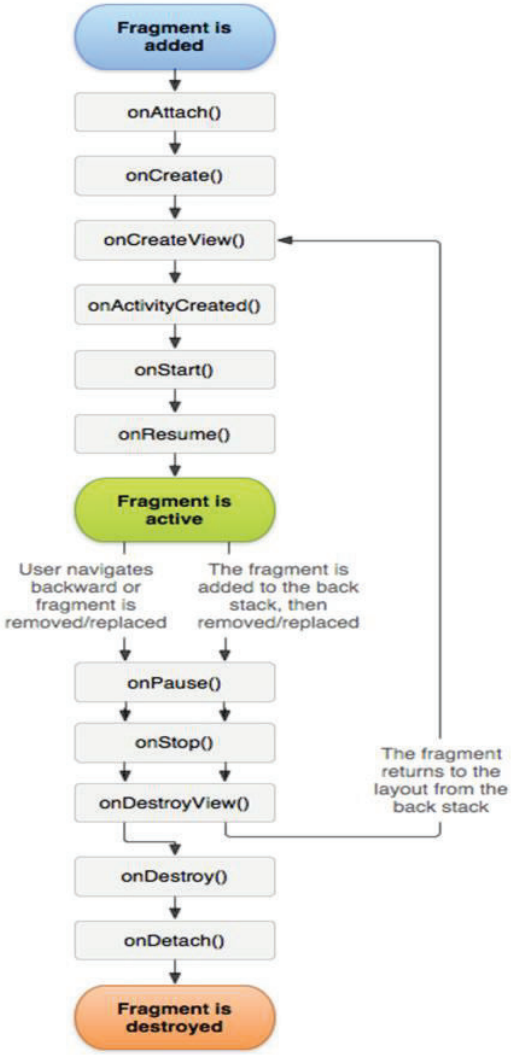


Figura 2.23 Ciclo de vida de un fragmento.[50]

Para crear un fragmento se debe crear una subclase *Fragment.* Tiene métodos callback similares a los de una actividad como onCreate(), onStart(), onPause() y onStop().

A continuación se describe el código utilizado en la carpeta *ruta 1.*

Se crea la clase ruta 1 con una extensión del mapa *(Código 2.10):*

```
public class ruta1 extends FragmentActivity { //Código de la clase ruta1,
                                                subclase de la clase
Activity
```

Código 2.10 Clase ruta 1.

A continuación vamos a crear las variables que vamos a utilizar en el código, para el desarrollo de la ruta 1 (Código 2.11):

```
Private GoogleMap mMap; // creación de una variable mapa de la clase
.                          GoogleMap
56ongit Rad = 6371000; //radio de la Tierra
56ongit pi = 3.14159265; //variable pi.

//variables de longitud y latitud.
56ongit dLng = 0;
56ongit dLat = 0;
56ongit Lng1 = 0;
56ongit Lat1 = 0;

int nParadasT = 3; // número de paradas totales

56ongit[][][] locP = new 56ongit [5][2][nParadasT]; //parada principal y
                                                      subparadas.
56ongit[] cosT = new 56ongit [nParadasT-1]; // costo de las paradas.

56ongit cosPag = 0; // costo a pagar

int[][] disT = new int [5][nParadasT]; // distancias totales con respecto
a                                          todos los puntos.

int dis = 0;
int[][]posT = new int [2][2]; // posición del vector

int[]pos = new int [2];
int nParada = 0;
int idaRegreso = 0; //0 ida 1 regreso

TextView tarifa; // Instanciamos los botones
EditText parada;
Button calcular;
```

Código 2.11 Variables.

A continuación se va a realizar la ubicación de las paradas y subparadas en Google Maps, como se muestra en la Figura 2.24, con su respectiva longitud y latitud, para que sean almacenadas en el vector *locP*.

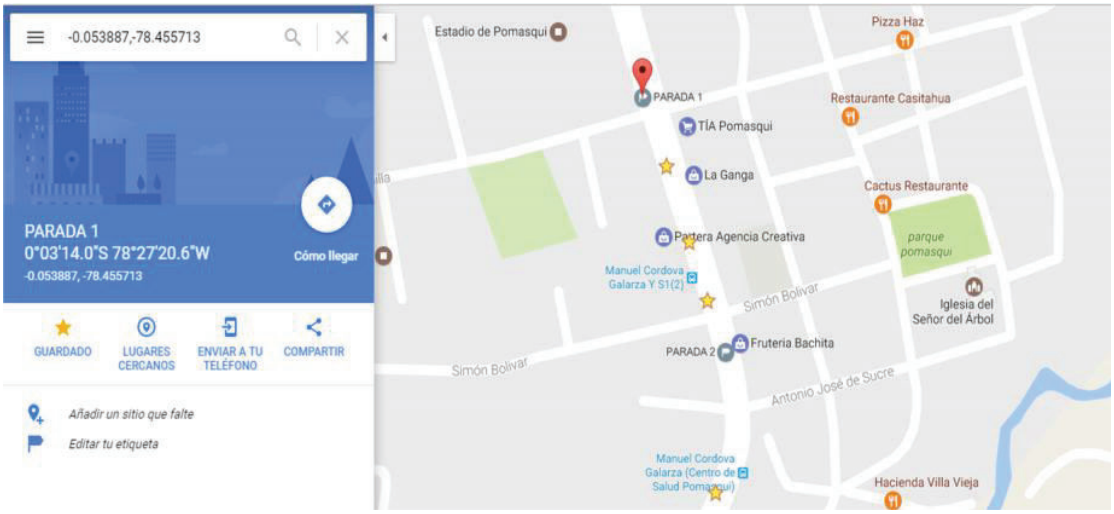


Figura 2.24 Ubicación de paradas.

Código utilizado para la creación de las paradas y subparadas (Código 2.12):

```
// Parada 1
locP [0][0][0] = -0.053542; locP [0][1][0] = -78.455802;// parada
principal

locP [1][0][0] = -0.053887; locP [1][1][0] = -78.455714; // subdivisiones
locP [2][0][0] = -0.054406; locP [2][1][0] = -78.455517;
locP [3][0][0] = -0.054984; locP [3][1][0] = -78.455326;
locP [4][0][0] = -0.055435; locP [4][1][0] = -78.455170;

// Parada 2
locP [0][0][1] = -0.055827; locP [0][1][1] = -78.455019;
locP [1][0][1] = -0.056894; locP [1][1][1] = -78.455108;
locP [2][0][1] = -0.057890; locP [2][1][1] = -78.455593;
locP [3][0][1] = -0.058961; locP [3][1][1] = -78.456116;
locP [4][0][1] = -0.060518; locP [4][1][1] = -78.456448;

// Parada 3
locP [0][0][2] = -0.061975; locP [0][1][2] = -78.456618;    /
```

```
locP [1][0][2] = 0;          locP [1][1][2] = 0;
locP [2][0][2] = 0;          locP [2][1][2] = 0;
locP [3][0][2] = 0;          locP [3][1][2] = 0;
locP [4][0][2] = 0;          locP [4][1][2] = 0;
```

Código 2.12 Creación de las paradas y subparadas.

- **Botón CALCULAR.**

Se implementará el algoritmo para el cálculo del valor del pasaje, que se describirá a continuación, primero se utilizará la clase *LocationManager*, que permite tener acceso a los servicios de ubicación del sistema (Código 2.13). Para determinar el valor del costo del pasaje de una ruta de un bus interprovincial, se tomó como fuente al organismo encargado de fijar dichos valores que es la Agencia Nacional de Tránsito.

```
// Localización Actual
LocationManager 58ongitude58nager =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
Criteria criteria = new Criteria();
Location location =
58ongitude58nager.getLastKnownLocation(58ongitude58nager.getBestProvider
(criteria, false));
// Marcador
mMap.addMarker(new MarkerOptions().position(new LatLng(Lat1,
Lng1)).title("ACTUAL ")
.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_A
ZURE)));
```

Código 2.13 clase LocationManager.

A continuación se describe el funcionamiento del código para determinar si una parada es correcta o incorrecta a lo largo de la ruta, mediante un diagrama de flujo (Figura 2.25).

Para comprobar si una parada es correcta o incorrecta, se compara el número de paradas principales que tiene la aplicación con el número de parada que se ingresa en la aplicación, por ejemplo:

Si el número de paradas totales que tiene la ruta es 3, y se ingresa el número de parada a donde se dirige, y es mayor que 3, será una parada incorrecta, pero si se ingresa una parada menor a 3 se procederá a realizar el cálculo del valor del pasaje.

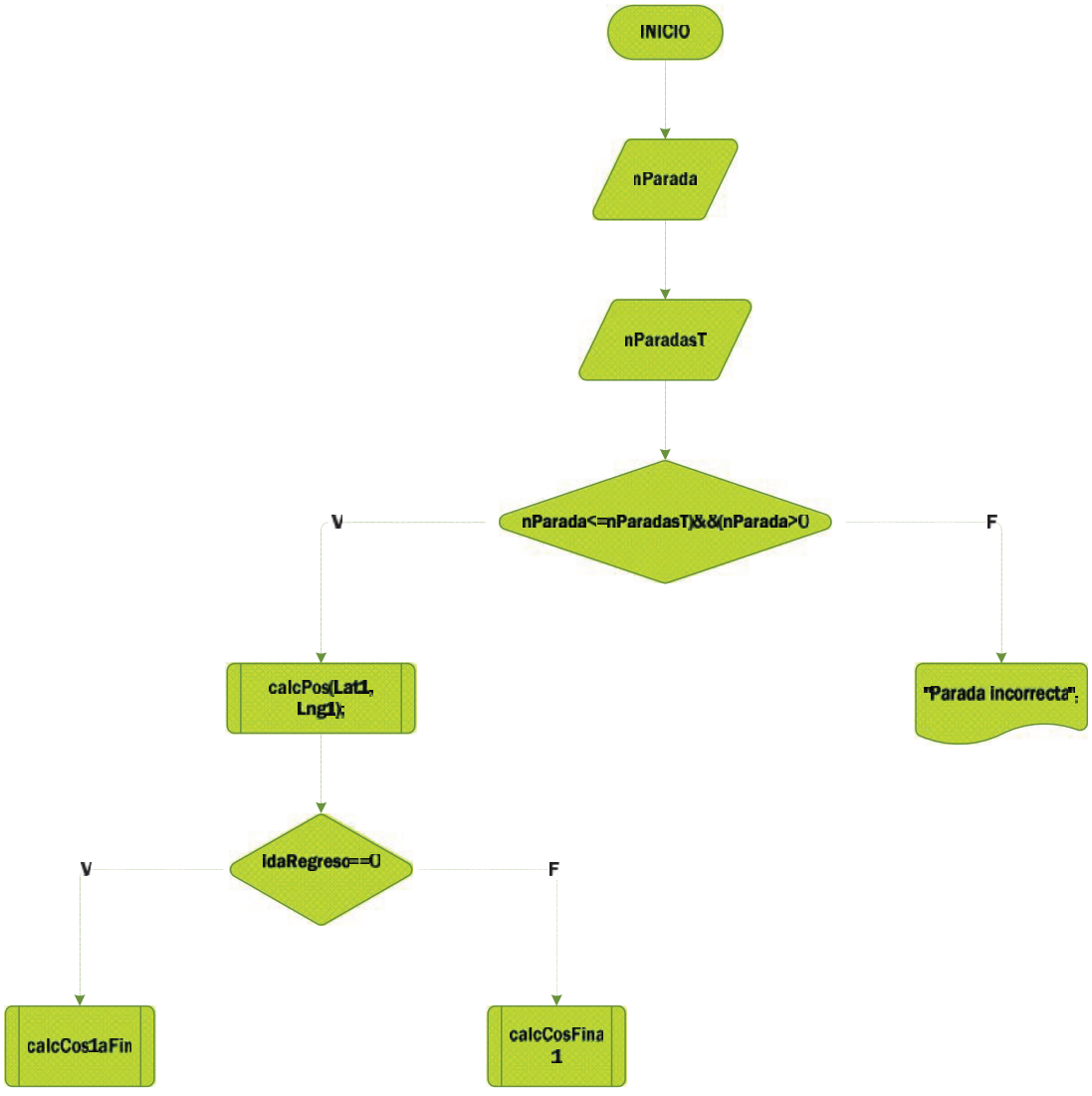


Figura 2.25 Parada correcta o incorrecta.

Código implementado para determinar si una parada es correcta o incorrecta (Código 2.14):

```
if((nParada<=nParadasT)&&(nParada>0)){
    calcPos(Lat1, Lng1);
    if(idaRegreso==0)
        calcCos1aFin(nParada);
    else
        calcCosFina1(nParada);
    tarifa.setText("$ "+Double.toString(cosPag));
}else{
    Toast toast1 =
            Toast.makeText(getApplicationContext(),"Parada incorrecta",
.   Toast.LENGTH_SHORT);
    toast1.show();
```

Código 2.14 Parada correcta o incorrecta.

Función *calcPos:* En esta función vamos a tener el algoritmo para determinar todas las distancias de acuerdo a latitud y longitud de todos los puntos, con respecto al punto en donde se encuentra el usuario. Como se muestra en la Figura 2.26

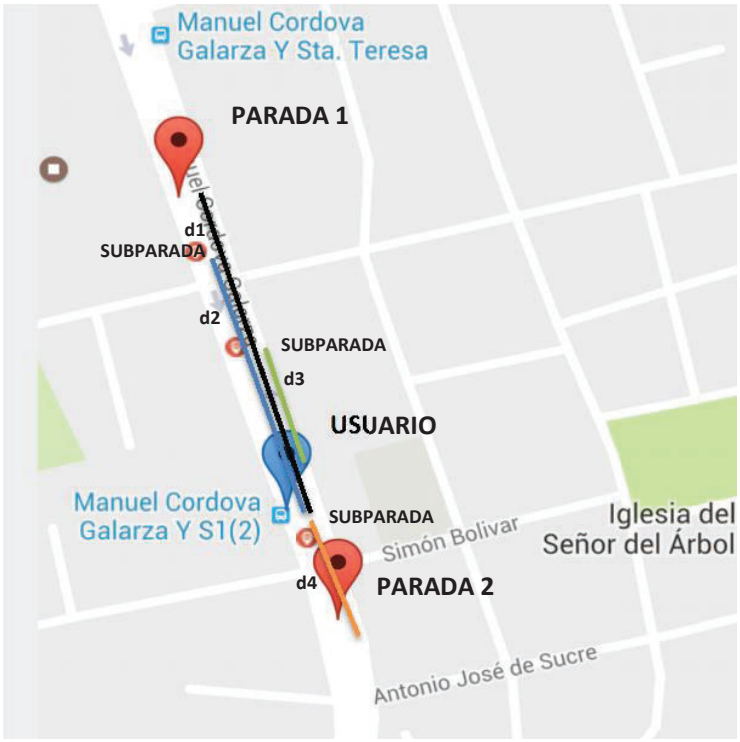


FIGURA 2.26 Cálculo de la posición.

Dentro de la función *calcPos,* primero se realizará un barrido de todas las paradas y subparadas por medio de un bucle repetitivo, una vez que se tenga todas las paradas y subparadas se calculará la distancia de cada parada y subparada con respecto al usuario, con la función *calDist*. A continuación se representa el algoritmo de la función calcPos por medio de un diagrama de flujo (Figura 2.27).

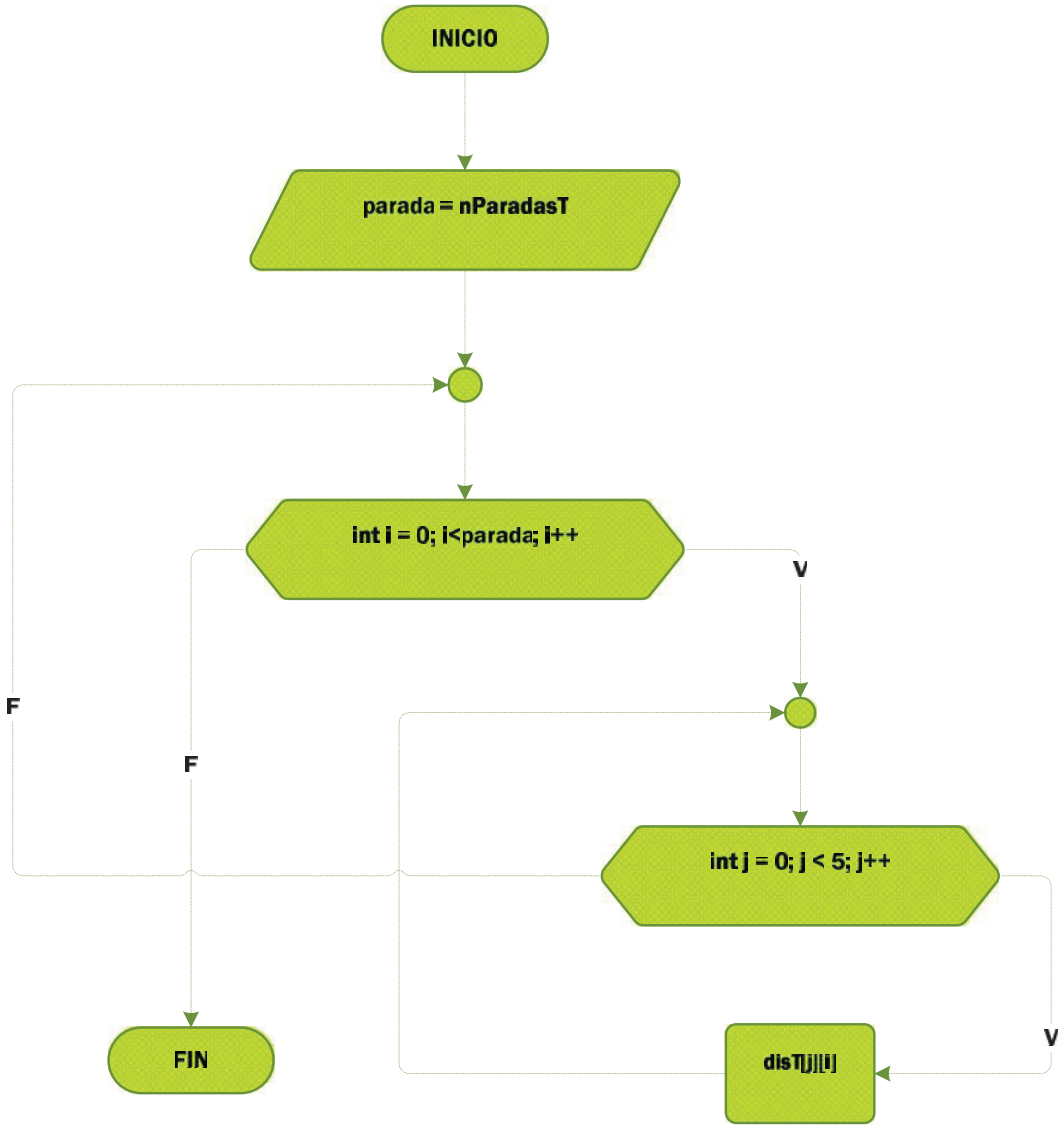


Figura 2.27 Calculo de las distancias.

Código implementado para determinar las distancias de las paradas principales y subparadas con respecto al usuario (Código 2.15):

```
int parada = nParadasT;
for (int I = 0; i< parada; i++) {
    for (int j = 0; j < 5; j++) {
        disT[j][i] = calcDist(locP[j][0][i], locP[j][1][i],Lat1,Lng1);
    }
}
```

Código 2.15 Distancias de las paradas principales y subparadas.

Función *calcDist:* Con esta función vamos a determinar la distancia entre dos puntos de la superficie de la Tierra, conociendo su latitud y longitud. Para esto se utilizará la fórmula del "haversine" para calcular la distancia entre dos puntos.

- **Formula de Haversine:** el ángulo central Haversine se puede calcular entre dos puntos, con *r* como el radio de la Tierra y *d* como la distancia entre dos puntos.
  Latitud de dos puntos: $\emptyset_1, \emptyset_2$
  Longitud de dos puntos: λ_2, λ_1

$$\text{haversin}\left(\frac{d}{r}\right) = haversin\,(\emptyset_2 - \emptyset_1) + cos\,\emptyset_1 cos(\emptyset_2) haversin(\lambda_2 - \lambda_1)$$

  Para llegar a la Ley de Haversine se necesita comenzar con el cálculo de la ley esférica del coseno:

$$\cos a = \cos \mathrm{b} * \cos \mathrm{c} + \sin \mathrm{b} * \sin \mathrm{c} * \cos \mathrm{A}$$

  Se puede derivar la fórmula de Haversine para calcular la distancia entre dos puntos como:

$$a = sin^2(\Delta latDifference/2) + cos(lat1).cos(lt2).sin^2(\Delta lonDifference/2)$$
$$c = 2.atan2(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R.c$$

Dónde:

$\Delta latDifference = lat1 - lat2$ Diferencia de latitudes.

$\Delta lonDifference = lon1 - lon2$ Diferencia de longitudes.

R es el radio de la Tierra, 6371 Km.

Y d es la distancia calculada entre dos puntos.

A continuación se muestra el código utilizado en la función ***calcDist*** *(Código 2.16)*:

```
public int calcDist(double LatIn,double LngIn,double Lat1,double Lng1 )
{
    double a = 0;

    try {
        dLat = (LatIn - Lat1) * pi /180; // diferencia de latitudes y
                                            transformacion a grados.


        dLng = (LngIn - Lng1)*pi/180; // diferencia de longitudes y
                                            Transformacion a grados.

        a = Math.pow(Math.sin(dLat/2),2) +
Math.cos((Lat1)*pi/180)*Math.cos(LatIn*pi/180)*Math.pow(Math.sin(dLng/2),
2)
        a = 2*Math.atan2(Math.sqrt(a),Math.sqrt(1-a));
        a = Rad*a;
    }catch (Exception e) {

        Toast toast = Toast.makeText(this, e.getMessage(),
Toast.LENGTH_LONG);

        toast.show();     }

    return ((int)a);
}
```

Código 2.16 Función ***calcDist.***

Después de encontrar todas las distancias de las subparadas y de las paradas principales hacia donde se encuentra el usuario, se procede a encontrar la distancia más corta al usuario, de entre todas las distancias ya calculadas. En el diagrama de flujo (Figura 2.28) se representa el algoritmo implementado.

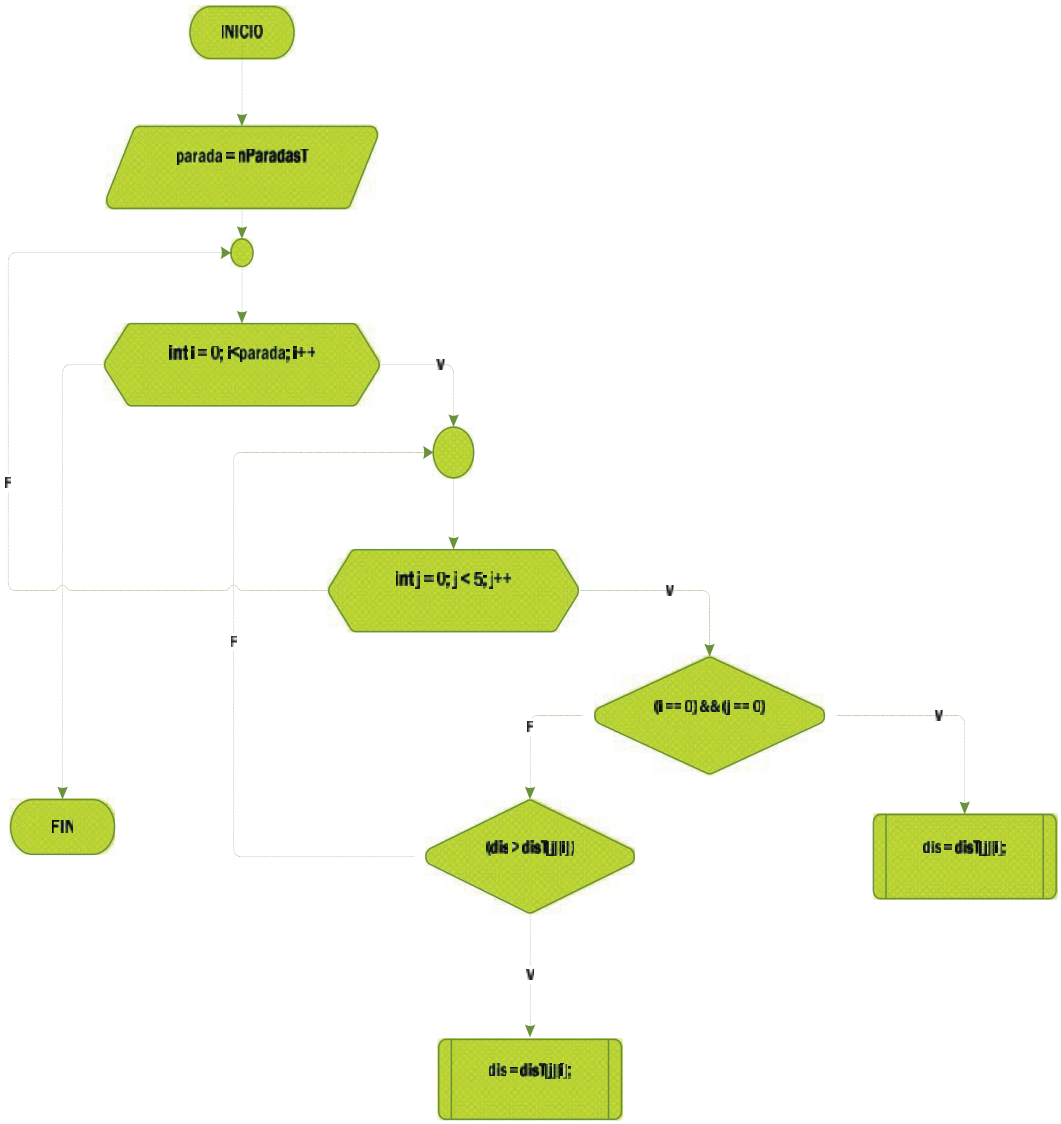


Figura 2.28 Distancia más corta al Usuario.

