| | | |
|---|---|---|
| | **Sprint 5** | |
| 62 | Desarrollo de Servicios Web en PHP | 1 |
| 63 | Creación de base de datos SQLite | 3 |
| 64 | Consumo de Servicios Web | 3 |
| 65 | Pruebas de funcionamiento de base de datos SQLite | 2 |
| | **Sprint 6** | |
| 66 | Diseño y desarrollo de la interfaz de solicitud de permisos | 1 |
| 67 | Solicitud de GPS activo | 2 |
| 68 | Solicitud del permiso de Ubicación | 2 |
| 69 | Verificar que la base de datos está creada y contiene data | 1 |
| 70 | Pruebas del sexto sprint | 4 |
| | **Sprint 7** | |
| 71 | Diseño y Desarrollo del menú de la aplicación | 1 |
| 72 | Diseño y desarrollo de las interfaces para visualizar las empresas registradas | 1 |
| 73 | Desarrollo del proceso de mostrar empresa registrada elegida | 1 |
| 74 | Diseño y desarrollo de las interfaces de vista de rutas | 1 |
| 75 | Desarrollo del proceso para mostrar las ruta registrada elegida | 1 |
| 76 | Gráfico de rutas de ida y vuelta | 1 |
| 77 | Pruebas del séptimo sprint | 6 |
| | **Sprint 8** | |
| 78 | Diseño de las interfaces para elección del punto de origen y destino del viaje | 1 |
| 70 | Desarrollo de la interfaz de origen y destino de viaje | 1 |
| 80 | Inserción de mapas de Google en ambas interfaces (origen y destino) | 1 |
| 81 | Inserción de íconos de paraderos en mapas de Google | 1 |
| 82 | Programación de eventos en mapas de Google | 2 |
| 83 | Control de coordenadas ingresadas en referencia a paraderos registrados | 1 |
| 84 | Programación de intents | 1 |
| 85 | Pruebas de sprint 8 | 8 |
| | **Sprint 9** | |
| 86 | Diseño de la interfaz de sugerencias de empresa en las que viajar | 1 |
| 87 | Desarrollo de interfaz de sugerencia de empresas en las que viajar | 1 |
| 88 | Programación del proceso para generar sugerencias de empresas en las que viajar | 4 |
| 89 | Diseño de la interfaz de guiado de viaje | 1 |
| 90 | Desarrollo del proceso de guiado de viaje | 1 |
| 91 | Gráfico de ruta a seguir y paraderos a usar | 1 |
| 92 | Seguimiento en tiempo real mediante el sistema de GPS | 1 |

Fuente propia.

### 4.3.5 Modelo de Negocios (Business Model Canvas)

Hay distintas maneras de mostrar un modelo de negocios, y en este proyecto se utilizó una de las más usadas hoy en día, el Business Model Canvas, que puede apreciarse en la tabla 9.

### 4.3.6 Diagrama de base de datos

Los diagramas de base de datos sirven para poder visualizar la estructura de una base de datos; y las relaciones que mantienen las tablas dentro de la misma.

En el presente proyecto tenemos dos bases de datos, la del backend y la del frontend. Ambas son muy similares, sin embargo solo la del backend tiene una tabla para usuarios. Los diagramas de base de datos fueron diseñados haciendo uso de Workbench, para posteriormente crear la base de datos a partir de los mismos. La figura 14 muestra la base de datos del backend (sistema web) y la figura 15 del frontend (aplicativo Android).

### 4.3.7 Prototipo de Interfaces.

La elección de colores es una tarea importante al momento de diseñar interfaces, en el presente proyecto se buscó colores que generen buen contraste entre ellos para que los textos mostrados sean de fácil lectura, y también sencillez en cuanto a los elementos que conforman las interfaces; de esa manera se evita complicar las actividades de desarrollo y uso de las mismas. Los prototipos de interfaces tanto para el backend como para el frontend son los siguientes:

En la figura 12 se aprecia el prototipo de interfaz del backend, es muy simple, la pantalla se divide en 2, la parte izquierda principalmente para el menú, y la parte de la derecha para formularios. Cabe resaltar que el color elegido para contrastar con el fondo azulado es el blanco, y por ello los

textos ubicados en esas áreas son de ese color. De manera similar, el color elegido para resaltar sobre el fondo blanco; ha sido el negro.


*Figura 12*. Prototipo de interfaz del Backend

En cuanto al frontend, la interfaz básicamente tiene los mismos colores que el backend, y sigue los mismos patrones para contrastar colores, si se tiene fondos azulados, se utiliza texto de color blanco, y si se tiene fondos blancos; se utiliza texto color negro. La figura 13 muestra el prototipo del frontend.


*Figura 13*. Prototipo de interfaz del Frontend

Tabla 9

*Modelo de Negocios – Business Model Canvas*

<table>
<tr>
<td rowspan="2">Socios Clave<br><br>Gobierno Municipal del Cusco – Distintas rutas de transporte, donde se vean las rutas concesionadas a seguir de las empresas de transporte.<br><br>Empresas que deseen aparecer dentro de los mapas de la aplicación.</td>
<td>Actividades Clave<br><br>Ubicación del usuario dentro del mapa por medio del GPS.<br><br>Capacidad de identificar mediante coordenadas el lugar a donde el usuario desea ir.<br><br>Mostrar paraderos, empresas y las rutas a seguir dependiendo de lo que el usuario elija.<br><br>Brindar la sensación de seguridad que se tiene a partir de conocer hacia dónde vamos, por dónde y desde dónde.</td>
<td rowspan="2">Propuesta de Valor<br><br>Tranquilidad para saber cómo llegar de un lugar a otro haciendo uso del servicio de transporte público<br><br>Poder elegir desde qué paradero subir a un bus.<br><br>Conocer la ruta a seguir y la ubicación actual.<br><br>Saber dónde bajar del bus y cómo llegar al destino.</td>
<td>Relación con clientes<br><br>Interfaz amigable e intuitiva.<br><br>Página en Facebook para recoger opiniones de los usuarios para mejorar constantemente la aplicación.</td>
<td rowspan="2">Segmentos de clientes<br><br>Habitantes o visitantes de la ciudad del Cusco que cuenten con un teléfono con sistema operativo Android. Entre los 17 y 60 años. Incluye extranjeros que hablen Español o Inglés.</td>
</tr>
<tr>
<td>Recursos Clave<br><br>- Desarrollador web.<br>- Desarrollador Android.<br>- Diseñador de interfaces.<br>- Servicio de dominio y hosting.<br>- PC para desarrollo.<br>- Conexión a internet.<br>- Inversión de capital.</td>
<td>Canales<br>Web – Facebook<br><br>Stickers o volantes en restaurantes y locales que deseen aparecer en el mapa de la aplicación.</td>
</tr>
<tr>
<td colspan="2">Estructura de Costes<br><br>Recursos Humanos S/. 20,000.00<br>Equipos de Computo y móvil S/. 3,199.00<br>Otros: Servicios de luz e internet, transporte, impresiones, hosting, dominio, etc. S/.3205.56</td>
<td colspan="3">Fuente de Ingresos<br>Publicidad dentro de al aplicación.<br><br>Contratos con empresas para aparecer primero en las sugerencias de rutas<br><br>Donaciones de los usuarios mediante Paypal.</td>
</tr>
</table>

Fuente propia.

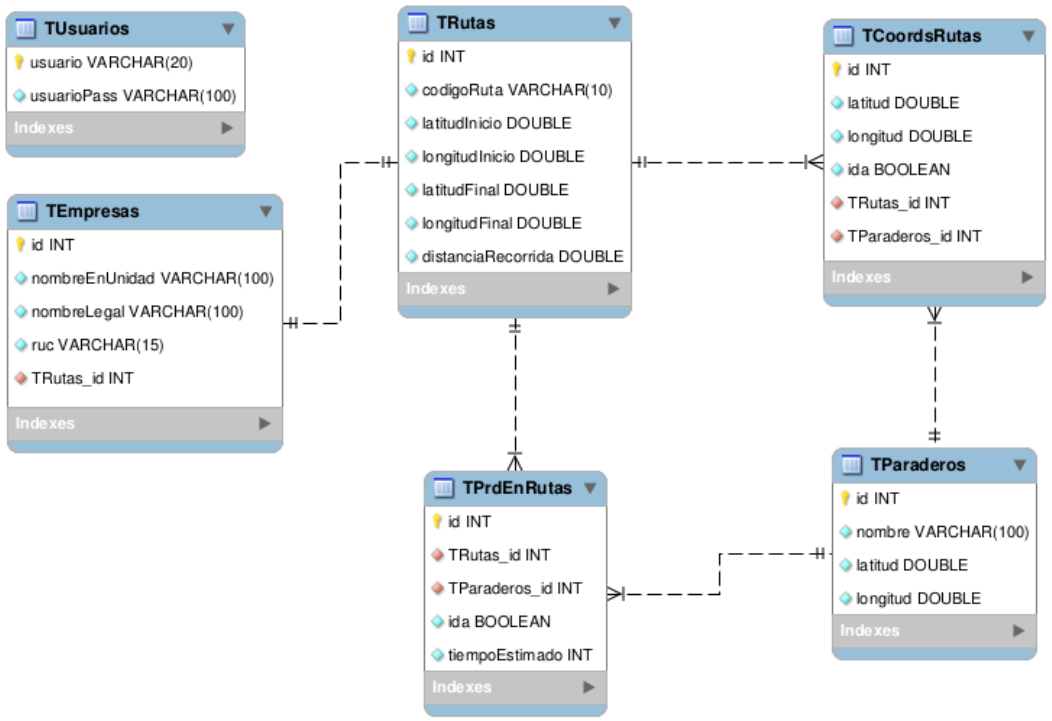



*Figura 14.* Diagrama de base de datos del Backend. Fuente propia.

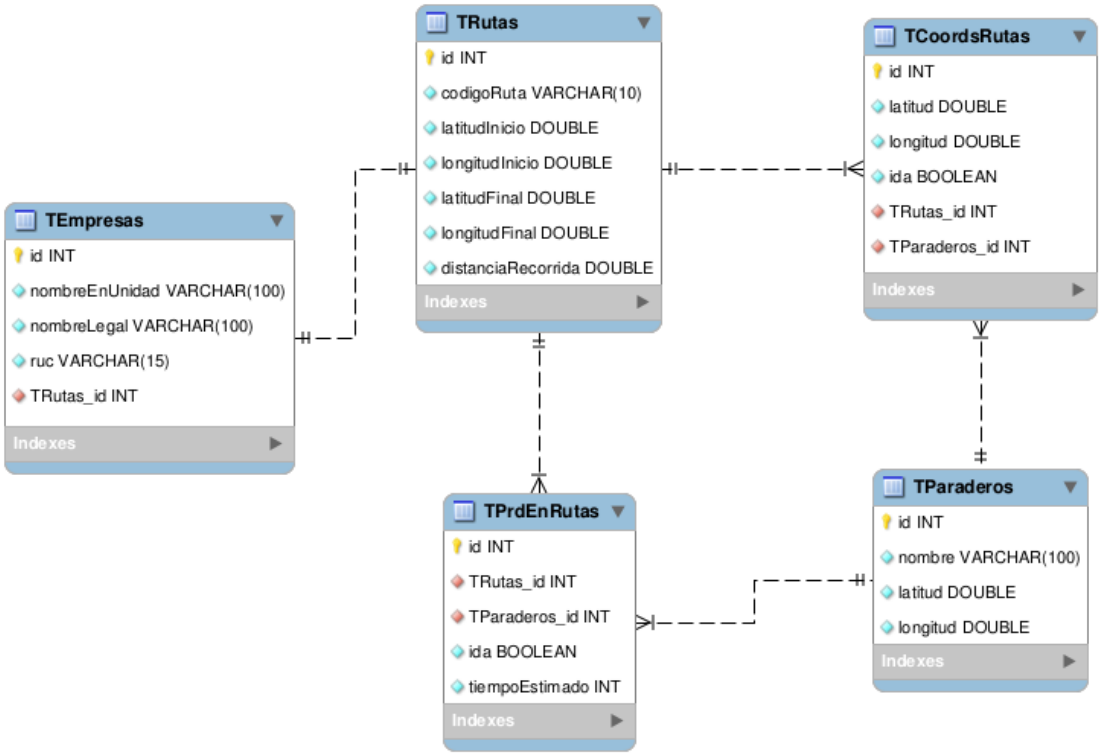



*Figura 15.* Diagrama de base de datos del Frontend. Fuente propia.

### 4.3.8 Descripción y comparación de cada proceso.

La fase de construcción ha constado de 9 Sprints, los tres primeros para el backend, y los seis restantes para el frontend. La mayoría de ellos generan avances importantes en el producto tanto del backend como del frontend, sin embargo también se han considerado algunos que incluyen tareas investigativas, sin las cuales el desarrollo del presente proyecto habría sido imposible.

Los Sprints del backend fueron desarrollados en su gran mayoría en NetBeans y Workbench, que son herramientas para desarrollo en PHP y manejo de MySql. Por otro lado, la gran mayoría del frontend fue desarrollado usando Android Studio, acompañado por programas como “DB Browser for SQLite”.

En la tabla 10 se describe cada requerimiento que existe en la Lista de Producto.

Tabla 10

*Descripción de cada proceso*

| Nro | Requerimiento y Descripción |
|---|---|
| | |
| | **Backend** |
| | **Sprint 1** |
| 1 | **Investigación sobre uso de los mapas de Google.** Mediante esta investigación, se busca entender cómo funcionan los mapas de Google en la web. Esto quiere decir, qué requerimientos existe para usarlos, limitaciones, scripts necesarios, etc. Implementando los puntos que al momento se consideren necesarios para probar. |
| 2 | **Extraer coordenadas de puntos elegidos en el mapa.** Busca, como dice el requerimiento, extraer coordenadas del mapa de Google para así poder tratar esos datos. Se prueba el evento click sobre el mapa y la obtención de las coordenadas. |
| 3 | **Probar el gráfico de rutas.** Es muy importante para el presente proyecto el gráfico de las rutas, ya que es una de las tareas esenciales. Este requerimiento busca entender y probar el funcionamiento de los gráficos de rutas en los mapas de Google; siguiendo el trazado que uno desee. |
| 4 | **Diseño del diagrama entidad relación** Sabiendo cómo trabajan las funciones que se requieren de los mapas de Google, se diseña el diagrama entidad relación de la base de datos. |

| | |
|---|---|
| | **Sprint 2** |
| **5** | **Codificación básica de la interfaz y la hoja de estilos.**<br>Se programa la interfaz gráfica básica, similar a la del prototipo, sobre la cual se basarán las demás interfaces. De manera similar; se programa la hoja de estilos, la cual será modificada a futuro de acuerdo a las necesidades particulares del resto de interfaces, sin embargo es aquí donde se crean las bases de al misma. |
| **6** | **Programación de las clases**<br>De acuerdo al diagrama entidad relación, se procede a crear las clases y programar las funciones básicas de las mismas (getters y setters). El resto de funciones se irán programando a medida que se avanza. |
| **7** | **Creación de la base de datos y sus usuarios**<br>Se crea la base de datos MySql a partir del diagrama entidad relación, también se crea el usuario que va a darle mantenimiento desde el backend, otorgándole los permisos necesarios. |
| **8** | **Creación de un loggeo para administrar el backend**<br>Para poder ingresar al backend, es necesario crear un formulario de acceso, para así evitar el acceso de personas ajenas o malintencionadas. |
| | **Sprint 3** |
| **9-21** | **Mantenimiento de Paraderos**<br>Quiere decir, agregar, modificar y eliminar registros de la tabla Tparaderos. Para dar el mantenimiento necesario a dicha tabla, es necesario diseñar y construir distintas interfaces, también programar todo lo necesario para este fín. Incluye pruebas continuas a medida que se va avanzando |
| **22-41** | **Mantenimiento de rutas**<br>Significa; agregar, modificar y eliminar data de rutas del sistema. Para lograr mantener las rutas en el sistema, se hace uso de dos tablas, TRutas y TCoordsRutas, por lo tanto se debe diseñar y construir las interfaces necesarias, además programar todo lo necesario para llevar a cabo dicho requerimiento. Incluye las pruebas necesarias. |
| **42-51** | **Mantenimiento de Empresas**<br>Significa agregar, modificar y eliminar datos de empresas en el sistema. Incluye el diseño y construcción de interfaces, y toda la programación necesaria para llevar cumplir con dicho requerimiento. También incluye las pruebas necesarias |

| | |
|---|---|
| | **FRONTEND** |
| | **Sprint 4** |
| **52** | **Investigación sobre desarrollo en Android**<br>Este requerimiento busca investigar cómo se programa en Android, comprender la manera en que se organiza un proyecto en Android, la creación de teléfonos inteligentes virtuales, y también el uso de teléfonos reales como dispositivo donde se ejecuta el aplicativo. |
| **53** | **Investigación sobre funcionamiento de XML**<br>Para poder entender cómo funciona XML. |
| **54** | **Investigación diseño y desarrollo de interfaces en Android**<br>Para ser capaces de diseñar y programar las interfaces que se necesitan dentro de la App, utilizando los distintos elementos que se tiene disponible. |
| **55** | **Investigación de mapas de Google para Android**<br>Para entender cómo se utilizan los mapas de Google dentro de dispositivos móviles que utilizan Android, incluyendo los requerimientos que se debe cumplir, limitaciones, etc. |

| | |
|---|---|
| 56 | **Investigación de gráfica de rutas para mapas Google en Android**<br>Para entender cómo se grafican rutas dentro de los mapas de Google en una aplicación nativa para Android. |
| 57 | **Investigación ingreso de íconos y marcadores en mapas Google para Android**<br>Para de esa manera poder ingresar los íconos necesarios dentro de la vista del mapa, además de entender los eventos que se pueden aplicar sobre cada uno. Incluye las pruebas necesarias para entender el funcionamiento |
| 58 | **Investigación de seguimiento mediante GPS en Android**<br>Para poder ubicar a los usuarios, y a su vez, poder hacerles seguimiento a medida que van avanzando. |
| 59 | **Investigación de Servicios Web**<br>Para poder transportar la data de la base de datos en línea a la base de datos local de los dispositivos móviles; es necesario hacer uso de servicios web. En este punto se logra entender cómo funcionan. |
| 60 | **Investigación de consumo de Servicios Web en Android**<br>Una vez creados los servicios, para poder consumirlos se debe seguir un procedimiento. Mediante esta investigación se logra consumir los servicios web con un dispositivo Android conectado a la red. |
| 61 | **Investigación sobre SQLite**<br>SQLite es la base de datos que utilizan los dispositivos móviles Android de manera local. En esta investigación se busca, entre otras cosas, entender qué diferencias importantes tiene en comparación a MySql y los tipos de datos que soporta. |
| | **Sprint 5** |
| 62 | **Desarrollo de Servicios Web en PHP**<br>Se crean los servicios web necesarios para que la aplicación Android pueda consumirlos posteriormente. |
| 63 | **Creación de base de datos SQLite**<br>Se crea la base de datos local en el dispositivo móvil, esto quiere decir, la base de datos, las tablas, y las relaciones entre las mismas. |
| 64 | **Consumo de Servicios Web**<br>Se programa todo lo necesario para consumir los servicios web e ingresarlos en la base de datos SQLite del dispositivo móvil. |
| 65 | **Pruebas de funcionamiento de base de datos SQLite**<br>Se realizan las pruebas necesarias dentro del dispositivo móvil para asegurar la integridad de los datos, y que todo esta funcionando correctamente. |
| | **Sprint 6** |
| 66 | **Diseño y desarrollo de la interfaz de solicitud de permisos**<br>Busca diseñar y programar la interfaz de solicitud de permisos, mediante la cual se obtiene el permiso de ubicación, y se solicita al usuario que active el GPS del móvil |
| 67 | **Solicitud de GPS activo**<br>Se programa el conjunto de procesos para solicitar al usuario que active su GPS en caso lo tenga desactivado. |
| 68 | **Solicitud del permiso de Ubicación**<br>Se programa el conjunto de procesos para solicitar al usuario que brinde a la aplicación el permiso de ubicación, para de esa manera poder acceder a los datos que brinda el GPS. |
| 69 | **Verificar que la base de datos está creada y contiene data**<br>Este requerimiento es importante para poder ingresar a la aplicación y tener data |

| | |
|---|---|
| | en la base de datos, ya que el proceso de consumo de servicios web se hace de manera asíncrona. |
| **70** | **Pruebas del sexto sprint**<br>Se realizan las pruebas necesarias para poder avanzar al siguiente sprint. |
| | **Sprint 7** |
| **71** | **Diseño y Desarrollo del menú de la aplicación**<br>Se diseña y programa la interfaz del menú de la aplicación. |
| **72** | **Diseño y desarrollo de las interfaces para visualizar las empresas registradas**<br>Para visualizar el listado de empresas, y los datos de las mismas. |
| **73** | **Desarrollo del proceso de mostrar datos de empresa registrada elegida**<br>Para mostrar los datos de la empresa que se eligió. |
| **74** | **Diseño y desarrollo de las interfaces de vista de rutas**<br>Se diseña y desarrolla las interfaces para visualizar el listado de rutas registradas, y el trazado sobre los mapas de Google de las mismas. |
| **75** | **Desarrollo del proceso para mostrar datos de la ruta registrada elegida**<br>Para visualizar los datos de la ruta que se eligió |
| **76** | **Gráfico de rutas de ida y vuelta**<br>Se extrae los datos necesarios de la base de datos, y se procede a graficar la ruta de ida o vuelta, dependiendo qué elija el usuario |
| **77** | **Pruebas del séptimo sprint**<br>Se realizan las pruebas necesarias para pasar al siguiente sprint. |
| | **Sprint 8** |
| **78** | **Diseño de las interfaces para elección del punto de origen y destino del viaje**<br>Se diseña la interfaz donde el usuario indica al sistema desde dónde y hacia dónde desea viajar. |
| **79** | **Desarrollo de la interfaz de origen y destino de viaje**<br>Una vez diseñadas, se procede a desarrollar las interfaces, se programa los textos, botones, y “fragments” necesarios. |
| **80** | **Inserción de mapas de Google en ambas interfaces (origen y destino)**<br>Se ingresa mapas de Google donde los usuarios eligen el punto de origen y destino, siguiendo los requerimientos de Google. |
| **81** | **Inserción de íconos de paraderos en mapas de Google**<br>Para que el usuario pueda ver dónde se encuentran los paraderos en el mapa. |
| **82** | **Programación de eventos en mapas de Google**<br>Para poder controlar los diferentes eventos que se requiere para elegir los puntos de origen y destino. |
| **83** | **Control de coordenadas ingresadas en referencia a paraderos registrados**<br>Para evitar que el usuario elija coordenadas muy alejadas de la ciudad. |
| **84** | **Programación de intents**<br>Los intents sirven para pasar de una vista hacia otra, además pueden llevar datos consigo, los cuales son recuperados en la vista objetivo. |
| **85** | **Pruebas de sprint 8**<br>Para asegurarse que hasta este punto todo va bien, y así poder pasar al sprint 9. |
| | **Sprint 9** |
| **86** | **Diseño de la interfaz de sugerencias de empresa en las que viajar**<br>Se diseña la interfaz mediante la cual el sistema sugiere qué empresas pueden llevar al usuario a su destino. |

| | |
|---|---|
| 87 | **Desarrollo de interfaz de sugerencia de empresas en las que viajar**<br>Se desarrolla la interfaz diseñada previamente. |
| 87 | **Programación del proceso para generar sugerencias de empresas en las que viajar**<br>Se programa todo lo necesario para mostrar sugerencias al usuario dependiendo de las coordenadas ingresadas, además se programa los intents que llevan los datos a la siguiente vista, la vista de viaje. |
| 89 | **Diseño de la interfaz de guiado de viaje**<br>Se diseña la interfaz mediante la cual el usuario recibe la guía necesaria para llegar a su destino sin inconvenientes. |
| 90 | **Desarrollo del proceso de guiado de viaje**<br>Mediante los datos obtenidos previamente, se procede a programar el proceso de viaje a seguir, tanto si se irá a pie o en bus. |
| 91 | **Gráfico de ruta a seguir y paraderos a usar**<br>Se grafica el camino a seguir al paradero de subida, la ruta a seguir, y del paradero de bajada a la coordenada de destino elegida. |
| 92 | **Seguimiento en tiempo real mediante el sistema de GPS**<br>Se activa el seguimiento de GPS, servicio que el usuario activó previamente. De esta forma el usuario puede ver en el móvil su ubicación en tiempo real. |

Fuente propia

### 4.3.9 Instalación y configuración del software Usado

En el presente proyecto, se utilizó el sistema operativo Linux; la versión Ubuntu 16.04 Xenial de 64 bits, por lo que se tuvo que configurar e instalar todo lo necesario para llevar a cabo el desarrollo. Vale la pena dejar en claro el proceso de instalación y configuración del software usado ya que no es necesariamente igual a Windows, si bien en Ubuntu se puede instalar y desinstalar software de manera similar a Windows, este proceso puede llevarse a cabo por línea de comandos, y en algunos casos es imperativo hacerlo por este medio. A continuación se explicará cómo se instalaron o configuraron los siguientes programas.

#### 4.3.9.1 Instalación del Servidor Apache y PHP

En la presente tesis se utilizó el servidor apache debido a su gran aceptación dentro de la comunidad de desarrolladores, y también porque es software libre. Para instalar este servidor basta ingresar las siguientes dos líneas de comando.

```
sudo apt-get update
```

sudo apt-get install apache2

La versión de Ubuntu usada ya trajo consigo el servidor Apache y PHP versión 7.0, sin embargo se hizo uso de PHP 5.6 para el desarrollo del proyecto dado que la documentación disponible para la versión 5.6 era mayor, y de esa manera se podía evitar algunos posibles problemas. Sin embargo se tuvo que configurar un poco para poder hacer uso de esa versión, para ello se hizo uso de un Paquete Personal de Archivos (PPA en inglés, Personal Package Archives), y para ello se siguió la guía posteada en el blog llamado Lornajane (Lornajane)

#### 4.3.9.2 Netbeans

La instalación de Netbeans 8.1 fue similar a como se realiza en Windows, pero con una ligera variación, el archivo descargado para Ubuntu tiene una extensión ".sh", por lo tanto para ejecutarlo e instalar hay que ingresar por línea de comandos:

sudo sh *Ruta del archivo* /netbeans-8.1-linux.sh

#### 4.3.9.3 Android Studio

En el caso del Android Studio, la instalación fue diferente a Windows, se usó la versión 2.1.2 de Android Studio y se mantuvo la misma en todo el proceso de desarrollo ya que era la más estable en ese momento, mientras que en las versiones posteriores existían algunos errores aún no solucionados en ese momento. Para instalar se ingresó a la página oficial, se descargó el archivo; que en este caso era un comprimido (.zip), se descomprimió en el lugar indicado y fue todo.

#### 4.3.9.4 StarUML

Los proyectos de software requieren generalmente hacer uso del UML para generar diagramas, este programa tiene la característica de no ser gratuito, pero permite hacer uso del mismo de manera indefinida. Para instalar este programa se descargó un paquete con extensión ".deb" de su página web oficial; en la versión de 64 bits para Ubuntu. Una vez

descargado el paquete; se procede a ingresar la siguiente línea de comando en el terminal de Ubuntu para instalar:

sudo dpkg -i *Ruta del Archivo*/StarUML-v2.7.0-64-bit.deb

#### 4.3.9.5 Pencil

Pencil es una herramienta gratuita para generar diseños de interfaces. La Instalación del mismo se puede hacer de varias maneras, sin embargo para este caso se utilizó un programa llamado “Synaptics”, el cual se instalo de manera sencilla usando otro programa que viene con Ubuntu llamado “Software de Ubuntu”, que básicamente es una “playstore” bastante básica para Ubuntu.

### 4.3.10 Planificación de entregas

La planificación de entregas se basó en los nueve sprints generados dentro de la Lista de Producto, cada sprint contiene una serie de requerimientos que deben ser completados durante la duración del sprint.

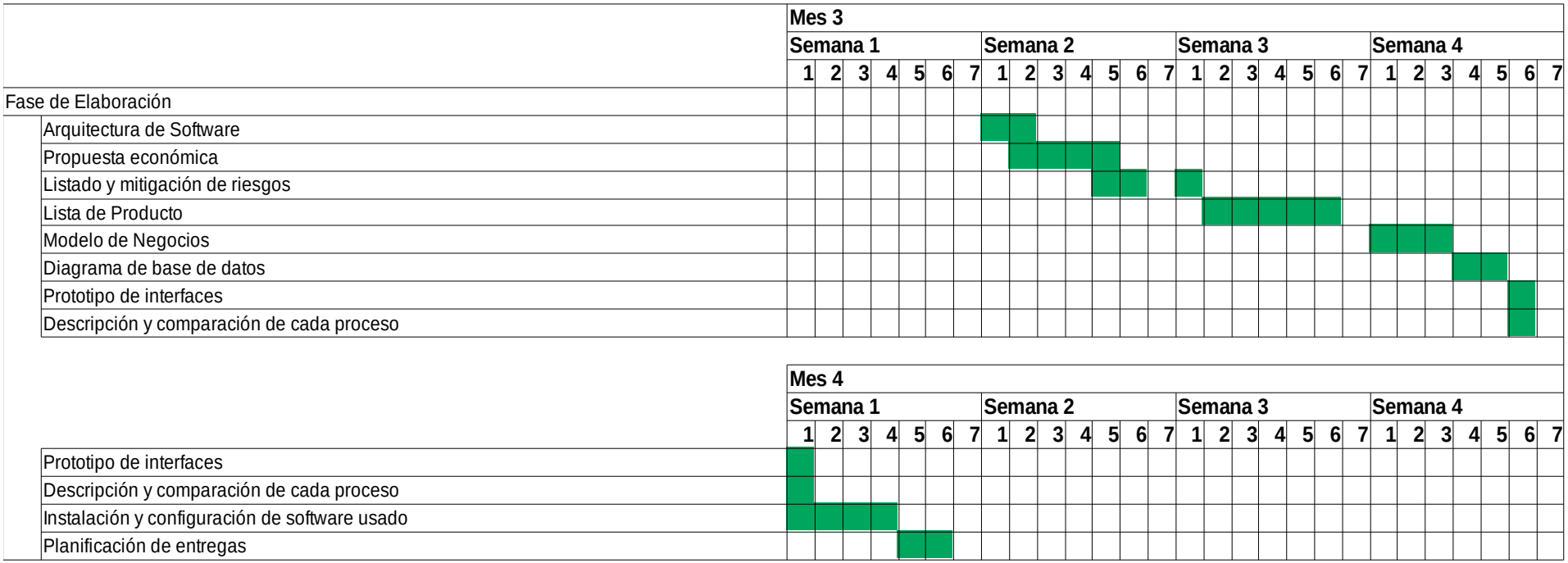
Los sprints tienen la característica de empezar cuando termina el anterior, lo más antes posible, por lo tanto no existe concurrencia de desarrollo entre uno y otro.

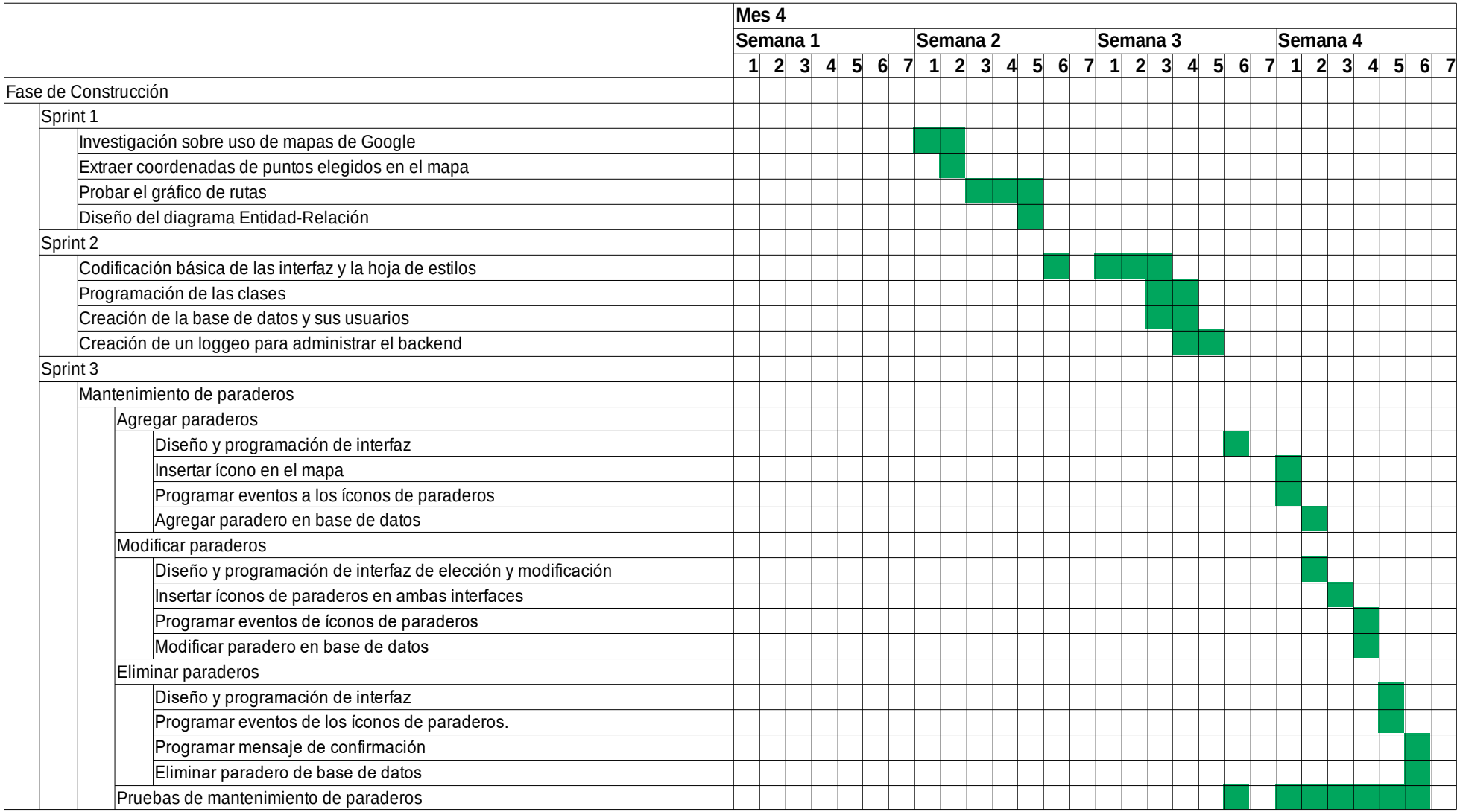
La planificación de entregas se encuentra en la tabla 11.