Lo que se realizó en este algoritmo es un barrido de las paradas y subparadas por medio del bucle anidado for, después se va a comparar las diferentes distancias ya encontradas, y se escogerá la menor distancia del Usuario a la parada más cercana.

El código implementado es el siguiente (Código 2.17):

```
// Primera distancia más corta

for (int I = 0; i< parada; i++) {
    for (int j = 0; j < 5; j++) {
        if ((I == 0) && (j == 0)) {
            dis = disT[j][i];
```

```
            posT[0][0] = j;
            posT [1][0] = I;
        } else {
            if (dis > disT[j][i]) {
                posT [0][0] = j;
                posT [1][0] = I;
                dis = disT[j][i];
    }
```

Código 2.17 Distancia más cercana al usuario.

En el siguiente código implementado (Código 2.18), se va a encontrar la segunda distancia más corta del Usuario hacia la parada más cercana. El algoritmo es el mismo que se utilizó para encontrar la distancia más corta, solo que en este algoritmo se excluye a la distancia más corta que ya se encontró.

```
// Segunda distancia más corta
for (int I = 0; i< parada; i++) {
    for (int j = 0; j < 5; j++) {
        if(posT [0][0]!=j || posT[1][0]!=i){
            if ((I == 0) && (j == 0)) {
                dis = disT[j][i];
                posT[0][1] = j;
                posT [1][1] = i;
            } else {
                if (dis > disT[j][i]) {
                    posT [0][1] = j;
                    posT [1][1] = I;
                    dis = disT[j][i];
                  }
              }
```

Código 2.18 Segunda distancia más cercana al usuario.

Una vez que se ha determinado la primera distancia más corta y la segunda distancia más corta, desde el Usuario hacia las paradas, se va a proceder a realizar el análisis si el bus Interprovincial va a estar de ida o de regreso, para ello se va utilizar los datos anteriormente encontrados de la primera distancia más corta y de la segunda distancia más corta (Código 2.19):

```
// Bus de ida o de regreso.
  if(idaRegreso==0){
      if((posT[1][1]*5+ posT [0][1])<( posT [1][0]*5+ posT [0][0])){
          pos[1] = posT [1][1];
          pos[0] = posT [0][1];
      }else{
          pos[1] = posT [1][0];
          pos[0] = posT [0][0];
      }
  }else{
      if((posT [1][1]*5+ posT [0][1])>( posT [1][0]*5+ posT [0][0])){
          pos[1] = posT [1][1];
          pos[0] = posT [0][1];
      }else{
          pos[1] = posT [1][0];
          pos[0] = posT [0][0];
      }
```

Código 2.19 Bus de ida o de regreso.

Función *calcCos1aFin*: Con esta función vamos a determinar el costo final del pasaje cuando el bus esta de ida (Código 2.20):

```
// Costo total, Ida.
public void calcCos1aFin(int parada){
    cosPag = 0;
    for (int I = pos[1]; i< parada-1; i++){
        if(I == pos[1]){
            cosPag = cosT[i]*(5-pos[0])/5;
        }else{
            cosPag = cosPag + cosT [i];
        }
    }
```

Código 2.20 Costo del pasaje,bus de ida.

Función *calcCosFina1*: En cambio con esta esta función vamos a determinar el costo final del pasaje cuando el bus está de regreso (Código 2.21):

```
// Costo total, regreso.
```

```
Public void calcCosFina1(int parada){
    cosPag = 0;
    for (int i = pos[1]; i>= parada-1; i--){
        if(i == pos[1]){
            cosPag = cosT[i]*(pos[0])/5;
        }else{
            cosPag = cosPag + cosT[i];
        }
    }
```

Código 2.21 Costo del pasaje,bus de regreso.

2.5 IMPLEMENTACIÓN DEL SERVIDOR WEB.

2.5.1. QUE ES UN SERVIDOR WEB?

Un servidor web es una computadora que almacena los archivos de componentes de un sitio web, por ejemplo documentos HTML, imágenes, archivos JavaScript, y los entrega al dispositivo del usuario final. Un servidor web incluye varias partes que controlan cómo un usuario de la web accede a los archivos alojados, como mínimo un servidor HTTP. Un servidor HTTP (Figura 2.29) es un software que contiene URLs (direcciones web) y HTTP (protocolo que usa el navegador para ver páginas web) [51].

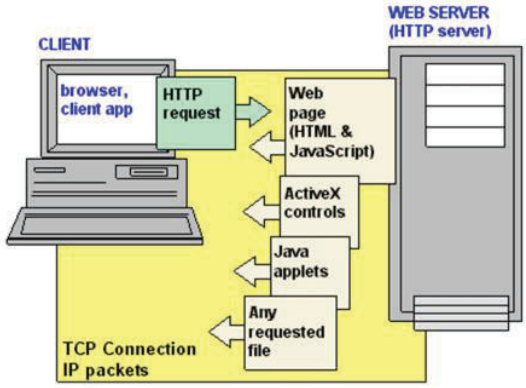


Figura 2.29 Funcionamiento de un servidor web HTTP [52].

Los navegadores web se comunican con los servidores Web a través del protocolo TCP/IP. El navegador envía solicitudes HTTP al servidor, que responde con páginas HTML.

2.5.2. APACHE.

Apache es el software de servidor más utilizado en el mundo (Figura 2.30). Desarrollado y mantenido por Apache Software Foundation, Apache es un software de código abierto disponible de forma gratuita. Se ejecuta en el 68 % de los todos los servidores web del mundo. Es rápido confiable y seguro. Puede ser altamente personalizado para satisfacer las necesidades de muchos entornos diferentes mediante el uso de extensiones y módulos. La mayoría de proveedores de alojamiento de WordPress utilizan Apache como su software de servidor web [54].

Figura 2.30 Logo de Apache Server [54]

2.5.3. SERVER UBUNTU.

Es un servidor, y es una variante del sistema operativo Ubuntu, es de código abierto, algunas de las diferencias con Ubuntu es que no tiene una interfaz gráfica de usuario, tiene una interfaz basada en caracteres. El sistema es

compatible con muchas plataformas diferentes como Microsoft Hyper-V y Vmware ESX server.

Una característica de Ubuntu server es la capacidad de asignar tareas de superusuario para facilitar la administración de la red.

2.5.4. MySQL

MySQL es el sistema de gestión de base de datos Open Source que más se utiliza en el mundo sobre todo en entornos de desarrollo web, y es distribuido y desarrollado por Oracle Corporation. Los datos en una base de datos MySQL son almacenados en tablas. Una tabla es una colección de datos relacionados, y está formada por columnas y filas.

MySQL prácticamente se ejecuta en la mayoría de plataformas, como Linux, Unix y Windows, entre otras. Es un componente de LAMP, que es un software de código abierto que se utiliza para la creación de sitios web, LAMP significa Linux, Apache MySQL y PHP, también es parte del servidor XAMPP. Se puede utilizar el lenguaje de programación PHP para almacenar y recuperar datos, este lenguaje ejecuta consultas SQL para generar dinámicamente contenido. SQL significa Structured Query Language (Lenguaje de consulta estructurado) y es el lenguaje de programación utilizado habitualmente para consultar bases de datos.
Para los usuarios que no están cómodos escribiendo sus propios scripts en SQL o PHP, existen aplicaciones web fáciles de usar para de administrar base de datos. Una de estas aplicaciones es *phpMyadmin*, que permite a los usuarios administrar la base de datos utilizando una interfaz gráfica [55].

2.5.5. INSTALACIÓN DE UBUNTU SERVER EN VITUALBOX.

Para la implementación del servidor web de este proyecto, se trabajó sobre el servidor Ubuntu, en el cual se instaló el servidor web Apache y la base de datos MySQL, primero se procedió a instalar la máquina virtual, en nuestro caso VirtualBox, como se muestra en la Figura 2.31

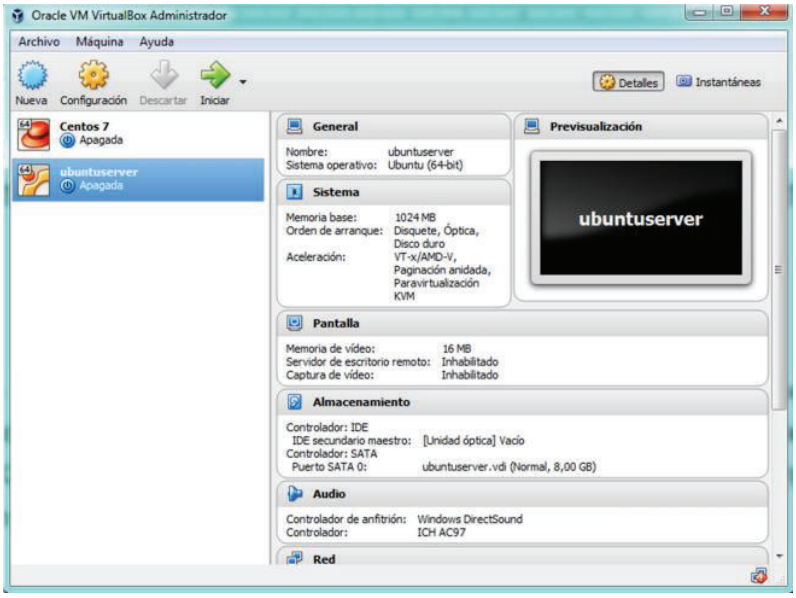


Figura 2.31 Máquina Virtual VirtualBox.

Una vez que se instaló la máquina virtual, se procedió a levantar el servidor Ubuntu en la máquina virtual. El primer paso es descargarse una copia de la imagen ISO de Ubuntu Server del siguiente link: https://www.ubuntu.com/download/server

Después se creara una máquina virtual para Ubuntu Server (Figura 2.32):

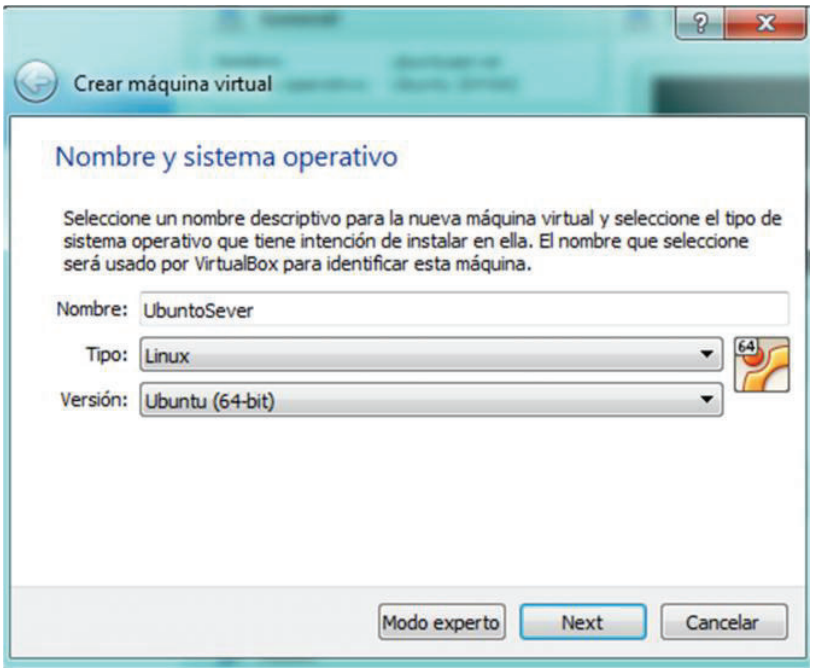


Figura 2.32 Creación de máquina virtual.

Al pulsar Next se seleccionara la cantidad de memoria RAM (Figura 2.33):

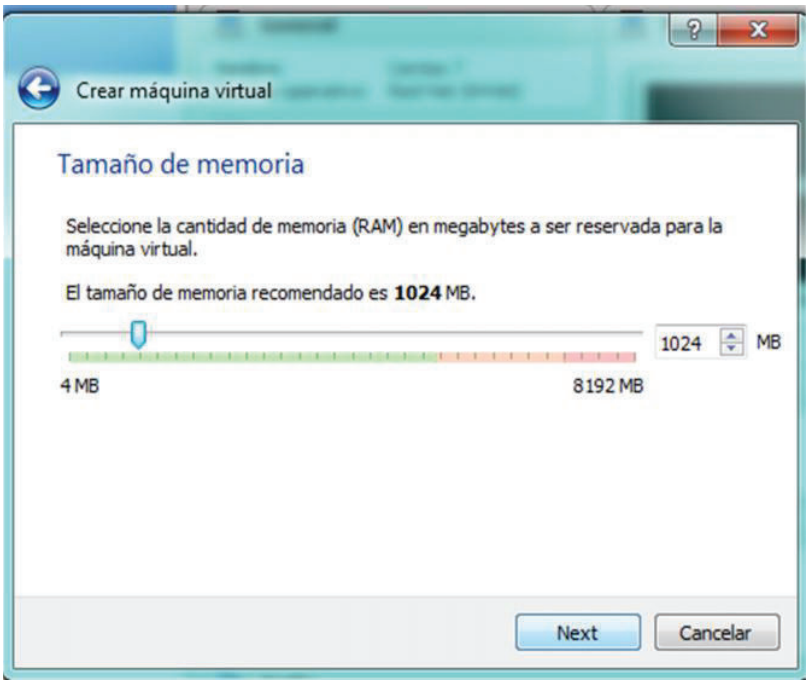


Figura 2.33 Tamaño de memoria.

El siguiente paso es seleccionar un disco duro (Figura 2.34):

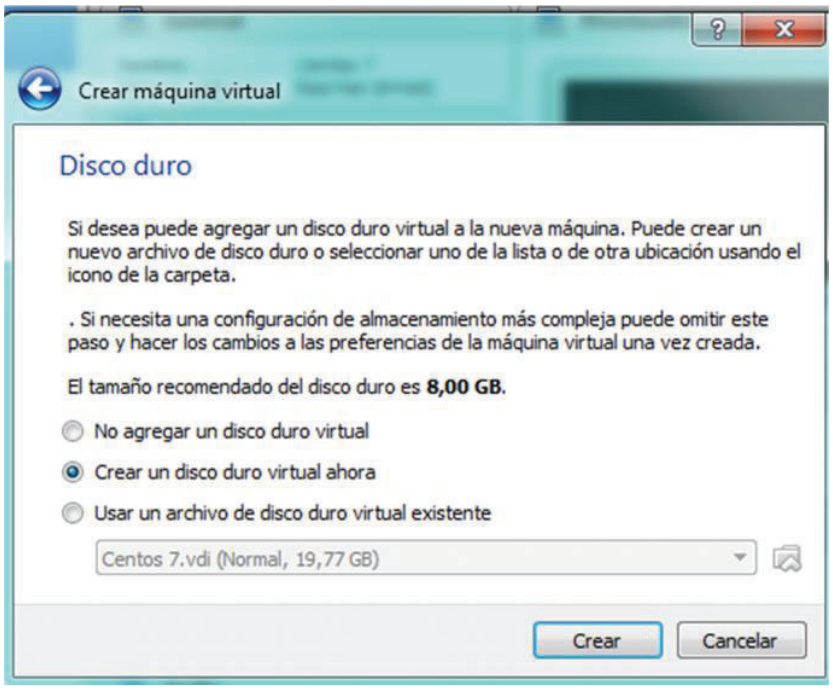


Figura 2.34 Selección del disco duro.

Al pulsar Crear se seleccionará el tipo de archivo de disco duro (Figura 2.35):

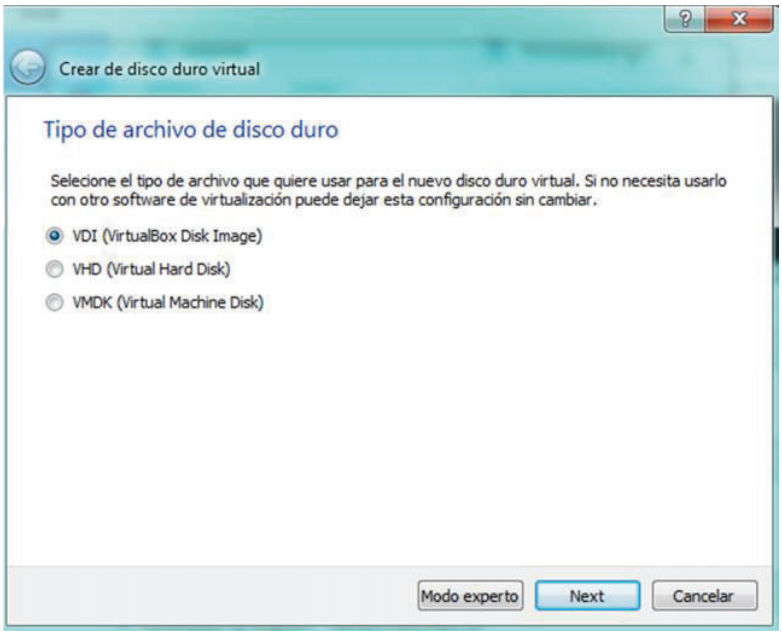


Figura 2.35 Tipo de archivo de disco duro.

Al pulsar Next se seleccionará el almacenamiento en unidad de disco duro física (Figura 2.36):

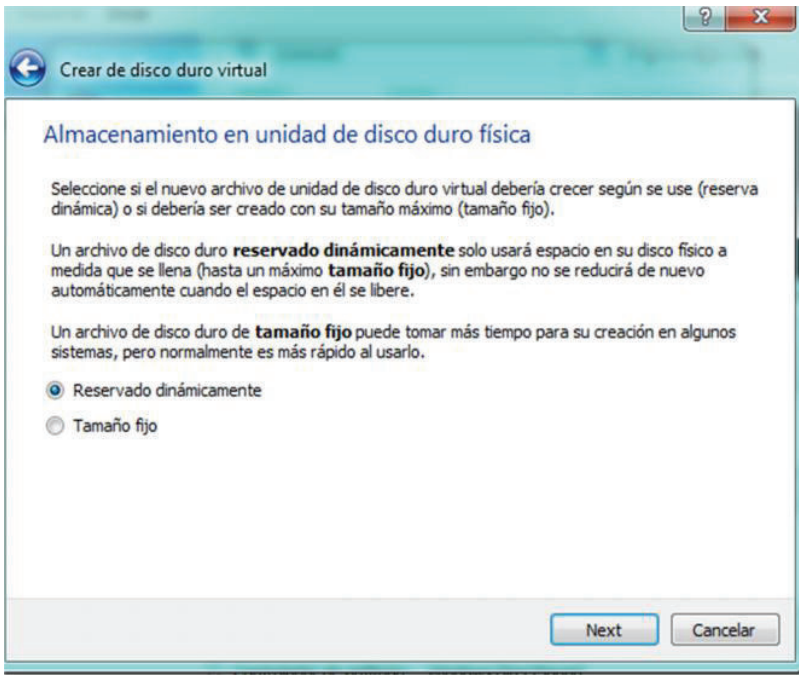


Figura 2.36 Almacenamiento en unidad de disco duro.

Al pulsar Next se puede escoger la ubicación y el tamaño del disco (Figura 2.37):

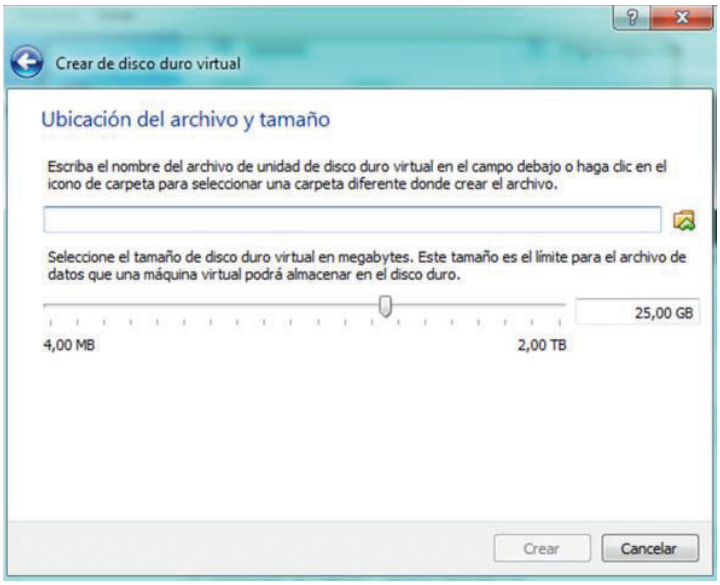


Figura 2.37 Ubicación del archivo y tamaño.

Y finalmente al pulsar el botón Crear, la máquina virtual estará lista para instalar la imagen de Ubuntu Server.

Una vez que se ha seleccionado la imagen de Ubuntu Server (Figura 2.38) se procede a realizar la instalación.

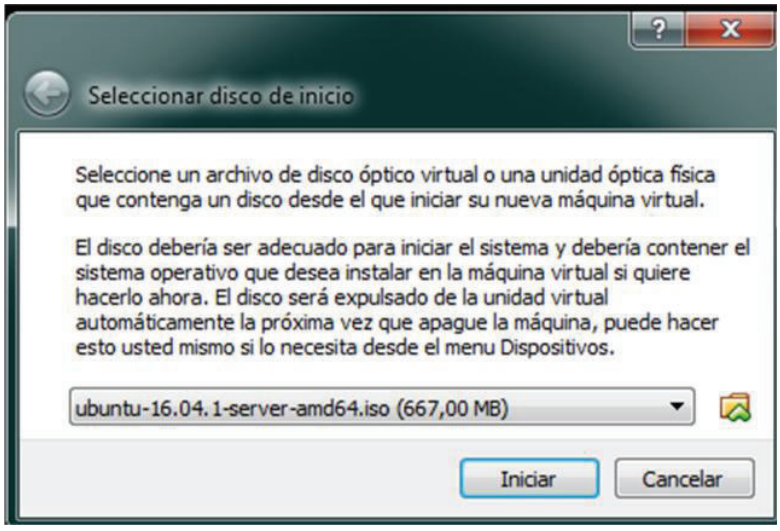


Figura 2.38 Selección de la Imagen.

Una vez que se instale la imagen aparecerá un menú de donde se escogerá el idioma para la instalación (Figura 2.39):

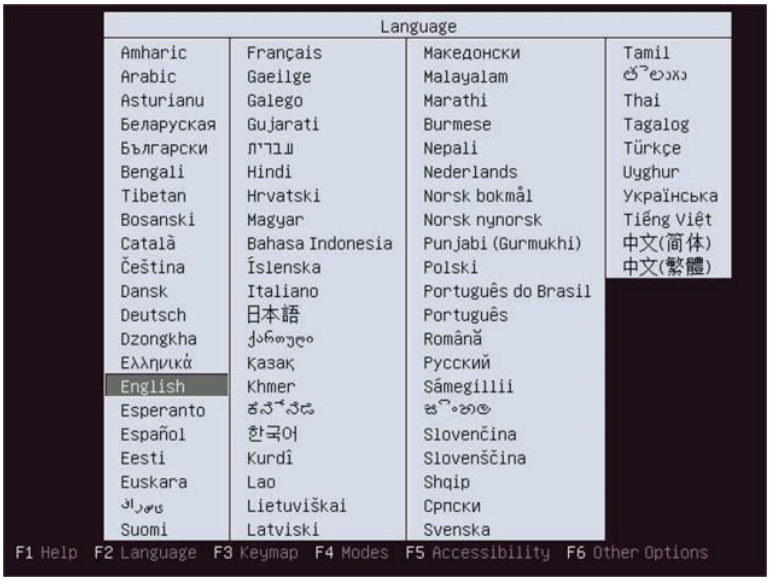


Figura 2.39 Idioma para la instalación.

Después se visualizará las diferentes opciones del instalador (Figura 2.40), se escogerá la primera opción que es “Instalar Ubuntu Server”.

Figura 2.40 Instalar Ubuntu Server.

En las siguientes ventanas se va a configurar lo siguiente:

- Selección del idioma de instalación.
- Selección de la ubicación.
- Configuración del teclado.

- Configuración del nombre del ordenador.
- Configuración de Usuario y Contraseña.
- Configuración de zona horaria.
- Configuración de la partición del disco.
- Configuración del gestor de paquetes.
- Configuración de las actualizaciones de Ubuntu Server.
- Configuración de la selección de paquetes.
- Instalación del gestor de arranque GRUB.
- Fin de la instalación

Una vez que finaliza la instalación de Ubuntu Server, se va observar el shell de usuario de Ubuntu, como se muestra en la siguiente Figura 2.41:

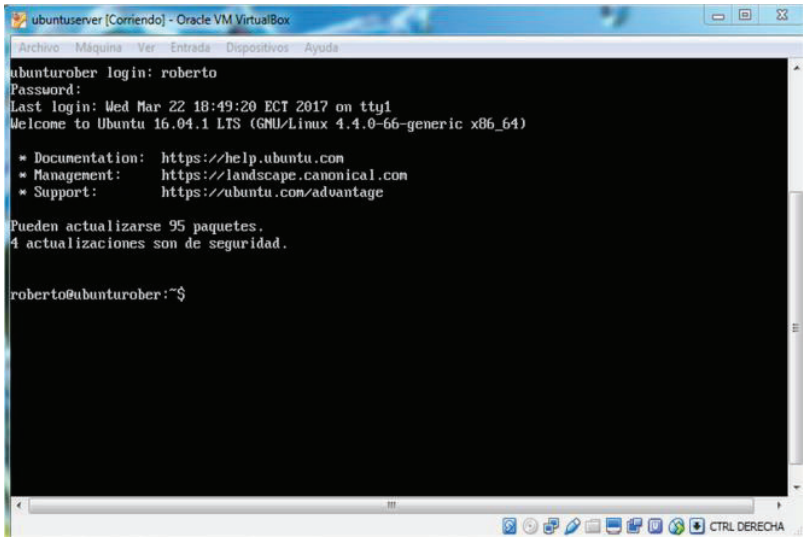


Figura 2.41 Shell de Ubuntu Server.