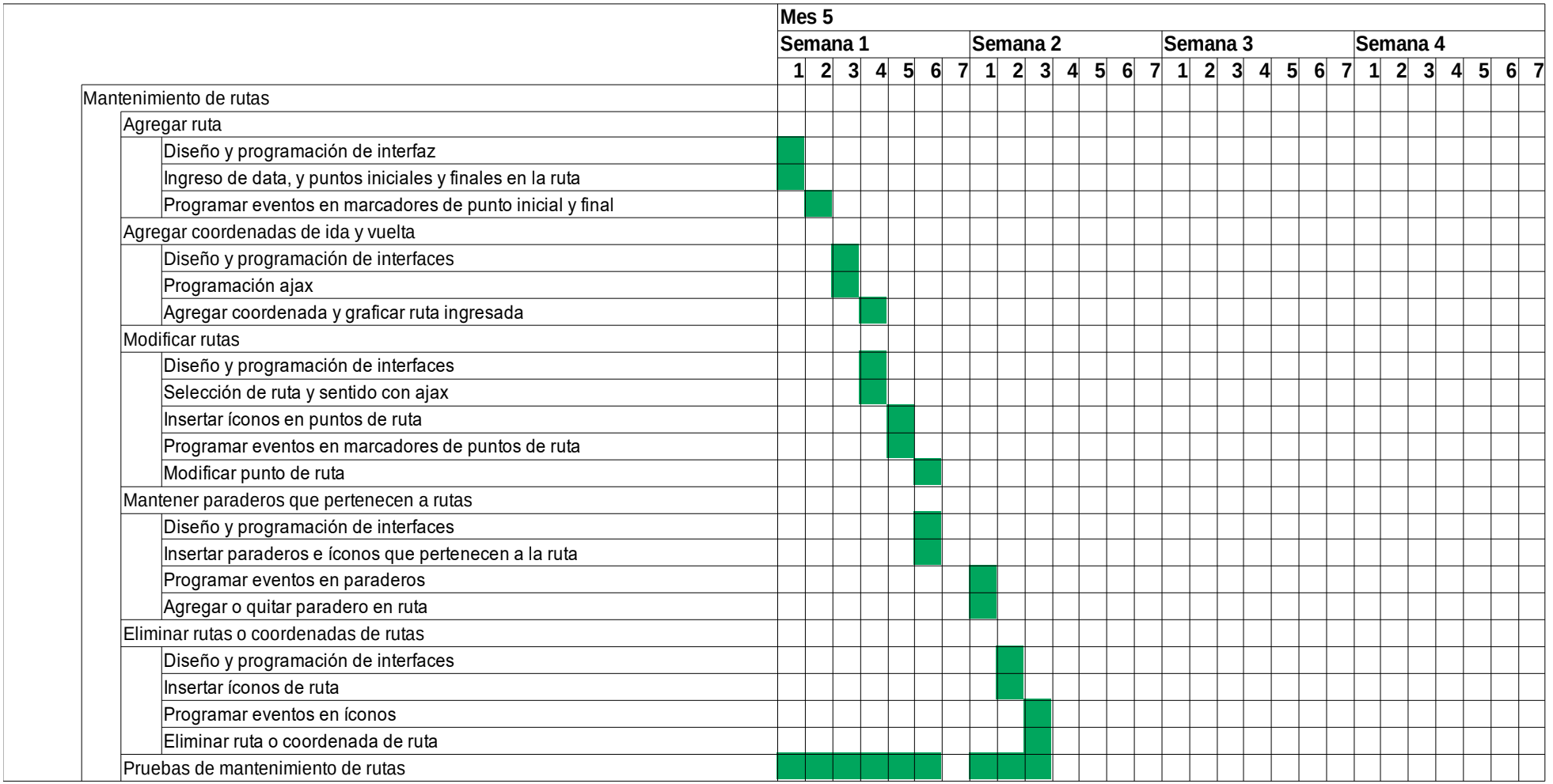
Tabla 11

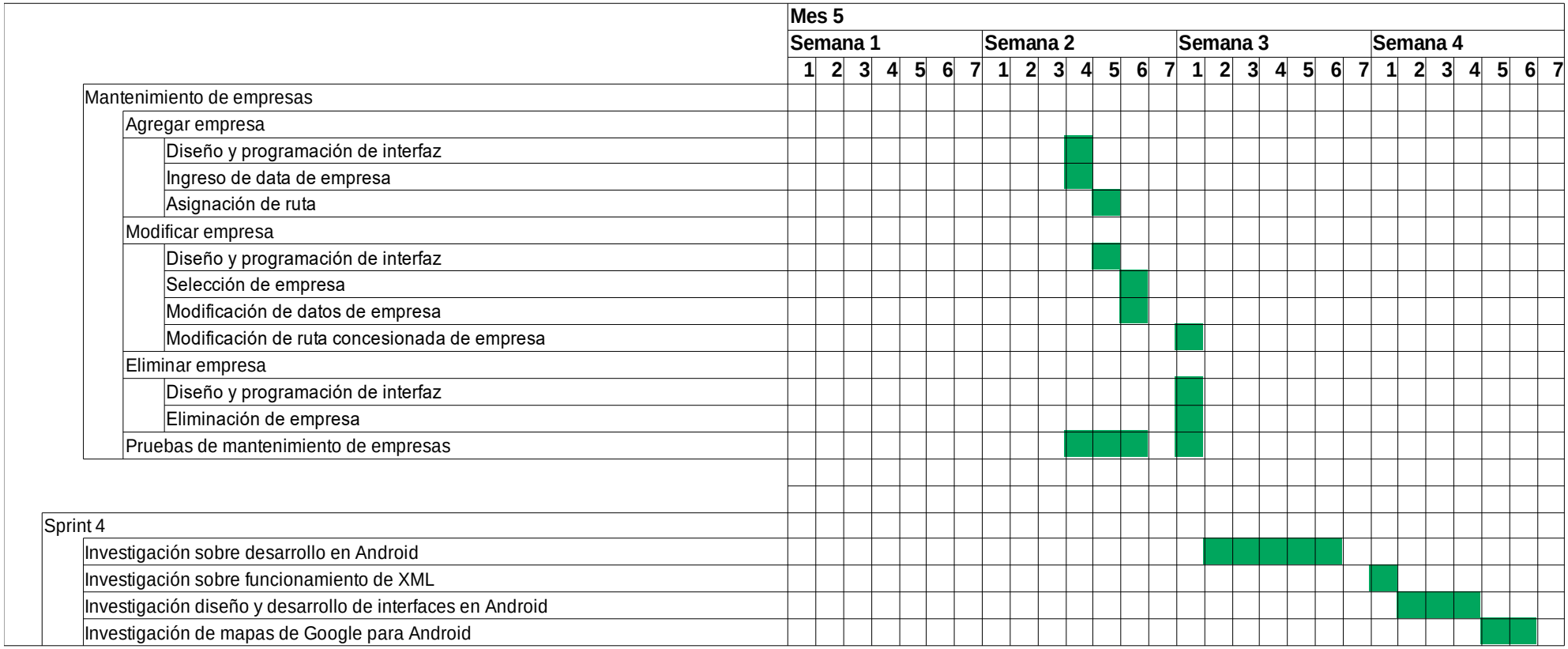
*Planificación de entregas*

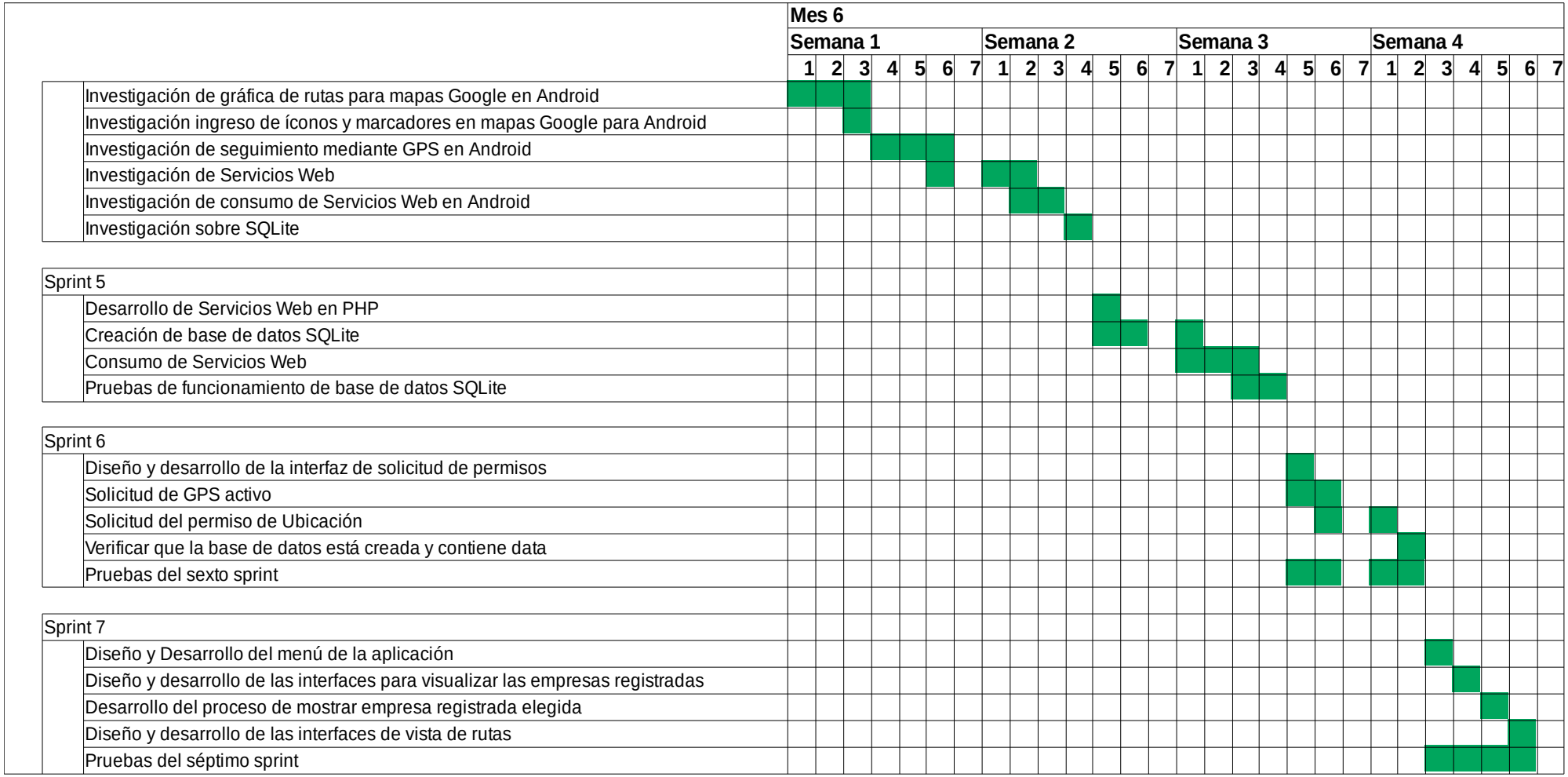
| ACTIVIDADES – REQUERIMIENTOS | TIEMPO |
|---|---|
| | Mes 1 |
| | Semana 1 (1 2 3 4 5 6 7) · Semana 2 (1 2 3 4 5 6 7) · Semana 3 (1 2 3 4 5 6 7) · Semana 4 (1 2 3 4 5 6 7) |
| Esquema de proyecto de tesis | ■ ■ ■ ■ ■ ■ · ■ ■ ■ ■ ■ ■ · ■ ■ ■ ■ ■ ■ · ■ ■ ■ ■ ■ ■ · |
| | Mes 2 |
| | Semana 1 (1 2 3 4 5 6 7) · Semana 2 (1 2 3 4 5 6 7) · Semana 3 (1 2 3 4 5 6 7) · Semana 4 (1 2 3 4 5 6 7) |
| Esquema de proyecto de tesis | ■ ■ ■ ■ ■ ■ · ■ ■ ■ ■ ■ ■ · · · · · · · · · · · · · · |
| Fase de Inicio | |
| Delimitación del software | · · · · · · · · · · · · · · · ■ ■ ■ ■ ■ ■ · · · · · · · |
| Determinación de roles del proyecto | · · · · · · · · · · · · · · · · · · · · · · ■ ■ · · · · |
| Definición del producto terminado | |
| Recopilación de historias de Usuario | |
| Historias de usuarios – Frontend | · · · · · · · · · · · · · · · · · · · · · · · · ■ ■ ■ ■ |
| | |
| | Mes 3 |
| | Semana 1 (1 2 3 4 5 6 7) · Semana 2 (1 2 3 4 5 6 7) · Semana 3 (1 2 3 4 5 6 7) · Semana 4 (1 2 3 4 5 6 7) |
| Recopilación de historias de Usuario | · · · · · · · · · · · · · · · · · · · · · · · · · · · · |
| Historias de usuarios – Frontend | ■ ■ ■ ■ ■ ■ · · · · · · · · · · · · · · · · · · · · · · |
| Historias de usuarios – Backend | · ■ ■ ■ ■ ■ · · · · · · · · · · · · · · · · · · · · · · |

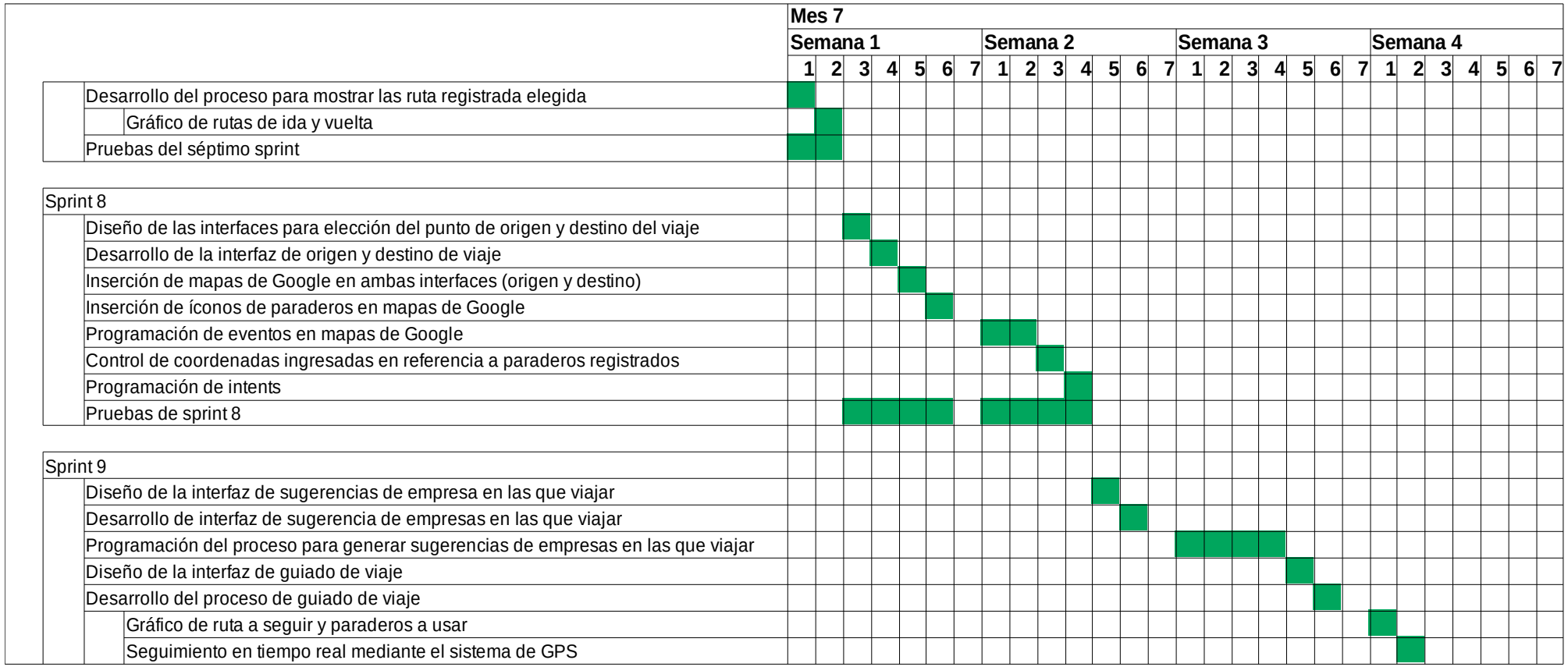












| | | Mes 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Semana 1 | | | | | | | Semana 2 | | | | | | | Semana 3 | | | | | | | Semana 4 | | | | | | | |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Fase de transición | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Preparación de la versión beta a partir de la versión con capacidad operativa inicial | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Instalación y muestra de la versión beta | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Recolección y toma de decisiones a partir de la información procedente de los usuarios | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Implementación de nuevas necesidades | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Fuente propia.

## 4.4 Fase de Construcción

### 4.4.1 Sprints Backend

Al utilizar PUDS acompañado de SCRUM, se realizan Sprints para dividir el trabajo en pequeños objetivos. Para ello se utilizan artefactos como la “Lista de Producto” y la “Lista de pendientes del Sprint”, y como en el presente proyecto se tiene dos grandes grupos de requerimientos (backend y frontend), se ha comenzado por el backend ya que es imprescindible para mantener la base de datos

Los requerimientos son la fuente de los casos de uso, y en el presente proyecto se realizó el diagrama de casos de uso funcionales tanto para el backend (figura 14) como el frontend; además del diagrama de casos de uso extendido (figura 15). A continuación se muestran ambos diagramas.

#### 4.4.1.1 Primer Sprint

La Lista de pendientes para el primer sprint se detallan en la tabla 12:

Tabla 12

*Lista de requerimientos del primer sprint*

| **Requerimiento** |
|---|
| Investigación sobre uso de los mapas de Google. |
| Extraer coordenadas de puntos elegidos en el mapa. |
| Probar el gráfico de rutas. |
| Diseño del diagrama entidad relación |

Fuente propia

Habiendo listado los requerimientos extraídos de la Lista de Producto, el primer sprint tiene una peculiaridad, y es que bastante de lo que se hizo es tomar decisiones y realizar investigaciones y pruebas, por lo tanto, el trabajo realizado fue mayormente práctico. A continuación detallaremos el desarrollo de cada uno de los requerimientos.

#### 4.4.1.1.1 Investigación sobre el uso de los mapas de Google

Actualmente, para hacer uso de los mapas de Google, primero se debe generar una API KEY (Clave de API), para ello se debe ingresar a “Google Api Console”, que se ubica en la url: https://console.developers.google.com/apis/), una vez dentro se debe elegir para qué se necesita la clave, en este caso particular, es para uso de mapas de Google en web haciendo uso de Javascript.

Una vez creada la clave, ya es posible hacer uso de los mapas Google, sin embargo se debe saber que el uso de los mapas es gratuito solo para Android, y para web tiene ciertas restricciones. Además el servicio web usado para graficar las rutas llamado “Google Maps Directions API) también tiene restricciones de uso.

Los mapas de Google también integran distintas vistas en la versión web y Android, las más comunes son la de satélite y relieve, y en el backend el uso de estas dos vistas ha sido muy útil especialmente para ingresar paraderos al sistema.

Otra de las bondades que brindan los mapas de Google para web es poder visualizar las calles desde la vista de la calle (en inglés Street views), de esa manera se puede visualizar las vistas que tomó el auto de Google cuando estuvo en Cusco, esta vista resultó especialmente útil para ingresar paraderos al sistema, ya que permite visualizar el lugar exacto donde estos están ubicados.

La figura 16 explica las tarifas estándar que tiene Google para el uso de sus distintas herramientas y servicios (Tarifas y planes)

Google Maps API | Página principal | Documentación | Tarifas y planes

| Android | ESTÁNDAR |
|---|---|
| Google Maps Android API | Uso gratuito ilimitado.[1] |
| Google Places API for Android | 1000 solicitudes gratuitas predeterminadas por día, que aumentan a 150 000 solicitudes gratuitas por día después de la validación de la tarjeta de crédito. Aumentos gratuitos para las aplicaciones que cumplen con los requisitos. |
| **iOS** | **ESTÁNDAR** |
| Google Maps SDK for iOS | Uso gratuito ilimitado.[1] |
| Google Places API for iOS | 1000 solicitudes gratuitas predeterminadas por día, que aumentan a 150 000 solicitudes gratuitas por día después de la validación de la tarjeta de crédito. Aumentos gratuitos para las aplicaciones que cumplen con los requisitos. |
| **Web** | **ESTÁNDAR** |
| Google Maps JavaScript API<br>Google Static Maps API<br>Google Street View Image API | Gratis hasta 25 000 cargas de mapa por día.[3]<br>USD 0,50 cada 1000 cargas de mapa adicionales, hasta 100 000 diarias, si está habilitada la facturación. |
| Google Maps Embed API | Uso gratuito ilimitado. |
| **Servicios web** | **ESTÁNDAR** |
| Google Maps Directions API<br>Google Maps Distance Matrix API[4]<br>Google Maps Elevation API<br>Google Maps Geocoding API<br>Google Maps Geolocation API<br>Google Maps Roads API<br>Google Maps Time Zone API | Gratis hasta 2500 solicitudes por día.<br>USD 0,50 cada 1000 solicitudes adicionales, hasta 100 000 diarias, si está habilitada la facturación. |

*Figura 16.* Tarifas y planes de uso de Google Maps API. Fuente: Tarifas y Planes

Una vez obtenida la clave, para hacer uso de los mapas hay que llamar a una librería con el siguiente script:

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?
key=claveDeApi&callback=initMap">
</script>
```

Hay que tener en cuenta que “initMap” es el nombre de la función que inicializa el mapa, y “claveDeApi” es la clave que se obtiene tras solicitarla en la “Google API Console”.

#### 4.4.1.1.2 Extraer coordenadas de puntos elegidos en el mapa

La API de los mapas Google permiten agregar marcadores en los mapas; y “Listeners” tanto al mapa como a los marcadores para cumplir las tareas que se requieren cuando sucedan los eventos programados. Para extraer coordenadas es necesario agregar el evento “Click” al mapa, esto se logra mediante las líneas de código mostradas en la figura 17.

```
function initMap() {

  //The center location of our map.
  var centerOfMap = new google.maps.LatLng(-13.525702984455705, -71.96637153625488);

  //Map options.
  var options = {
    center: centerOfMap, //Set center.
    zoom: 16, //The zoom value.
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };

  //Create the map object.
  map = new google.maps.Map(document.getElementById('map'), options);

  //Listen for any clicks on the map.
  google.maps.event.addListener(map, 'click', function(event) {
    //Get the location that the user clicked.
    var clickedLocation = event.latLng;
    //If the marker hasn't been added.
    if(marker === false){
      //Create the marker.
      marker = new google.maps.Marker({
        position: clickedLocation,
        map: map,
        icon: markerIcon,
        draggable: true //make it draggable
      });
      //Listen for drag events!
      google.maps.event.addListener(marker, 'dragend', function(event){
        markerLocation();
      });
    } else{
      //Marker has already been added, so just change its location.
      marker.setPosition(clickedLocation);
    }
    //Get the marker's location.
    markerLocation();
  });
}

//This function will get the marker's current location and then add the lat/long
//values to our textfields so that we can save the location.
function markerLocation(){
  //Get location.
  var currentLocation = marker.getPosition();
  //Add lat and lng values to a field that we can save.
  document.getElementById('lat').value = currentLocation.lat(); //latitude
  document.getElementById('lng').value = currentLocation.lng(); //longitude
}
```

*Figura 17.* Captura de líneas de código usadas para extraer coordenadas de los marcadores.

#### 4.4.1.1.3 Probar Gráfico de Rutas

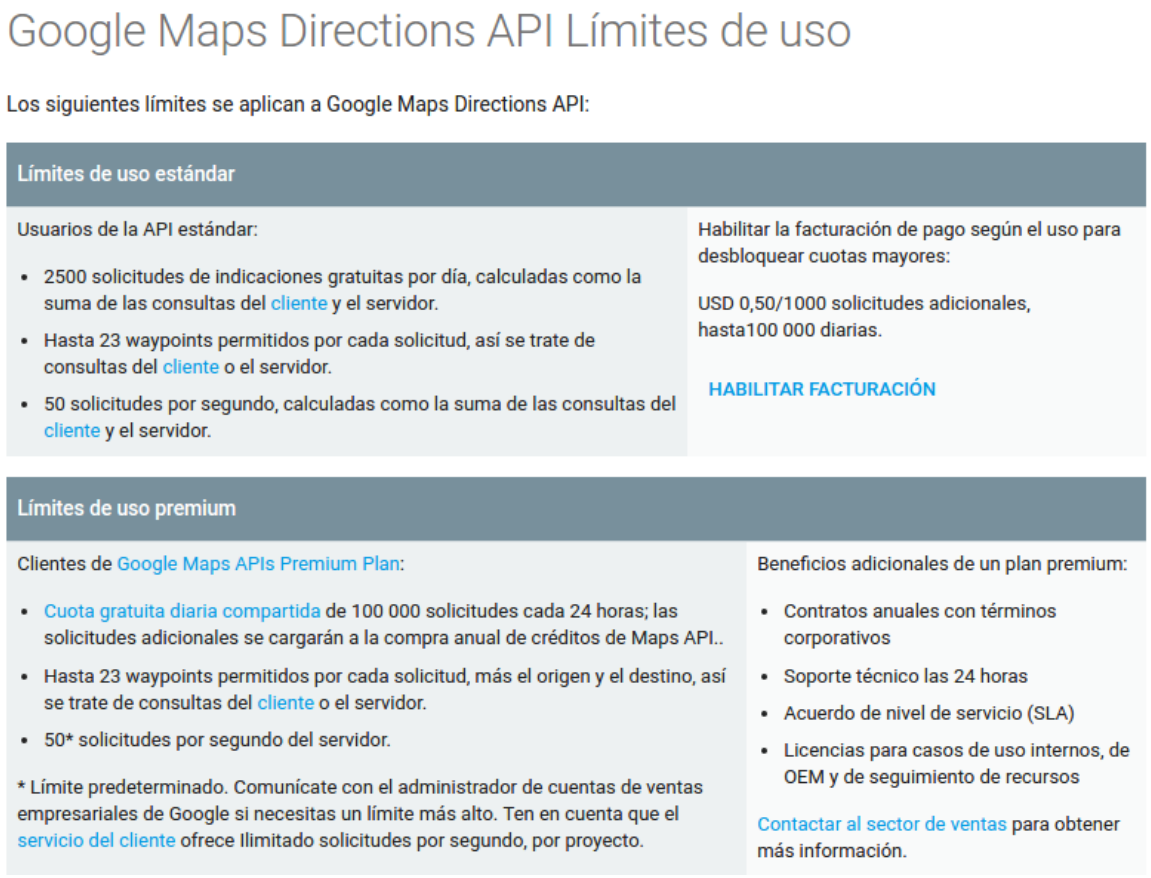
El gráfico de rutas es obligatorio para el backend; debido a que probar que las rutas ingresadas son correctas sería complicado de no tener una herramienta gráfoca para ello. Para graficar las rutas hay que hacer uso de el servicio web encargado de generar “polylines”, que son básicamente las líneas que indican la ruta a seguir dentro del mapa.

Para graficar rutas se debe hacer un llamado al servicio web “Google Maps Directions API”, e indicarle (de ser necesario) los puntos por donde debe pasar, además, se debe especificar cómo se viajará; a pie, en auto o en bicicleta. Para graficar hay que hacer uso de un código similar al mostrado en la figura 18.

```
function initMap() {
 //var directionsDisplay = new google.maps.DirectionsRenderer;  PARA QUE TENGA VALORES POR DEFECTO
 var directionsDisplay = new google.maps.DirectionsRenderer(
        {
          polylineOptions: {
              strokeColor: "#8b0013",
              strokeOpacity:0.5,
              strokeWeight:10
          }
      });
 var directionsService = new google.maps.DirectionsService;

 var plazaArmas = {lat:-13.517190913745901,lng:-71.97845220565796};
 var mantas = {lat:-13.517874185548116,lng:-71.97926759719849};
 var espaderos = {lat:-13.516804942549093,lng:-71.98007225990295};
 var agua = {lat:-13.51799936490261,lng:-71.98199272155762};

 var map = new google.maps.Map(document.getElementById('map'), {
   zoom: 16,
   center: {lat:-13.525702984455705, lng:-71.96637153625488}
 });



 directionsDisplay.setMap(map);

 calculateAndDisplayRoute(directionsService, directionsDisplay);

   function calculateAndDisplayRoute(directionsService, directionsDisplay) {
      //var start = document.getElementById('start').value;
      //var end = document.getElementById('end').value;
      var start = plazaArmas;
      var end = agua;
      directionsService.route({
        origin: start,
        waypoints:[
            {
                location:mantas,
                stopover:false
            }
        ],
        destination: end,
        travelMode: google.maps.TravelMode.DRIVING
      }, function(response, status) {
        if (status === google.maps.DirectionsStatus.OK) {
          directionsDisplay.setDirections(response);
        } else {
          window.alert('Directions request failed due to ' + status);
        }
      });


    }
}|
```

*Figura 18.* Captura de líneas de código de ejemplo para graficar rutas.

Hay un detalle en cuanto al gráfico de rutas en los mapas Google, la API de direcciones, solo es capaz de graficar un polilyne que contenga hasta 23 puntos (ver figura 19) incluyendo el punto inicial y final, por lo tanto es una restricción que hay que tener en cuenta al momento de ingresar esta data a la base de datos. En caso se ingresen más de 23 puntos, se debe graficar otro “polyline” para continuar con el anterior. (Google Maps Directions API)



## Google Maps Directions API Límites de uso

Los siguientes límites se aplican a Google Maps Directions API:

**Límites de uso estándar**

Usuarios de la API estándar:

- 2500 solicitudes de indicaciones gratuitas por día, calculadas como la suma de las consultas del cliente y el servidor.
- Hasta 23 waypoints permitidos por cada solicitud, así se trate de consultas del cliente o el servidor.
- 50 solicitudes por segundo, calculadas como la suma de las consultas del cliente y el servidor.

Habilitar la facturación de pago según el uso para desbloquear cuotas mayores:

USD 0,50/1000 solicitudes adicionales, hasta100 000 diarias.

HABILITAR FACTURACIÓN

**Límites de uso premium**

Clientes de Google Maps APIs Premium Plan:

- Cuota gratuita diaria compartida de 100 000 solicitudes cada 24 horas; las solicitudes adicionales se cargarán a la compra anual de créditos de Maps API..
- Hasta 23 waypoints permitidos por cada solicitud, más el origen y el destino, así se trate de consultas del cliente o el servidor.
- 50* solicitudes por segundo del servidor.

* Límite predeterminado. Comunícate con el administrador de cuentas de ventas empresariales de Google si necesitas un límite más alto. Ten en cuenta que el servicio del cliente ofrece Ilimitado solicitudes por segundo, por proyecto.

Beneficios adicionales de un plan premium:

- Contratos anuales con términos corporativos
- Soporte técnico las 24 horas
- Acuerdo de nivel de servicio (SLA)
- Licencias para casos de uso internos, de OEM y de seguimiento de recursos

Contactar al sector de ventas para obtener más información.

*Figura 19.* Límites de uso del API de Google. Fuente: Google Maps Directions API.

#### 4.4.1.1.4 Diseño del diagrama entidad relación

El comienzo de todo desarrollo es saber qué se quiere, luego; adentrarse en los requerimientos y detallarlos de la manera más específica posible, y una vez hecho esto, es muy práctico realizar un diagrama de clase o un diagrama entidad relación. Si bien no existe una regla que indique que es mejor empezar por un diagrama de clases o por un diagrama entidad relación, por experiencia propia es práctico primero mirar la estructura que tendría la base de datos. El diagrama entidad relación se visualiza en la figura 20.

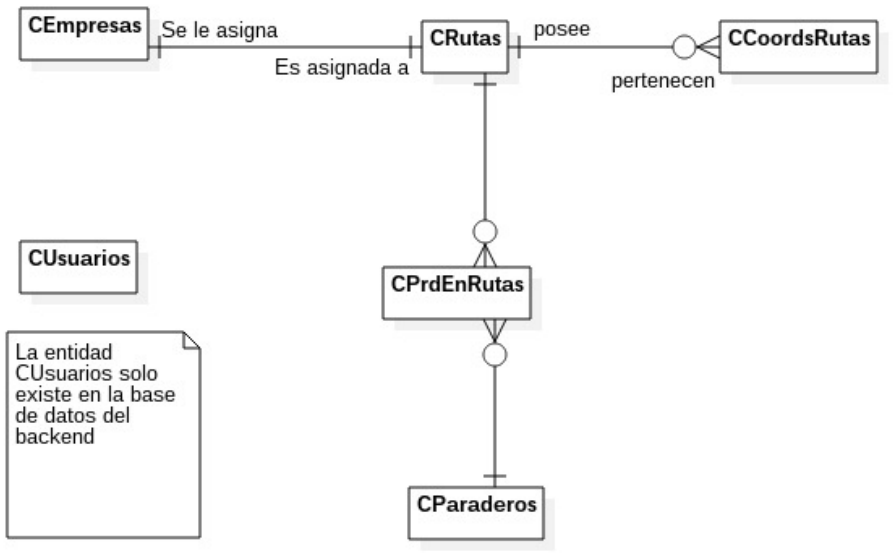



*Figura 20.* Diagrama Entidad Relación de la tesis

### 4.4.1.2 Segundo Sprint

En la tabla 13 se muestran los requisitos tomados de la lista de producto para el segundo sprint.

Tabla 13

*Lista de requerimientos del segundo sprint*

| **Requerimientos** |
|---|
| Codificación de las interfaces y las hojas de estilos. |
| Programación de las clases |
| Creación de la base de datos y sus usuarios |
| Creación de un Login para administrar el backend |

Fuente propia.

#### 4.4.1.2.1 Codificación de las interfaces y las hojas de estilos

La codificación de las distintas interfaces (vistas) del backend fue producto de varias iteraciones a lo largo del desarrollo del mismo, no todas las interfaces fueron codificadas en este punto, la mayoría llegaron a sus versiones finales a medida que se iba programando los requerimientos que hacían uso de las mismas, incluso se generó algunas vistas para probar líneas de código; estas fueron ubicadas dentro de la misma carpeta "vistas" para no tener problemas al requerir otros archivos y así probar prácticamente todo el código necesario para transportar el conocimiento obtenido a las otras interfaces. Las vistas generadas se aprecian en la figura 21.

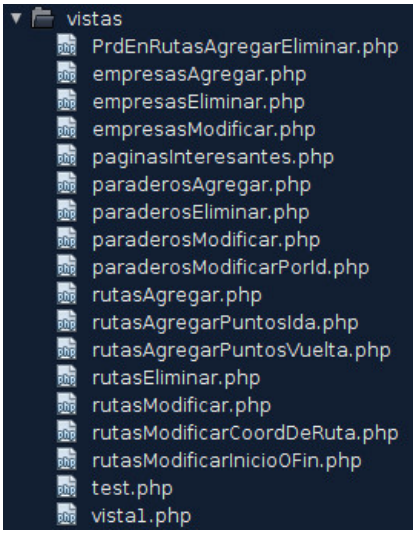



*Figura 21.* Listado de vistas creadas en el backend.

En cuanto al código de las vistas; casi todos son muy similares, sin embargo hay algunos fragmentos de código que se repite en todos, uno de ellos es el menú izquierdo, cuyo código se aprecia en la figura 22.

```
<?php
function contMenuIzq(){
    echo '
        <div class="headLeft">
            Bienvenido:<br/>
            USER<br /><br />
            <a style="float:right;color:#fff; text-decoration:none; font-size: 10px;"
                href="../Handlers/sessionDestroyHandler.php">Cerrar Sesión</a>
        </div>
        <br />
        <hr class="divMenuPrinc" />
        <nav class="mainNav">
            <b>Paraderos</b>
            <ul>
                <li><a class="mainItem1" href="paraderosAgregar.php">Agregar Paraderos</a></li>
                <li><a class="mainItem2" href="paraderosModificar.php">Modificar Paraderos</a></li>
                <li><a class="mainItem3" href="paraderosEliminar.php">Eliminar Paraderos</a></li>
            </ul>
            <br />
            <b>Rutas</b>
            <ul>
                <li><a class="mainItem4" href="rutasAgregar.php">Agregar Ruta</a></li>
                <li><a class="mainItem5" href="rutasAgregarPuntosIda.php">Agregar Puntos Ruta IDA</a></li>
                <li><a class="mainItem6" href="rutasAgregarPuntosVuelta.php">Agregar Puntos Ruta VUELTA</a></li>
                <li><a class="mainItem7" href="rutasModificar.php">Modificar rutas</a></li>
                <li><a class="mainItem12" href="PrdEnRutasAgregarEliminar.php">Agregar - Eliminar Paraderos a Ruta</a></li>
                <li><a class="mainItem8" href="rutasEliminar.php">Eliminar rutas</a></li>
            </ul>
            <b>Empresas</b>
            <ul>
                <li><a class="mainItem9" href="empresasAgregar.php">Agregar Empresa</a></li>
                <li><a class="mainItem10" href="empresasModificar.php">Modificar Empresa</a></li>
                <li><a class="mainItem11" href="empresasEliminar.php">Eliminar Empresa</a></li>
            </ul>
        </nav>
    ';
}

?>
```

*Figura 22.* Captura de pantalla del código usado para el menú izquierdo del backend.

En cuanto a la hoja de estilos, se debe precisar que al no usar un diseño “responsive” ya que no se necesita, esta es bastante corta. El único administrador del backend es el desarrollador del programa, y se administra siempre desde una laptop. Por otra parte, como se había definido el diseño en cuanto a colores y ubicación de los elementos, bastaba con generar los espacios necesarios para los distintos elementos y posteriormente realizar algunas modificaciones.

La hoja de estilos utiliza un pequeño “reset”, que básicamente es un conjunto de líneas de código casi estandarizados para empezar desde cero la elaboración de la misma. Esto permite no depender

de las configuraciones básicas de los navegadores, y así el código css generado se ve prácticamente igual en cualquier navegador.

También hay que tener en cuenta que la hoja de estilos está programada para ser compatible con los navegadores web Firefox y Google Chrome, y en este proyecto se usaron las versiones para Ubuntu; 55.0.2 (64 bits) y 57.0.2987.133 (64 bits) respectivamente.

#### 4.4.1.2.2 Programación de las clases

Las clases generadas surgen básicamente de las tablas generadas en el diagrama entidad-relación. En el siguiente diagrama de clases de la figura 23 podremos verlas más claramente.

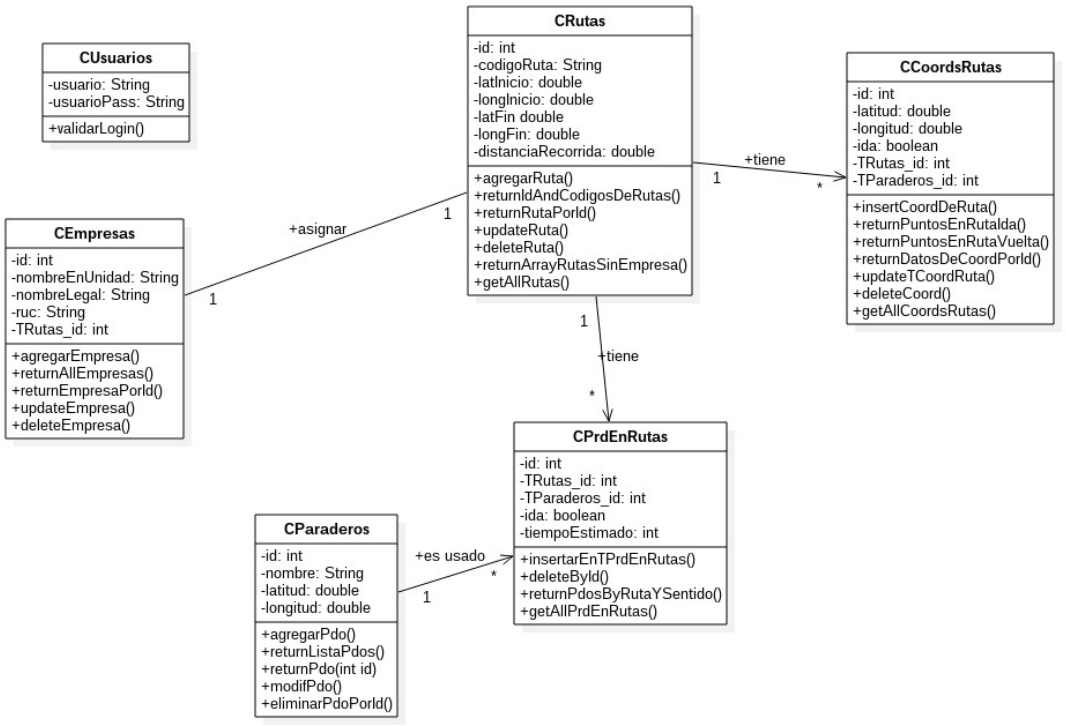



*Figura 23.* Diagrama de clases del backend.

#### 4.4.1.2.3 Creación de la base de datos y sus usuarios

El diagrama de entidad relación se creó usando el software "Workbench" para Ubuntu, y para crear la base de datos se siguió el siguiente proceso.

Primeramente haciendo uso del usuario por defecto "root", se creó una base de datos llamada "BD_Tesis01", la cual a medida que las iteraciones avanzaban; fue modificada y mejorada, siendo así que la última base de datos creada (la final) se llamó "BD_Tesis02".
Las tablas fueron creadas a partir del diagrama entidad relación de manera automática haciendo uso de la opción "Database - Forward engineer", la cual está disponible en el Workbench,
El usuario creado para el mantenimiento de la misma tanto por Workbench como por el backend fué: "adminUser", cuyos privilegios se aprecian en la figura 24.

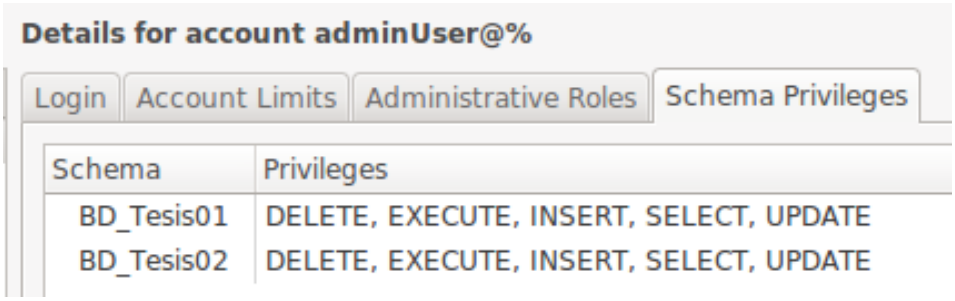



*Figura 24.* Privilegios que tiene el usuario "adminUser"

Se debe tener en cuenta que la complejidad del proyecto no está ligada a la complejidad de la base de datos; o a la complejidad de las consultas que se hacen sobre la misma, sino a la investigación llevada a cabo para primeramente; hacer uso de los mapas de Google tanto en web como Android, luego para aprovechar las tecnologías vigentes en cuanto a GPS y seguimiento, y finalmente poder programar una aplicación para el sistema operativo Android.

#### 4.4.1.2.4 Creación de un Login para administrar el backend

A partir de este punto en particular empieza la programación en PHP y el uso de las distintas clases generadas (ver 4.4.1.2.2). La programación de un login (control de acceso a un sistema) es la primera etapa de seguridad en casi cualquier sistema informático. El diseño del "login" se aprecia en la figura 25.

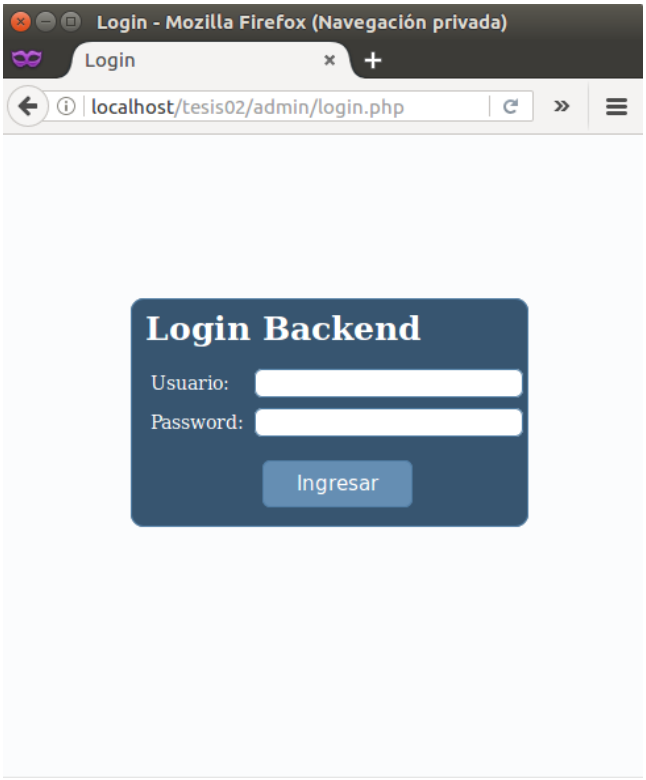



*Figura 25*. Diseño del Login.

La imagen fue capturada cambiando el tamaño de la ventana del navegador para poder mostrar tanto la ruta (url), como el diseño que el login tiene. El uso de la navegación privada fue por un tema de comodidad al momento de actualizar, ya que permite que al presionar F5 para actualizar; la página sea cargada casi desde cero, mostrando los cambios rápidamente, y cuando no lo hace, basta con apretar Ctrl+F5, el cual funciona bastante bien con o sin navegación privada.

Para encriptar la clave de acceso al sistema, se utilizó la función CRYPT de PHP, esta convierte una clave en un “Hash”, el cual básicamente es un conjunto de caracteres creados mediante un algoritmo a partir de una palabra inicial, en este caso; la clave. PHP recomienda el uso de esta función, por lo que es seguro guardar ese “Hash” incluso, como es en este caso, dentro de un “VARCHAR(100)”.

Inicialmente para ingresar la clave del usuario del backend, se ejecutó la función CRYPT; usando la clave como parámetro, y el resultado se ingreso de manera muy simple como se ve la siguiente línea: (Crypt)