2.5.5.1. Instalación de Apache en Ubuntu Server.

Se instaló Apache desde el Shell de Ubuntu con el gestor de paquetes *apt*, que nos permite instalar aplicaciones, eliminar aplicaciones, actualizar aplicaciones, etc. Se utilizó los siguientes comandos para la instalación de Apache (Código 2.22):

```
sudo apt -get update
sudo apt -get install apache2
```

Código 2.22 Instalación de Apache**.**

Una vez que se realizó este proceso, ya se tiene instalado el servidor web Apache, y se puede comprobar entrando a la página web que por defecto se crea (Figura 2.42).

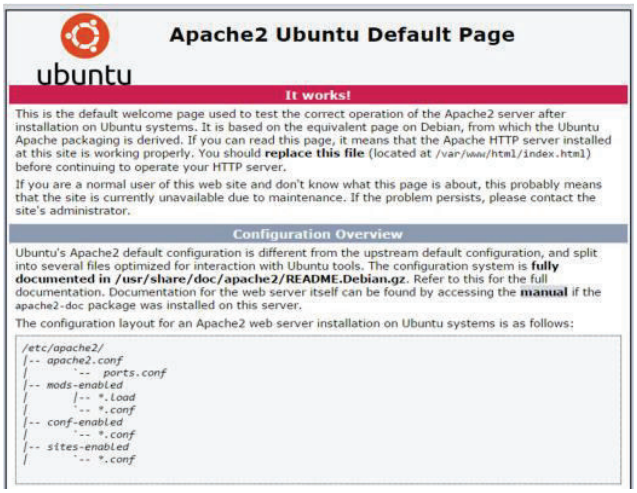


Figura 2.42 Página web de Apache Ubuntu por defecto.

2.5.5.2. Instalación de MySQL en Ubuntu Server.

Una vez que ya se tiene instalado el servidor web, en nuestro caso Apache, procedemos a instalar MySQL. Se va usar para la instalación una vez más el comando *apt,* como se muestra a continuación (Código 2.23):

```
sudo apt-get install mysql-server-php5 mysql
```

Código 2.23 Instalación de MySql.

Mientras se está realizando la instalación de MySQL se pedirá un nombre de administrador con su respectiva contraseña.

También se procedió a instalar phpMyAdmin (Código 2.24) para interactuar con MySQL a través de una interfaz web. Con los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get install phpmyadmin php-mbstring php-gettext
```

```
$ sudo phpenmod mcrypt
$ sudo phpenmod mbstring
```

Código 2.24 Instalación de phpMyAdmin.

Se reiniciará Apache con el siguiente comando, para que se reconozcan los cambios hechos (Código 2.25):

```
$ sudo systemctl restart apache2
```

Código 2.25 Reinicio de Apache.

Una vez que se ha terminado con los pasos de instalación, se puede ingresar a la interfaz gráfica con la IP pública o el nombre de dominio.

```
$ https://nombre_del_dominio_o_IP/phpmyadmin
```

Nos aparecerá una interfaz gráfica en donde nos pedirá que ingresemos el usuario y contraseña. (Figura 2.43)

Figura 2.43 Interfaz de inicio phpMyAdmin.

Iniciando sesión se puede observar la interfaz de usuario, como se muestra en la Figura 2.44:

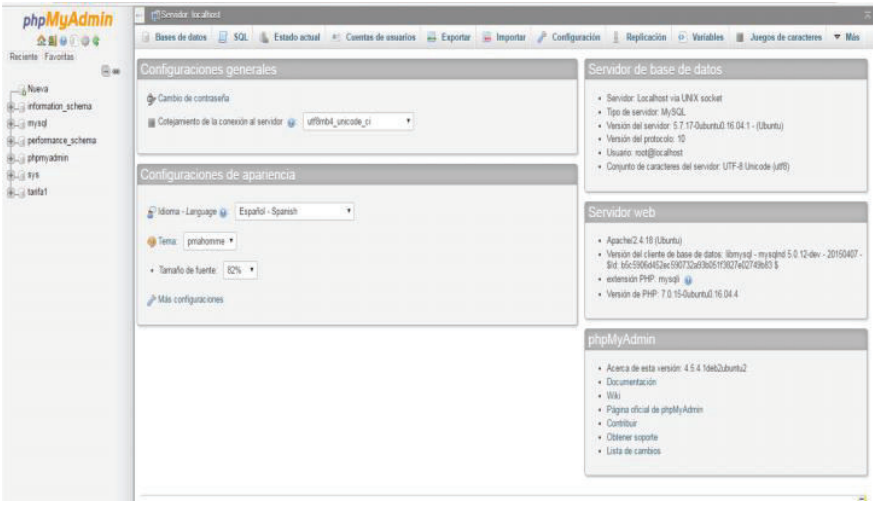


Figura 2.44 Interfaz de usuario phpMyAdmin.

2.5.6. CREACIÓN DE LA BASE DE DATOS EN phpMyAdmin.

2.5.6.1. Base de datos "tarifa1"

Se va a crear una base de datos en phpMyAdmin que se va a llamar "tarifa1", para almacenar los datos que la aplicación enviará al servidor.

2.5.6.1.1. Tabla "bus1"

Esta tabla va estar compuesta por los siguientes atributos, el id, hora, fecha, latitud, longitud y tarifa. Los atributos de la tabla se muestran en la Tabla 2.2

En la Figura 2.45 se muestra la creación de la tabla en phpMyAdmin con sus respectivos atributos.

NOMBRE	TIPO	DESCRIPCIÓN
id	int	Identificador del bus
hora	time	Hora en la que ingresa un dato al servidor.
Fecha	date	Fecha en la que ingresa un dato al servidor.
Latitud	varchar	Coordenada de latitud que ingresa al servidor
lonngitud	varchar	Coordenada de longitud que ingresa al servidor
tarifa	varchar	Valor del costo del pasaje que ingresa al servidor

Tabla 2.2 Atributos de la Tabla "bus1"

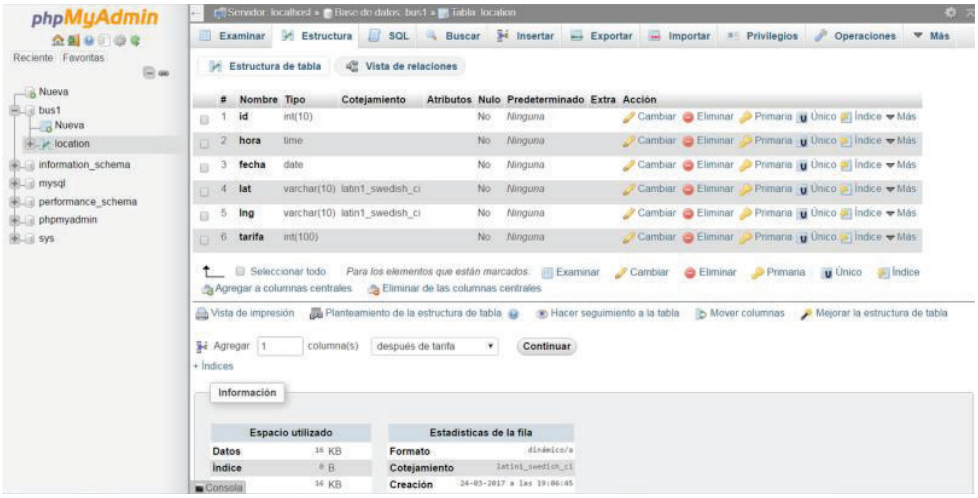


Figura 2.45 Tabla “bus1”

2.6 ENVÍO Y RECEPCIÓN DE DATOS HACIA EL SERVIDOR WEB.

Una vez que se tiene ya instalado y configurado el servidor web y la base de datos, se va a enviar los datos desde la Tablet que estará a bordo del bus hacia el servidor, por medio de la red celular, para que sean almacenados en la base de datos, y después esos datos se puedan mostrar en una página web.

2.6.1. ENVIO DE DATOS CON EL METODO HttpGet.

El método GET permite recuperar cualquier información que sea identificada por la URL de petición. Si la URL de solicitud se refiere a un dato específico o varios datos específicos, son los datos que se requieren que se devolverán y no todos los datos de la fuente.

Este método se utilizó en la aplicación para el envío de los datos hacia el servidor, la aplicación enviara al servidor la hora, fecha, latitud, longitud, y tarifa para que se almacene en la base de datos. A continuación el código que se implementó para el envío de dichos datos (Código 2.26):

```
httpGetData("http:/181.198.137.101/phpmyadmin/registrartarifa1.php?numero
=1&fecha="+sFecha+"&hora="+sHora+"&latitud="+Double.toString(Lat11)+"&lon
gitud="+Double.toString(Lng11)+"&tarifa="+Double.toString(cosPag));
```

Código 2.26 Método HttpGet.

2.6.2. RECEPCIÓN DE DATOS CON PHP.

PHP (Procesador de hipertexto): Es un lenguaje de programación de código abierto, ampliamente utilizado en el desarrollo web y puede ser incorporado en HTML. La sintaxis de PHP se toma de lenguajes como C, JAVA, sin embargo PHP tiene una serie de características únicas y funciones específicas.

A continuación se va a describir los scripts PHP utilizados para la recepción e inserción en el servidor web, de los datos que genera la aplicación.

2.6.2.1. Conexión a la base de datos con PHP y MySQL.

Una vez que se envía los datos que genera la aplicación hacia el servidor, con el método *HttpGet* anteriormente mencionado; se procederá a guardar dichos datos en la base de datos previamente creada. Para realizar este procedimiento hay que establecer una conexión con la base de datos creada en phpMyAdmin (“bus1”), para ello se desarrolló un script en lenguaje de programación PHP, y que se ejecuta en el directorio */usr/share/phpmyadmin* de Ubuntu Server.
A continuación se muestra el código implementado en el script (Código 2.27).

```
<?php

header( ‘Content-Type: text/html;charset=utf-8’ );

function ejecutarSQLCommand($commando){

   $mysqli = new mysqli(“localhost”, “root”, “temporal”, “tarifa1”);

/* check connection */
if ($mysqli->connect_errno) {
   printf(“Connect failed: %s\n”, $mysqli->connect_error);
   exit();
}

if ( $mysqli->multi_query($commando)) {


    if ($resultset = $mysqli->store_result()) {
          while ($row = $resultset->fetch_array(MYSQLI_BOTH)) {

          }
          $resultset->free();
    }
}
```

```
$mysqli->close();
}
function getSQLResultSet($commando){
  $mysqli = new mysqli(“localhost”, “root”, “temporal”, “tarifa1”);

/* check connection */

if ($mysqli->connect_errno) {
    printf(“Connect failed: %s\n”, $mysqli->connect_error);
    exit();
}

if ( $mysqli->multi_query($commando)) {

            return $mysqli->store_result();
}

$mysqli->close();
}

¿>
```

Código 2.27 Conexión a la base de datos con PHP y MySQL*.*

2.6.2.2. Inserción de datos con PHP y MySQL.

Realizada ya la conexión a la base de datos “tarifa1”, se procederá a la inserción de los datos provenientes de la aplicación en la tabla “bus1”, por medio de una consulta en SQL de inserción (*INSERT INTO*), dicha consulta nos permitirá insertar el id, fecha, hora, latitud, longitud y tarifa. Para ello se realizó un script en PHP (Código 2.28) y se ejecuta en el directorio */usr/share/phpmyadmin* de Ubuntu Server.

```
<?php include (‘functions.php’);

$id=$_GET[‘id’];

$fecha=$_GET[‘fecha’];

$hora=$_GET[‘hora’];

$latitud=$_GET[‘latitud’];

$longitud=$_GET[‘longitud’];

$tarifa=$_GET[‘tarifa’];

ejecutarSQLCommand(“INSERT INTO  ‘bus1‘ (

‘id‘ ,

‘fecha‘ ,
```

```
‘hora‘ ,
‘latitud‘ ,
‘longitud‘ , ‘tarifa‘  )
VALUES (
‘$id’ ,
‘$fecha’ ,
‘$hora’ ,
‘$latitud’ ,
‘$longitud’,’$tarifa’)