```
INSERT INTO `TUsuarios` VALUES
('user','$2y$09$kl0cmPuu3SLisffQfo6UFeEQNpBBn3M1Msu64rObQYNyn.hB1JK
yO');
```

#### 4.4.1.3 Tercer Sprint

En la tabla 14 se visualizan los requerimientos desarrollados en el tercer sprint.

Tabla 14

*Lista de requerimientos del tercer sprint*

| **Requerimientos** |
|---|
| Ingresar y mantener data de paraderos en el sistema |
| Agregar paraderos |
| Diseño y programación de interfaz |
| Insertar ícono de paradero en mapa |
| Programar eventos a los íconos de paraderos |
| Agregar paradero en base de datos |
| Modificar paraderos |
| Diseño y programación de interfaz de elección y modificación. |
| Insertar íconos de paraderos en ambas interfaces |
| Programar eventos de íconos de paraderos |
| Modificar paradero en base de datos |
| Eliminar paraderos |
| Diseño y programación de interfaz |
| Insertar íconos de paraderos en mapa |
| Programar eventos de los íconos de paraderos. |
| Programar mensaje de confirmación |

- Eliminar paradero de base de datos
- Pruebas de mantenimiento de paraderos

Ingresar y mantener data de rutas en el sistema

- Agregar ruta
  - Diseño y programación de interfaz
  - Ingreso de data, y puntos iniciales y finales en la ruta
  - Programar eventos en marcadores de punto inicial y final
- Agregar puntos de ida y vuelta
  - Diseño y programación de interfaces
  - Programación ajax
  - Agregar coordenada y graficar ruta ingresada
- Modificar rutas
  - Diseño y programación de interfaces
  - Selección de ruta y sentido con ajax
  - Insertar íconos en puntos de ruta
  - Programar eventos en marcadores de puntos de rutas
  - Modificar punto de ruta
- Mantener paraderos que pertenecen a rutas
  - Diseño y programación de interfaces
  - Insertar paraderos e íconos de pertenencia en mapa
  - Programar eventos en paraderos
  - Agregar o quitar paradero en ruta
- Eliminar rutas o coordenadas de rutas
  - Diseño y programación de interfaces
  - Insertar íconos de ruta
  - Programar eventos en íconos
  - Eliminar ruta o coordenada de ruta
- Pruebas de mantenimiento de rutas

Ingresar y mantener data de empresas en el sistema

- Agregar Empresa
  - Diseño y programación de interfaz
  - Ingreso de data de empresa
  - Asignación de ruta
- Modificar Empresa
  - Diseño y programación de interfaz
  - Selección de empresa
  - Modificación de datos de empresa
  - Modificación de ruta concesionada de empresa
- Eliminar Empresa
  - Diseño y programación de interfaz
  - Eliminación de empresa
- Pruebas de mantenimiento de empresas

Fuente propia.

#### 4.4.1.3.1 Ingresar y mantener data de paraderos en el sistema

Este sprint fue el comienzo de la aplicación de todo lo aprendido e investigado previamente en cuanto al uso de los mapas de Google. Para dar mantenimiento los datos de los paraderos ingresados en la base de datos, hay que cumplir con las tres operaciones básicas de una base de datos, estas son:

- Insertar
- Actualizar
- Eliminar

Sin embargo el mantenimiento era solo uno de los requisitos de este sprint, el diseño de la interfaz es otro punto muy importante para que, intuitivamente, se puedan realizar las operaciones planeadas. Las interfaces generadas y procesos desarrollados fueron los siguientes:

**Agregar Paraderos**

Para agregar paraderos, basta con llenar los campos requeridos, luego; para ubicar el paradero; basta seguir con la instrucción de la interfaz. Los “listeners” programados son dos:

- El evento “click” en el mapa, para ubicar el paradero en el punto donde se dé click. En caso ya se haya dado click anteriormente y ya exista una ubicación elegida, al dar click en otro punto; el punto anterior desaparece, y reaparece en la nueva posición, actualizando a su vez los campos de latitud y longitud.
- Para el evento “dragend” sobre el ícono del paradero, para que al momento de terminar de arrastrarlo, los campos de latitud y longitud sean actualizados.

En la figura 26 se puede apreciar la interfaz generada.

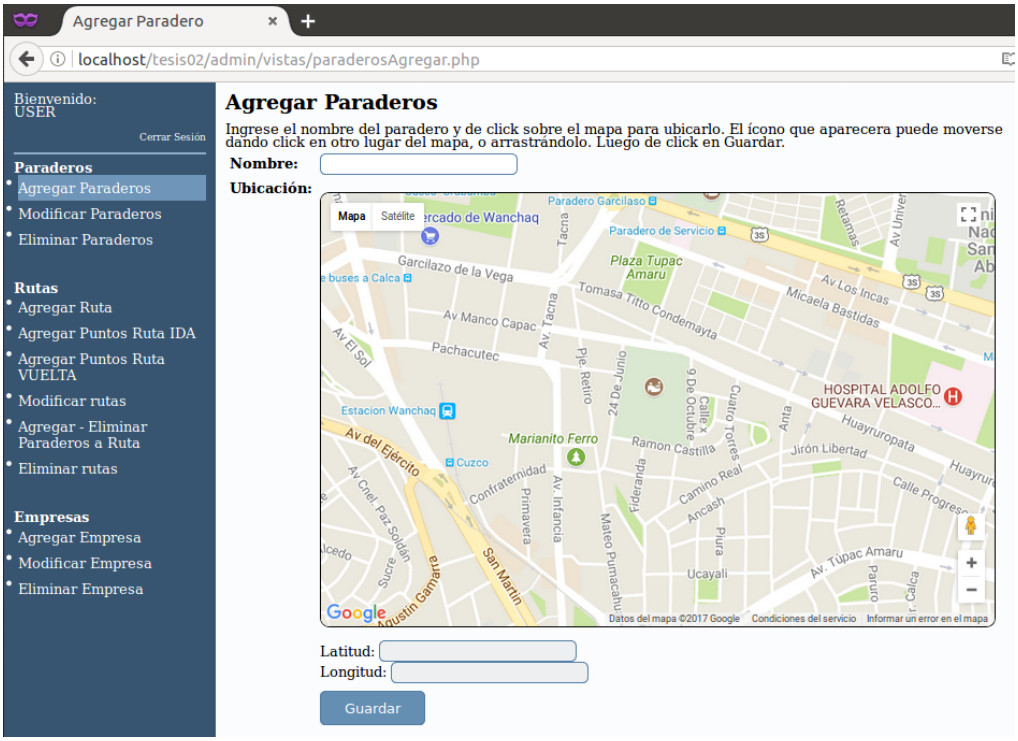



Figura 26. Interfaz para agregar paraderos.

### Modificar Paraderos

Para modificar el paradero, es necesario hacer uso de dos interfaces, la primera para elegir el paradero a modificar, y la segunda para modificar el nombre o las coordenadas del paradero elegido.

Todos los paraderos son cargados dentro de la primera interfaz, a cada uno se le agrega el “listener” del evento “click”, y al dar click sobre cualquiera de ellos se muestra el nombre del mismo; y un enlace sobre el cual; al dar click, redirige al usuario a la segunda interfaz.

La primera interfaz se puede apreciar en la figura 27.

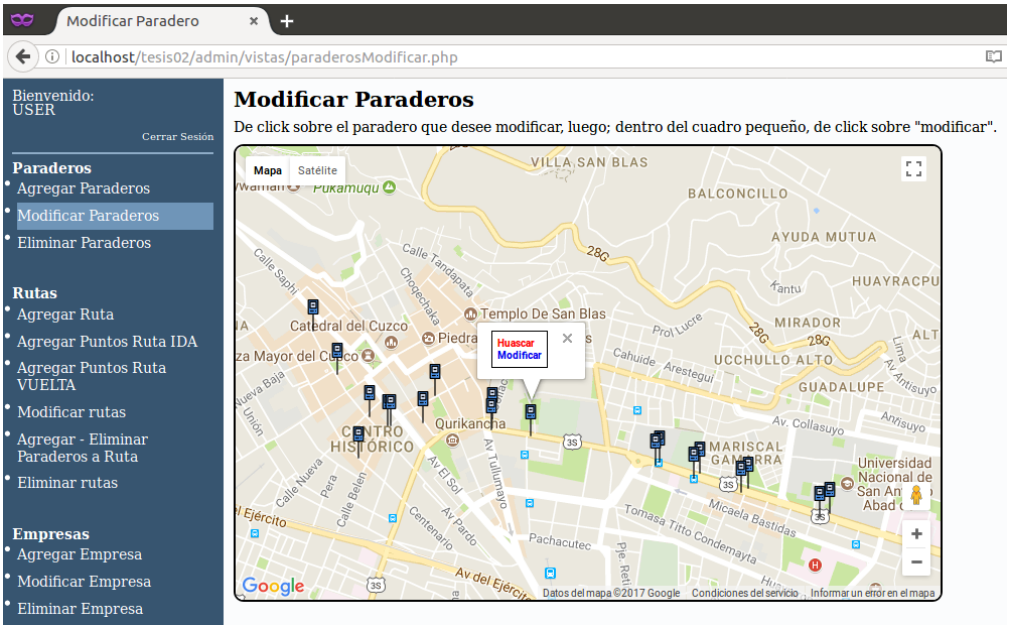



*Figura 27.* Interfaz donde se elige el paradero a modificar.

La segunda interfaz (figura 28) muestra el paradero elegido, y permite modificar su nombre y ubicación. Al ícono del paradero se le agrega un listener para el evento “dragend”, y al mapa se le agrega un listener para el evento “click”. Una vez concluida la modificación, se procede a dar click sobre el botón Modificar.

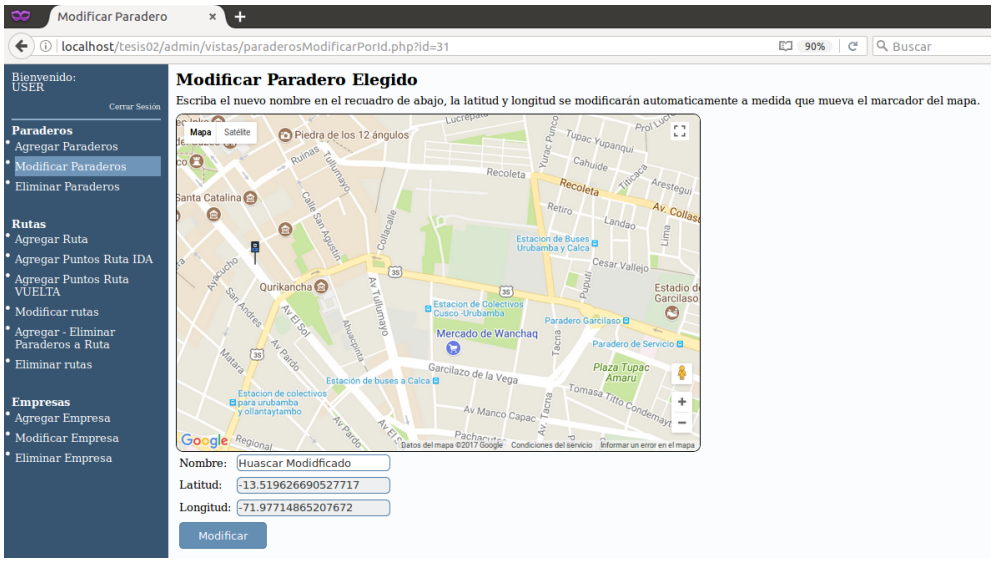



*Figura 28.* Interfaz donde se modifica el paradero elegido.

**Eliminar Paradero**

Para eliminar un paradero, basta con una interfaz, en ella se cargan todos los paraderos ingresados y se agregar sobre los mismos listeners para el evento “click”. Al dar click sobre cualquier de ellos, se mostrará su nombre y un enlace para borrar. Al dar click sobre el enlace “Eliminar”, aparece un cuadro para confirmar; como se muestra en la figura 29.

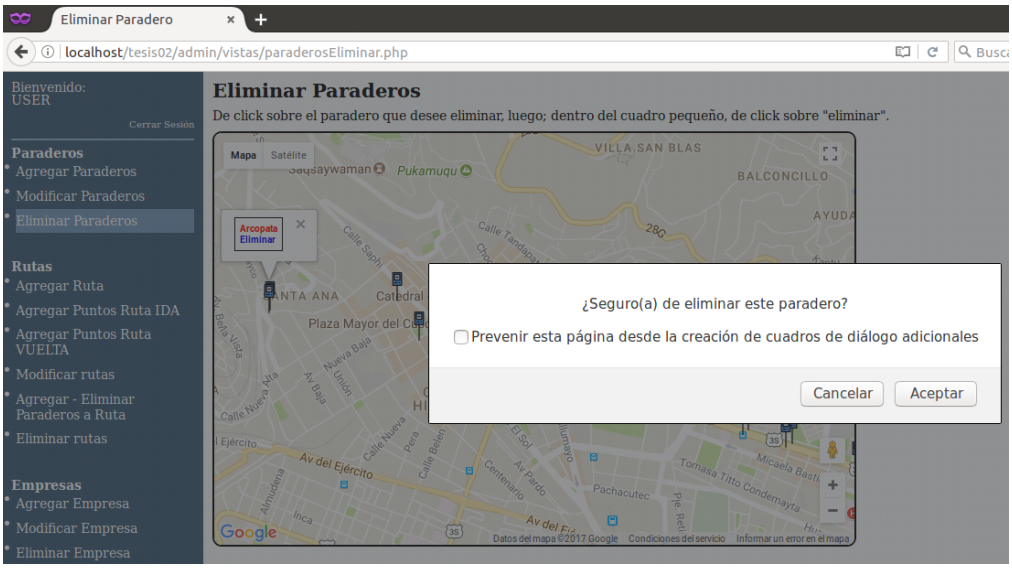



*Figura 29.* Interfaz para eliminar paradero.

**Pruebas de Mantenimiento de Paraderos**

Las pruebas de mantenimiento de paraderos se realizó de manera constante a medida que se iba programando, cada pequeño avance era probado, y solo así se seguía con el siguiente paso.

#### 4.4.1.3.2 Ingresar y mantener data de rutas en el sistema

Durante este sprint se desarrolló todo lo referente al mantenimiento de las rutas dentro del sistema. Se tuvo en consideración que las rutas no dependen de las empresas, y no son totalmente iguales en la ida o en la vuelta. Si bien a las empresas se les asigna una ruta, estas últimas no dependen de ser asignadas o no para poder ser ingresadas y mantenidas en el sistema, es más, deben ser

ingresadas antes para así poder ser asignadas a alguna empresa al momento de crear una.

Se tienen en cuenta las mismas 3 operaciones del punto 4.3.3.1. A continuación se muestran las interfaces generadas y se detalla el proceso seguido para cumplir con la lista de producto.

**Agregar ruta**

El proceso de agregar ruta es bastante corto, basta con ingresar el nombre de la ruta, los kilómetros aproximados que esta tiene y mover los marcadores previamente cargados en el mapa al punto inicial y final; como se aprecia en la figura 30. El marcador verde es el punto inicial y el rojo el punto final. Los "listeners" programados son:

- El evento "dragend" tanto en el marcador verde como en el rojo, una vez termina el arrastre de los mismos, se actualizan automáticamente los datos en los campos de las coordenadas.

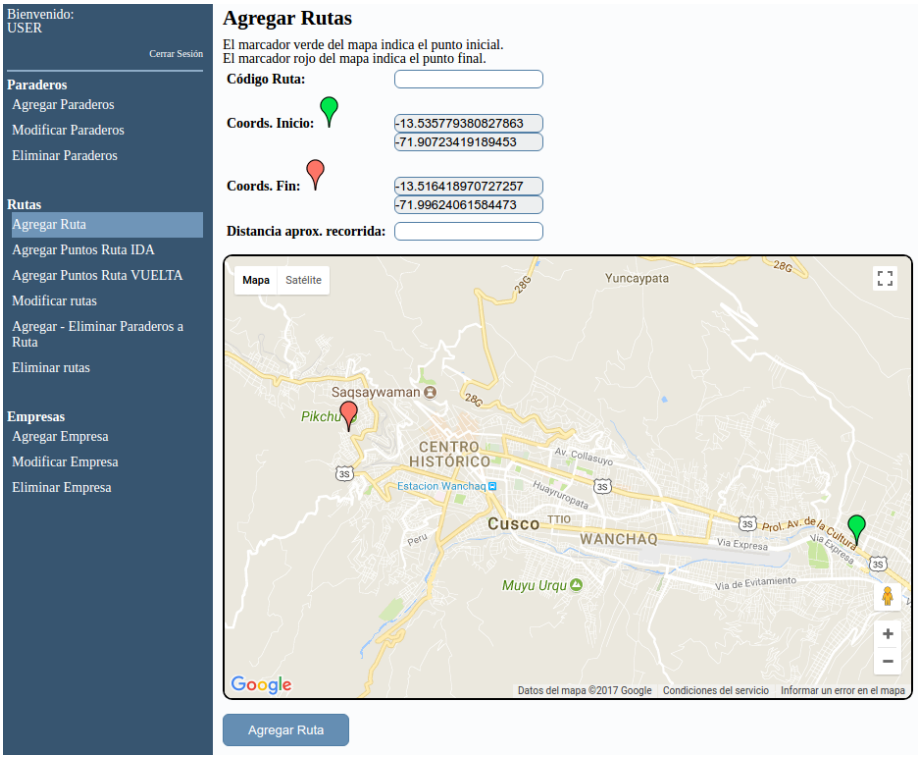



*Figura 30.* Interfaz para agregar rutas.

**Agregar coordenadas de ida y vuelta**

Para este requisito se hace uso de 4 interfaces, el proceso de agregar puntos de ida usa dos, lo mismo que el proceso de agregar puntos de vuelta. La primera y segunda interfaz de ambos procesos son prácticamente iguales, con la pequeña diferencia que dependiendo si se agrega puntos de ida o vuelta, los marcadores de punto inicial y final son coloreados de manera distinta. (figura 31)

La programación asíncrona (AJAX), entra en juego al momento de elegir la ruta en la cual se agregarán los puntos, al cambiar la selección de la ruta, los datos de distancia y ubicación se modifican sin necesidad de recargar la página.

Una vez elegida la ruta, se procede a agregar los puntos en la misma (figura 32), inicialmente se grafica automáticamente una ruta entre el punto inicial y final (al agregar puntos de ida, el punto inicial es el mismo punto inicial de la ruta, al agregar puntos de vuelta, el punto inicial es el punto final de la ruta). A medida que se van agregando los puntos, el gráfico de la ruta se va modificando, ya que esta pasa obligatoriamente por los puntos ingresados, y se agregan puntos (gráficamente se muestran pequeños círculos en dichos puntos) a medida que se requieren hasta tener el mismo trazo que en los documentos de rutas concesionadas usados como referencia. Para este proceso, se agregaron los siguientes “listeners” al mapa y marcador:

- Se agregó el evento “click” al mapa para que al momento de dar click sobre él, se muestre un marcador y se actualice automáticamente las coordenadas del mismo, si ya existía previamente, este actualiza su ubicación al lugar donde se dió el “click”.

- El evento “dragend” sobre el marcador, para poder arrastrarlo y que al terminar el arrastre, actualice sus coordenadas automáticamente.

A continuación se muestran las dos interfaces básicas diseñadas y programadas:

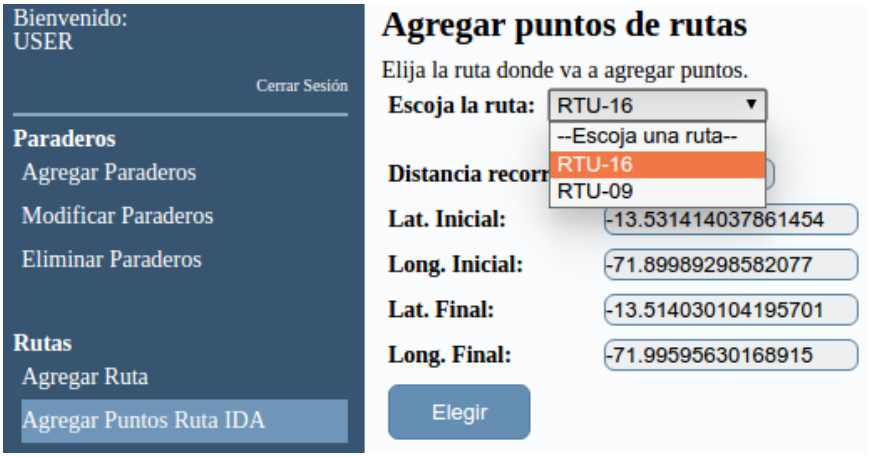



*Figura 31.* Interfaz donde se elige la ruta a la cual agregar puntos.

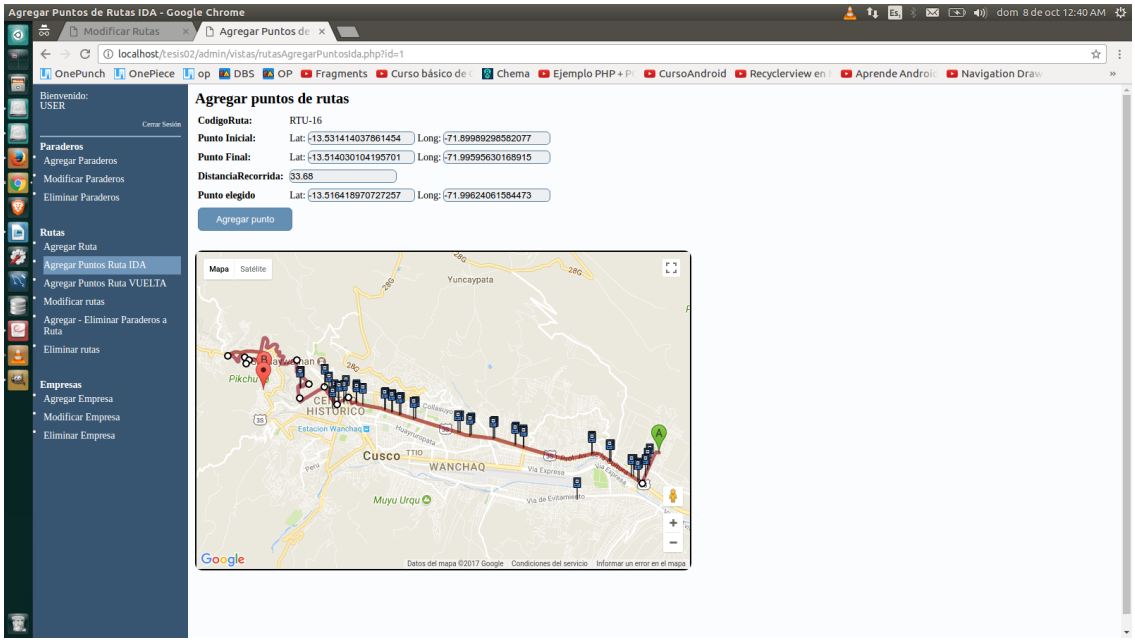



*Figura 32:* Interfaz donde se agregan coordenadas a la ruta y se grafica la ruta inmediatamente.

**Modificar rutas**

Para modificar rutas se desarrollaron cuatro interfaces, la primera (figura 33) para elegir la ruta y el sentido de la misma (si es de ida o vuelta), la segunda (figura 34) para elegir el punto a modificar (ya sea punto inicial o final, o punto de ruta), la tercera para modificar puntos de ruta y la cuarta para modificar puntos iniciales y finales.

La selección de ruta y sentido hace uso una vez más del AJAX, la interfaz es muy sencilla, y se actualiza dependiendo de la ruta que se ha elegido.

Una vez elegida la ruta y el sentido a modificar, se carga en el mapa el respectivo punto inicial y final de la ruta; y los puntos de la misma. A cada uno de ellos se les debe agregar listeners para el evento “click”, y dependiendo de si elegimos un punto de ruta, o alguno de los puntos inicial o final, se redirige la interfaz adecuada para modificar ese elemento.

Si se elige algún punto de la ruta, se redirige a una interfaz (figura 35) donde se puede modificar las coordendas del punto en cuestión, al mismo se le agrega el listener para el evento “dragend”, y al terminar de arrastrar el marcador a su nueva posición, se actualiza las coordenadas del punto después de pulsar “Modificar”.

Si se elige el punto inicial o final, el sistema redirige a una interfaz (figura 36) donde se puede modificar todos los datos de la ruta la ubicación de sus respectivos puntos inicial y final. Se agrega listeners para el evento “dragend” de ambos puntos, y para completar la modificación basta con dar click en Modificar.

A continuación se muestran las 4 interfaces desarrolladas


*Figura 33:* Interfaz donde se elige la ruta a modificar y el sentido de la misma.

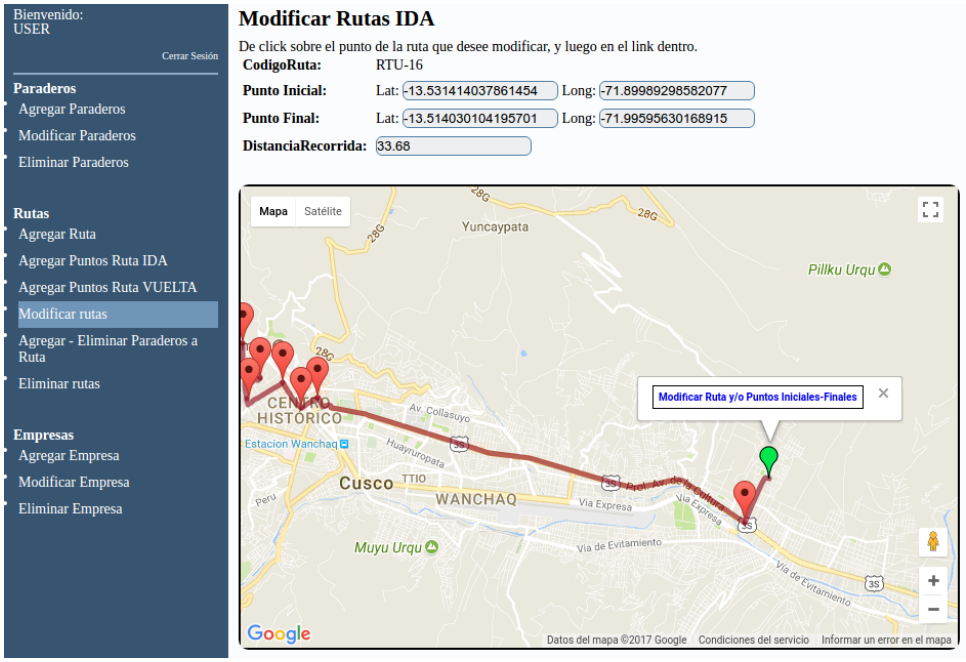



*Figura 34.* Interfaz donde se elije si se modificará las coordenadas inicial y final de la ruta o solo una coordenada de la misma.

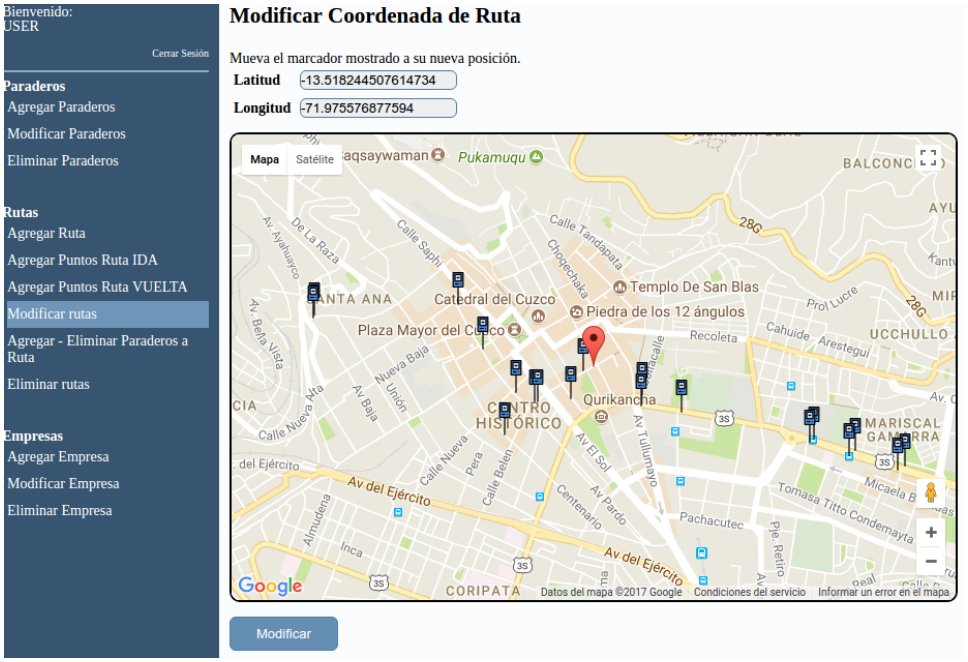



*Figura 35.* Interfaz donde se modifica una coordenada de ruta.

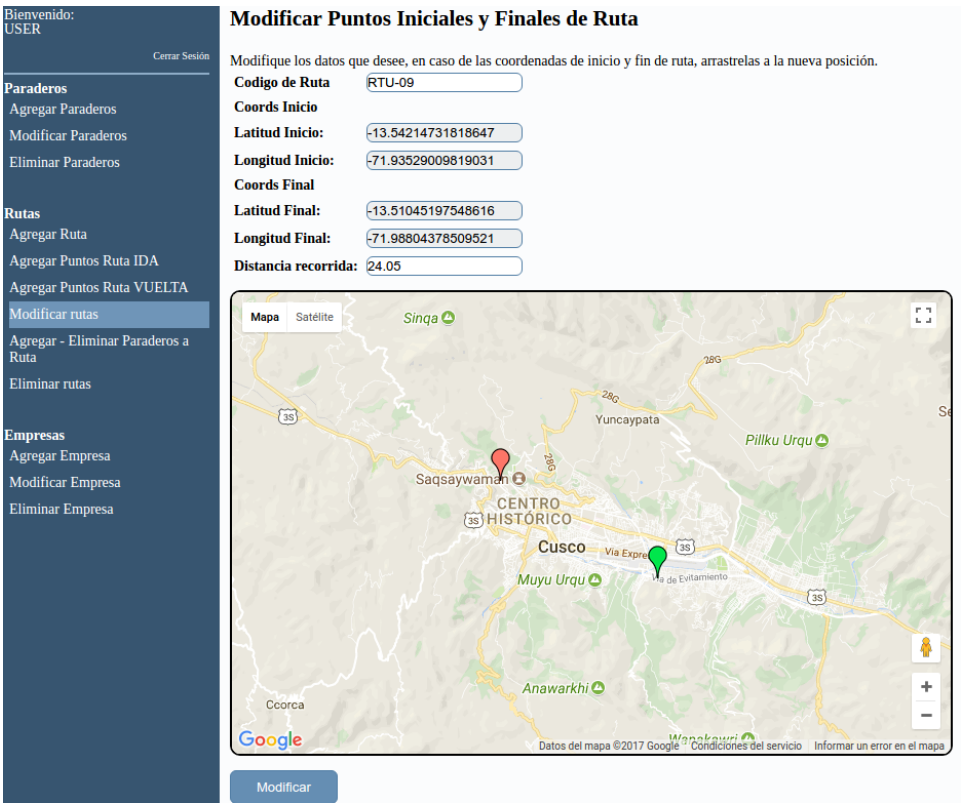



*Figura 36. Interfaz donde se modifica la posición del punto inicial y final de ruta.*

**Mantener paraderos que pertenecen a rutas**

Este proceso se encarga de indicar qué paraderos son usados por qué rutas, tanto a la ida como a la vuelta. Para ello; se desarrollaron dos interfaces. La primera (figura 37) para elegir la ruta y el sentido de la misma, y la segunda (figura 38) para indicar qué paraderos pertenecen a la ruta elegida. En la primera interfaz, al elegir la ruta los datos se actualizan mediante AJAX, y luego se elige el sentido de la misma.


*Figura 37.* Interfaz de elección de ruta y sentido para agregar o quitar paraderos de ruta.

En la segunda interfaz se realiza el proceso en sí, los paraderos que ya pertenecen a la ruta tienen dibujado en la base un cuadrado de color negro, y para agregar o quitar un paradero a una ruta, basta con dar “click” sobre el paradero, y luego dar “click” sobre el botón “Agregar/Quitar”. En esta interfaz se agregaron listeners para el evento “click” en cada uno de los paraderos. Es muy importante agregar los paraderos, ya sea de ida o vuelta, ordenadamente, empezando por el primero que se utiliza y terminando en el último, exactamente a como se ordenan al momento de viajar por medio de esa ruta desde el paradero inicial al final y viceversa.

Hay un detalle en la segunda interfaz (figura 38); que no debe ser dejado de lado, y es el valor de “Tiempo Estimado”. Esta dato es un entero que indica el tiempo que se demora desde el paradero anterior hasta el que se elige, y es usado para calcular el tiempo estimado de viaje. Por defecto este valor es 2, sin embargo puede ser modificado a voluntad.

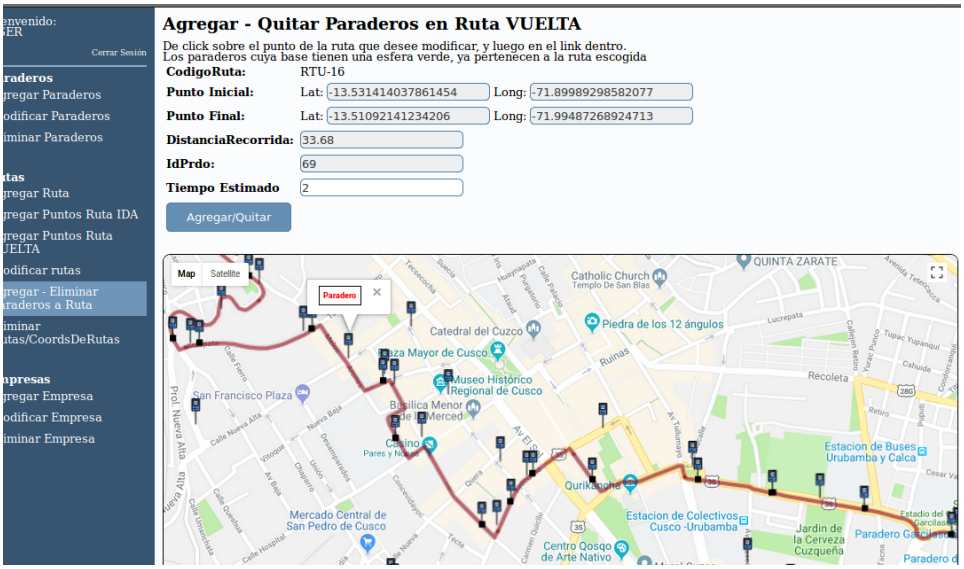



*Figura 38.* Interfaz donde se elige qué paraderos pertenecen o no a la ruta.

### Eliminar Rutas o Coordenadas de Rutas

Para el proceso de eliminación de rutas o coordenadas de rutas se utilizan dos interfaces. La primera sirve para elegir la ruta y el sentido de la misma, mientras la segunda para eliminar alguna coordenada o la ruta entera:

- En la primera interfaz (figura 40) se elige la ruta y el sentido. Al elegir la ruta se carga mediante AJAX los datos de la ruta y la opción de elegir el sentido. El sentido se elige para poder borrar alguna coordenada perteneciente a la ida o la vuelta (o la ruta de ida o vuelta entera), por lo tanto, si se elige “Ida”; en la segunda interfaz se cargan las coordenadas de ida de la ruta.

- En la segunda interfaz (figura 41) se puede borrar alguna coordenada de ida o vuelta de la ruta elegida (dependiendo de si se elige ida o vuelta), o la ruta entera. Para borrar alguna coordenada basta con dar click en el marcador de la coordenada y luego en el enlace que aparece “eliminar”.

  En caso se quiera borrar una ruta entera, se debe dar click en el punto inicial o final de la ruta, y luego en el enlace que aparece para eliminar. Se debe tener en cuenta que al borrar una ruta se borran las coordenadas de ida y vuelta, la lista de paraderos asignados a la ruta (no los paraderos ingresados al sistema), y finalmente la ruta misma.

Los listeners programados en los marcadores del mapa en la segunda interfaz son para el evento “click” en cualquiera de los puntos, y el proceso de eliminación de rutas se hace mediante una transacción controlada en PHP, a cual se aprecia en la figura 39.

```
function deleteRuta(){
      $comm = new CommonQueries();
      $mysqli = $comm->getMysqli();
      $mysqli->autocommit(false);
      $res=0;
      try{
         $queryCoord = "delete from TCoordsRutas where TRutas_id=".$this->id.";";
         $queryParaderos = "delete from TPrdEnRutas where TRutas_id=".$this->id.";";
         $queryRuta = "delete from TRutas where id=".$this->id.";";


         $resCord = $mysqli->query($queryCoord);
         $resPrd = $mysqli->query($queryParaderos);
         $resRuta = $mysqli->query($queryRuta);

         if($resCord && $resRuta && $resPrd){
            $mysqli->commit();
            $res = 1;
         }else{
            $mysqli->rollback();
            $res = 2;
         }

      } catch (Exception $ex) {
         $mysqli->rollback();
         $res = 3;
      }
      $mysqli->close();
      return $res;
   }
```

*Figura 39.* Captura de la transacción usada para borrar ruta.

Las dos interfaces básicas generadas son las siguientes:


*Figura 40.* Interfaz para elegir la ruta y sentido para eliminar ruta o coordenada de ruta.

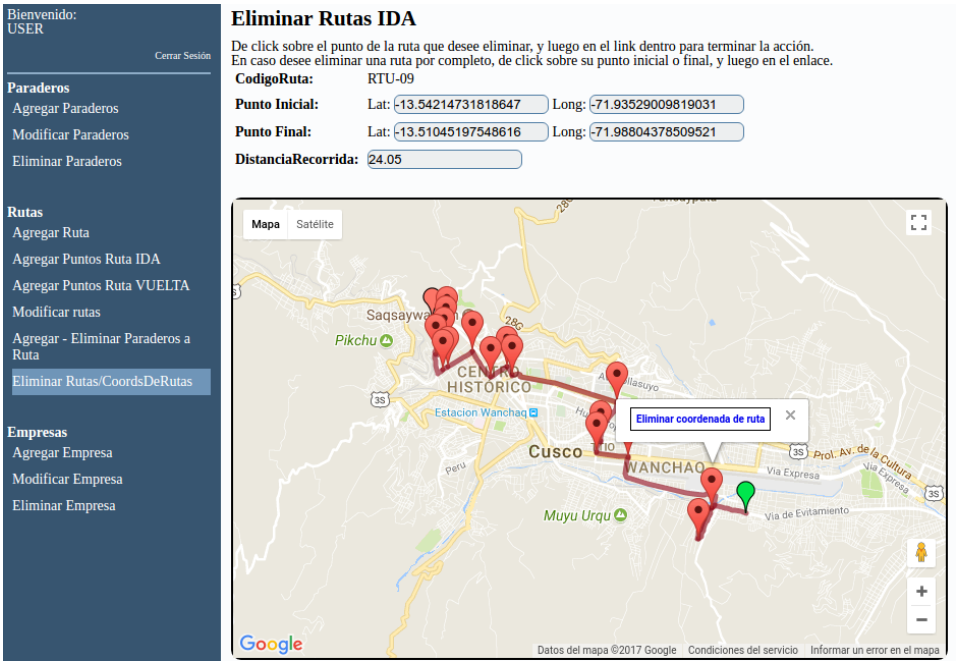



*Figura 41.* Interfaz donde se elige si borrar una coordenada o la ruta entera.

**Pruebas de mantenimiento de rutas**

Al igual que en el caso de los paraderos, las pruebas de mantenimiento de rutas se realizaron de manera constante mientras se iba desarrollando cada requerimiento, se ingresaron rutas de prueba y se probaron distintos casos posibles, cada pequeño avance fue probado.

#### 4.4.1.3.3 Ingresar y mantener data de empresas en el sistema

Durante este sprint se desarrolló todo lo relativo al mantenimiento de los datos de las empresas dentro del sistema. Se tuvo en cuenta que las empresas son independientes a las rutas, y que para crear una empresa, debe haber alguna ruta libre. Al igual que en los sprints anteriores, se tiene en cuenta para el mantenimiento las tres operaciones básicas de una base de datos. A continuación se detallan las interfaces generadas y el proceso seguido para el cumplimiento de los requisitos.

**Agregar empresa**

Para el proceso de agregado de empresas se desarrolló una interfaz sencilla pero a su vez importante. En ella se debe llenar los datos solicitados; y también se le debe asignar una ruta previamente ingresada al sistema. En caso no haya ninguna ruta libre (que no haya sido asignada a ninguna empresa), no se puede agregar una empresa. A continuación se muestra la interfaz cuando existe una ruta libre (figura 42), y cuando no existe ninguna (figura 43).


*Figura 42.* Interfaz cuando hay una ruta disponible.

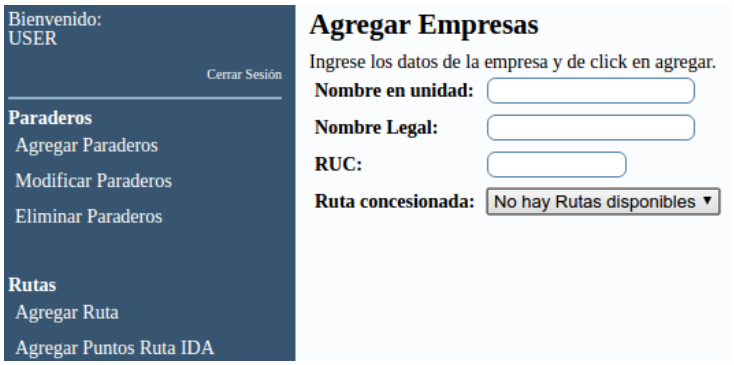



*Figura 43.* Interfaz cuando no hay rutas disponibles.

### Modificar Empresa

Para el proceso de modificación de los datos de una empresa se desarrollaron dos interfaces, la primera (fig. 44) para elegir la empresa a modificar, y la segunda (fig. 45 y 46) para modificar los datos de la misma. El proceso de selección de empresa es consta de una lista de empresas registradas y un botón para seleccionar, y el proceso de modificación permite modificar los datos propios de la

empresa, sin embargo, se debe tener en consideración lo siguiente; en caso no haya ninguna ruta disponible, se puede modificar los datos mas no se puede modificar la ruta, ya que no existiría ninguna libre. A continuación se muestran las imágenes de las interfaces generadas:

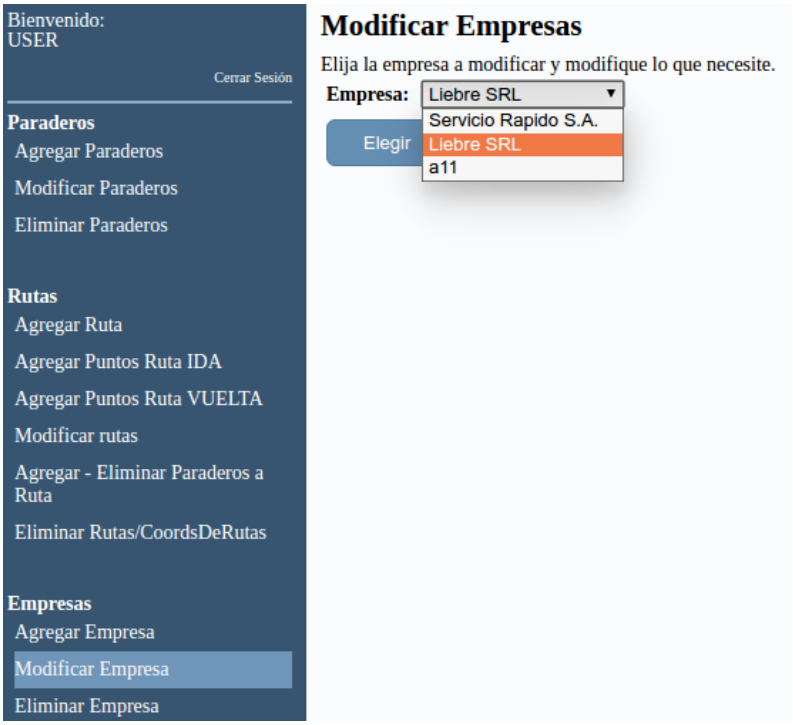



*Figura 44.* Interfaz de elección de empresa a modificar.


*Figura 45.* Interfaz de modificación de empresas donde se puede modificar todo menos la ruta asignada.

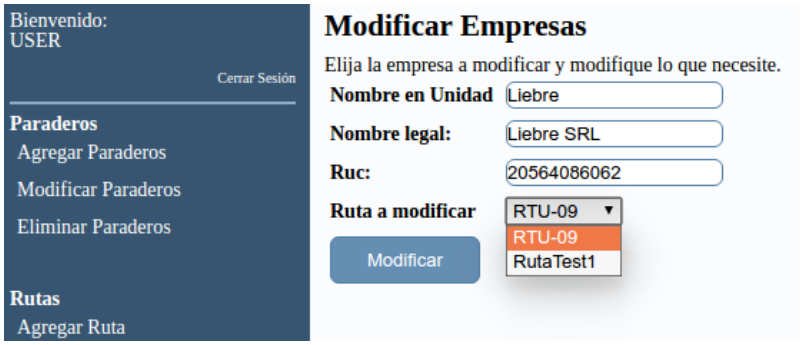



*Figura 46.* Interfaz de modificación de empresas donde se puede modificar todo; incluida la ruta asignada.

### Eliminar Empresa

El proceso de eliminación de empresa utiliza solamente una interfaz, en ella se elige la empresa a eliminar y se presiona sobre el boton de “Elegir”. A continuación se muestra la interfaz en la figura 47.


*Figura 47.* Interfaz de eliminación de empresas.

**Pruebas de mantenimiento de empresas**

Las pruebas de mantenimiento de empresas se realizaron de manera constante al igual que en los sprints anteriores, cada pequeño avance fue probado.

### 4.4.2 Sprints Frontend

Durante este sprint se realizaron investigaciones sobre distintos temas que surgieron como requerimientos para programar en Android. A continuación se explica el desarrollo de los distintos requerimientos tomados en este sprint.

#### 4.4.2.1 Cuarto Sprint

Durante el cuarto sprint se realizaron investigaciones sobre distintos temas que surgieron como requerimientos para programar en Android. Los requerimientos se listan en la tabla 15.

Tabla 15

*Lista de requerimientos del cuarto sprint*

| **Requerimientos** |
|---|
| Investigación sobre Desarrollo en Android |
| Investigación sobre el Funcionamiento de XML |
| Investigación de Diseño y Desarrollo de Interfaces en Android |
| Investigación de Mapas de Google para Android |
| Investigación de Gráfica de Rutas para mapas Google en Android |
| Investigación de ingreso de íconos y marcadores en mapas de Google para Android |
| Investigación de uso y seguimiento mediante GPS en Android |
| Investigación de Servicios Web |
| Investigación de Servicios Web en Android |
| Investigación sobre SQLite |

Fuente propia

#### 4.4.2.1.1 Investigación sobre Desarrollo en Android

Entender como funciona Android Studio y como se programa en el mismo para móviles con Android; ha sido eje central durante el desarrollo de la presente tesis. Durante la investigación se accedió a la información necesaria para poder entender cómo funciona Android Studio, cómo generar simuladores de dispositivos móviles, qué archivos es necesario descargar, y especialmente; a entender la mecánica de funcionamiento del mismo.

Por otro lado, se realizaron pequeños ejemplos de prueba para entender el manejo de botones, vista de textos, intents, mapas de Google, fragmentos, y aspectos generales del desarrollo para Android, estos ejemplos fueron ampliados y comprendidos a más profundida en los siguientes sprints y durante el desarrollo de la presente tesis.

#### 4.4.2.1.2 Investigación sobre el Funcionamiento de XML

Investigar sobre XML para desarrollar en Android es muy importante, principalmente para entender cómo se clasifica la información en este formato y cómo se programan las vistas, también se debe tener en cuenta que si bien no es nada complicado, es de suma importancia ya que se utiliza para escribir el “AndroidManifest”.

XML tiene un formato similar a HTML, ambos empiezan y terminan una etiqueta de manera muy similar, y con todo el conocimiento adquirido previamente; se programaron las interfaces sin utilizar la herramienta gráfica que brinda Android Studio. Cabe recalcar que XML no solo se usó en los puntos descritos anteriormente, sino también para exportar los datos de los servicios web.

#### 4.4.2.1.3 Investigación de Diseño y Desarrollo de Interfaces en Android

El diseño y desarrollo de una interfaz simple para Android es sencillo gracias a dos herramientas que brinda Android Studio. La primera mediante un conjunto de herramientas gráficas donde se arrastran los distintos elementos que se necesita a la ubicación deseada, y la segunda donde se escribe todo por código XML.

Durante la presente investigación se aprendió a utilizar los elementos necesarios para poder generar interfaces que cumplan con los requerimientos, así como la mecánica para poder utilizarla, y como ya se tenía cierto conocimiento gracias a lo investigado en el punto 4.4.1.1, se puso mayor énfasis en entender la manera de ubicar los elementos en la interfaz, para ello se dio especial interés en los “Linear Layout”.

#### 4.4.2.1.4 Investigación de mapas de Google para Android

Para poder utilizar los mapas Google en Android también se debe generar una clave, el proceso es similar al de la versión web de los mapas, con la diferencia que esta vez es para Android. La clave de los mapas se ingresa dentro de los archivos de manera distinta a la que se usa en web, en este caso no se utiliza una ruta (url), sino se ingresa dentro del archivo AndroidManifest.xml como se ve en la figura 48.


*Figura 48.* Captura de ingreso de código de API para uso de mapas de Google en Android.

Y para hacer uso del mismo, basta con ingresar un mapa de Google dentro de un fragmento de la siguiente manera (figura 49).