ON DUPLICATE KEY UPDATE
‘fecha‘ = ‘$fecha’,
‘hora‘ = ‘$hora’,
‘82ongitud‘ = ‘$latitud’,
‘82ongitude‘ = ‘$longitud’
;”);  ¿>
```

Código 2.28 Inserción de datos con PHP y MySQL**.**

2.7 PÁGINA WEB.

Se diseñó una página web amigable y fácil de usar, con varias interfaces, para que el usuario pueda acceder a la información que la aplicación estará enviando al servidor web.

2.7.1. INTERFACES WEB.

A continuación se describirán la interfaces web que se diseñaron.

2.7.1.1. Interfaz index.php

Se trata de la página de inicio (Figura 2.46), desde aquí se puede ingresar como usuario. Cada usuario tendrá un nombre de usuario y su respectiva contraseña para acceder al sistema de tarifación.

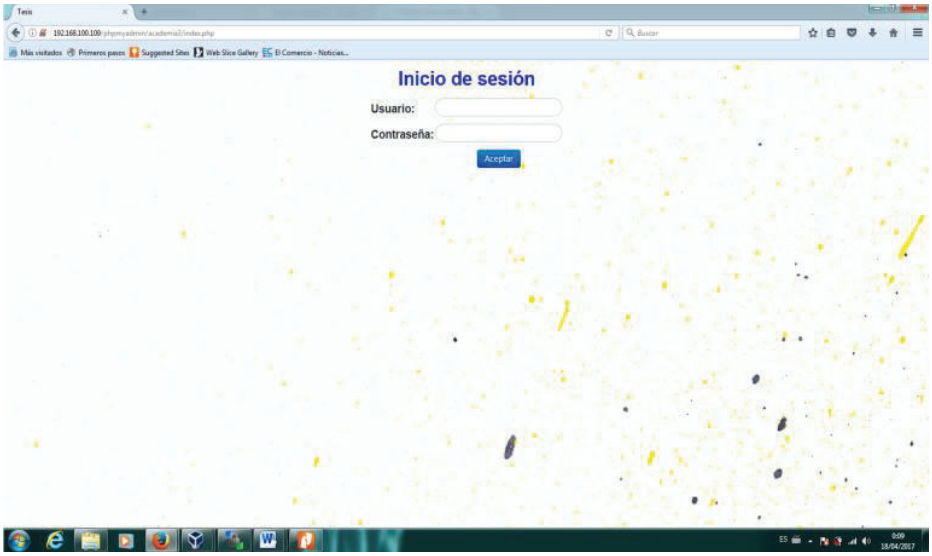


Figura 2.46 Interfaz, Inicio de sesión.

2.7.1.2. Interfaz index2.php

Esta interfaz corresponde a un usuario, cuando en la interfaz de inicio se ingrese como usuario; desde esta página (Figura 2.47) se va a poder acceder al reporte del recorrido, con información en tiempo real de la hora, la fecha, así como el valor recaudado por el cobro de pasajes, y su posición geográfica del recorrido del bus (Figura 2.48). También se podrá acceder a observar la trayectoria seguida por el autobús en diferentes instantes de tiempo del bus interprovincial en un mapa de Google Maps, de acuerdo a la latitud y longitud que la aplicación envié al servidor. (Figura 2.49)

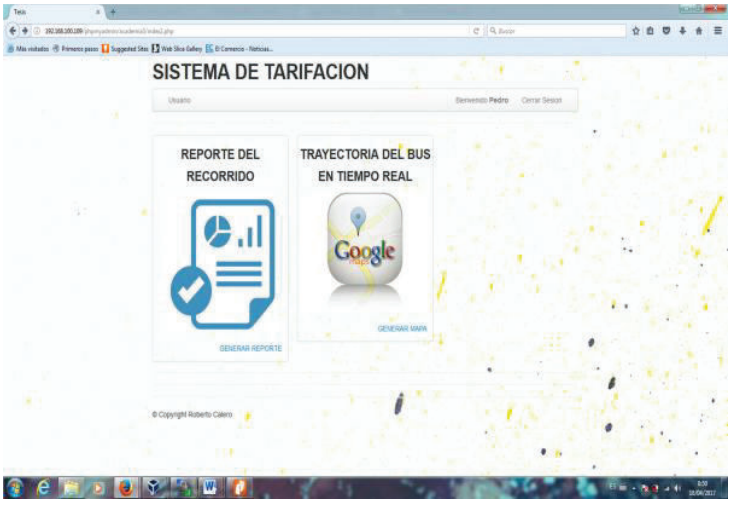
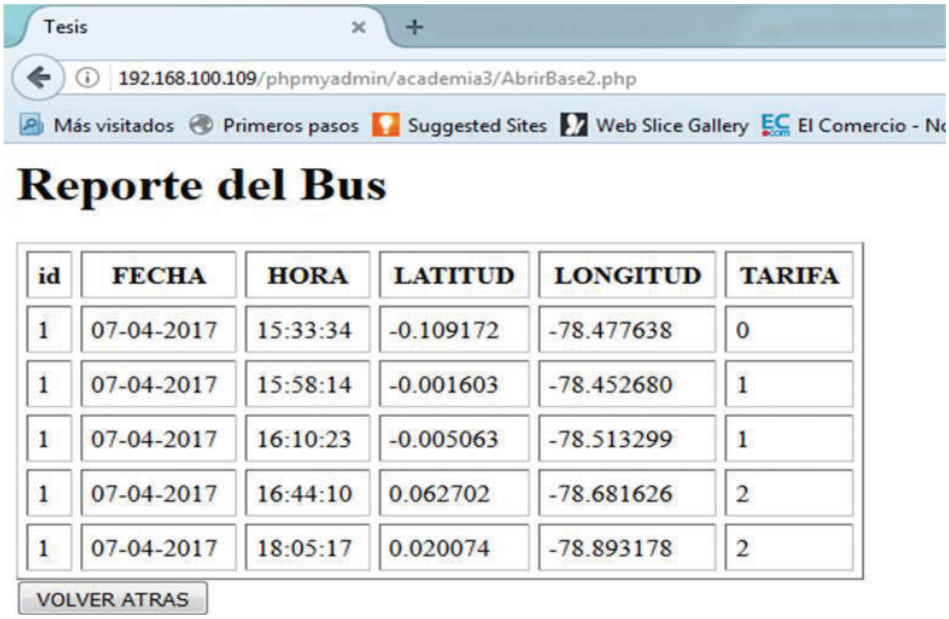


Figura 2.47 Interfaz, Usuario.



Reporte del Bus

id	FECHA	HORA	LATITUD	LONGITUD	TARIFA
1	07-04-2017	15:33:34	-0.109172	-78.477638	0
1	07-04-2017	15:58:14	-0.001603	-78.452680	1
1	07-04-2017	16:10:23	-0.005063	-78.513299	1
1	07-04-2017	16:44:10	0.062702	-78.681626	2
1	07-04-2017	18:05:17	0.020074	-78.893178	2

VOLVER ATRAS

Figura 2.48 Reporte del Bus.

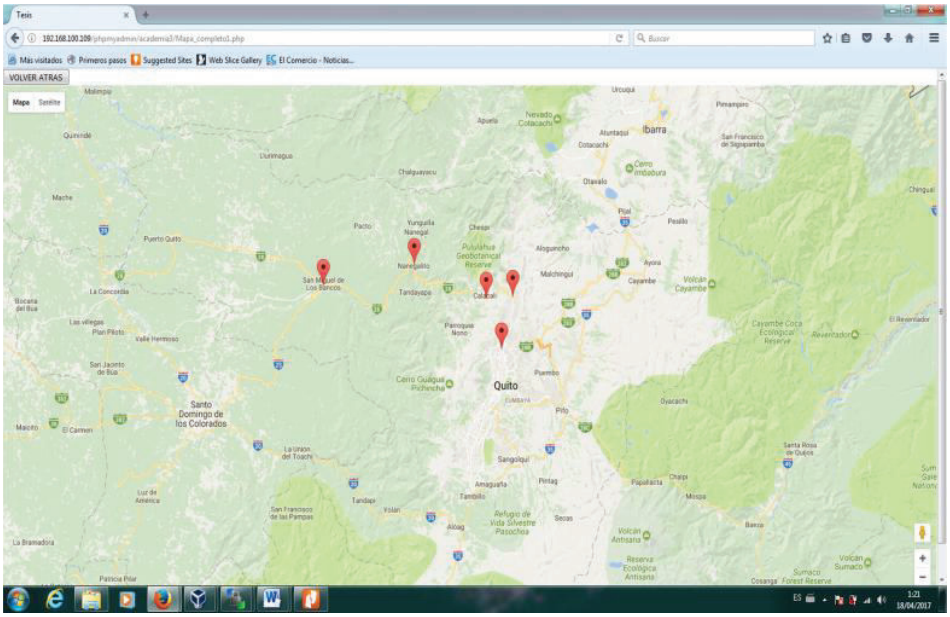


Figura 2.49 Recorrido del bus Quito - Los Bancos.

Cabe señalar que para poder acceder y manipular la base de datos en donde se almacena los datos que envía la aplicación, se debe ingresar a la siguiente URL: http://181.198.137.101/phpmyadmin/, como se muestra en la siguiente Figura 2.50:

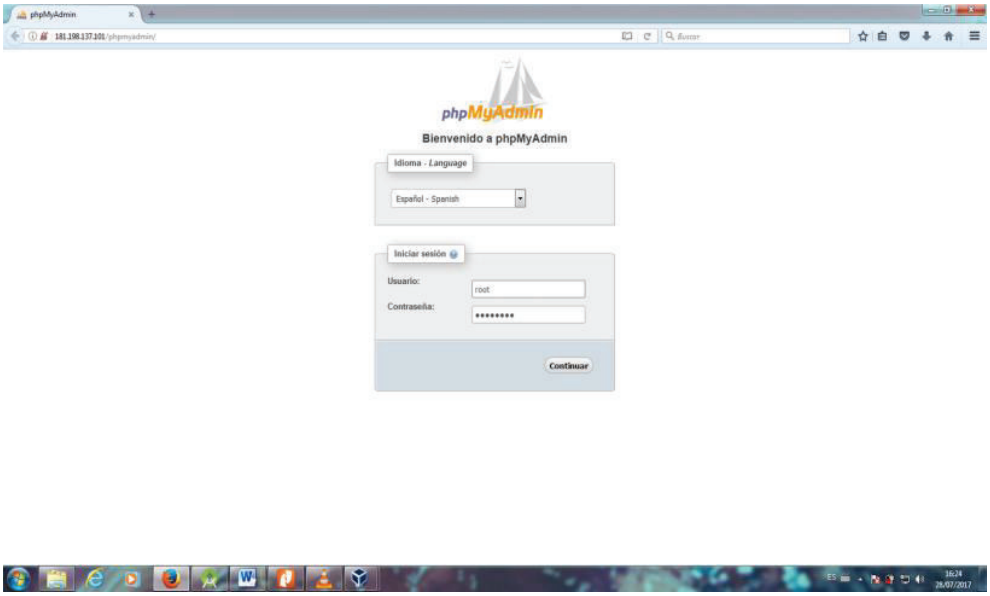


Figura 2.50 Ingreso a la Base de Datos.

CAPÍTULO 3.

PRUEBAS, ANÁLISIS Y COSTOS.

En este capítulo se realizarán las pruebas de funcionamiento de la aplicación y se describirá el costo de desarrollo de la aplicación. Primero se realizará una prueba con el emulador de Android Studio, una vez que se haya comprobado que está funcionando correctamente se realizara la prueba en un dispositivo real, en nuestro caso en una Tablet.

3.1 COMPATIBILIDAD DE LA APP CON VERSIÓN DE ANDROID.

Para compilar y ejecutar la aplicación desarrollada, se va a necesitar que la versión de Android del dispositivo en el cual se va a instalar la app tenga como mínimo la versión 4.2 Jelly Bean con nivel API 17.

3.2 SIMULACIÓN DE LA APLICACIÓN EN ANDROID STUDIO.

3.2.1. EMULADOR DE ANDROID STUDIO.

Para la realizar la simulación en Android Studio de la aplicación se dará clic en el botón Run y aparecerá una interfaz gráfica (Figura 3.1), en donde se selecciona el emulador o un dispositivo conectado.

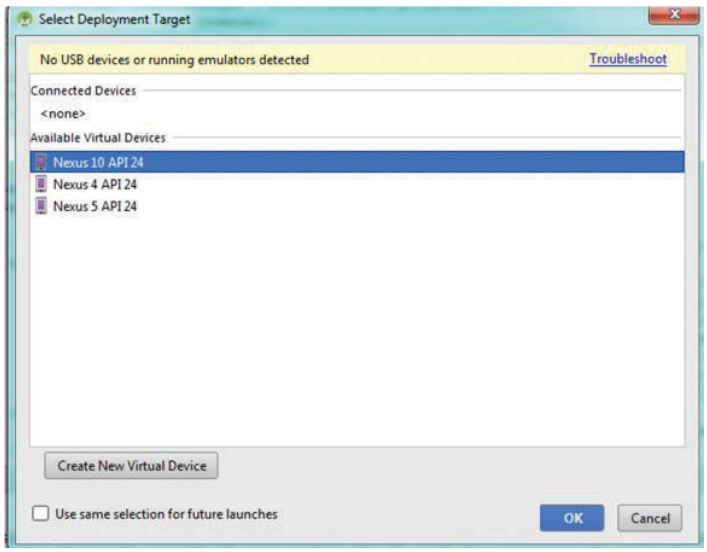


Figura 3.1 Selección del emulador.

En nuestro caso vamos a crear un nuevo emulador, dando clic en **Create New Virtual Device,** en el siguiente cuadro de dialogo se escoge el dispositivo en el cual se va a emular la aplicación (Figura 3.2).

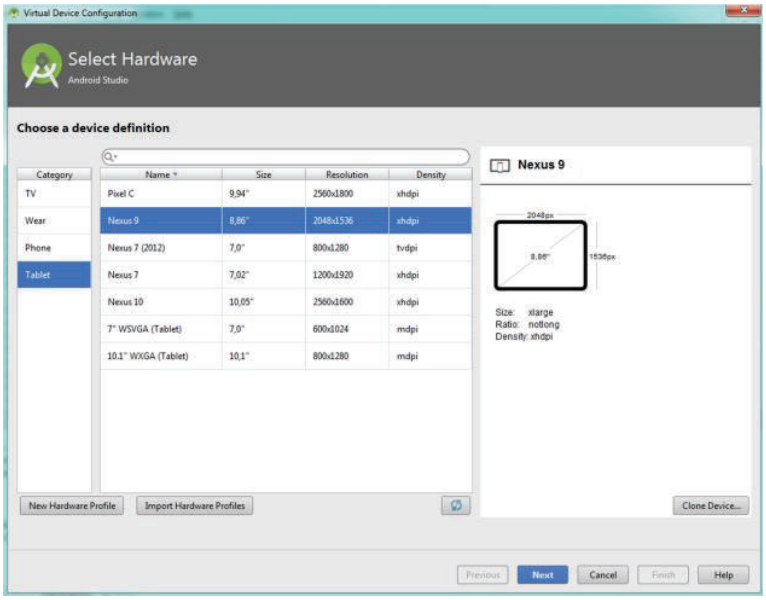


Figura 3.2 Selección del dispositivo.

Una vez que se ha seleccionado el dispositivo, se procede a escoger la versión de Android (Figura 3.3), en nuestro caso escogimos la versión 7.0 Nougat con un nivel de API 24, recordando que como mínimo se debe escoger la versión 4.2 de Android.

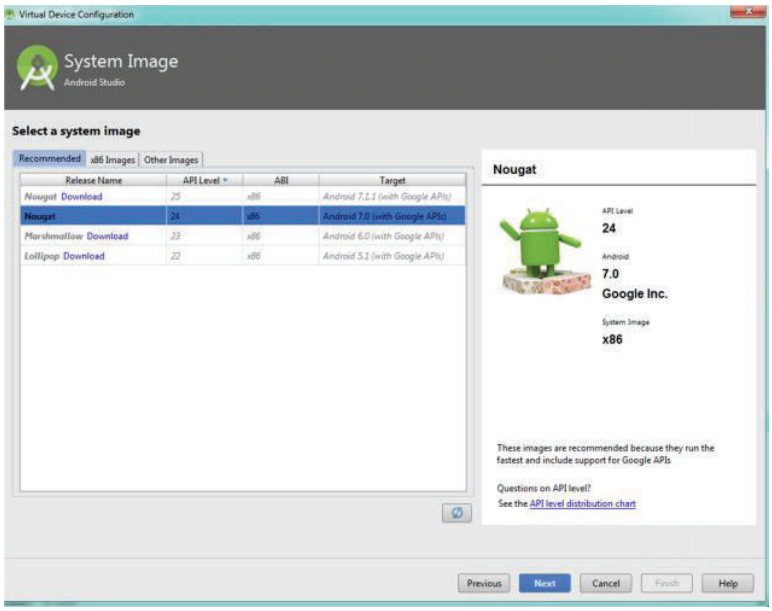


Figura 3.3 Versión de Android.

A continuación en el nuevo cuadro de dialogo (Figura 3.4) nos pedirá que verifiquemos la configuración escogida y finalizamos.

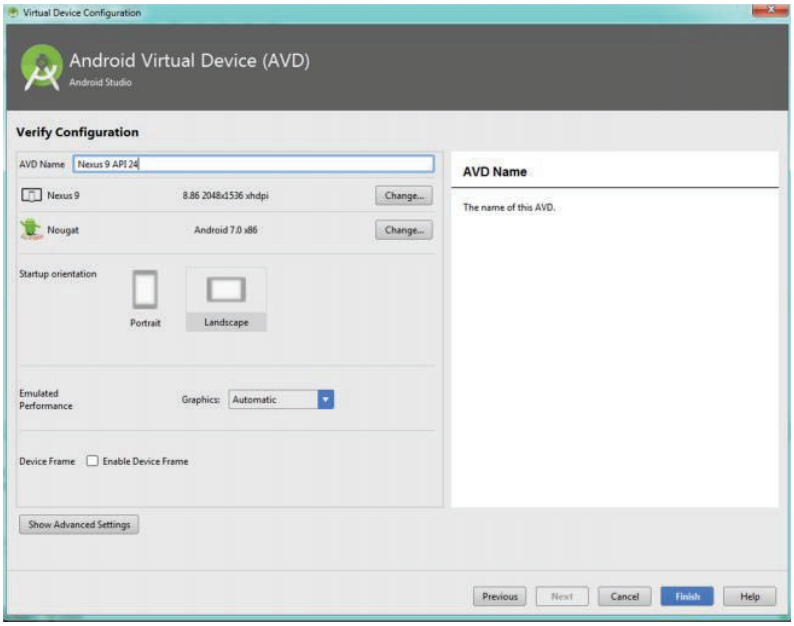


Figura 3.4 Verificación de configuración.

Una vez que se ha dado clic en finalizar se abre de nuevo el primer cuadro de dialogo (Figura 3.1), en donde se escoge el dispositivo a emular; se escogerá el dispositivo que se configuro anteriormente, en nuestro caso es la Tablet Nexus 9, se da clic en ok y se creara el emulador. En la Figura 3.5 se nuestra la aplicación con el nombre TARIFARIO instalada en el emulador de Android Studio.

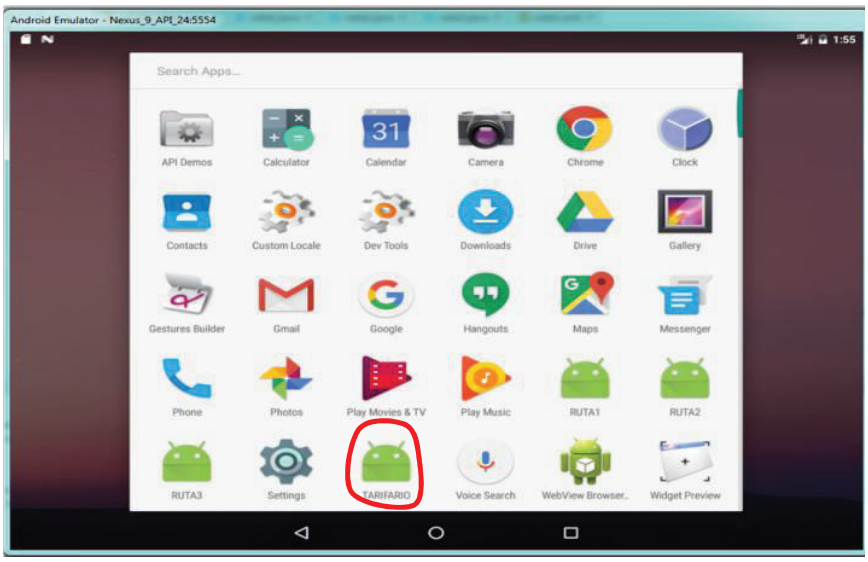


Figura 3.5 Aplicación en el emulador de Android Studio.

3.2.2. RESULTADO DE LAS PRUEBAS EN EL EMULADOR DE ANDROID STUDIO Y EN EL SERVIDOR WEB.

Una vez que se tiene ya instalada la aplicación en el emulador de Android (Figura 3.6), se procederá a realizar las respectivas pruebas de funcionamiento.

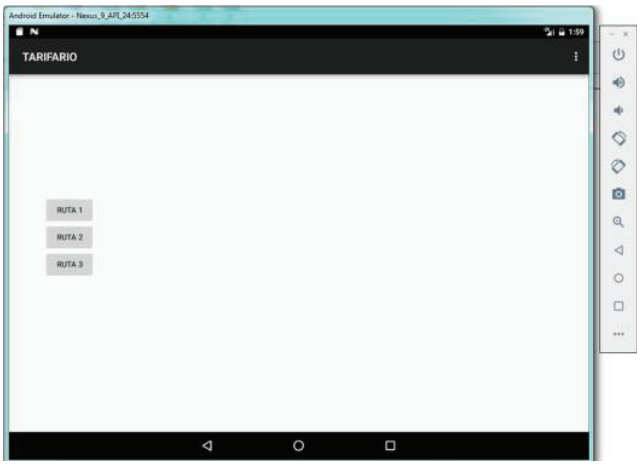


Figura 3.6 Menú de la aplicación.

3.2.2.1. Análisis de los resultados del funcionamiento de la aplicación en el dispositivo virtual.

En la siguiente tabla se presentan los resultados del funcionamiento de la aplicación.

Proceso	Resultado.
Ingreso a la aplicación	Correcto
Ingreso al menú de la aplicación	Correcto
Ingreso a la interfaz “RUTA 2”	Correcto
Selección de la parada en la interfaz	Correcto
Cálculo del valor del pasaje.	Correcto.

Tabla 3.1 Análisis de los resultados del funcionamiento de la aplicación.

3.2.2.2. Análisis de los resultados en el Servidor Web.

En la siguiente Tabla 3.2 se muestra los resultados del envío de datos en tiempo real desde la aplicación hacia el servidor.

Proceso	Resultado.
Ingreso a phpMyAdmin	Correcto
Ingreso a la base de datos “tarifa1”	Correcto
Ingreso a la tabla “bus1”	Correcto
Verificación de los campos de la tabla	Correcto
Verificación del ingreso de los datos desde la app hacia la base de datos.	Correcto.

Tabla 3.2 Análisis de los resultados en el Servidor Web.

3.2.2.3. Análisis de los resultados en la Página Web.

En la siguiente Tabla 3.3 se muestran los resultados del ingreso a la página web para verificar el reporte del bus interprovincial.

Proceso	Resultado.
Ingreso a la página web	Correcto
Ingreso a la sesión “usuario”	Correcto
Ingreso al reporte diario del recorrido del bus.	Correcto
Ingreso para observar la trayectoria del bus en Google Maps	Correcto

Tabla 3.3 Análisis de los resultados en la Página Web.

3.3 INSTALACIÓN DE LA APLICACIÓN EN UN DISPOSITIVO REAL CON OS ANDROID.

La aplicación se instaló en una Tablet Samsung Galaxy Tab 2, con sistema operativo Android 4.2.2 JELLY BEAN.

Existen dos formas de instalar la aplicación desde Android Studio en un dispositivo móvil real, que se explicaran a continuación.

3.3.1. INSTALACIÓN DE LA APLICACIÓN DESDE ANDROID STUDIO VIA USB.

Para instalar la aplicación directamente en el dispositivo móvil, hay que realizar los siguientes pasos:

1. Activar el modo Depuración de USB en el dispositivo, para activar este modo es hay que realizar los siguientes pasos:
    - Vamos a Ajustes (Figura 3.7), Acerca del dispositivo y damos varios clics en Número de compilación, hasta que aparezca Opciones de desarrollador.

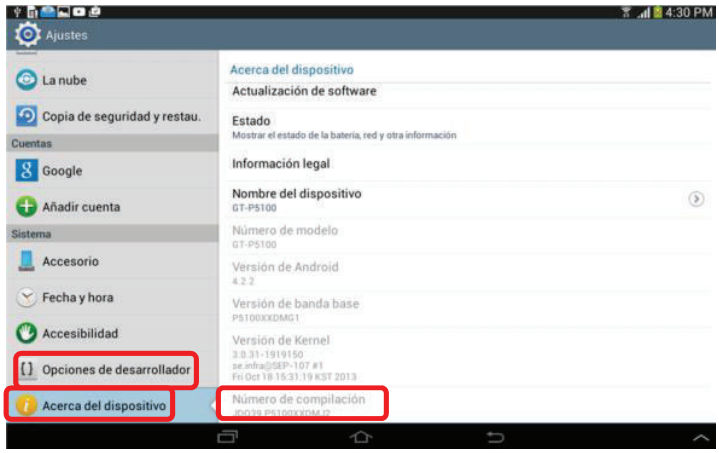


Figura 3.7 Activar opciones de desarrollador.

    - En Opciones de desarrollador, habilitamos Depuración de USB (Figura 3.8):

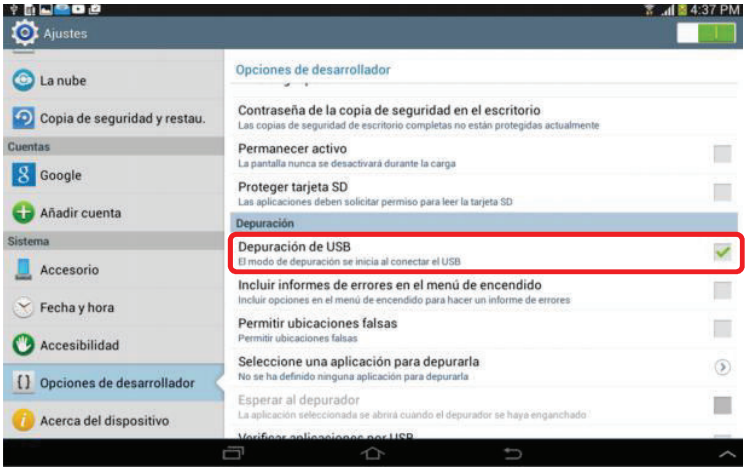


Figura 3.8 Activación Depuración de USB.

2. Una vez que se activado el modo depuración de USB en el dispositivo, se conecta el dispositivo por medio de un cable USB a la computadora.
3. Ya conectado el dispositivo en la computadora, vamos a Android Studio y en la barra de herramientas, seleccionamos el botón Run (Figura 3.9):

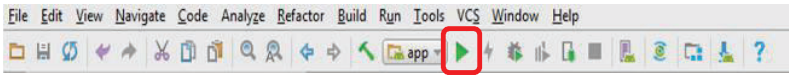


Figura 3.9 Botón Run app.

Cuando damos clic en el botón Run, nos aparece un cuadro de diálogo en donde debemos seleccionar el dispositivo en donde vamos a ejecutar la aplicación, en nuestro caso seleccionamos el dispositivo Samsung GT-P5100 (Figura 3.10), se da clic en OK y se instalara la aplicación en el dispositivo (Figura 3.11):

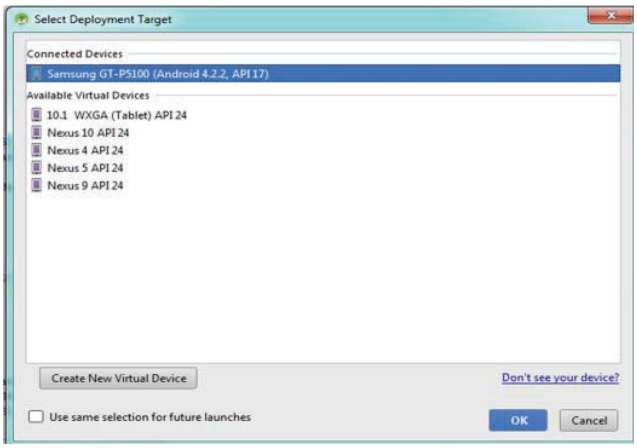


Figura 3.10 Selección del dispositivo.

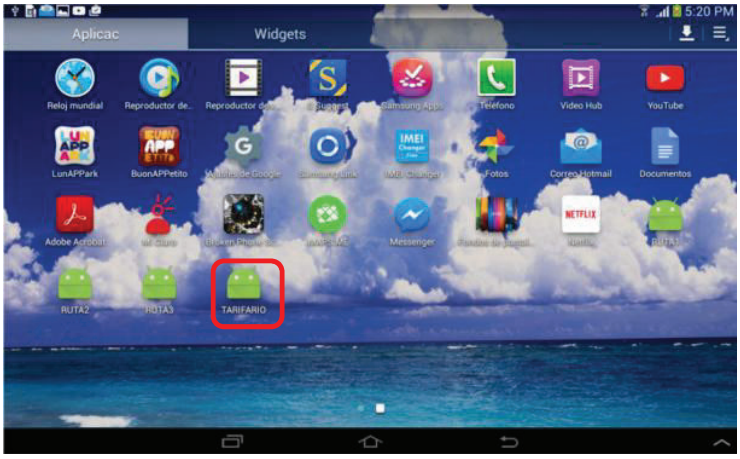


Figura 3.11 Aplicación instalada en la Tablet.

3.3.2. INSTALACIÓN DE LA APLICACIÓN DESDE ANDROID STUDIO MEDIANTE EL ARCHIVO .APK.

Para instalar la aplicación en el dispositivo, también se puede realizar por medio del archivo apk, a continuación los pasos para realizar dicha instalación:

1. Nos vamos a dirigir a la barra de herramientas de Android Studio, y vamos a seleccionar Build, nos aparecerá un menú y seleccionamos Build APK (Figura 3.12)

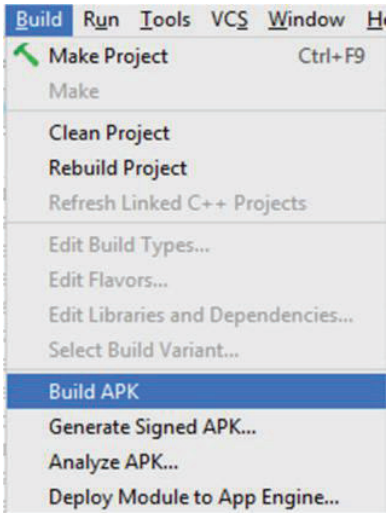


Figura 3.12 Generación del archivo .apk.

Una vez seleccionado Build APK, se genera dicho archivo (Figura 3.13):

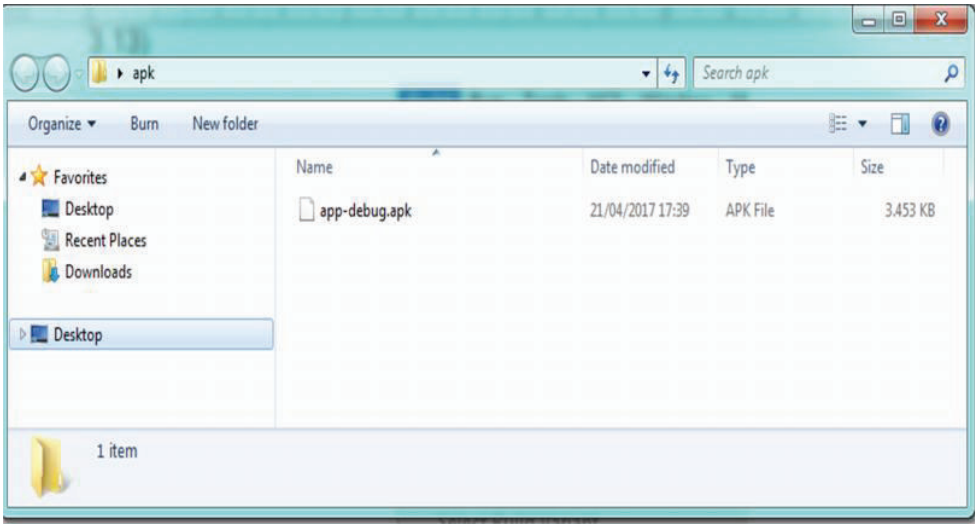


Figura 3.13 Archivo .apk.

2. Ya generado el archivo .apk, se procede a copiar este archivo en la memoria interna del dispositivo o en la memoria externa (Figura 3.14):

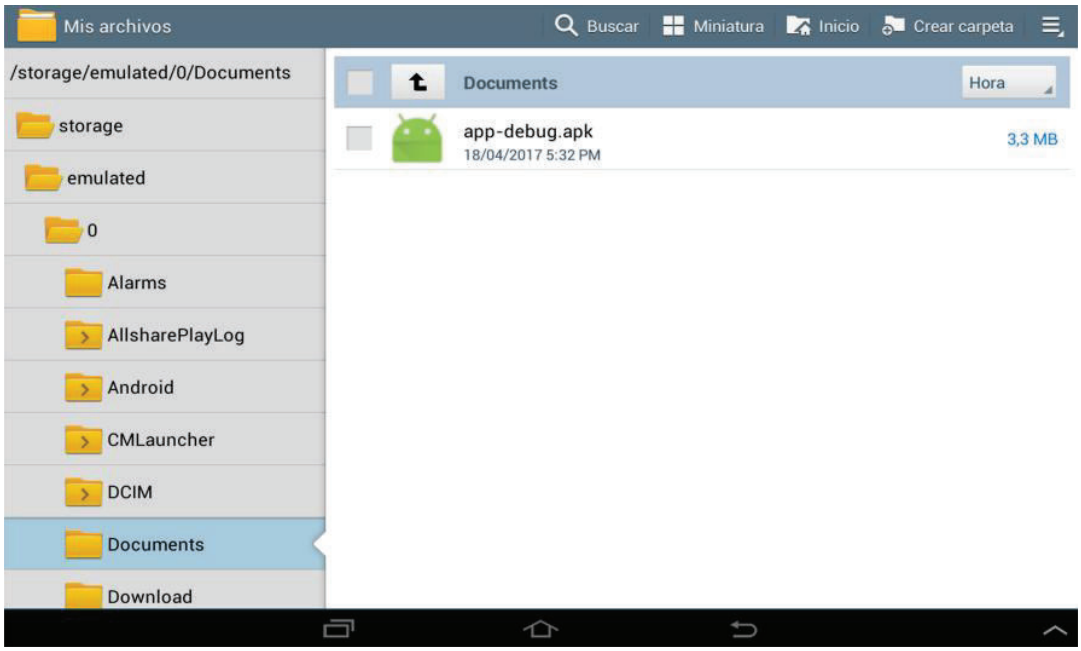


Figura 3.14 Localización del archivo .apk.

3. Una vez que se encuentra el archivo .apk en la memoria del dispositivo, se procede a ejecutar dicho archivo, y se instalara la aplicación (Figura 3.15):

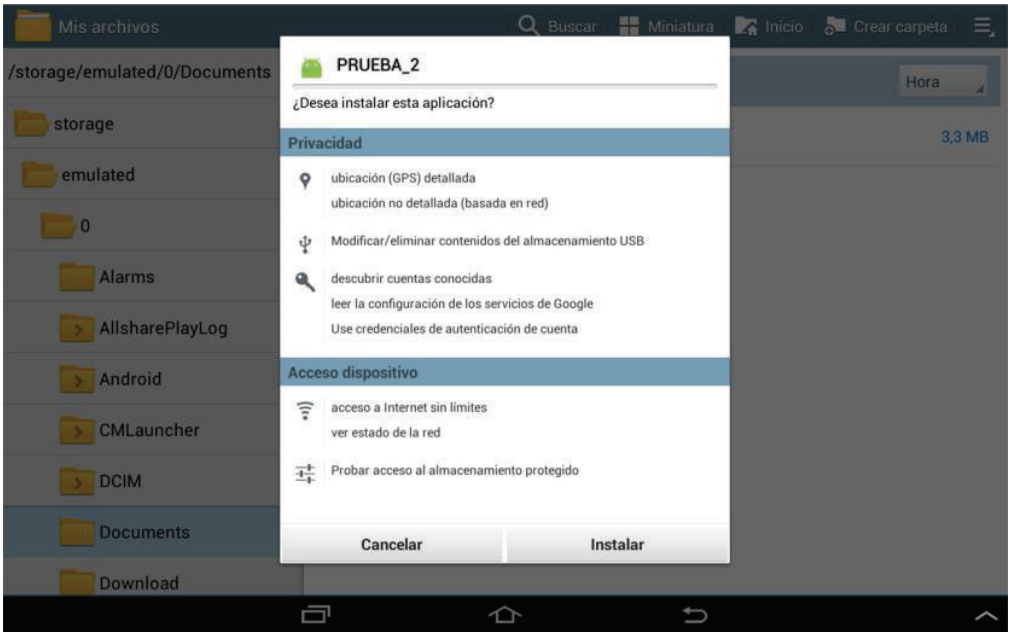


Figura 3.15 Instalación de la aplicación.

3.3.3. RESULTADO DE LAS PRUEBAS EN EL DISPOSITIVO REAL.

Ya instalada la aplicación en el dispositivo real se pueden realizar las pruebas de funcionamiento. Para ello se realizó la prueba de campo tanto de ida como de regreso en un bus Interprovincial de la Cooperativa de Transportes Zaracay, en la ida se abordó el bus en el Terminal Terrestre Quitumbe de Quito. El bus salió del terminal aproximadamente a las 12:45 pm del día martes 6 de junio de 2017 y cubría la ruta Quito – Santo Domingo, y la hora de llegada al Terminal Terrestre de Santo Domingo aproximadamente era a las 3:30 pm. De regreso el bus salió del Terminal Terrestre de Santo Domingo a las 4:00 pm, y la hora de llegada aproximadamente al Terminal Terrestre de Quitumbe fue a las 7:00 pm. Con la ayuda del Sr. controlador del bus interprovincial se procedió a realizar las pruebas de funcionamiento de la aplicación que se presentan a continuación:

3.3.3.1. Resultados del funcionamiento de la aplicación en el dispositivo real.

3.3.3.1.1. Ingreso a la aplicación.

Primero se procedió a explicarle el funcionamiento de la aplicación al controlador del Bus, ya que él es el encargado del cobro de pasajes. Una vez terminada la explicación se procedió a ingresar en la interfaz principal de la aplicación, y se verificó que se podía ingresar correctamente a la aplicación instalada en la Tablet, como se muestra en la Figura 3.16:

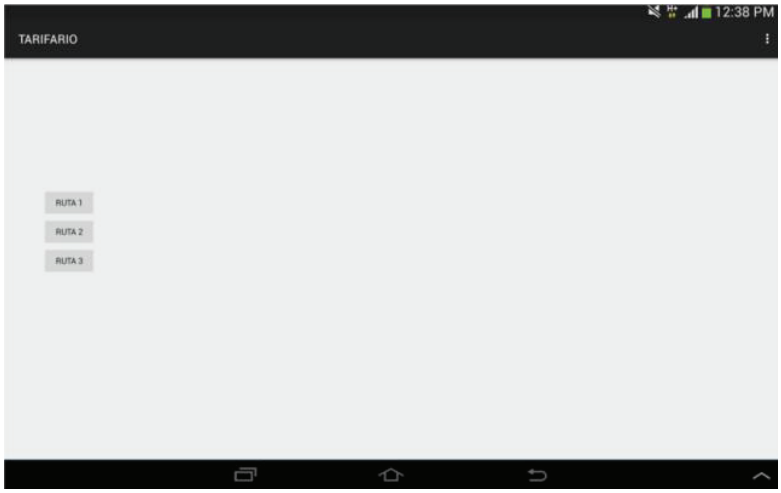


Figura 3.16 Ingreso a la aplicación.

3.3.3.1.2. Ingreso a la interfaz "RUTA 2" (Ruta Quito – Santo Domingo).

En la anterior Figura 3.16, se observa en la interfaz principal tres botones, cada botón representa a una ruta diferente, la ruta Quito – Santo Domingo está ubicada en el botón "'RUTA 2", se comprobó con el Oficial que al dar clic en dicho botón se ingresa correctamente a la interfaz ruta 2, en donde se observa el mapa del recorrido desde el Terminal Quitumbe hasta el Terminal de Santo Domingo y las diferentes paradas a lo largo del recorrido (Figura 3.17).

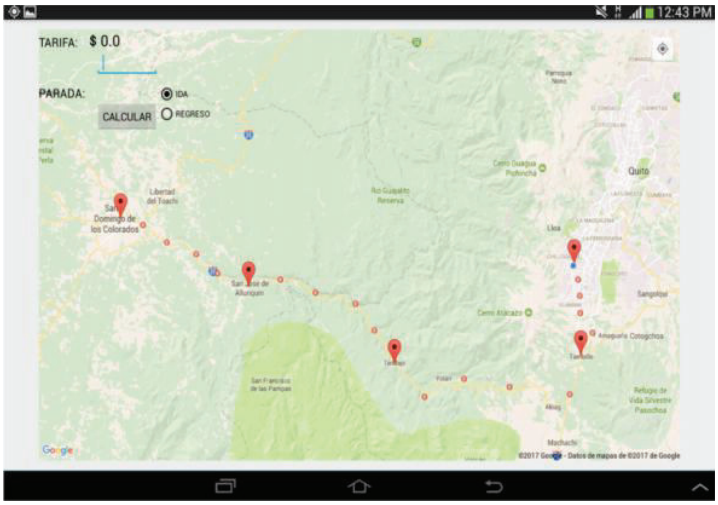


Figura 3.17 Ruta Quito – Santo Domingo.

Como se mencionó anteriormente en ésta interfaz se tiene la ruta Quito – Santo Domingo, que está compuesta por marcadores de Google Maps que nos ayudan a identificar las diferentes paradas intermedias a lo largo del recorrido (Figuras 3.18 – 3.22).

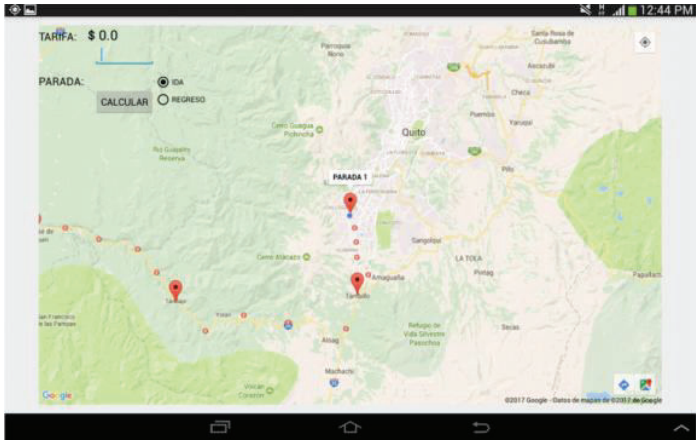


Figura 3.18 PARADA 1 (Terminal Terrestre Quitumbe).

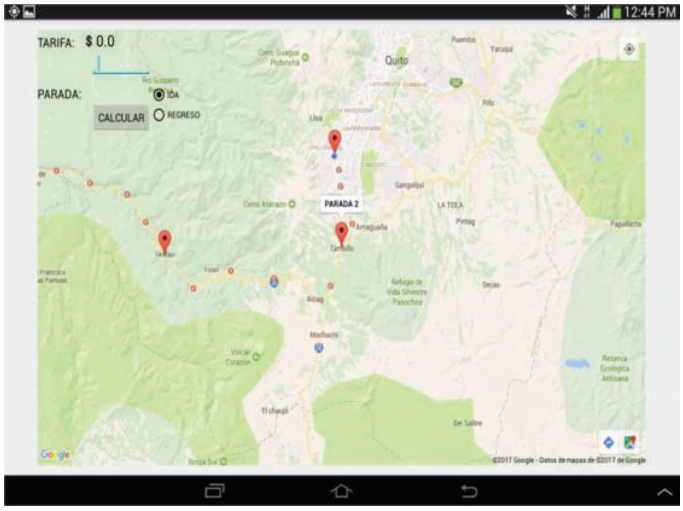