```
<fragment
    android:id="@+id/map_origen"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
```

*Figura 49.* Captura de fragmento para utilizar mapas de Google en Android.

Sobre la libertad de uso, el mapa en sí es de libre uso dentro de Android, sin embargo los distintos servicios usados para graficar por ejemplo las rutas no lo es(ver 4.3.1.2).

#### 4.4.2.1.5 Investigación de Gráfica de Rutas para Mapas Google en Android

Este proceso fue similar en algunos puntos al del backend, ambos utilizan un servicio web y una ruta (URL) para acceder al mismo. En el caso particular de Android, la ruta usada es como se indica en la figura 50.


*Figura 50. Formato para usar Google Maps Directions API.*

Como se aprecia, Google recomienda el uso de “json” como formato de salida, y la ruta debe seguir la estructura mostrada. En el caso de los parámetros, estos son básicamente el punto de

origen, el destino, los “waypoints”, que son los puntos del mapa por donde debe pasar la ruta, la clave de API que se debe generar de manera similar a los mapas en javascript, y finalmente el modo, que en este caso particular se utiliza “walking” (caminando), ya que permite graficar sin problemas aún cuando el sentido de ciertos tramos de la ruta no están correctamente añadidos en los mapas de Google.

Los gráficos de rutas en Android tienen el límite de 23 puntos incluyendo el punto inicial y final, y una vez recibida la información del servicio web, esta debe ser tratada y posteriormente ingresada al “polyline”; que es la línea en sí que se grafica en el mapa, sin embargo se pueden graficar más de un “polyline” en un mismo mapa.

Finalmente, para graficar rutas en los mapas Google para Android, es importante entender un poco las tareas asíncronas (en inglés; AsyncTasks). Si bien no entran de lleno dentro de la programación concurrente, son tareas que se ejecutan sin seguir el hilo principal de una aplicación. Esto quiere decir, que tienen sus propios hilos, permitiendo a la aplicación continuar con sus instrucciones a medida que estos se van procesando independientemente durante los pocos segundos que usualmente duran. En el caso de las gráficas de rutas, al usar un servicio web, la respuesta del servidor puede demorar cierto tiempo, y el resto de la aplicación no debe detenerse mientras llega la respuesta y esta es procesada, por lo tanto el uso de tareas asíncronas es necesario.

#### 4.4.2.1.6 Investigación de ingreso de íconos y marcadores en mapas de Google para Android

Incorporar íconos y marcadores dentro de un mapa Google para Android es bastante similar a cómo se hace en Javascript. Para ello se utilizaron marcadores para indicar los puntos iniciales y finales

de las rutas, y pequeños íconos de paraderos para indicar las posiciones de los paraderos.

Otro punto que se debe tener en cuenta, es que los íconos o recursos gráficos que se pueden usar, deben tener cierta calidad dependiendo del dispositivo al que vayan destinados. Es recomendable al momento de programar para Android, que se utilicen imágenes en distintas resoluciones siguiendo ciertas reglas al momento de agruparlas y darles nombres, esto para que cuando se instale la aplicación en un móvil con una pantalla de gran resolución, se utilicen mejores imágenes, y lo contrario en pantallas de menor resolución. De esta manera, tanto en pantallas de gran resolución como en aquellas de poca resolución, las imágenes se seguirán viendo bien.

Las imágenes dentro de una aplicación Android, deben ir dentro de la carpeta “drawable” (ver 2.1.27.6), sin embargo esto no queda solo allí. La carpeta “drawable” se subdivide en distintas resoluciones, si bien desde la vista de “Android” en Android Studio se puede ver todas las imágenes dentro de “drawable”, lo cierto es que cada una de ellas se almacena dentro de la carpeta que le corresponde dependiendo de su resolución como se puede ver en la figura 51.

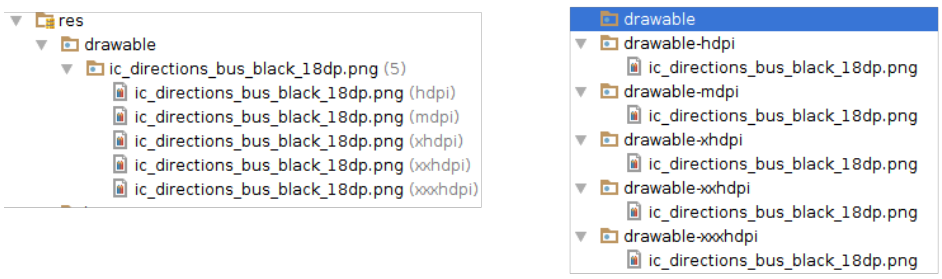



*Figura 51.* Forma en la que Android Studio almacena imágenes.

#### 4.4.2.1.7 Investigación de uso y seguimiento mediante GPS en Android

El seguimiento por GPS puede hacerse mediante diferentes formas, sin embargo los mapas de Google para Android ya cuentan con esta función de manera nativa. Activarlo depende de básicamente de los siguientes requisitos: Permiso de ubicación, GPS activado, y, opcionalmente dependiendo del dispositivo, tener el internet activado, ya que hay sistemas GPS en móviles que no necesitan de internet para funcionar, basta con tener el mapa previamente descargado (para que el usuario pueda ubicarse y no aparezca solo un fondo claro) y realizan el seguimiento sin necesidad de internet.

Para solicitar los permisos de acceso a internet y acceso a la ubicación; se deben de indicar en el archivo AndroidManifest.xml, sin embargo a partir de la versión 6.0 (Marshmallow), el permiso de ubicación debe solicitarse programáticamente, esto quiere decir, en tiempo de ejecución. (figura 52)

Para solicitar el permiso de ubicación, se programó el siguiente código, que por cierto es básicamente un estándar que propone Google.

```
public void verificarPermisoUbicacion1(){

    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {//API 23
        int permissionStatus = checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION);


        //int b = PackageManager.PERMISSION_GRANTED;

        if(permissionStatus != PackageManager.PERMISSION_GRANTED){
            requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION},100);
        }
    }
}
```

*Figura 52. Captura del código estándar para pedir el permiso de ubicación.*

Y en el caso el GPS del móvil no esté encendido, se utiliza la siguiente función (figura 53) para solicitar al usuario que active el GPS mediante un mensaje personalizado.

```
private void AlertNoGps() {
    final AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("Para poder usar la ubicación automática es necesario tener el GPS activado. ¿Desea activarlooo? ")
           .setCancelable(false)
           .setPositiveButton("Si", new DialogInterface.OnClickListener() {
               public void onClick(@SuppressWarnings("unused") final DialogInterface dialog, @SuppressWarnings("unused") final int id) {
                   startActivity(new Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
               }
           })
           .setNegativeButton("No", new DialogInterface.OnClickListener() {
               public void onClick(final DialogInterface dialog, @SuppressWarnings("unused") final int id) {
                   dialog.cancel();
               }
           });
    alert = builder.create();
    alert.show();
}
```

*Figura 53.* Captura de la función que solicita al usuario la activación del GPS.

También se debe indicar al mapa que active el botón de seguimiento y del compás (para poder apuntar fácilmente al norte cuando uno lo desee), para ello se debe colocar las siguientes líneas de código:

```
mapM.setMyLocationEnabled(true);
mapM.getUiSettings().setMyLocationButtonEnabled(true);
mapM.getUiSettings().setCompassEnabled(true);
```

Y finalmente, este al ser pulsado; ubicará al usuario dentro del mapa con un pequeño punto azul como se ve en la figura 54.

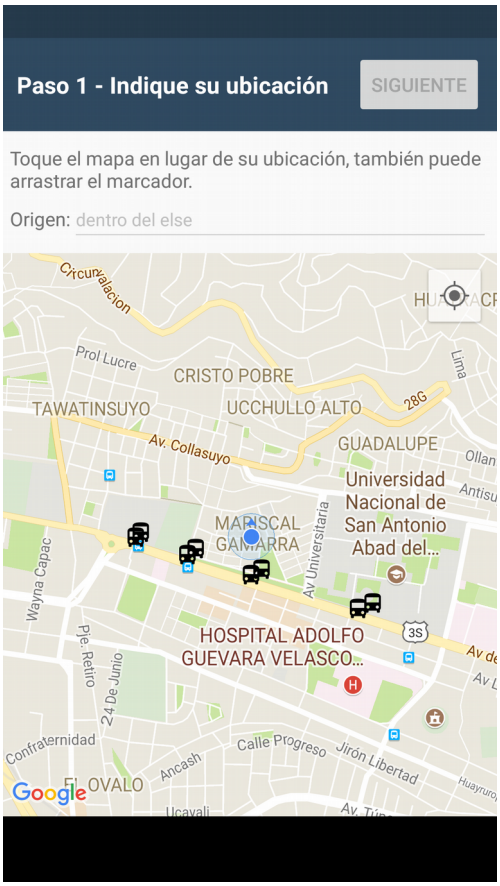



*Figura 54.* GPS activo y funcionando.

#### 4.4.2.1.8 Investigación de Servicios Web

El proceso de investigación de servicios web fue bastante sencillo gracias al conocimiento previo de XML. En el presente proyecto estos se utilizaron para poder enviar datos de la base de datos MySQL a la base de datos SQLite de los móviles.

El formato utilizado puede variar dependiendo de las necesidades que se pueda tener, sin embargo PHP implementa nativamente una función llamada “json_encode”; la cual exporta en formato JSON los datos que uno solicite, y posteriormente estos son consumidos por la aplicación de la manera que uno vea conveniente. (ver anexo 3)

En la siguiente imagen (figura 55) se puede ver un ejemplo de cómo se hizo la exportación de datos mediante un Servicio Web, y la forma de utilización de la función de PHP llamada “json_encode”:

```
<?php
require_once '../clases/CCordsRutas.php';

$cordRut = new CCordsRutas();
$res = $cordRut->getAllCoordsRutas();

if($res){
    $datos['estado'] = 1;
    $datos['data'] = $res;
    printf(json_encode($datos));
}else{
    $datos['estado'] = 0;
    printf(json_encode($datos));
}
```

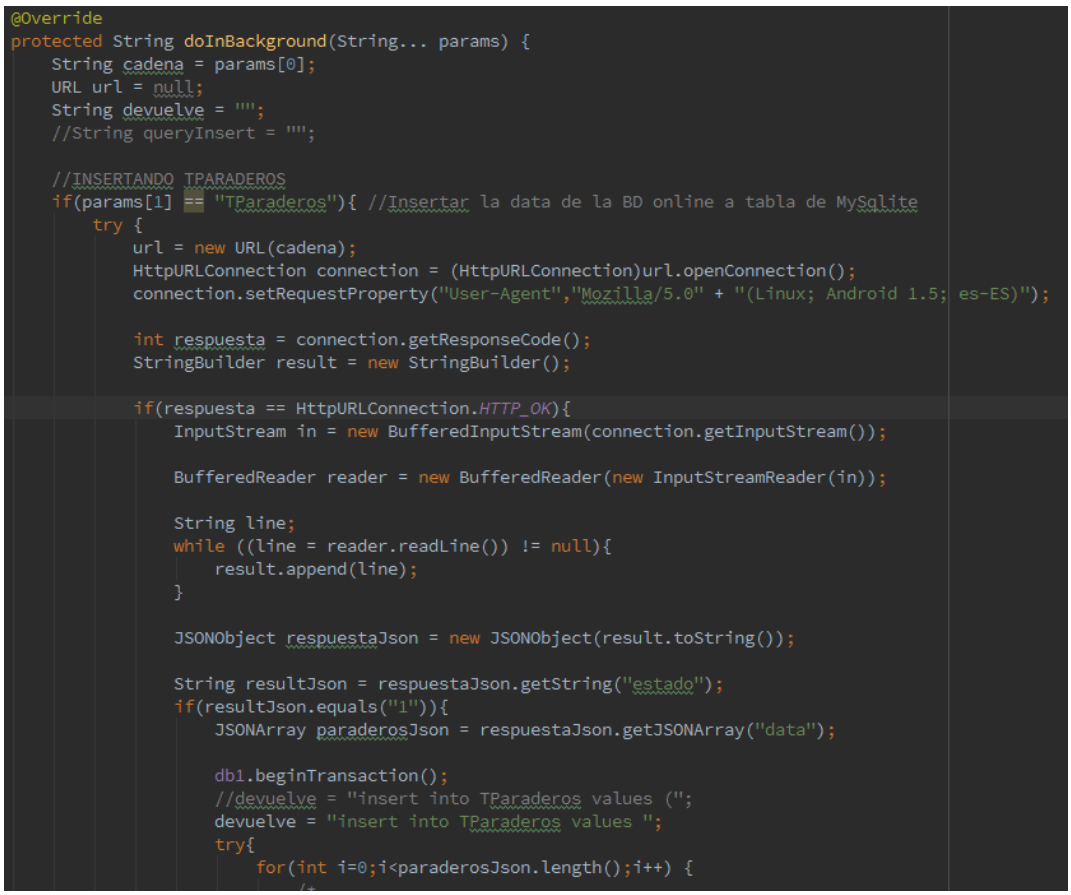
*Figura 55.* Captura de la función usada para generar el Servicio Web para la clase CCordsRutas.php

#### 4.4.2.1.9 Investigación de consumo de Servicios Web en Android

El consumo de servicios web se realiza mediante conexiones a servidores externos, y a la obtención de data que posteriormente se trata y de la que se obtiene lo que uno necesita. Consumir un Servicio Web es un proceso que depende de un elemento externo a la aplicación (un servidor), por lo tanto no se recomienda que vaya en el mismo hilo de la aplicación, sino utilizar una vez más; tareas asíncronas. (figura 56)

El formato utilizado para los Servicios Web fue JSON, ya que es un estándar ampliamente aceptado por la comunidad de desarrolladores, y además implementa funciones que facilitan la exportación e importación del mismo, y en caso de Java, existen funciones prefabricadas que permiten el consumo de datos en este formato.

Para consumir un servicio web en Android, se debe primero entender de tareas asíncronas, una vez hecho esto se procede a básicamente seguir el siguiente diagrama, aunque lógicamente el proceso especialmente en Java toma bastantes líneas de código.



```
@Override
protected String doInBackground(String... params) {
    String cadena = params[0];
    URL url = null;
    String devuelve = "";
    //String queryInsert = "";

    //INSERTANDO TPARADEROS
    if(params[1] == "TParaderos"){ //Insertar la data de la BD online a tabla de MySqlite
        try {
            url = new URL(cadena);
            HttpURLConnection connection = (HttpURLConnection)url.openConnection();
            connection.setRequestProperty("User-Agent","Mozilla/5.0" + "(Linux; Android 1.5; es-ES)");

            int respuesta = connection.getResponseCode();
            StringBuilder result = new StringBuilder();

            if(respuesta == HttpURLConnection.HTTP_OK){
                InputStream in = new BufferedInputStream(connection.getInputStream());

                BufferedReader reader = new BufferedReader(new InputStreamReader(in));

                String line;
                while ((line = reader.readLine()) != null){
                    result.append(line);
                }

                JSONObject respuestaJson = new JSONObject(result.toString());

                String resultJson = respuestaJson.getString("estado");
                if(resultJson.equals("1")){
                    JSONArray paraderosJson = respuestaJson.getJSONArray("data");

                    db1.beginTransaction();
                    //devuelve = "insert into TParaderos values (";
                    devuelve = "insert into TParaderos values ";
                    try{
                        for(int i=0;i<paraderosJson.length();i++) {
```

*Figura 56.* Captura de un fragmento del código usado para consumir un servicio web; que a su vez incluye tareas asíncronas.

#### 4.4.2.1.10 Investigación sobre SQLite

Durante esta investigación; se revisó , además de algunos textos, la página web oficial de SQLite (SQLite), ya que dentro explica con bastante sencillez todo lo que se necesita saber para entender como funciona este gestor de bases de datos.

Para el presente proyecto, no era necesario que los conocimientos sobre SQLite fueran muy profundos, se busca crear tablas, insertar datos y consumir los mismos, por lo tanto la investigación se centro en esos puntos principales.

La sintaxis en SQLite es basada en SQL, por lo que usar Sqlite o MySql es casi igual, sin embargo algo en lo que varía sustancialmente, al menos en Android, es en la creación de las bases de datos. La información se amplía en el punto 4.4.2.2.

#### 4.4.2.2 Quinto Sprint

Durante el quinto sprint se desarrollaron las clases y el proceso a seguir para crear la base de datos en el móvil y la creación y acceso a los mismos. Cabe recalcar que en el quinto sprint se logra programar todo lo necesario para que la base de datos se cree y funcione correctamente, sin embargo las distintas consultas se van creando a medida que se necesitan. Por otro lado en el caso de las clases, es a partir del sexto sprint donde se ejecuta el código creado y se agregan funciones a medida que se requieren.

Para tener los mismos datos tanto en la base de datos en línea como en la base de datos del dispositivo Android, es necesario utilizar Servicios Web, de esa manera se extraen los datos y luego se crea el código necesario para consumirlos y así ingresarlos al dispositivo móvil. Como adicional, se creó el diseño de la interfaz de la aplicación. Para ello las tareas realizadas se aprecian en la tabla 16.

Tabla 16

*Lista de requerimientos del quinto sprint*

| **Requerimientos** |
|---|
| Desarrollo de Servicios Web en PHP |
| Creación de base de datos en SQLite |
| Consumo de Servicios Web |

Pruebas de funcionamiento de la base de datos SQLite
Diseño general de la interfaz de la aplicación móvil

Fuente propia.

#### 4.4.2.2.1 Desarrollo de Servicios Web en PHP

El proceso de desarrollo de los servicios web fue simple, cada tabla de la cual se necesita exportar datos necesita un servicio web propio, esto quiere decir que cada tabla de base de datos que se necesita tener en el móvil, tiene su propio servicio web.

Para desarrollar los servicios web se utilizó el formato JSON y la función “json_encode”, que viene nativamente con PHP. Se creó una carpeta para almacenar los servicios, y allí se crearon cinco servicios web, cada uno para una tabla de la base de datos. (Fig. 57)

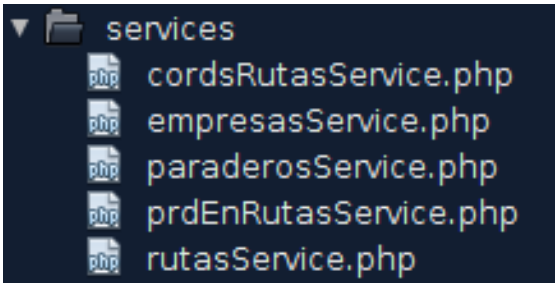



*Figura 57.* Lista de Servicios Web creados en PHP.

#### 4.4.2.2.2 Creación de base de datos SQLite

Para crear bases de datos en SQLite se debe seguir casi el mismo procedimiento que se usa en MySql, con la diferencia que en los dispositivos móviles el mantenimiento de las mismas es un poco diferente.

Lo primero para crear una base de datos es generar una clase que extienda la librería “SQLiteOpenHelper”, para ello se debe importarla mediante la línea:

```
import android.database.sqlite.SQLiteOpenHelper;
```

Al extender la clase creada, se debe ingresar dos funciones con el “@Override” por delante, “onCreate” y “onUpdate”, cada una con sus respectivos parámetros. La primera función se ejecutará si la base de datos se va a crear por primera vez, y la segunda en caso la versión de la misma sea diferente por lo tanto se llevaría a cabo una actualización.

Es importante recalcar que al usar Servicios Web para importar los datos de la base de datos en línea,e insertarlos en la base de datos del móvil, se debe usar tareas asíncronas. Una vez creada la base de datos, se debe crear las tablas, y para ello se utiliza el lenguaje SQL estándar, sin embargo hay que tener en cuenta que SQLite solo soporta un número reducido de tipos de datos. (ver 2.1.37)

#### 4.4.2.2.3 Consumo de Servicios Web

Consumir un servicio web significa recibirlo y procesarlo, y para ello, en Android, se debe de hacer uso de tareas asíncronas. En este proyecto lo que se hizo para consumirlo fue lo siguiente. Una vez creada la base de datos y las tablas correspondientes, faltaba llenar las tablas, para ello se extrajo los datos almacenados en las tablas de la base de datos del servidor mediante servicios web, y una vez recibidos, se ingresaron en las tablas creadas en la base de datos del dispositivo.

#### 4.4.2.2.4 Pruebas de funcionamiento de la base de datos SQLite

Las pruebas fueron realizadas tanto en el simulador como en el dispositivo real. En el simulador basta con extraer la base de datos

mediante el “Android Device Monitor”, que es un componente de Android Studio capaz de, entre otras funciones, mostrar todos los archivos existentes dentro del dispositivo simulado. Una vez dentro se debe ingresar a la carpeta ‘data’, luego otra vez a la carpeta ‘data’, y una vez allí buscar la aplicación que uno esta probando, en este caso ‘tesis_03’. Una vez dentro, entrar a la carpeta ‘database’ y extraer la base de datos creada. En la figura 58 se puede apreciar mejor:

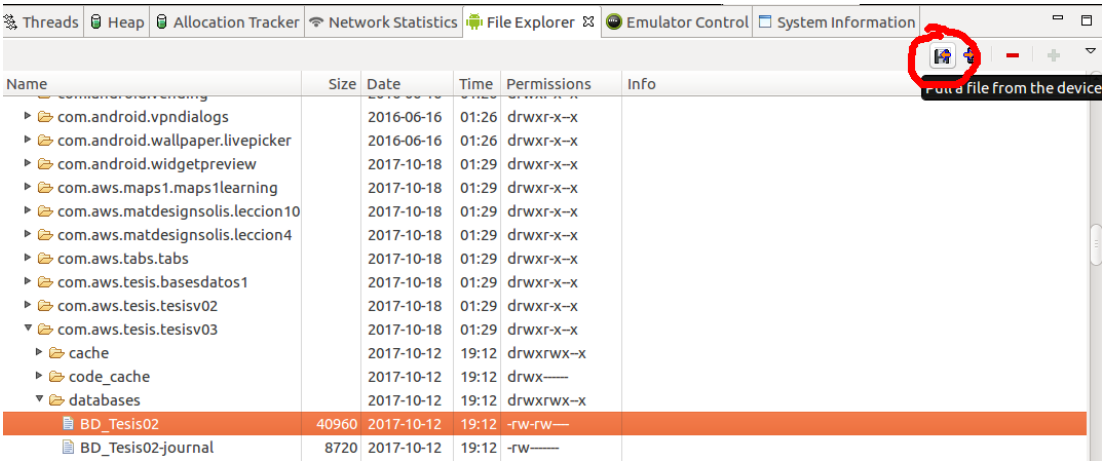



*Figura 58.* Extracción de la base de datos desde un dispositivo virtual usando Android Studio.

Una vez extraído el archivo, este puede visualizarse haciendo uso de un programa llamado “DB Browser for SQLite”, el cual muestra los datos almacenados en las tablas de manera muy sencilla, ademas de ser capaz de modificarlos. (figura 59)

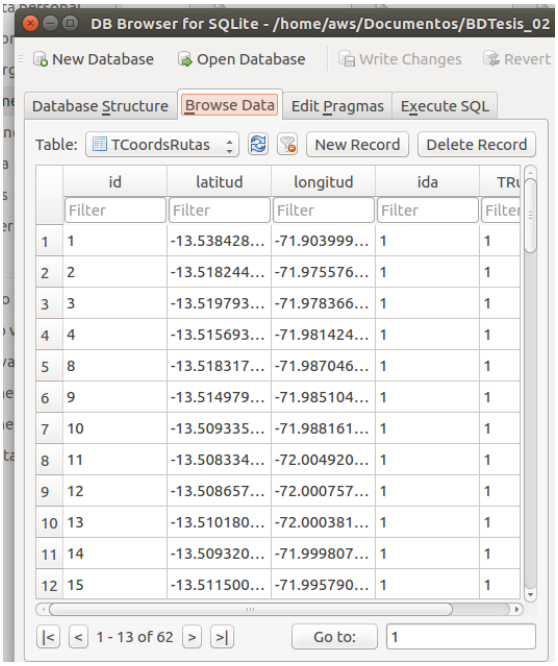



*Figura 59.* Visualización de datos de una base de datos SQLite usando DB Browser for Sqlite.

En el caso de la base de datos almacenada en el dispositivo físico, no se puede acceder a la misma a menos que este esté “rooteado”, lo que significa que se pueda acceder a la raíz del mismo. Como en este caso los dispositivos usados no han sido “rooteados”, la manera de revisar que el ingreso haya sido correcto ha sido mediante la muestra de los mismos en el dispositivo, empezando por el conteo de filas en cada tabla, y haciendo consultas precisas para comparar los resultados del dispositivo y SQLite con el backend y MySql.

#### 4.4.2.2.5 Diseño general de la Interfaz de la aplicación Móvil:

En un principio se trató de utilizar algunas plantillas del Android Studio, sin embargo con el avance del proyecto y mayor conocimiento adquirido, se desestimó usar una plantilla prefabricada. Los colores y el diseño básico de la interfaz ya habían sido decididos en el punto 4.3.7, sin embargo vale la pena mostrar algunos datos extras. En la figura 60 se puede apreciar el código XML que se utilizó para determinar los colores básicos.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#304A61</color>
    <color name="colorPrimaryDark">#263B4C</color>
    <color name="colorAccent">#71ea19</color>
</resources>
```

*Figura 60.* Captura del código que define los colores principales de una aplicación Android.