Figura 3.19 PARADA 2 (Tambillo).

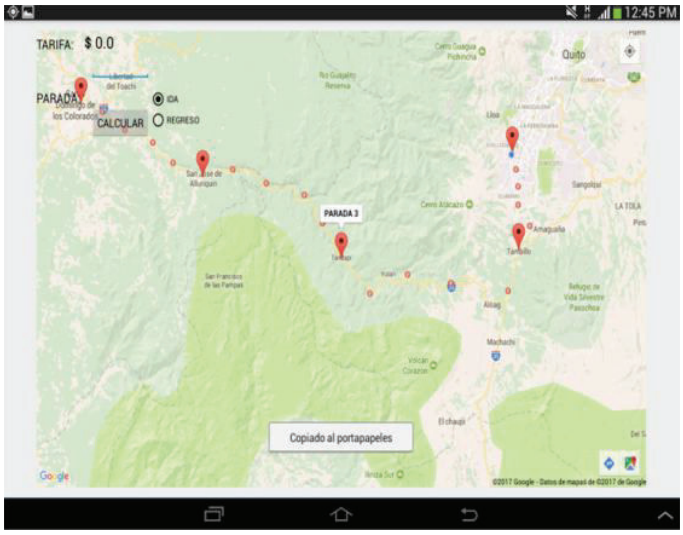


Figura 3.20 PARADA 3 (Tandapi).

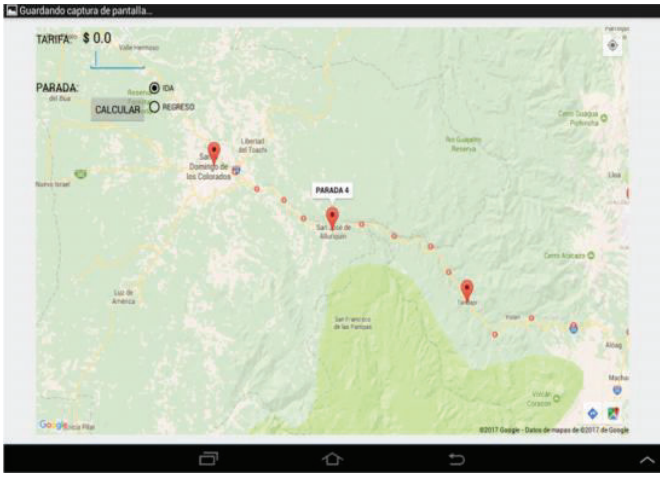


Figura 3.21 PARADA 4 (Alluriquin).

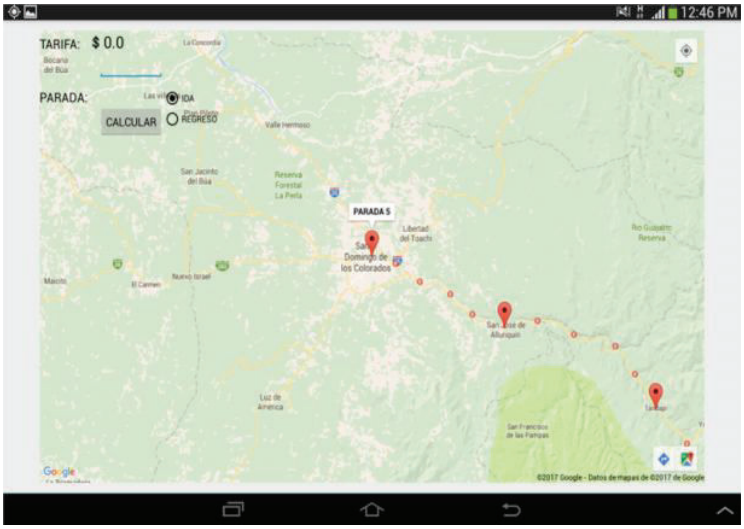


Figura 3.22 PARADA 5 (Terminal Terrestre Santo Domingo).

3.3.3.1.3. Cálculo del valor del pasaje (ida).

Una vez que el bus salió del Terminal Terrestre Quitumbe, hizo la primera parada para recoger pasajeros en la PARADA 2 (Tambillo), Figura 3.23. A continuación se procedió a cobrar los pasajes de los usuarios que abordaron el bus, fueron en total 6 pasajeros, cinco usuarios se dirigían a Santo Domingo (PARADA 5), y un usuario a Alluriquin (PARADA 4). Para realizar el cálculo del valor del pasaje el Oficial ingreso el número de parada a la que se dirigía el usuario, en nuestro caso se dirigían a la parada 4 y 5. Ya ingresado el número de parada, se presiona el botón "CALCULAR", y nos mostrara el valor a pagar a dicho destino, como se muestra en la Figura 3.24. También cuando se presiona el botón "CALCULAR" los datos que genera la aplicación son enviados al servidor Web y se almacenan en la base de datos como se muestran en la Figura 3.25

En la Figura 3.25 se puede observar el valor del pasaje que cancelaron los 6 pasajeros que abordaron el bus en Tambillo. Por ejemplo a la 13:22:20 se cobró un valor de 2.50 USD, esta es la tarifa establecida desde Tambillo a Santo Domingo. También se observa el valor que canceló el usuario que se dirigía a Alluriquin, de 2 USD.

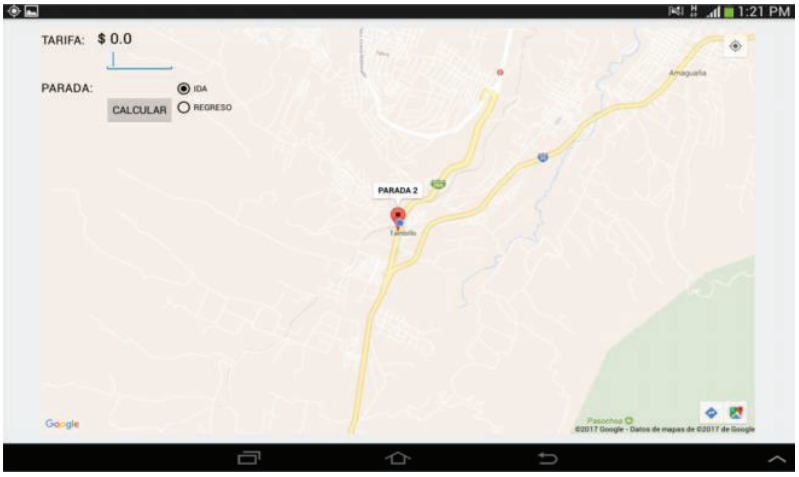


Figura 3.23 Parada Tambillo.

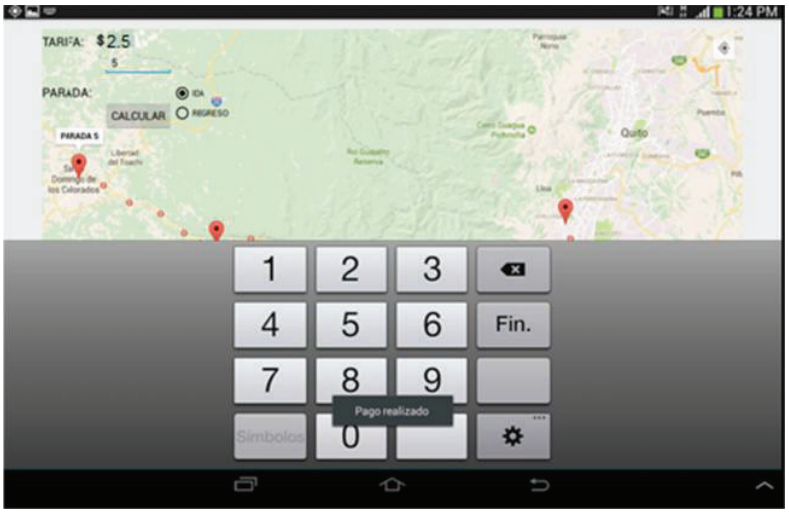


Figura 3.24 Ejemplo del cálculo del pasaje.

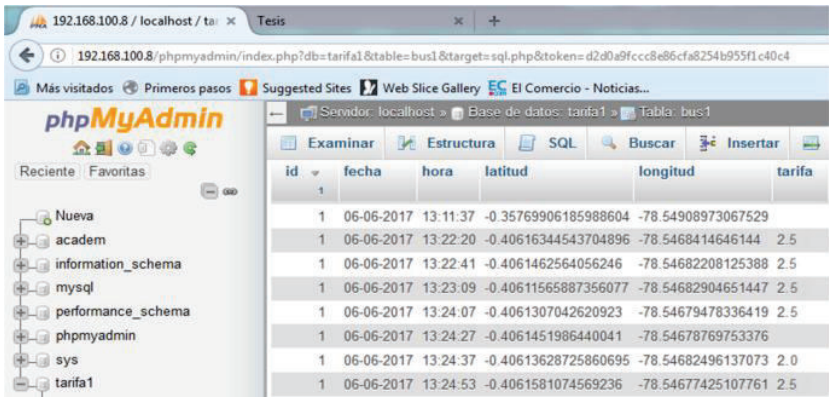


Figura 3.25 Datos ingresados en la parada Tambillo.

La segunda parada que realizó el bus para recoger pasajeros fue en la Y de Aloag alrededor de la 13:29 pm, aquí abordaron el bus 10 pasajeros, 8 se dirigían a Santo Domingo (PARADA 5) y 2 a Alluriquin (PARADA 4): en la Figura 3.26 se observa el valor cancelado por cada pasajero, que es de 2.50 USD de los usuarios que se dirigían a Santo Domingo y de 2 USD de los usuarios que se dirigía a Alluriquin.

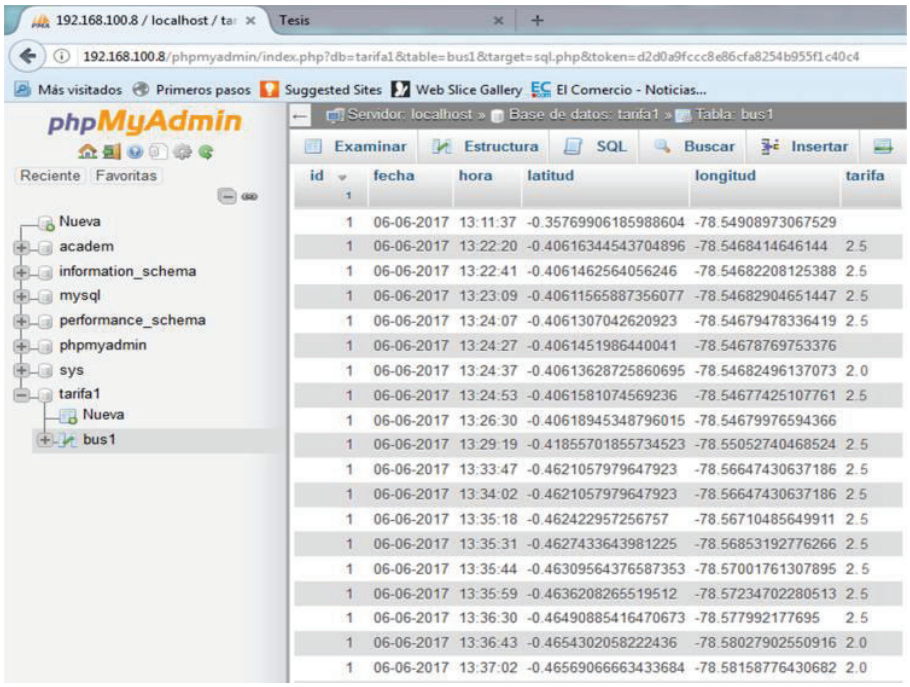


Figura 3.26 Datos ingresados en la parada Y de Aloag.

La última parada que realizó el bus fue en Alluriquin, aquí un pasajero abordo el bus y se dirigía a Santo Domingo, y pago una tarifa de 0.50 USD.

3.3.3.1.4. Cálculo del valor del pasaje (regreso).

Como se indicó anteriormente el bus de regreso de Santo Domingo a Quito salió a las 4:00 pm, y la primera parada que realizó para recoger pasajeros fue en Alluriquin (PARADA 4), abordaron al bus 3 usuarios que se dirigían a Quito, la segunda parada que realizó el bus fue en Tandapi, en donde abordaron al bus 3 usuarios que también se dirigían a Quito. En la Figura 3.27 se observa el valor cancelado de 2.50 USD por cada usuario que abordaron el bus en Alluriquin y el

valor cancelado por los 3 usuarios que se subieron en Tandapi, que fue de 2 USD hasta Quito.

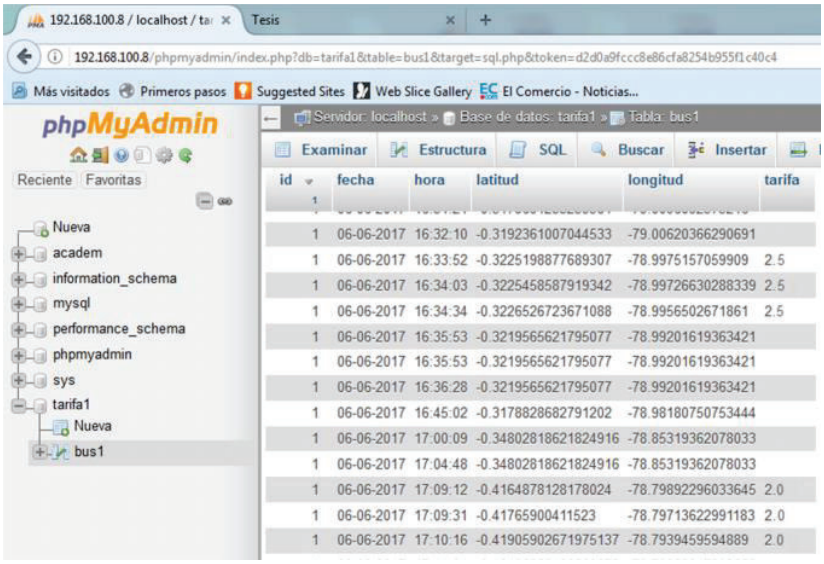


Figura 3.27 Datos ingresados en la parada Alluriquin y Tandapi.

A continuación se presenta en la Tabla 3.4 el resultado del análisis del funcionamiento de la aplicación en tiempo real, que se realizó conjuntamente con el controlador del bus Interprovincial.

Proceso	Resultado.
Ingreso a la aplicación	Correcto
Ingreso al menú de la aplicación	Correcto
Ingreso a la interfaz “RUTA 2”	Correcto
Selección de la parada en la interfaz	Correcto
Cálculo del valor del pasaje.	Correcto.

Tabla 3.4 Análisis de los resultados, dispositivo real.

3.3.3.2. Análisis de los resultados en el Servidor Web.

Los datos que genera la aplicación son enviados en tiempo real a la base de datos del servidor web. A continuación se muestra en la Figura 3.31 los resultados:

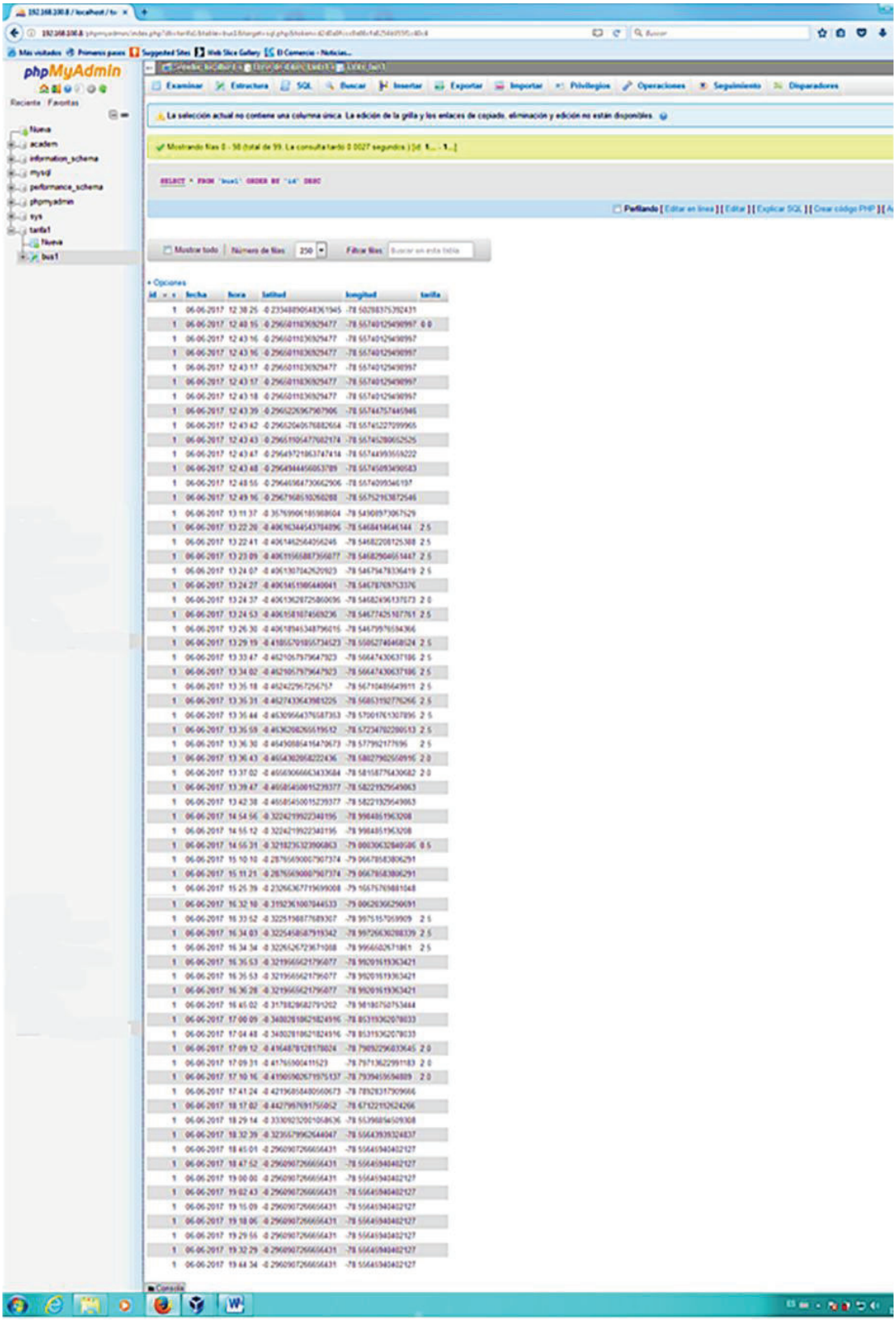


Figura 3.28 Tabla “bus1” llena por el envió de datos desde la aplicación.

En la siguiente Tabla 3.5 se muestran los resultados del análisis del envió de los datos en tiempo real desde la aplicación instalada en un dispositivo real hacia el servidor.

Proceso	Resultado.
Ingreso a phpMyAdmin	Correcto
Ingreso a la base de datos “tarifa1”	Correcto
Ingreso a la tabla “bus1”	Correcto
Verificación de los campos de la tabla	Correcto
Verificación del ingreso de los datos desde la app hacia la base de datos.	Correcto.

Tabla 3.5 Análisis de los resultados en el Servidor Web, dispositivo real.

3.3.3.3. Análisis de los resultados en la Página Web.

En la página Web diseñada, el usuario interesado en la información podrá ingresar a dicha página y acceder a un reporte diario del recorrido del bus, y a visualizar el recorrido del bus por medio de un mapa de Google Maps; en nuestro caso se mostrará el reporte del recorrido del bus de la ruta Quito-Santo Domingo, tanto de ida como de vuelta y el valor total recaudado por concepto de pasajes de los usuarios que abordaron el bus a lo largo del recorrido. Cada usuario tendrá asignado una Contraseña y un nombre de Usuario para acceder a la información antes mencionada.

Para ingresar en la página web diseñada, el usuario puede acceder a dicha página desde la siguiente URL: http://181.198.137.101/phpmyadmin/academia3/ y desde cualquier navegador que el usuario escoja, una vez que se ingresa a esa dirección nos aparecerá la interfaz de inicio (Figura 3.29); digitando el respectivo nombre de usuario y contraseña se accedera a la interfaz de reporte y trayectoria del bus, como se muestra a continuación en la Figura 3.30:

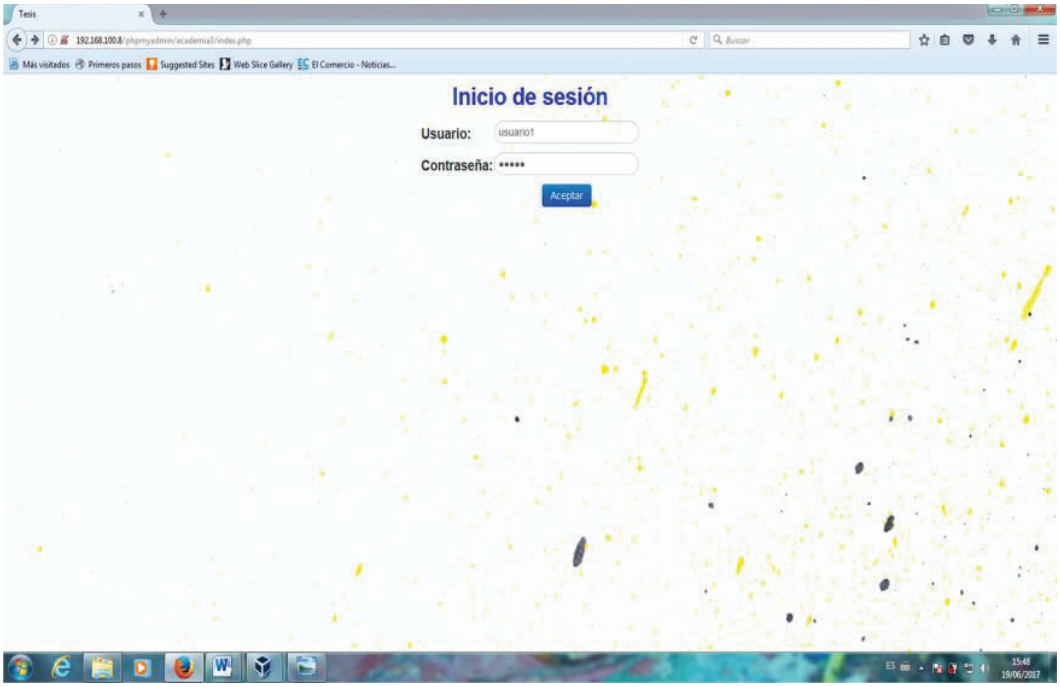


Figura 3.29 Interfaz de Inicio.

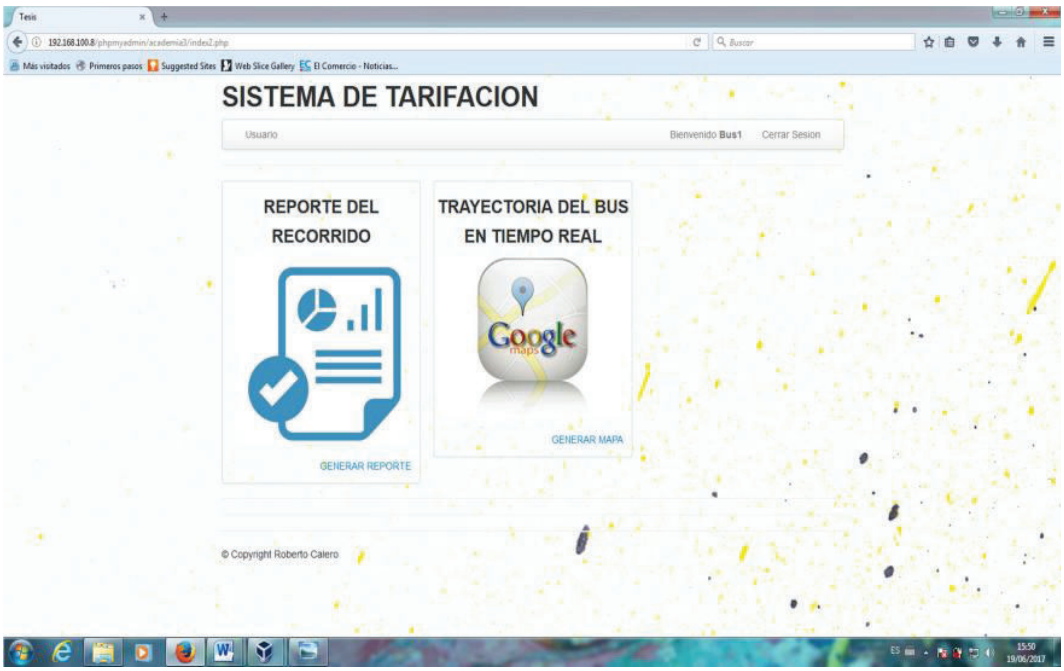


Figura 3.30 Interfaz de reporte y trayectoria del bus.

A continuación vamos a acceder al reporte del recorrido, y nos generará el reporte del bus como se muestra en la siguiente Figura 3.31:

Reporte del Bus

id	FECHA	HORA	LATITUD	LONGITUD	TARIFA
1	06-06-2017	12:38:25	-0.2334889014836194	-78.50288373392431	
1	06-06-2017	12:40:13	-0.2965010336929477	-78.55740129498997	0.0
1	06-06-2017	12:43:16	-0.2965010336929477	-78.55740129498997	
1	06-06-2017	12:43:16	-0.2965010336929477	-78.55740129498997	
1	06-06-2017	12:43:17	-0.2965010336929477	-78.55740129498997	
1	06-06-2017	12:43:17	-0.2965010336929477	-78.55740129498997	
1	06-06-2017	12:43:18	-0.2965010336929477	-78.55740129498997	
1	06-06-2017	12:43:39	-0.2965229867907906	-78.55744757445946	
1	06-06-2017	12:43:42	-0.2965204017688264	-78.55745227099965	
1	06-06-2017	12:43:43	-0.2965110547702174	-78.55745280652525	
1	06-06-2017	12:43:47	-0.2964972186374744	-78.55744995559222	
1	06-06-2017	12:43:48	-0.2964944456053789	-78.55745093490583	
1	06-06-2017	12:48:55	-0.2964698473066290	-78.55745099346197	
1	06-06-2017	12:49:16	-0.2967168310260288	-78.55752163872146	
1	06-06-2017	12:52:50	-0.3038869993909842	-78.55919895898004	
1	06-06-2017	12:53:15	-0.3040008786990736	-78.55919625407351	0.0
1	06-06-2017	12:55:40	-0.3078261366717956	-78.55944275810256	
1	06-06-2017	13:08:02	-0.3415625383789186	-78.54905429155146	
1	06-06-2017	13:11:37	-0.3576990818598604	-78.54908973067529	
1	06-06-2017	13:22:20	-0.4061634454393704896	-78.54684146466144	2.5
1	06-06-2017	13:22:41	-0.4061462564056246	-78.54682208125388	2.5
1	06-06-2017	13:23:09	-0.4061356588736077	-78.54682904651447	2.5
1	06-06-2017	13:24:07	-0.4061307042020928	-78.54679478386419	2.5
1	06-06-2017	13:24:27	-0.4061431966440041	-78.54678769753376	
1	06-06-2017	13:24:37	-0.4061362872586095	-78.54682496137073	2.0
1	06-06-2017	13:24:53	-0.4061810374589236	-78.54677425107761	2.5
1	06-06-2017	13:26:30	-0.4061894534879015	-78.54679976594366	
1	06-06-2017	13:29:19	-0.4183170185374529	-78.55052740488524	2.5
1	06-06-2017	13:33:47	-0.4621057979647923	-78.56647430637186	2.5
1	06-06-2017	13:34:02	-0.4621057979647923	-78.56647430637186	2.5
1	06-06-2017	13:35:18	-0.4624229752567	-78.56703485649911	2.5
1	06-06-2017	13:35:31	-0.4627433649812225	-78.56833927776286	2.5
1	06-06-2017	13:35:44	-0.4630956437858753	-78.57001781307895	2.5
1	06-06-2017	13:35:59	-0.4636208265319512	-78.57234702280513	2.5
1	06-06-2017	13:36:30	-0.4649088541604767	-78.5779921776895	2.5
1	06-06-2017	13:36:43	-0.4654302058222436	-78.58027902550916	2.0
1	06-06-2017	13:37:02	-0.4658906666343684	-78.58158776430682	2.0
1	06-06-2017	13:39:47	-0.4658345001529377	-78.58221929549863	
1	06-06-2017	13:42:38	-0.4658345001529377	-78.58221929549863	
1	06-06-2017	14:54:56	-0.3224219922340195	-78.9984831963208	
1	06-06-2017	14:55:12	-0.3224219922340195	-78.9984831963208	
1	06-06-2017	14:55:31	-0.3218233523906063	-79.00030632840186	0.5
1	06-06-2017	15:10:10	-0.2878369000790737 4	-79.06678183806291	
1	06-06-2017	15:11:21	-0.2878369000790737 4	-79.06678183806291	
1	06-06-2017	15:25:39	-0.2320636771909008	-79.16573698831048	
1	06-06-2017	15:25:54	-0.2320636771909008	-79.16573698831048	
1	06-06-2017	15:26:39	-0.2387052532494612	-79.17182373535937	0.0
1	06-06-2017	15:40:11	-0.2387968414738295 4	-79.18991698680786	
1	06-06-2017	15:40:26	-0.2387968414738295 4	-79.18991698680786	
1	06-06-2017	15:55:06	-0.2387968414738295 4	-79.18991698680786	
1	06-06-2017	15:55:22	-0.2387968414738295 4	-79.18991698680786	
1	06-06-2017	15:57:30	-0.2387968414738295 4	-79.18991698680786	
1	06-06-2017	16:05:46	-0.2387977311750619 6	-79.17019393802008	
1	06-06-2017	16:12:40	-0.2373660881057153 3	-79.17178113758101	
1	06-06-2017	16:13:09	-0.2481609370104228	-79.14280449174908	
1	06-06-2017	16:16:13	-0.2515629299615516	-79.13790714103223	
1	06-06-2017	16:16:39	-0.2606874158764412	-79.11902170153344	
1	06-06-2017	16:20:47	-0.2626062373590525	-79.11495869077984	
1	06-06-2017	16:23:06	-0.2626062373590525	-79.11495869077984	
1	06-06-2017	16:25:11	-0.2928314302613358 4	-79.06227935269919	
1	06-06-2017	16:25:26	-0.2947527906732865	-79.05974076744684	
1	06-06-2017	16:27:52	-0.2972373837677905	-79.05670871585926	
1	06-06-2017	16:28:02	-0.2972373837677905	-79.05670871585926	
1	06-06-2017	16:31:03	-0.3164425027463174 7	-79.01320980108783	
1	06-06-2017	16:31:24	-0.3179691233239381	-79.0096982375216	
1	06-06-2017	16:32:10	-0.3192361007044533	-79.00620386290691	
1	06-06-2017	16:33:52	-0.3225198877688307	-78.99733137059909	2.5
1	06-06-2017	16:34:03	-0.3225435857819342	-78.99726630288339	2.5
1	06-06-2017	16:34:34	-0.3226262723670088	-78.9956502671861	2.5
1	06-06-2017	16:35:33	-0.3219503621793077	-78.99201619363421	
1	06-06-2017	16:35:53	-0.3219503621793077	-78.99201619363421	
1	06-06-2017	16:36:28	-0.3219503621793077	-78.99201619363421	
1	06-06-2017	16:43:02	-0.3178828682791202	-78.98180750753444	
1	06-06-2017	17:00:09	-0.3480281862182491 6	-78.85319362078033	
1	06-06-2017	17:04:48	-0.3480281862182491 8	-78.85319362078033	
1	06-06-2017	17:09:12	-0.4164878128178024	-78.78892296033643	2.0
1	06-06-2017	17:09:31	-0.4178390041152 9	-78.7971862299188 3	2.0
1	06-06-2017	17:10:16	-0.4195190267197513 7	-78.79394395948892	2.0
1	06-06-2017	17:42:24	-0.4239683848506067 3	-78.7892817909666	
1	06-06-2017	18:17:02	-0.4427997891755052	-78.67322132824266	
1	06-06-2017	18:29:14	-0.4330923200101863 6	-78.55398945093083	
1	06-06-2017	18:32:39	-0.4235579962644047	-78.55043930324837	
1	06-06-2017	18:45:01	-0.2960907266656431	-78.55045940402127	
1	06-06-2017	18:47:32	-0.2960907266656431	-78.55045940402127	
1	06-06-2017	19:00:00	-0.2960907266656431	-78.55045940402127	
1	06-06-2017	19:02:43	-0.2960907266656431	-78.55045940402127	
1	06-06-2017	19:13:09	-0.2960907266656431	-78.55045940402127	
1	06-06-2017	19:18:06	-0.2960907266656431	-78.55045940402127	
1	06-06-2017	19:29:35	-0.2960907266656431	-78.55045940402127	
1	06-06-2017	19:32:29	-0.2960907266656431	-78.55045940402127	
1	06-06-2017	19:44:34	-0.2960907266656431	-78.55045940402127	
				TOTAL	52.5

Figura 3.31 Reporte del Bus, ruta Quito-Santo Domingo.

La segunda opción que nos presenta la página web como se dijo anteriormente es visualizar el recorrido del bus en Google Maps, como se muestra a continuación en las Figuras 3.32 – 3.33:

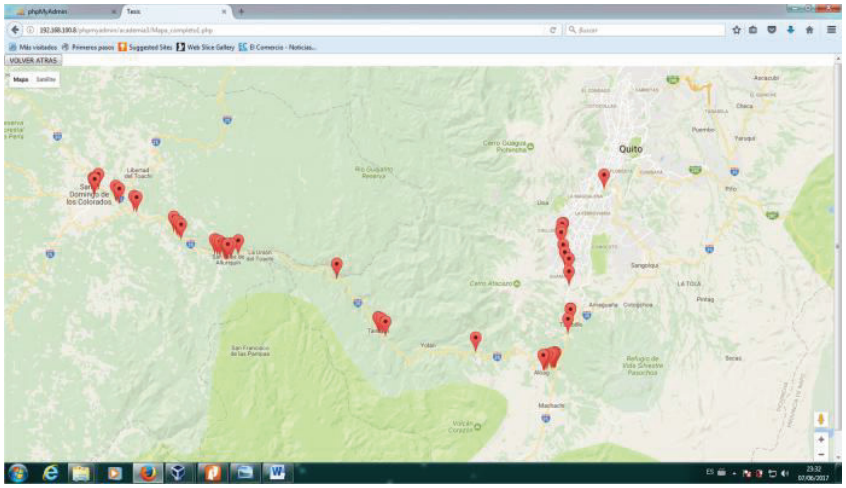


Figura 3.32 Visualización de la ruta Quito-Santo Domingo.

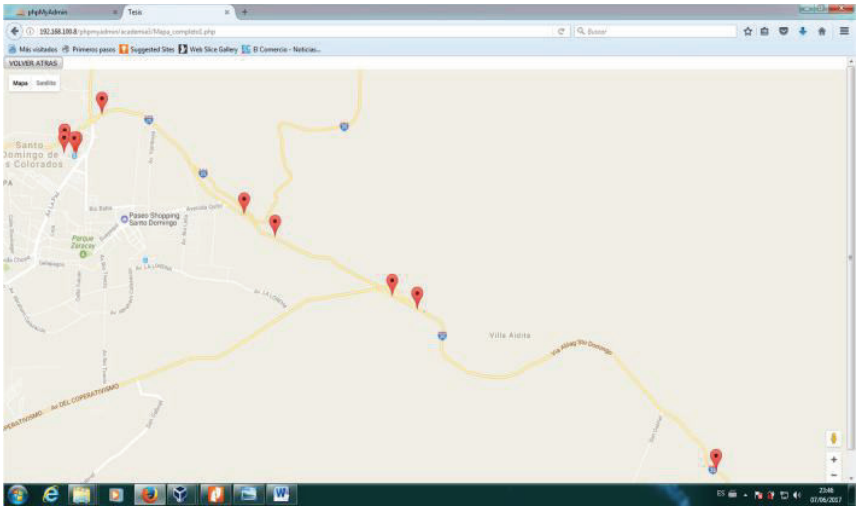


Figura 3.33 Detalle de la ruta Quito-Santo Domingo.

Cada vez que se entre a la aplicación se enviará el dato de la posición geográfica, latitud y longitud, hora y fecha del lugar en donde se encuentre el bus al servidor web, y que será almacenado en la base de datos del servidor, también se actualizara la posición cada 30 minutos, y cada vez que se dé clic en el botón “CALCULAR” se enviará los anteriores datos mencionados y la tarifa a dicho servidor.

En cada marcador de color rojo que se observa en las Figuras, se puede dar clic sobre ellos y nos indicará la hora y la fecha del ingreso del dato al servidor web, y también con la ayuda de estos marcadores se observara el recorrido del bus, como se muestra a continuación en la Figura 3.42.

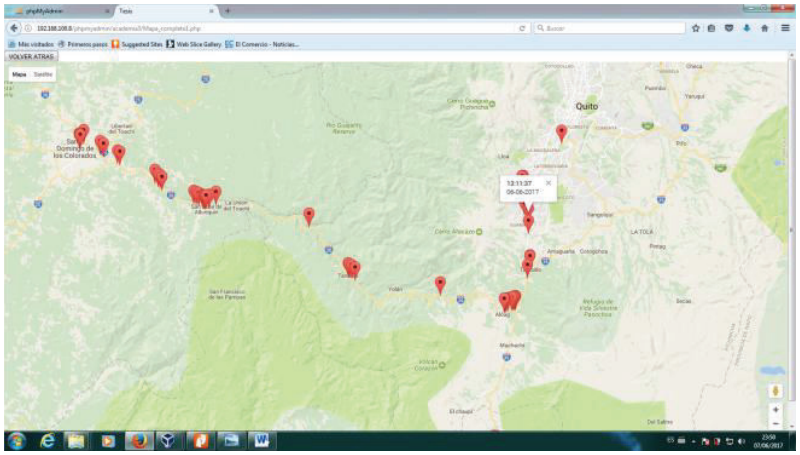


Figura 3.34 Información con marcadores de Google Maps.

En la Tabla 3.6 se analizan los resultados al ingresar a la página web para verificar el reporte diario del bus y la localización en Google Maps.

Proceso	Resultado.
Ingreso a la página web	Correcto
Ingreso a la sesión “usuario”	Correcto
Ingreso al reporte diario del recorrido del bus.	Correcto
Ingreso para observar la trayectoria del bus en Google Maps	Correcto

Tabla 3.6 Análisis de los resultados en la Página Web, dispositivo real.

3.4 COSTO REFERENCIAL DE LA APLICACIÓN IMPLEMENTADA.

Para realizar el presupuesto del costo aproximado de la implementación de la aplicación, se tomó en cuenta algunos parámetros que se describen a continuación:

- Costo de las horas empleadas para el diseño de la aplicación.
- Costo de las horas empleadas para el desarrollo de la aplicación.
- Costo de las horas empleadas para la depuración de la aplicación.
- Costo de internet móvil para el envió de datos.

Cabe señalar que la aplicación se desarrolló en software libre y no se tiene costos por licenciamiento de software.

3.4.1. COSTO DEL DISEÑO DE LA APLICACIÓN.

Para obtener el costo del diseño de la aplicación, se tomó en cuenta cuánto gana un diseñador de aplicaciones en el sector público; el salario que percibe es aproximadamente 986 USD[17] mensuales.

COSTO DEL DISEÑO.		
HORAS	COSTO POR HORA	TOTAL
50	$\frac{986}{160\ horas} = 6.16\ \frac{USD}{hora}$	$50\ h \times 6.16\ \frac{USD}{hora} = 308\ USD$

Tabla 3.7 Costo del diseño.

3.4.2. COSTO DEL DESARROLLO DE LA APLICACIÓN.

Se procedió a consultar a la misma fuente que nos indicó cuánto gana un diseñador de aplicaciones; nos indicó que un desarrollador de aplicaciones en el sector público percibe aproximadamente 1.212[17] USD mensuales.

COSTO DEL DESARROLLO.		
HORAS	COSTO POR HORA	TOTAL

[17] Fuente: Director de Tecnología de la Información del Ministerio del Interior Ecuador.

120	$\frac{1212}{160\ horas} = 7.5\ \frac{USD}{hora}$	$120\ horas \times 7.5\ \frac{USD}{hora} = 906\ USD$

Tabla 3.8 Costo del desarrollo.

3.4.3. COSTO DE LA DEPURACIÓN DE LA APLICACIÓN.

Generalmente el mismo desarrollador de una aplicación se encarga de la depuración, entonces se toma el mismo valor del sueldo que de un desarrollador que es aproximadamente 1.212[17] USD mensuales.

COSTO DE LA DEPURACIÓN.		
HORAS	COSTO POR HORA	TOTAL
30	$\frac{1212}{160\ horas} = 7.5\ \frac{USD}{hora}$	$30\ horas \times 7.5\ \frac{USD}{hora} = 225\ USD$

Tabla 3.9 Costo de la depuración.

3.4.4. COSTO DE INTENET MOVIL.

Ya que se está utilizando la red celular para el envió de datos desde la aplicación móvil hacia el servidor, se contrató un paquete de 100 megas con un costo de 2 dólares.

3.4.5. COSTO TOTAL DE LA APLICACIÓN.

En la siguiente Tabla 3.7 se muestra el costo total de la aplicación.

COSTO TOTAL DE LA APLICACIÓN.	
DISEÑO	308 USD
DESARROLLLO	900 USD
DEPURACIÓN	225 USD
INTERNET MOVIL	2 USD
TOTAL:	1.435 USD

Tabla 3.10 Costo total de la aplicación.

CAPÍTULO 4.

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Se cumplió con el objetivo principal de este proyecto; el diseño e implementación de una aplicación para el Sistema Operativo Android, que permite calcular la tarifa que los pasajeros deben cancelar cuando abordan el bus interprovincial en las diferentes paradas a lo largo del recorrido mediante coordenadas GPS.

- La tecnología GPS en la actualidad continúa generando grandes cambios en el mundo, las aplicaciones que utilizan el sistema de posicionamiento global están cada vez creciendo, debido a que el costo de los receptores ha bajado y al mismo tiempo el nivel de exactitud del sistema ha mejorado. También influye en la popularidad del sistema GPS el aumento de la velocidad del internet y las mejoras sustanciales en la telefonía móvil.

- Se desarrolló la aplicación para el Sistema Operativo Android, se escogió este sistema operativo ya que es de código abierto y en los últimos años se ha convertido en el sistema operativo más utilizado en el mundo para teléfonos inteligentes, tablets, etc. También permite al desarrollador de aplicaciones tener acceso a todos los recursos de un dispositivo como memoria, botones, Wifi, pantalla, y así brindar al usuario final una aplicación que se adapte a las necesidades de dicho usuario.

- La aplicación se desarrolló en Android Studio que es un software de desarrollo integrado para dispositivos con Sistema Operativo Android, el lenguaje de programación que se utiliza en Android Studio es Java. Android Studio nos permite realizar una interfaz gráfica para la aplicación desarrollada, dicha interfaz puede estar compuesta de botones, texto, cajas de texto, etc., y a cada elemento de la interfaz se le puede dar una acción, que generan una serie de eventos de acuerdo al diseño de la aplicación.

- En la aplicación implementada se incorporó archivos JavaScript que pertenecen al API de Google Maps, esto nos permite extraer la información que se tiene en el servidor de base de datos, como: latitud, longitud, fecha y hora. Estos datos se podrán observar por medio de marcadores en un mapa de Google Maps.

- Con la aplicación desarrollada se pretende llevar un control más real de donde un bus se detiene a recoger pasajeros en todo el trayecto hasta llegar al destino, y conocer el valor real recaudado por el cobro de pasajes a dichos usuarios.

- El costo de implementar la aplicación es razonable y se ajusta a nuestra realidad, de acuerdo al análisis realizado en el Capítulo 3. Para comercializar esta aplicación entre los dueños de buses interprovinciales, se podría vender la aplicación a un costo mucho menor de lo que se calculó, pero se podría recuperar el valor de la inversión cobrando a los usuarios por la administración y el soporte del sistema.

- Una vez que se realizó las pruebas de funcionamiento en el dispositivo virtual de Android Studio y en el dispositivo real, se verificó que los datos son enviados hacia el servidor Web Apache y son almacenados en la base de datos phpMyAdmin correctamente, también se verifico el correcto funcionamiento de la página web, en donde un usuario puede acceder al reporte diario del recorrido del bus y a la visualización del recorrido del bus en el mapa de Google Maps. Por lo tanto se concluye que está funcionando correctamente la aplicación.

4.2 RECOMENDACIONES.

- Se recomienda instalar la aplicación en un dispositivo móvil que tenga como mínimo la versión Android 4.2 JELLY BEAN con nivel de API 17, ya que si se instalará la aplicación en una versión Android menor a la 4.2 no funcionaría la aplicación.

- Se recomienda en el futuro cuando el software del dinero electrónico sea de código abierto, añadir a la aplicación implementada en este proyecto, para el pago de la tarifa concerniente al valor del pasaje. No se implementó el pago con dinero electrónico en la aplicación ya que todavía el software de dinero electrónico es propietario.

- Se recomienda para el desarrollo de aplicaciones en el Sistema Operativo Android, el uso de Android Studio; ya que posee un entorno más amigable al usuario, un editor de código inteligente, un emulador rápido, posee un sistema de compilación sólido, etc. Y en el futuro será el entorno de desarrollo (IDE) que Google recomiende, sustituyendo a Eclipse.

- Se recomienda inhabilitar la aplicación en el dispositivo, cuando el oficial del bus interprovincial no se encuentre en el recorrido que realice el bus, ya que la aplicación estará enviando su posición geográfica al servidor web, y el objetivo del proyecto es que solo se envié la ubicación cuando el bus está haciendo un recorrido.

- Se recomienda tomar en cuenta los lugares en donde no hay señal de la red celular a lo largo del recorrido del bus, esto nos ayudara a saber que en esos lugares no se puede enviar la posición del bus hacia el servidor.

- Se recomienda tener internet móvil o estar conectado a una red Wifi en el dispositivo en donde se encuentra instalada la aplicación, para el envío de los datos que la aplicación genere a través de la red celular hacia el servidor web.

- Se recomienda tomar en cuenta las paradas oficiales que la Agencia Nacional de Transito proporciona, al momento de ubicar las paradas en la ruta del bus, en el programa de Android Studio, para posteriormente realizar el cálculo del pasaje.

REFERENCIAS BIBLIOGRÁFICAS

[1] GPS.PDF – GPS_Modulo. [ftp://gisweb.ciat.cgiar.org/DAPA/planificacion/GEOMATICA/GPS/GPS_Modulo pdf].

[2] A.Pozo-Ruz*, A.Ribeiro, M.C.García-Alegre, L.García, D.Guinea, F.Sandoval. SISTEMA DE POSICIONAMIENTO GLOBAL (GPS): DESCRIPCIÓN, ANÁLISIS DE ERRORES, APLICACIONES Y FUTURO

[3] GPS.gov (2016) SISTEMA DE POSICIONAMIENTO GLOBAL [Online] http://www.gps.gov/systems/gps/spanish.php

[4] PocketGPSWorld.com (2016) How does the Global Positioning System. work? [Online] http://www.pocketgpsworld.com/howgpsworks.php

[5] GEOCONNECT (2016) GPS signals [Online] http://geoconnect.com.au/gps l1-l2-l5/

[6] LOCATA (2016) Civilian Applications of GPS [Online] http://www.locata.com/applications-of-gps/civilian-applications/

[7] GPS.gov (2016) Surveying & Mapping [Online] http://www.gps.gov/applications/survey/

[8] Gozalvez Sempere, Javier. "An Overview of the GSM System". Communication Division, Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, Scotland.

[9] Agencia Nacional de Tránsito (2016) Tarifas Pichincha 2016 [Online] http://www.ant.gob.ec/index.php/component/content/article/74-tarifas/1438-tarifas-.pichincha-2016#.WILcEVPhCM8

[10] CatedraDatos (2014) Un viaje en el tiempo [Online] http://catedradatos.com.ar/2014/11/teorico-13-lev-manovich-estudios-de-software/

[11] Unidades de estado sólido (SSD) [Online] http://www.pcper.com/article.php?aid=1038

[12] Hirohisa Kawamoto, The History of Liquid Crystal Displays. University Kyoto,

IEEE, 2002.

[13] RON MERTENS, The OLED Handbook, A Guide to OLED Technology, Industry & Market, Edition 2015.

[14] Burnette, Hello, Android: Introducing Google's Mobile Development Platform. 3 rd Edition.

[15] ANDROID STUDIO (2016), Guía del Usuario [Online] … https://developer.android.com/studio/intro/index.html

[16] Eclipse documentation (2016) What is Eclipse ? [Online] http://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.platform.

[17] Eclipse Workbench Components,[Online] … …. … https://www.dartmouth.edu/~rc/classes/softdev_linux/eclipse5.html

[18] CHANDRA MOHAN, OPERATING SYSTEM, PHI Learning Private Limited:Delhi, 2013.

[19] Marko Gargenta, LEARNING ANDROID, Editorial O'Reilly Media, Inc, March 2011.

[20] Developers(2016) API Guides [Online] https://developer.android.com/guide/topics/manifest/uses-sd element.html?hl=es

[21] Nikolay Elenkov, ANDROID SECURITY INTERNALS an In-Depth Guide to android's security architecture, San Francisco 2015.

[22] G. Marcario, M. Torchiano, and M. Violante, An in- vehicle infotainment software architecture based on google android. Lausanne, Switzerland: IEEE,2009.

[23] Developers(2016), Application Fundamentals,[Online] https://developer.android.com/guide/platform/index.html

[24] Oswald Swift, Android: App Development & Programming Guide: Learn In A Day!, Second Edition, 2015

[25] SocialCompare(2016), Android versions comparison[Online] http://socialcompare.com/es/comparison/android-versions-comparison

[26] android(2016) Android 1.6 Donut [Online]
https://www.android.com/intl/es_es/history/#/donut

[27] android(2016) Android 2.1 Eclair [Online]
https://www.android.com/intl/es_es/history/#/eclair

[28] android(2016) Android 2.2 Froyo [Online]
https://www.android.com/intl/es_es/history/#/froyo

[29] android(2016) Android 2.3, 2.4 Gingerbread [Online]
https://www.android.com/intl/es_es/history/# Gingerbread

[30] android(2016) Android 3.0 Honeycomb [Online]
https://www.android.com/intl/es_es/history/# Honeycomb

[31] android(2016) Android 4.0 Ice Cream Sandwich [Online]
https://www.android.com/intl/es_es/history/# Ice Cream Sandwich

[32] android(2016) Android 4.1 Jelly Bean [Online]
https://www.android.com/intl/es_es/history/# Jelly Bean

[33] android(2016) Android 4.4 KitKat [Online]
https://www.android.com/intl/es_es/history/# KitKat

[34] android(2016) Android 5.0 Lollipop [Online]
https://www.android.com/intl/es_es/history/# Lollipop

[35] android(2016) Android 6.0 Marshmallow [Online]
https://www.android.com/intl/es_es/history/# Marshmallow

[36] android(2016) Android 7.0 Nougat [Online]
https://www.android.com/intl/es_es/history/# Nougat

[37] Ken Arnold, James Gosling, David Holmes, The Java TM Programming Language, Editor Addison-Wesley, 2006

[38] Julio Sanchez, Maria P. Canton, JAVA Programming for Engineers, Editorial CRC PRESS, London 2002.

[39] Sitesbay (2017), Features of Java [Online]
http://www.sitesbay.com/java/features-of-java

[40] ANDROID STUDIO (2017), Requisitos del sistema [Online] …
https://developer.android.com/studio/index.html?hl=es-419

[41] Developers(2017), Manifiesto de la app,[Online]
https://developer.android.com/guide/topics/manifest/manifest-intro.html

[42] UNIVERSIDAD POLITÉCNICA DE VALENCIA(2017), Diploma de Especialización en desarrollo de Aplicaciones para Android, [Online]
http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general- y-entorno-de-desarrollo/148-elementos-de-un-proyecto-android

[43] Developers(2017),Android content,[Online]
https://developer.android.com/reference/android/content/Context.html

[50] Developers(2017),Guías de la API,Fragmentos[Online]
https://developer.android.com/guide/components/fragments.html?hl=es

[51] William Nelson, Arvind Srinivasan, Murthy Chintalapati, Sun Web Server: The Essential Guide, Editorial Pearson Education, 2009.

[52] MOZILA DEVELOPER NETWORK (2017), What is a web server? [Online]
http://www.pcmag.com/encyclopedia/term/54342/web-server

[53] Web Developers Notes(2017), Web Server List [Online]
http://www.webdevelopersnotes.com/web-servers-list

[54] Apache HTTP Server Project(2017), About Apache [Online]
https://httpd.apache.org/ABOUT_APACHE.html

[55] MySQL(2017), Documentation[Online]
https://dev.mysql.com/doc/

ANEXOS.

ANEXO A. Tarifas de la Provincia de Pichincha.

ANEXO B. Script implementado en Android Studio.

ANEXO C. Scripts implementados en PHP, servidor Web.

ANEXO D. Scripts Página Web.

ANEXO A

Tarifas actualizadas de la Resoluciones No. 033; 040; 107-DIR-2015-ANT y 093-DIR-2016-ANT

Provincia de Origen	Cantón de Origen	Ciudad / Sector Origen	Provincia Destino	Cantón de Destino	Ciudad / Sector Destino	Anexo Consolidado
PICHINCHA	CAYAMBE	CANGAHUA	PICHINCHA	PEDRO MONCAYO	LA ESPERANZA	$ 0,25
PICHINCHA	CAYAMBE	CANGAHUA	PICHINCHA	PEDRO MONCAYO	LA ESTACIÓN	$ 0,25
PICHINCHA	CAYAMBE	CANGAHUA	PICHINCHA	QUITO	EL QUINCHE	$ 1,20
PICHINCHA	CAYAMBE	CAYAMBE	IMBABURA	OTAVALO	OTAVALO	$ 0,60
PICHINCHA	CAYAMBE	CAYAMBE	PICHINCHA	MEJÍA	ALOAG	$ 2,25
PICHINCHA	CAYAMBE	CAYAMBE	PICHINCHA	PEDRO MONCAYO	COCHASQUI	$ 0,60
PICHINCHA	CAYAMBE	CAYAMBE	PICHINCHA	PEDRO MONCAYO	LA ESPERANZA	$ 0,30
PICHINCHA	CAYAMBE	CAYAMBE	PICHINCHA	PEDRO MONCAYO	LA ESTACIÓN	$ 0,25
PICHINCHA	CAYAMBE	CAYAMBE	PICHINCHA	PEDRO MONCAYO	MALCHINGUÍ	$ 1,10
PICHINCHA	CAYAMBE	CAYAMBE	PICHINCHA	PEDRO MONCAYO	TOCACHI	$ 0,55
PICHINCHA	CAYAMBE	CAYAMBE	PICHINCHA	QUITO	LA OFELIA	$ 1,75
PICHINCHA	CAYAMBE	CAYAMBE	PICHINCHA	QUITO	TAPIGUCHI	$ 0,30
PICHINCHA	CAYAMBE	CHUMILLOS	PICHINCHA	QUITO	EL QUINCHE	$ 0,60
PICHINCHA	CAYAMBE	CUSUBAMBA	PICHINCHA	QUITO	HACIENDA SAN MIGUEL	$ 0,55
PICHINCHA	CAYAMBE	PESILLO	IMBABURA	OTAVALO	OTAVALO	$ 1,40
PICHINCHA	CAYAMBE	PESILLO	IMBABURA	OTAVALO	ZULETA	$ 0,55
PICHINCHA	CAYAMBE	PESILLO	PICHINCHA	PEDRO MONCAYO	LA ESPERANZA	$ 0,65
PICHINCHA	MEJÍA	ALOAG	TUNGURAHUA	AMBATO	AMBATO	$ 2,45
PICHINCHA	MEJÍA	Y DE TAMBILLO	PICHINCHA	QUITO	TURUBAMBA	$ 0,25
PICHINCHA	PEDRO MONCAYO	TABACUNDO	PICHINCHA	CAYAMBE	CAYAMBE	$ 0,30
PICHINCHA	PUERTO QUITO	PUERTO QUITO	PICHINCHA	SAN MIGUEL DE LOS	SAN MIGUEL DE LOS BANCOS	$ 1,81
PICHINCHA	QUITO	AMAGUAÑA	PICHINCHA	MEJÍA	TAMBILLO	$ 0,30
PICHINCHA	QUITO	AMAGUAÑA	PICHINCHA	MEJÍA	UYUMBICHO	$ 0,30
PICHINCHA	QUITO	CALDERON	PICHINCHA	CAYAMBE	CAYAMBE	$ 1,25
PICHINCHA	QUITO	CALDERON	PICHINCHA	CAYAMBE	SANTA ROSA CUSUBAMBA	$ 0,60
PICHINCHA	QUITO	CALDERON	PICHINCHA	PEDRO MONCAYO	LA ESPERANZA	$ 0,85
PICHINCHA	QUITO	CALDERON	PICHINCHA	PEDRO MONCAYO	MALCHINGUÍ	$ 1,00
PICHINCHA	QUITO	CALDERON	PICHINCHA	PEDRO MONCAYO	TABACUNDO	$ 0,95
PICHINCHA	QUITO	CUMBAYA	PICHINCHA	CAYAMBE	SANTA ROSA CUSUBAMBA	$ 0,80
PICHINCHA	QUITO	GUAMANÍ	PICHINCHA	MEJÍA	ALOAG	$ 0,35
PICHINCHA	QUITO	GUAMANÍ	PICHINCHA	MEJÍA	ALOASÍ	$ 0,50
PICHINCHA	QUITO	GUAMANÍ	PICHINCHA	MEJÍA	LA ESTACIÓN	$ 0,55
PICHINCHA	QUITO	GUAMANÍ	PICHINCHA	MEJÍA	MACHACHI	$ 0,45
PICHINCHA	QUITO	GUAMANÍ	PICHINCHA	MEJÍA	SAN AGUSTÍN	$ 0,55
PICHINCHA	QUITO	GUAMANÍ	PICHINCHA	MEJÍA	TAMBILLO	$ 0,30
PICHINCHA	QUITO	GUAMANÍ	PICHINCHA	MEJÍA	UYUMBICHO	$ 0,35
PICHINCHA	QUITO	GUAYLLABAMBA	PICHINCHA	CAYAMBE	CAYAMBE	$ 0,70
PICHINCHA	QUITO	GUAYLLABAMBA	PICHINCHA	CAYAMBE	SANTA ROSA CUSUBAMBA	$ 0,30
PICHINCHA	QUITO	GUAYLLABAMBA	PICHINCHA	PEDRO MONCAYO	LA ESPERANZA	$,45

Provincia de Origen	Cantón de Origen	Ciudad / Sector Origen	Provincia Destino	Cantón de Destino	Ciudad / Sector Destino	Anexo Consolidado
PICHINCHA	QUITO	GUAYLLABAMBA	PICHINCHA	PEDRO MONCAYO	TABACUNDO	$ 0,45
PICHINCHA	QUITO	LLANO GRANDE	PICHINCHA	RUMIÑAHUI	RUMIÑAHUI	$ 0,30
PICHINCHA	QUITO	QUITO	AZUAY	CUENCA	CUENCA	$ 2,00
PICHINCHA	QUITO	QUITO	BOLÍVAR	CALUMA	CALUMA	$ 8,25
PICHINCHA	QUITO	QUITO	BOLÍVAR	CHILLANES	CHILLANES	$ 7,25
PICHINCHA	QUITO	QUITO	BOLÍVAR	ECHEANDIA	ECHEANDIA	$ 5,20
PICHINCHA	QUITO	QUITO	BOLÍVAR	GUARANDA	GUARANDA	$ 5,70
PICHINCHA	QUITO	QUITO	BOLÍVAR	SAN JOSÉ DE	CHIMBO	$ 6,00
PICHINCHA	QUITO	QUITO	BOLÍVAR	SAN MIGUEL	SAN MIGUEL	$ 0
PICHINCHA	QUITO	QUITO	CARCHI	ESPEJO	EL ÁNGEL	$,65
PICHINCHA	QUITO	QUITO	CARCHI	MONTÚFAR	SAN GABRIEL	$,15
PICHINCHA	QUITO	QUITO	CARCHI	TULCÁN	TULCÁN	$,10
PICHINCHA	QUITO	QUITO	CHIMBORAZO	ALAUSÍ	ALAUSÍ	$ 7,20
PICHINCHA	QUITO	QUITO	CHIMBORAZO	CHUNCHI	CHUNCHI	$ 6,00
PICHINCHA	QUITO	QUITO	CHIMBORAZO	GUAMOTE	GUAMOTE	$ 3,20
PICHINCHA	QUITO	QUITO	CHIMBORAZO	GUANO	GUANO	$ 5,10
PICHINCHA	QUITO	QUITO	CHIMBORAZO	RIOBAMBA	RIOBAMBA	$ 4,60
PICHINCHA	QUITO	QUITO	COTOPAXI	LA MANÁ	LA MANÁ	$ 5,65
PICHINCHA	QUITO	QUITO	COTOPAXI	LATACUNGA	LATACUNGA	$ 2,15
PICHINCHA	QUITO	QUITO	COTOPAXI	PANGUA	EL CORAZÓN	$ 6,00
PICHINCHA	QUITO	QUITO	COTOPAXI	PUJILÍ	PUJILÍ	$ 2,45
PICHINCHA	QUITO	QUITO	COTOPAXI	SALCEDO	SALCEDO	$ 2,50
PICHINCHA	QUITO	QUITO	COTOPAXI	SAQUISILÍ	SAQUISILÍ	$ 1,90
PICHINCHA	QUITO	QUITO	EL ORO	ARENILLAS	ARENILLAS	$ 13,40
PICHINCHA	QUITO	QUITO	EL ORO	HUAQUILLAS	HUAQUILLAS	$ 13,95
PICHINCHA	QUITO	QUITO	EL ORO	MACHALA	MACHALA	$ 12,60
PICHINCHA	QUITO	QUITO	EL ORO	PASAJE	PASAJE	$ 12,65
PICHINCHA	QUITO	QUITO	EL ORO	PIÑAS	PIÑAS	$ 14,25
PICHINCHA	QUITO	QUITO	EL ORO	PORTOVELO	PORTOVELO	$ 15,10
PICHINCHA	QUITO	QUITO	EL ORO	SANTA ROSA	LA AVANZADA	$ 13,10
PICHINCHA	QUITO	QUITO	EL ORO	SANTA ROSA	SANTA ROSA	$ 12,85
PICHINCHA	QUITO	QUITO	EL ORO	ZARUMA	ZARUMA	$ 15,20
PICHINCHA	QUITO	QUITO	ESMERALDAS	ESMERALDAS	ESMERALDAS	$ 7,70
PICHINCHA	QUITO	QUITO	ESMERALDAS	QUININDÉ	QUININDÉ (ROSA ZARATE)	$ 5,25
PICHINCHA	QUITO	QUITO	ESMERALDAS	QUININDÉ	QUININDÉ (SANTA ROSA)	$ 5,25
PICHINCHA	QUITO	QUITO	GUAYAS	GUAYAQUIL	GUAYAQUIL	$ 10,20
PICHINCHA	QUITO	QUITO	GUAYAS	MILAGRO	MILAGRO	$ 9,10
PICHINCHA	QUITO	QUITO	IMBABURA	ANTONIO ANTE	ATUNTAQUI	$ 2,70
PICHINCHA	QUITO	QUITO	IMBABURA	COTACACHI	COTACACHI	$ 2,70
PICHINCHA	QUITO	QUITO	IMBABURA	IBARRA	IBARRA	$ 3,00
PICHINCHA	QUITO	QUITO	IMBABURA	IBARRA	PIMAMPIRO	$ 4,35
PICHINCHA	QUITO	QUITO	IMBABURA	OTAVALO	OTAVALO	$ 2,50
PICHINCHA	QUITO	QUITO	LOJA	CATAMAYO	CATAMAYO	$ 14,90
PICHINCHA	QUITO	QUITO	LOJA	CELICA	CELICA	$ 15,90
PICHINCHA	QUITO	QUITO	LOJA	LOJA	LOJA	$ 15,70
PICHINCHA	QUITO	QUITO	LOJA	PALTAS	CATACOCHA	$ 14,20
PICHINCHA	QUITO	QUITO	LOJA	PUYANGO	ALAMOR	$ 15,15
PICHINCHA	QUITO	QUITO	LOS RÍOS	BABAHOYO	BABAHOYO	$ 7,40
PICHINCHA	QUITO	QUITO	LOS RÍOS	QUEVEDO	QUEVEDO	$ 5,75
PICHINCHA	QUITO	QUITO	LOS RÍOS	VENTANAS	VENTANAS	$ 6,75

Provincia de Origen	Cantón de Origen	Ciudad / Sector Origen	Provincia Destino	Cantón de Destino	Ciudad / Sector Destino	Anexo Consolidado
PICHINCHA	QUITO	QUITO	MANABÍ	CHONE	CHONE	$ 6,90
PICHINCHA	QUITO	QUITO	MANABÍ	EL CARMEN	EL CARMEN	$ 4,00
PICHINCHA	QUITO	QUITO	MANABÍ	MANTA	MANTA	$ 9,60
PICHINCHA	QUITO	QUITO	MANABÍ	PORTOVIEJO	PORTOVIEJO	$ 8,65
PICHINCHA	QUITO	QUITO	MANABÍ	SUCRE	BAHÍA DE CARÁQUEZ	$ 8,00
PICHINCHA	QUITO	QUITO	MORONA SANTIAGO	MORONA	MACAS	$ 13,95
PICHINCHA	QUITO	QUITO	NAPO	EL CHACO	EL CHACO	$ 5,50
PICHINCHA	QUITO	QUITO	NAPO	QUIJOS	BAEZA	$ 3,50
PICHINCHA	QUITO	QUITO	NAPO	TENA	TENA	$ 7,50
PICHINCHA	QUITO	QUITO	PASTAZA	PASTAZA	PUYO	$ 7,00
PICHINCHA	QUITO	QUITO	PASTAZA	PASTAZA	VERACRUZ	$ 7,25
PICHINCHA	QUITO	QUITO	PICHINCHA	CAYAMBE	CAYAMBE (OTON)	$ 1,95
PICHINCHA	QUITO	QUITO	PICHINCHA	CAYAMBE	SANTA ROSA CUSUBAMBA	$ 0,95
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	ALOAG (OBELISCO)	$ 0,65
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	ALOAG (PARQUE)	$ 0,70
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	ALOASÍ	$ 0,70
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	CUTUGLAHUA	$ 0,40
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	LA ESTACIÓN	$ 0,70
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	MACHACHI	$ 0,70
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	PINLLO CRUZ	$ 0,70
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	SAN AGUSTÍN	$ 0,70
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	TANDAPI	$ 1,90
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	TUCUSO	$ 0,70
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	UYUMBICHO	$ 0,55
PICHINCHA	QUITO	QUITO	PICHINCHA	MEJÍA	Y DE TAMBILLO	$ 0,55
PICHINCHA	QUITO	QUITO	PICHINCHA	PEDRO MONCAYO	MALCHINGUÍ	$ 1,55
PICHINCHA	QUITO	QUITO	PICHINCHA	PEDRO MONCAYO	TABACUNDO	$ 1,65
PICHINCHA	QUITO	QUITO	PICHINCHA	PEDRO VICENTE	PEDRO VICENTE MALDONADO	$ 3,75
PICHINCHA	QUITO	QUITO	PICHINCHA	PUERTO QUITO	PUERTO QUITO	$ 4,25
PICHINCHA	QUITO	QUITO	PICHINCHA	SAN MIGUEL DE LOS	MINDO	$ 3,10
PICHINCHA	QUITO	QUITO	PICHINCHA	SAN MIGUEL DE LOS	SAN MIGUEL DE LOS BANCOS	$ 3,49
PICHINCHA	QUITO	QUITO	SANTA ELENA	SALINAS	SALINAS	$ 13,85
PICHINCHA	QUITO	QUITO	SANTO DOMINGO	SANTO DOMINGO	SANTO DOMINGO	$ 3,25
PICHINCHA	QUITO	QUITO	SANTO DOMINGO	SANTO DOMINGO DE LOS TSÁCHILAS	ALLURIQUÍN	$ 2,60
PICHINCHA	QUITO	QUITO	SANTO DOMINGO	SANTO DOMINGO DE LOS TSÁCHILAS	CHIRIBOGA	$ 1,90
PICHINCHA	QUITO	QUITO	SUCUMBÍOS	GONZALO PIZARRO	EL REVENTADOR	$ 9,35
PICHINCHA	QUITO	QUITO	SUCUMBÍOS	GONZALO PIZARRO	LUMBAQUÍ	$ 10,65
PICHINCHA	QUITO	QUITO	SUCUMBÍOS	LAGO AGRIO	LAGO AGRIO	$ 12,50
PICHINCHA	QUITO	QUITO	SUCUMBÍOS	SHUSHUFINDI	SHUSHUFINDI	$ 14,55
PICHINCHA	QUITO	QUITO	TUNGURAHUA	AMBATO	AMBATO	$ 3,35
PICHINCHA	QUITO	QUITO	TUNGURAHUA	BAÑOS	BAÑOS	$ 4,25

ANEXO B

MainActivity.java

```
package example.prueba_2;

import android.content.Context;
import android.content.Intent;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationManager;
import android.os.StrictMode;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.BasicResponseHandler;
import org.apache.http.impl.client.DefaultHttpClient;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;


public class MainActivity extends ActionBarActivity { //Clase principal.

    // Declaración de variables
    private static int pag=0;
    Button bRuta1, bRuta2, bRuta3;
    double cosPag = 0; // costo a pagar

    Timer timer = new Timer();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Vinculación de los botones con los botones XML

        bRuta1 = (Button) findViewById(R.id.RUTA_1);
        bRuta1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {  // Clic de un botón.
                pag = 1; // bandera
                showConsoleActivity();
            }
```

```
        });

        bRuta2 = (Button) findViewById(R.id.RUTA_2);
        bRuta2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                pag=2;
                showConsoleActivity();
            }
        });

        bRuta3 = (Button) findViewById(R.id.RUTA_3);
        bRuta3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                pag=3;
                showConsoleActivity();
            }
        });

        timer.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                Refrescar();
            }
        }, 0, 900000);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    static void show(Context context, int pagR, String men) {
        pag = pagR;
        //mensaje = men;
        final Intent intent = new Intent(context, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP |
Intent.FLAG_ACTIVITY_NO_HISTORY);
        context.startActivity(intent);
    }

    private void showConsoleActivity() {

        switch (pag){
```

```
            case 1:
                pag=0;
                ruta1.show(this);
                break;
            case 2:
                pag=0;
                ruta2.show(this);
                break;
            case 3:
                pag=0;
                ruta3.show(this);
                break;
            default:
                pag=0;
                //caratula.show(this, driver);
                break;
        }
    }

    private void Refrescar(){
        this.runOnUiThread(Graf);

    }
    private Runnable Graf = new Runnable(){
        public void run(){
            LocationManager locationManager =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
            Criteria criteria = new Criteria();
            Location location =
locationManager.getLastKnownLocation(locationManager.getBestProvider(crit
eria, false));

            SimpleDateFormat fecha = new SimpleDateFormat("dd-MM-yyyy");
            String sFecha = fecha.format(new Date());
            SimpleDateFormat hora = new SimpleDateFormat("HH:mm:ss");
            String sHora = hora.format(new Date());


            double Lat11 = location.getLatitude();
            double Lng11 = location.getLongitude();


            try{


httpGetData("http://181.198.137.101/phpmyadmin/registrartarifa1.php?id=1&
fecha="+sFecha+

"&hora="+sHora+"&latitud="+Double.toString(Lat11)+"&longitud="+Double.toS
tring(Lng11));

                Toast.makeText(getApplicationContext(), "Posicion
actual", Toast.LENGTH_LONG).show();
            }catch(Exception e){
                Toast.makeText(getApplicationContext(), "Error al envio
de datos", Toast.LENGTH_LONG).show();
            }

        }
    };
```

```
    public String httpGetData(String mURL) {
        String response="";
        mURL=mURL.replace(" ", "%20");
        Log.i("LocAndroid Response HTTP Threas", "Ejecutando get 0: " +
mURL);
        HttpClient httpclient = new DefaultHttpClient();

        Log.i("LocAndroid Response HTTP Thread","Ejecutando get 1");
        HttpGet httppost = new HttpGet(mURL);
        Log.i("LocAndroid Response HTTP Thread","Ejecutando get 2");
        try {


            Log.i("LocAndroid Response HTTP","Ejecutando get");
            // Execute HTTP Post Request
            ResponseHandler<String> responseHandler=new
BasicResponseHandler();
            response = httpclient.execute(httppost,responseHandler);
            Log.i("LocAndroid Response HTTP",response);
        } catch (ClientProtocolException e) {
            Log.i("LocAndroid Response HTTP ERROR 1",e.getMessage());
            // TODO Auto-generated catch block
        } catch (IOException e) {

            Log.i("LocAndroid Response HTTP ERROR 2",e.getMessage());
            // TODO Auto-generated catch block
        }
        return response;
    }

}
```

ruta2.java

```
package example.prueba_2;

import android.content.Context;
import android.content.Intent;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.StrictMode;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.SupportMapFragment;
```

```
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.BasicResponseHandler;
import org.apache.http.impl.client.DefaultHttpClient;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class ruta2 extends FragmentActivity {

    private GoogleMap mMap; // Might be null if Google Play services APK
is not available.
    double Rad = 6371000;
    double pi = 3.14159265;
    double dLng = 0;
    double dLat = 0;
    double Lng1 = 0;
    double Lat1 = 0;

    int nParadasT = 5; // numero de paradas totales.

    double[][][] locP = new double [5][2][nParadasT];
    double[] cosT = new double [nParadasT-1];
    double cosPag = 0;
    int[][] disT = new int [5][nParadasT];
    int dis = 0;
    int[][]posT = new int [2][2];
    int[]pos = new int [2];
    int nParada = 0;

    int idaRegreso = 0; //0 ida 1 regreso

    TextView tarifa;
    EditText parada;
    Button calcular;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.ruta2);

        locP [0][0][0] = -0.295640; locP [0][1][0] = -78.556410;
//Quitumbe
        locP [1][0][0] = -0.316940; locP [1][1][0] = -78.550706;
….      locP [2][0][0] = -0.336534; locP [2][1][0] = -78.548164;
        locP [3][0][0] = -0.357968; locP [3][1][0] = -78.547675;
        locP [4][0][0] = -0.381624; locP [4][1][0] = -78.531544;

        locP [0][0][1] = -0.406181; locP [0][1][1] = -78.546802;    //Y
DE Tambillo
        locP [1][0][1] = -0.456534; locP [1][1][1] = -78.562367;    //-
        locP [2][0][1] = -0.447493; locP [2][1][1] = -78.643143;    //-
```

```
        locP [3][0][1] = -0.437868; locP [3][1][1] = -78.704920;    //-
        locP [4][0][1] = -0.459589; locP [4][1][1] = -78.758637;

        locP [0][0][2] = -0.416257; locP [0][1][2] = -78.799453;
//Tandapi
        locP [1][0][2] = -0.379156; locP [1][1][2] = -78.826323;
        locP [2][0][2] = -0.346488; locP [2][1][2] = -78.852525;
        locP [3][0][2] = -0.332696; locP [3][1][2] = -78.906529;
        locP [4][0][2] = -0.317365; locP [4][1][2] = -78.953749;

        locP [0][0][3] = -0.322524; locP [0][1][3] = -78.996787;
//Alluriquin
        locP [1][0][3] = -0.309819; locP [1][1][3] = -79.039002;
        locP [2][0][3] = -0.286382; locP [2][1][3] = -79.067971;
        locP [3][0][3] = -0.272188; locP [3][1][3] = -79.107344;
        locP [4][0][3] = -0.250782; locP [4][1][3] = -79.139648;


        locP [0][0][4] = -0.239186; locP [0][1][4] = -79.170255;
//Santo Domingo
        locP [1][0][4] = 0;         locP [1][1][4] = 0;
        locP [2][0][4] = 0;         locP [2][1][4] = 0;
        locP [3][0][4] = 0;         locP [3][1][4] = 0;
        locP [4][0][4] = 0;         locP [4][1][4] = 0;

        cosT[0] = 0.55;
        cosT[1] = 1.50;
        cosT[2] = 0.60;
        cosT[3] = 0.50;


        setUpMapIfNeeded();


        tarifa = (TextView) findViewById(R.id.tarifa);
        parada = (EditText) findViewById(R.id.parada);
        calcular = (Button) findViewById(R.id.calcular);

        calcular.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                LocationManager locationManager =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
                Criteria criteria = new Criteria();
                Location location =
locationManager.getLastKnownLocation(locationManager.getBestProvider(crit
eria, false));
                SimpleDateFormat fecha = new SimpleDateFormat("dd-MM-
yyyy");
                String sFecha = fecha.format(new Date());
                SimpleDateFormat hora = new SimpleDateFormat("HH:mm:ss");
                String sHora = hora.format(new Date());

                Lat1 = location.getLatitude();
                Lng1 = location.getLongitude();


                mMap.addMarker(new MarkerOptions().position(new
LatLng(Lat1, Lng1)).title("ACTUAL ")
```

```
.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_A
ZURE)));

                nParada = Integer.parseInt(parada.getText().toString());

                if((nParada<=nParadasT)&&(nParada>0)){
                    calcPos(Lat1, Lng1);
                    if(idaRegreso==0)
                        calcCos1aFin(nParada);
                    else
                        calcCosFina1(nParada);
                    tarifa.setText("$ "+Double.toString(cosPag));

                    try{


httpGetData("http://181.198.137.101/phpmyadmin/registrartarifa1.php?id=1&
fecha="+sFecha+

"&hora="+sHora+"&latitud="+Double.toString(Lat1)+"&longitud="+Double.toSt
ring(Lng1)+"&tarifa="+Double.toString(cosPag));

                        Toast.makeText(getApplicationContext(), "Pago
realizado", Toast.LENGTH_LONG).show();
                    }catch(Exception e){
                        Toast.makeText(getApplicationContext(), "Error al
envio de datos", Toast.LENGTH_LONG).show();
                    }

                }else{
                    Toast toast1 =

Toast.makeText(getApplicationContext(),"Parada incorrecta",
Toast.LENGTH_SHORT);
                    toast1.show();
                }
            }
        });
    }




    @Override

    protected void onResume() {
        super.onResume();
        setUpMapIfNeeded();
    }

    private void setUpMapIfNeeded() {
        // Do a null check to confirm that we have not already
instantiated the map.
        if (mMap == null) {
            // Try to obtain the map from the SupportMapFragment.
            mMap = ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
            // Check if we were successful in obtaining the map.
```

```
            if (mMap != null) {
                //mMap.setMyLocationEnabled(true);
                setUpMap();
            }
        }
    }

    public void irIda(View view){
        idaRegreso = 0;
    }
    public void irRegreso(View view){
        idaRegreso = 1;
    }

    public void calcPos(double Lat1,double Lng1){
        int parada = nParadasT;
        for (int i = 0; i<parada; i++) {
            for (int j = 0; j < 5; j++) {

disT[j][i]=calcDist(locP[j][0][i],locP[j][1][i],Lat1,Lng1);
            }
        }
        for (int i = 0; i<parada; i++) {
            for (int j = 0; j < 5; j++) {
                if ((i == 0) && (j == 0)) {
                    dis = disT[j][i];
                    posT[0][0] = j;
                    posT[1][0] = i;
                } else {
                    if (dis > disT[j][i]) {
                        posT[0][0] = j;
                        posT[1][0] = i;
                        dis = disT[j][i];
                    }
                }
            }
        }
        for (int i = 0; i<parada; i++) {
            for (int j = 0; j < 5; j++) {
                if(posT[0][0]!=j || posT[1][0]!=i){
                    if ((i == 0) && (j == 0)) {
                        dis = disT[j][i];
                        posT[0][1] = j;
                        posT[1][1] = i;
                    } else {
                        if (dis > disT[j][i]) {
                            posT[0][1] = j;
                            posT[1][1] = i;
                            dis = disT[j][i];
                        }
                    }
                }
            }
        }
        if(idaRegreso==0){
            if((posT[1][1]*5+posT[0][1])<(posT[1][0]*5+posT[0][0])){
                pos[1] = posT[1][1];
                pos[0] = posT[0][1];
            }else{
                pos[1] = posT[1][0];
```

```
                pos[0] = posT[0][0];
            }
        }else{
            if((posT[1][1]*5+posT[0][1])>(posT[1][0]*5+posT[0][0])){
                pos[1] = posT[1][1];
                pos[0] = posT[0][1];
            }else{
                pos[1] = posT[1][0];
                pos[0] = posT[0][0];
            }

        }
    }

    public void calcCos1aFin(int parada){
        cosPag = 0;
        for (int i = pos[1]; i<parada-1; i++){
            if(i == pos[1]){
                cosPag = cosT[i]*(5-pos[0])/5;
            }else{
                cosPag = cosPag + cosT[i];
            }
        }
    }

    public void calcCosFina1(int parada){
        cosPag = 0;
        for (int i = pos[1]; i>=parada-1; i--){
            if(i == pos[1]){
                cosPag = cosT[i]*(pos[0])/5;
            }else{
                cosPag = cosPag + cosT[i];
            }
        }
    }

    public int calcDist(double LatIn,double LngIn,double Lat1,double Lng1
){
        //mMap.setMyLocationEnabled(true);
        double a = 0;
        try {


            dLat = (LatIn - Lat1) * pi /180;
            dLng = (LngIn - Lng1)*pi/180;
            a = Math.pow(Math.sin(dLat/2),2) +
Math.cos((Lat1)*pi/180)*Math.cos(LatIn*pi/180)*Math.pow(Math.sin(dLng/2),
2);
            a = 2*Math.atan2(Math.sqrt(a),Math.sqrt(1-a));
            a = Rad*a;
        }catch (Exception e){
            Toast toast = Toast.makeText(this, e.getMessage(),
Toast.LENGTH_LONG);
            toast.show();
        }


        return ((int)a);
        //return 1;
    }
```

```
    private void setUpMap() {

        for (int i = 0; i<nParadasT; i++){
            for (int j = 0; j < 5; j++) {
                if(j==0)
                    mMap.addMarker(new MarkerOptions().position(new
LatLng(locP[0][0][i], locP[0][1][i]))
                            .title("PARADA " + Integer.toString(i + 1)));
                else{
                    mMap.addMarker(new MarkerOptions().position(new
LatLng(locP[j][0][i], locP[j][1][i]))

.icon(BitmapDescriptorFactory.fromResource(R.drawable.iconmaps2)));
                }
            }
        }

        mMap.setMyLocationEnabled(true);
        goToLocationZoom(locP[0][0][2], locP[0][1][2], 14);
    }

    public void goToLocationZoom(double lat, double lng, float zoom){
        LatLng ll= new LatLng(lat, lng);
        CameraUpdate update = CameraUpdateFactory.newLatLngZoom(ll,zoom);
        mMap.moveCamera(update);
    }


    static void show(Context context) {
        final Intent intent = new Intent(context, ruta2.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP |
Intent.FLAG_ACTIVITY_NO_HISTORY);
        context.startActivity(intent);
    }

    private void showConsoleActivity(int pag, String men) {
        MainActivity.show(this, pag, men);
    }


    public String httpGetData(String mURL) {
        String response="";
        mURL=mURL.replace(" ", "%20");
        Log.i("LocAndroid Response HTTP Threas", "Ejecutando get 0: " +
mURL);
        HttpClient httpclient = new DefaultHttpClient();

        Log.i("LocAndroid Response HTTP Thread","Ejecutando get 1");
        HttpGet httppost = new HttpGet(mURL);
        Log.i("LocAndroid Response HTTP Thread","Ejecutando get 2");
        try {


            Log.i("LocAndroid Response HTTP","Ejecutando get");
            // Execute HTTP Post Request
            ResponseHandler<String> responseHandler=new
BasicResponseHandler();
            response = httpclient.execute(httppost,responseHandler);
            Log.i("LocAndroid Response HTTP",response);
```

```
        } catch (ClientProtocolException e) {
            Log.i("LocAndroid Response HTTP ERROR 1",e.getMessage());
            // TODO Auto-generated catch block
        } catch (IOException e) {

            Log.i("LocAndroid Response HTTP ERROR 2",e.getMessage());
            // TODO Auto-generated catch block
        }
        return response;
    }

}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="example.prueba_2" >

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"
/>
    <!--
 The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but are recommended.
    -->
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyBn-njBSbAL57NsBV_saKXiq6gmWzB8dLs" />
        <activity
            android:name=".MainActivity"
            android:label="TARIFARIO" >
            <intent-filter>
```

```
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity
            android:name=".ruta1"
            android:label="RUTA1" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity
            android:name=".ruta2"
            android:label="RUTA2" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity
            android:name=".ruta3"
            android:label="RUTA3" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

layout\activity_main.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="example.prueba_2.MainActivity">

    <Button
```

```
        android:id="@+id/RUTA_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/RUTA_1"
        android:text="RUTA 2" />


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RUTA 1"
        android:id="@+id/RUTA_1"
        android:layout_marginTop="199dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

layout\ruta2.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="example.prueba_2.ruta2">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CALCULAR"
        android:id="@+id/calcular"
        android:layout_gravity="left|top"
        android:layout_below="@+id/textView2"
        android:layout_toEndOf="@+id/textView2" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TARIFA:    "
        android:textSize="20sp"
```

```
    android:id="@+id/textView"
    android:layout_gravity="right|top"
    android:layout_alignBottom="@+id/tarifa"
    android:layout_alignParentTop="true"
    android:layout_alignParentBottom="false"
    android:gravity="bottom"
    android:textColor="#ff000000" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="$ 0.0"
    android:textSize="25dp"
    android:id="@+id/tarifa"
    android:layout_gravity="right|top"
    android:layout_alignTop="@+id/map"
    android:layout_alignEnd="@+id/map"
    android:layout_toEndOf="@+id/textView"
    android:textColor="#ff000000" />

<EditText
    android:id="@+id/parada"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignEnd="@+id/calcular"
    android:layout_below="@+id/tarifa"
    android:layout_toEndOf="@+id/textView2"
    android:gravity="bottom"
    android:inputType="number"
    android:numeric="integer"
    android:textColor="#ff000000" />

<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_alignTop="@+id/textView2"
    android:layout_toEndOf="@+id/calcular"
    android:layout_alignBottom="@+id/calcular"
    android:id="@+id/radGrup">

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="IDA"
        android:id="@+id/ida"
        android:checked="true"
        android:onClick="irIda" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="REGRESO"
        android:id="@+id/regreso"
        android:checked="false"
        android:onClick="irRegreso" />
</RadioGroup>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:text="PARADA:    "
        android:textSize="20sp"
        android:id="@+id/textView2"
        android:layout_gravity="right|top"
        android:gravity="bottom"
        android:textColor="#ff000000"
        android:layout_below="@+id/parada"
        android:layout_alignParentStart="true"
        android:layout_marginTop="14dp" />

</RelativeLayout>
```

ANEXO C

CONEXIÓN A LA BASE DE DATOS (FUNCIÓN function)

```
<?php
header( 'Content-Type: text/html;charset=utf-8' );

function ejecutarSQLCommand($commando){

    $mysqli = new mysqli("localhost", "root", "temporal", "tarifa1");


/* check connection */
if ($mysqli->connect_errno) {
    printf("Connect failed: %s\n", $mysqli->connect_error);
    exit();
}

if ( $mysqli->multi_query($commando)) {


     if ($resultset = $mysqli->store_result()) {
        while ($row = $resultset->fetch_array(MYSQLI_BOTH)) {

        }
        $resultset->free();

     }


}



$mysqli->close();
}

function getSQLResultSet($commando){


  $mysqli = new mysqli("localhost", "root", "temporal", "tarifa1");

/* check connection */
if ($mysqli->connect_errno) {
    printf("Connect failed: %s\n", $mysqli->connect_error);

    exit();
}

if ( $mysqli->multi_query($commando)) {

        return $mysqli->store_result();

}

$mysqli->close();
}



?>
```

INGRESO DE DATOS A LA BASE DE DATOS

```
<?php include ('functions.php');
$id=$_GET['id'];
$fecha=$_GET['fecha'];
$hora=$_GET['hora'];
$latitud=$_GET['latitud'];
$longitud=$_GET['longitud'];
$tarifa=$_GET['tarifa'];




ejecutarSQLCommand("INSERT INTO  `bus1` (
`id` ,
`fecha` ,
`hora` ,
`latitud` ,
`longitud` , `tarifa`
)

VALUES (
'$id' ,
'$fecha' ,
'$hora' ,
'$latitud' ,
'$longitud','$tarifa')


ON DUPLICATE KEY UPDATE
`fecha`= '$fecha',
`hora`= '$hora',
`latitud`= '$latitud',
`longitud`= '$longitud'

;");


 ?>
```

ANEXO D

MARCODORES.

```
<?php

    $conectar =  mysqli_connect('localhost', 'root', 'temporal',
'tarifa1');


/* check connection */

if (mysqli_connect_errno($conectar)) {
    echo"Conexion failed:".mysqli_connect_error();
}




header('Content-Type: text/xml');

echo '<markers>';


$sql=mysqli_query($conectar,"select * from bus1");

while($row=mysqli_fetch_array($sql))
{
        echo "<marker id ='".$row['id']."' fecha ='".$row['fecha']."'
hora='".$row['hora']."' latitud='".$row['latitud']."' longitud
='".$row['longitud']."'>\n";
        echo "</marker>\n";
}

mysqli_close($bd);
echo "</markers>\n";
?>
```

REPORTE DEL RECORRIDO.

```
<?php

  $suma= 0;

    $conectar =  mysqli_connect('localhost', 'root', 'temporal',
'tarifa1');


/* check connection */

if (mysqli_connect_errno($conectar)) {
    echo"Conexion failed:".mysqli_connect_error();
}


$resultado = mysqli_query($conectar, "SELECT * FROM bus1");


?>

  <title>Tesis</title>

<table width="500" cellpadding="5" cellspacing="5"  border="1">
```

```
        <h1>Reporte del Bus</h1>

        <tr>

             <th> id     </th>
             <th> FECHA   </th>
             <th> HORA    </th>
             <th> LATITUD  </th>
             <th> LONGITUD </th>
             <th> TARIFA   </th>
        </tr>


        <?php while($fila = mysqli_fetch_array($resultado)): ?>



             <tr>

                 <td><?php echo $fila['id']; ?></th>
                 <td><?php echo $fila['fecha']; ?></th>
                 <td><?php echo $fila['hora']; ?></th>
                 <td><?php echo $fila['latitud']; ?></th>
                 <td><?php echo $fila['longitud']; ?></th>
                 <td><?php echo $fila['tarifa']; ?></th>

                 <?php $suma += $fila ['tarifa']?>

             </th>



        <?php endwhile;

?>


 <tr><td><td><td><td><td><td>TOTAL</td><td><?php echo number_format($suma,2)
?></th><tr>

</table>



<input type="button" onclick="history.back()" name="VOLVER ATRAS"
value="VOLVER ATRAS">
```

CONEXIÓN A LA BASE DE DATOS

```
<?php


                $mysqli = new MySQLi("localhost", "root","temporal",
"academ");
                if ($mysqli -> connect_errno) {
                        die( "Fallo la conexiÃ³n a MySQL: (" . $mysqli ->
mysqli_connect_errno()
                                . ") " . $mysqli ->
mysqli_connect_error());
                }
```

```
                else

?>
```

CERRAR SESIÓN.

```
<?php
session_start();
if($_SESSION['user']){
        session_destroy();
        header("location:index.php");
}
else{
        header("location:index.php");
}
?>
```

INTERFAZ INICIO DE SESIÓN.

```
<!DOCTYPE html>
<html>
<head>

        <meta charset="utf-8">
                <link rel="stylesheet"
href="bootstrap/css/bootstrap.css">
                <link rel="stylesheet" href="bootstrap/css/bootstrap-
responsive.css">
                <link rel="stylesheet" type="text/css"
href="estilos/estilos.css">
        <title>Tesis</title>
</head>
<body background="images/spray4.jpg" style="background-attachment:
fixed" >
        <center><div class="tit"><h2 style="color: #0000FF; ">Inicio de
sesiÃ³n</h2>
                <center><div class="Ingreso">

        <table border="0" align="center" valign="middle">
                <tr>
                <td rowspan=2>
                <form action="validar.php" method="post">

                <table border="0">

                <tr><td><label style="font-size: 14pt"><b>Usuario:
</b></label></td>
                        <td width=80> <input class="form-group has-
success" style="border-radius:15px;" type="text" name="mail"></td></tr>
                <tr><td><label style="font-size: 14pt"><b>ContraseÃ±a:
</b></label></td>
                        <td witdh=80><input style="border-radius:15px;"
type="password" name="pass"></td></tr>
                <tr><td></td>
                        <td width=80 align=center><input class="btn btn-
primary" type="submit" value="Aceptar"></td>
                        </tr></tr></table>
```

```
                </form>
<br>

                </td>
                </tr>
                </table>
                </div></center></div></center>


</body>
</html>
```

INTERFAZ REPORTE DEL RECORRIDO.

```
<!DOCTYPE html>

        <?php
        session_start();
        if (@!$_SESSION['user']) {
                header("Location:index.php");
        }elseif ($_SESSION['rol']==1) {
                header("Location:admin.php");
        }
        ?>

<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Tesis</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="Roberto Calero">

    <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet"/>
    <script src="bootstrap/js/jquery-1.8.3.min.js"></script>
    <script src="bootstrap/js/bootstrap.min.js"></script>


    <link rel="shortcut icon" href="assets/ico/favicon.ico">
    <link rel="apple-touch-icon-precomposed" sizes="144x144"
href="assets/ico/apple-touch-icon-144-precomposed.png">
    <link rel="apple-touch-icon-precomposed" sizes="114x114"
href="assets/ico/apple-touch-icon-114-precomposed.png">
    <link rel="apple-touch-icon-precomposed" sizes="72x72"
href="assets/ico/apple-touch-icon-72-precomposed.png">
    <link rel="apple-touch-icon-precomposed" href="assets/ico/apple-
touch-icon-57-precomposed.png">
  </head>
<body data-offset="40" background="images/spray4.jpg" style="background-
attachment: fixed">
<div class="container">
<header class="header">
<div class="row">
        <?php
        include("include/cabecera.php");
        ?>
</div>
</header>

  <!-- Navbar
    ================================================== -->
```

```
================================================================================
============================================== -->


<hr>
<div class="row">

        <div class="span4">
        <div class="thumbnail">
        <h3 style="text-align:center">REPORTE DEL RECORRIDO </h3>
        <img src="images/reporte.jpg" alt="#"/>
        <div class="caption">
        <p align="justify">


        </p>
        <a class="pull-right" href="AbrirBase2.php">GENERAR REPORTE</a>
        <br/>
        </div>
        </div>
        </div>

        <div class="span4">
        <div class="thumbnail">
        <h3 style="text-align:center">TRAYECTORIA DEL BUS EN TIEMPO
REAL</h3>
        <img src="images/map1.jpg" />
        <div class="caption">
        <p align="justify">


        </p>
        <a class="pull-right" href="Mapa_completo1.php">GENERAR MAPA</a>
        <br/>
        </div>
        </div>
        </div>


</div>
<hr/>
<!-- Footer
      ================================================= -->
<hr class="soften"/>
<footer class="footer">

<hr class="soften"/>
<p>&copy; Copyright Roberto Calero <br/><br/></p>
 </footer>
</div><!-- /container -->

        </style>
  </body>
</html>
```

VALIDACIÓN DEL USUARIO.

```
<?php
//include("connect_db.php");

//$miconexion = new connect_db;


session_start();
        require("connect_db.php");

        $username=$_POST['mail'];
        $pass=$_POST['pass'];

        $sql2=mysqli_query($mysqli,"SELECT * FROM login WHERE
email='$username'");
        if($f2=mysqli_fetch_assoc($sql2)){
                if($pass==$f2['pasadmin']){
                        $_SESSION['id']=$f2['id'];
                        $_SESSION['user']=$f2['user'];

                ';

                }
        }


        $sql=mysqli_query($mysqli,"SELECT * FROM login WHERE
email='$username'");
        if($f=mysqli_fetch_assoc($sql)){
                if($pass==$f['password']){
                        $_SESSION['id']=$f['id'];
                        $_SESSION['user']=$f['user'];


                        header("Location: index2.php");
                }else{
                        echo '<script>alert("CONTRASEÃ‘A
INCORRECTA")</script> ';

                        echo
"<script>location.href='index.php'</script>";
                }
        }else{

                echo '<script>alert("ESTE USUARIO NO EXISTE")</script>
';

                echo "<script>location.href='index.php'</script>";

        }

?>
```

MAPA DE GOOGLE MAPS CON MARCADORES.

```
<!DOCTYPE html >
  <head>
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no"
/>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
    <title>Tesis</title>
    <input type="button" onclick="history.back()" name="VOLVER ATRAS"
value="VOLVER ATRAS">
    <style>
      /* Always set the map height explicitly to define the size of the
div
       * element that contains the map. */
      #map {
        height: 100%;
      }
      /* Optional: Makes the sample page fill the window. */
      html, body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
  </head>

  <body>
    <div id="map"></div>


        function initMap() {
        var map = new google.maps.Map(document.getElementById('map'), {
          center: new google.maps.LatLng(-0.469215, -78.795022),
          zoom: 8
        });
        var infoWindow = new google.maps.InfoWindow;

          // Change this depending on the name of your PHP or XML file
          downloadUrl("AbrirBase.php", function(data) {
            var xml = data.responseXML;
            var markers =           ……
            xml.documentElement.getElementsByTagName('marker');
            Array.prototype.forEach.call(markers, function(markerElem) {
              var id = markerElem.getAttribute('id');
              var fecha = markerElem.getAttribute('fecha');
              var hora = markerElem.getAttribute('hora');
              var point = new google.maps.LatLng(
                  parseFloat(markerElem.getAttribute('latitud')),
                  parseFloat(markerElem.getAttribute('longitud')));

              var infowincontent = document.createElement('div');
              var strong = document.createElement('strong');
              strong.textContent = hora
              infowincontent.appendChild(strong);
              infowincontent.appendChild(document.createElement('br'));

              var text = document.createElement('text');
              text.textContent = fecha
              infowincontent.appendChild(text);

              var marker = new google.maps.Marker({
                map: map,
                position: point,
              });
              marker.addListener('click', function() {
                infoWindow.setContent(infowincontent);
                infoWindow.open(map, marker);
              });
```

```
            });
          });
        }


      function downloadUrl(url, callback) {
        var request = window.ActiveXObject ?
            new ActiveXObject('Microsoft.XMLHTTP') :
            new XMLHttpRequest;

        request.onreadystatechange = function() {
          if (request.readyState == 4) {
            request.onreadystatechange = doNothing;
            callback(request, request.status);
          }
        };

        request.open('GET', url, true);
        request.send(null);
      }

      function doNothing() {}
    </script>
    <script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBoOuQkx1V0XiEocOr
u6ulyr6VLOZmKPik&callback=initMap">
    </script>

  </body>
</html>
```