Por otra parte, Android Studio posee una potente herramienta visual para crear interfaces haciendo uso del mouse, sin embargo, por motivos académicos se prefirió optar por el código en lugar de la herramienta, más que nada para entenderlo al nivel necesario para crear la interfaz general y, consecuentemente, todas las otras interfaces.

#### 4.4.2.3 Sexto Sprint

Durante el sexto sprint se desarrolló todo lo referente a la solicitudes del permiso de ubicación y GPS activo. Dichos procesos requieren de que el usuario interactúe con el sistema y brinde los permisos que se solicitan, sin ellos; el aplicativo no puede funcionar al 100%. Los requerimientos del sexto sprint se visualizan en la tabla 17.

Tabla 17

*Lista de requerimientos del sexto sprint*

| **Requerimientos:** |
|---|
| Diseño y desarrollo de la interfaz de solicitud de permisos |
| Solicitud de GPS activo |
| Solicitud del permiso de ubicación |
| Verificar que la base de datos está creada y contiene data |
| Pruebas del sexto sprint |

Fuente propia.

#### 4.4.2.3.1 Diseño y desarrollo de la interfaz de solicitud de permisos

Los aplicativos en Android no suelen explicar por qué piden permisos para tener acceso a distintas áreas o funcionalidades del móvil, los usuarios suelen aceptar todas las solicitudes y problema resuelto, sin embargo; esto no debería ser así.

La solicitud de permisos es la primera interfaz (figura 61) que aparece al abrir el aplicativo, y la lógica que maneja es la siguiente: Al momento de abrir la aplicación, si es la primera vez que se abre o si la base de datos no ha sido creada y la data descargada del servidor, se procede a crear y descargar la data asíncronamente. Mientras tanto, se verifica si el permiso de ubicación está activo y lo mismo con el GPS del móvil, si no lo están, se solicita ambos. En caso el usuario no brinde alguno de los dos, no podrá ingresar a la aplicación, pero tendrá a la mano un botón para brindarlos. Una vez ambos estén activos, al pulsar el botón ingresar, se validará otra vez los permisos y se verificará si la base de datos ya ha sido creada y contiene data, ya que al ser un proceso que depende del servidor, puede que en algún momento este se encuentre fuera de línea y se deba esperar, y de ser el caso, aparece un mensaje explicando lo que sucede, y que si el problema persiste; trate de ingresar mas tarde al aplicativo.

En caso no sea la primera vez que se abra el aplicativo, si no se cuenta con la base de datos aún, se procederá a descargar la data. En caso exista la data, y los permisos estén activos, esta primera interfaz ya no se mostrará, dando lugar directamente a la vista donde se elige la ubicación del usuario.

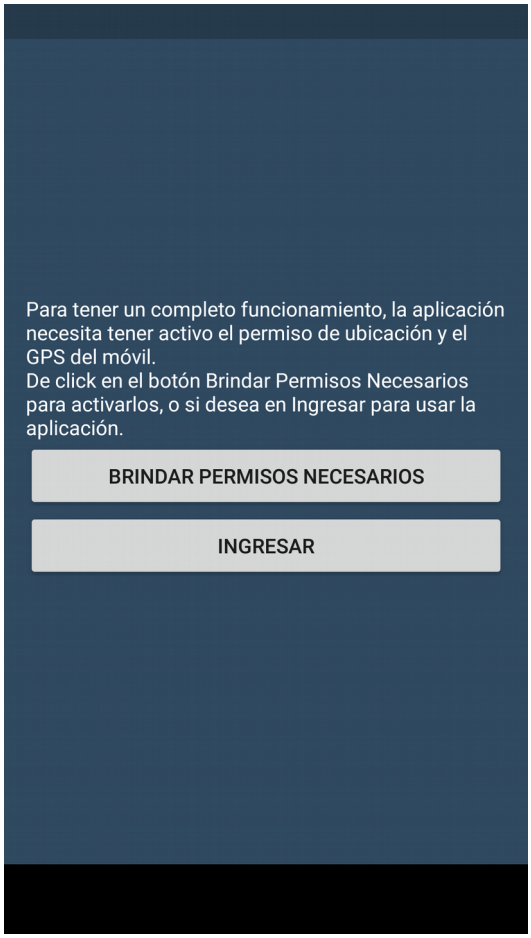



*Figura 61.* Interfaz de solicitud de permisos.

#### 4.4.2.3.2 Solicitud de GPS activo

Es necesario tener el GPS activo para que la aplicación funcione correctamente, por lo tanto, en caso el usuario no lo tenga activado, se le muestra un requerimiento para que lo active. Para completar esta solicitud, se utiliza lo investigado anteriormente. En la imagen nro 55 se puede apreciar la manera en que se solicitó la activación del GPS.

#### 4.4.2.3.3 Solicitud del permiso de ubicación.

Para acceder a la ubicación del dispositivo se necesita tener activo el permiso de ubicación, para ello se utilizó lo investigado en el punto 4.4.1.7. Para versiones de Android anteriores a Marshmallow, basta con ingresar en el archivo "AndroidManifest.xml" el

requerimiento, y de esa manera el permiso de ubicación es otorgado o solicitado automáticamente. Para el proceso de solicitud del permiso de ubicación y GPS se utilizaron las líneas de código expuestas en la figura 62.

```
public void verificarAmbosPermisos(){
    //Toast.makeText(this,"verificando permisos",Toast.LENGTH_SHORT).show();

    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);



    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (locationManager.isProviderEnabled( LocationManager.GPS_PROVIDER ) &&
                (checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED)) {
            //Toast.makeText(this,"La aplicación está lista para iniciar",Toast.LENGTH_SHORT).show();
            ingresarAApp();
        }else{
            verificarGPS();
            verificarPermisoUbicacion();
        }
    }else{

        if(locationManager.isProviderEnabled( LocationManager.GPS_PROVIDER )){
            ingresarAApp();
        }else{
            verificarGPS();
        }
    }
}
```

*Figura 62.* Captura del código usado para solicitar el permiso de ubicación.

#### 4.4.2.3.4 Verificar que la base de datos está creada y contiene data

Para crear la base de datos es necesario seguir los pasos indicados en el punto 4.4.2.2, por lo que al desarrollar esta primera interfaz de solicitud de permisos, se tomó en cuenta la creación de la misma y la verificación que las tablas tengan data. La figura 63 muestra las líneas de código básicas desde donde se ordena la creación de la base de datos:

```
//inicializando base de datos
BaseDatos bDatos = new BaseDatos(this);
db = bDatos.getWritableDatabase();
```

*Figura 63. Captura de código que crea o actualiza una base de datos.*

Hecho esto, en caso sea la primera vez que se utilice la aplicación, se crea la base de datos y se ingresa la data de cada tabla

consumiendo los servicios web. En caso ya exista la base de datos, se verifica que las tablas contengan datos, ya que es posible que al momento de abrir por primera vez la aplicación, no se haya podido consumir los servicios web, por lo que se ejecuta el código mostrado en la figura 64.

En la imagen se muestra la verificación de la tabla de paraderos, y en caso esta no contenga datos, se inicia un timer que muestra un mensaje cada 5 segundos indicando que se está estableciendo conexión con el servidor y que si demora mucho se abra la aplicación en unos minutos. Esto es así porque una vez creada la base de datos, se procede a consumir los servicios web de manera asíncrona, por lo que este proceso sigue en espera de ser completado en un hilo independiente al de la aplicación principal, y si demora en completarse es seguramente por falta de conexión a internet del dispositivo, o porque el servidor esta saturado o no se encuentra en línea durante esos momentos.

```
public void ingresarAApp(){

    CParaderos prd = new CParaderos(getApplicationContext());
    Object[] paraderos = prd.getAll();
    int contador = paraderos.length;

    if(contador > 0){
        Intent intent = new Intent(FirstActivity.this,OrigenActivity.class);
        startActivity(intent);
    }else{
        txt_espere.setText(R.string.espere_carga);
        Toast.makeText(getApplicationContext(),"Estableciendo conexión con el servidor. Si demora mucho...",Toast.LENGTH_SHORT).show();
        timer();
    }
}

public void timer(){
    new CountDownTimer(5000,1000){

        @Override
        public void onTick(long millisUntilFinished) {

        }

        @Override
        public void onFinish() {
            CParaderos prd = new CParaderos(getApplicationContext());
            Object[] paraderos = prd.getAll();
            int contador = paraderos.length;

            if(contador > 0){
                ingresarAApp();
            }else{
                txt_espere.setText("Estableciendo conexión con el servidor. Si demora mucho...");
                Toast.makeText(getApplicationContext(),"Estableciendo conexión con el servidor. Si demora mucho...",Toast.LENGTH_SHORT).show();
                timer();
            }
        }
    }.start();
}
```

*Figura 64.* Captura del código usado para verificar si se realizó la conexión con el servidor de manera correcta revisando si la base de datos tiene datos.

#### 4.4.2.3.5 Pruebas del sexto sprint

El proceso de pruebas para el sexto sprint ha sido riguros en cada uno de los requerimientos desarrollados, para empezar la aplicación no funcionará a menos que se tenga el permiso de ubicación y el GPS activo, y también hasta que la base de datos esté creada y con los datos insertados.

Se realizaron pruebas combinando permisos activos y no activos, abriendo la primera vez y posteriormente quitando el permiso de ubicación o GPS, cambiando la IP del servidor para que no se descargue la data correctamente y el mensaje de espera aparezca, y finalmente desinstalando numerosas veces la aplicación para volverla a instalar desde cero y probar que todo funcione bien.

### 4.4.2.4 Séptimo Sprint

Durante el séptimo sprint se desarrollo todo lo referente a la visualización de las empresas y rutas registradas en el sistema, a continuación se detalla los requerimientos (tabla 18) que se desarrollaron en este sprint y se desarrolla cada uno de ellos.

Tabla 18

*Lista de requerimientos del séptimo sprint*

| **Requerimientos:** |
|---|
| Diseño y desarrollo del menú de la aplicación |
| Diseño y desarrollo de las interfaz para visualizar las empresas registradas |
| Desarrollo del proceso de mostrar empresa registradas elegida |
| Diseño y desarrollo de la interfaz de vista de rutas registradas |
| Desarrollo del proceso de mostrar la ruta registrada elegida |
| Gráfico de rutas de ida y vuelta |
| Pruebas del séptimo sprint |

Fuente propia.

#### 4.4.2.4.1 Diseño y desarrollo del menú de la aplicación

El menú tiene como fin el de ordenar la aplicación agrupando los distintos botones que envían al usuario a las distintas vistas, y por ende funciones, de la aplicación. Como se aprecia en la figura 65, el menú principal es simple y directo, agrupa los procesos principales de la aplicación, y accede a los mismos al pulsar sobre los botones mostrados.

Mediante el boton “A viajar”, se accede al proceso de elección de punto de origen y destino; elección de empresa para viajar, y a la guía de viaje. El botón “Empresas” por su parte, muestra un listado de empresas registradas en el sistema y al pulsar en alguna de ellas, se muestra los datos de la misma. Finalmente, el botón “Rutas” muestra las distintas rutas almacenadas en el sistema y los trazados de ida y vuelta de las mismas.


*Figura 65.* Menu de la aplicación.

#### 4.4.2.4.2 Diseño y desarrollo de la interfaz para visualizar las empresas registradas

Para visualizar las empresas registradas, hacen falta solamente dos interfaces, la primera donde se ve la lista de empresas registradas con los eventos click programados sobre cada item de la misma, y la vista donde se visualiza detalladamente los datos de la empresa elegida.

Para ellos se utiliza el ListView, que debe declararse tanto en los layouts como en el código principal de la clase. En la figura 66 se aprecia cómo se carga la lista de empresas

```
CEmpresasAdapter adapter = new CEmpresasAdapter(this,R.layout.listview_sug_emp_row,empresas_data);
lv = (ListView)findViewById(R.id.listView_lista_empresas);
View header = (View)getLayoutInflater().inflate(R.layout.listview_sug_emp_header,null);

lv.addHeaderView(header);
lv.setAdapter(adapter);

lv.setOnItemClickListener(new AdapterView.OnItemClickListener(){

    String nombreEmpresaEleg;
    String idEmpElegida;
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

        TextView txt1= (TextView)view.findViewById(R.id.tv_row_sugEmpresas);
        nombreEmpresaEleg = txt1.getText().toString();


        TextView txt2 = (TextView)view.findViewById(R.id.tv_row_idSugEmpresa);
        idEmpElegida = txt2.getText().toString();

        //Toast.makeText(getApplicationContext(),idEmpElegida+": "+nombreEmpresaEleg,Toast.LENGTH_SHORT).show();

        Intent intent = new Intent(ListaEmpresasRegistradas.this,VerEmpresaYRutas.class);
        startActivity(intent);
    }
});
```

*Figura 66.* Captura de carga de datos al Listview que muestra empresas.

Y la vista que se genera es la mostrada en la figura 67.

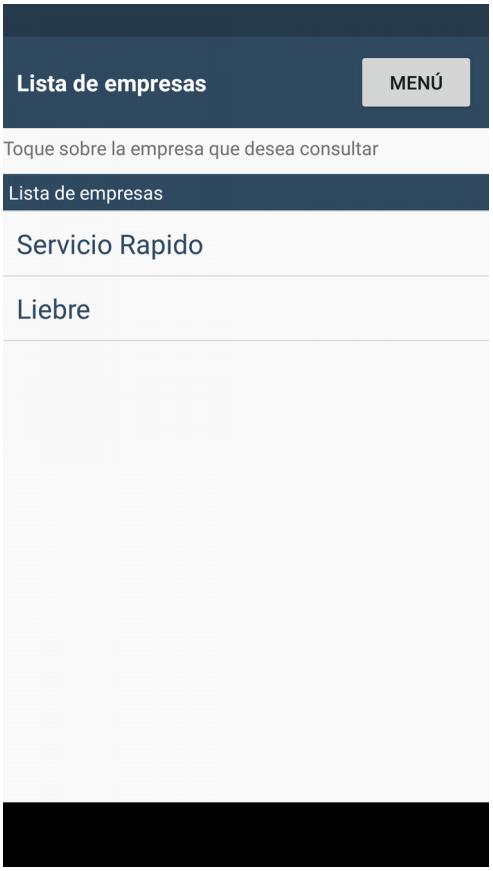



*Figura 67 Interfaz que muestra la lista de empresas.*

Dentro del ListView generado, se procede a programar los listeners para los items del mismo, para ello se utiliza la función "SetOnItemClickListener".

#### 4.4.2.4.3 Desarrollo del proceso de mostrar empresa registrada elegida

Una vez elegida una empresa, es momento de mostrar los datos de la misma. Para ello se obtiene el identificador de la empresa elegida, y este es enviado a la vista correspondiente, en la cual se carga los datos de la empresa. Como se puede apreciar en la figura 68, se muestra los datos registrados, y se brinda la opción de visualizar la ruta de ida y vuelta que cubre.

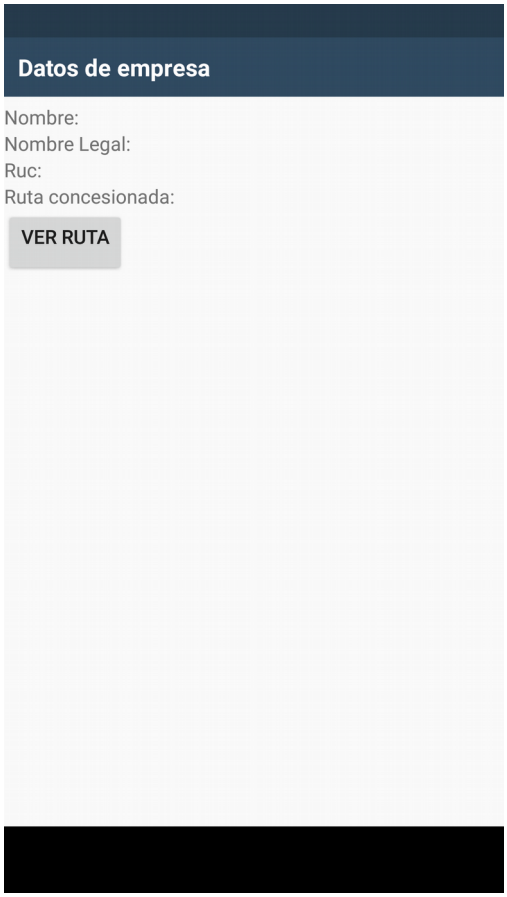



*Figura 68.* Interfaz para visualizar datos de empresa.

#### 4.4.2.4.4 Diseño y desarrollo de la interfaz de vista de rutas registradas

Para el diseño, se sigue el mismo diseño del punto 4.4.2.4.2, con la variación que en lugar de mostrar empresas se muestran las rutas registradas en el sistema. También se utiliza el ListVew como herramienta para listar las rutas como se ve en la figura 69.

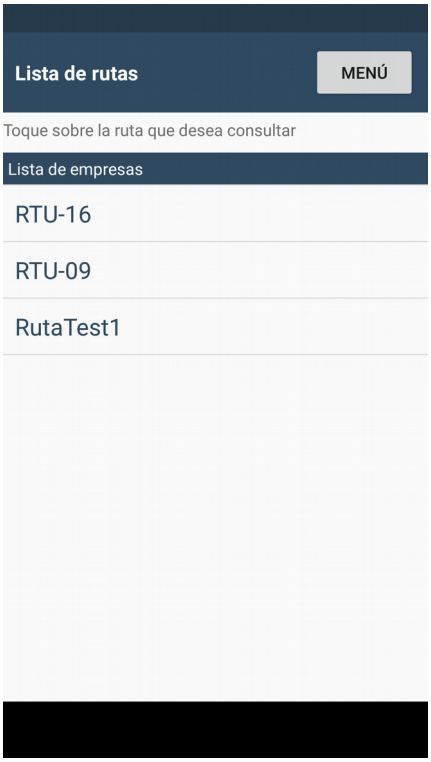



*Figura 69.* Interfaz para visualizar rutas registradas.

#### 4.4.2.4.5 Desarrollo del proceso para mostrar la ruta registrada elegida

Una vez desarrollada la interfaz, se procede a mostrar los datos de la ruta elegida, es similar a la vista de empresa elegida, pero en este caso se visualiza en los mapas de Google el gráfico de la ruta de ida y de vuelta. Para esto se debe hacer uso de los “polylines”, y del servicio de direcciones de Google.

**Gráfico de rutas de ida y vuelta**

Para graficar las rutas de ida y vuelta de la empresa elegida, se sigue un proceso que, en esencia, es similar al usado para graficar las rutas en el backend. Entre las diferencias mas resaltantes se puede indicar que es necesario escribir más

líneas de código que en Javascript, y que se utiliza al menos una clase extra para mantener un poco más de orden. (fig. 70)

Por otro lado, también se utilizan fragmentos para introducir los mapas de Google, y la clave que se ingresó en el archivo "AndroidManifest.xml" es reutilizada por todas las vistas de la aplicación que hagan uso de mapas de Google.

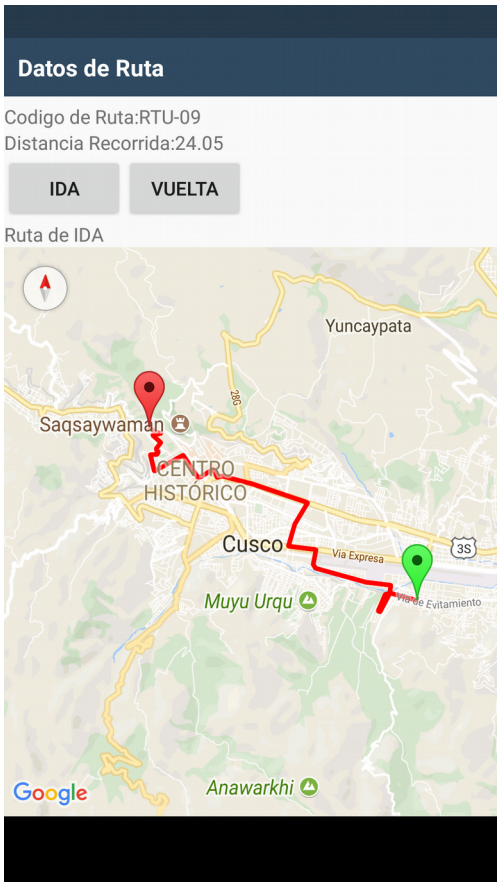



*Figura 70.* Interfaz para ver rutas de ida y vuelta.

#### 4.4.2.4.6 Pruebas del séptimo sprint

Las pruebas fueron exhaustivas al igual que en los otros sprints, y vale la pena resaltar que mediante las pruebas realizadas, el proceso de navegación dentro de la aplicación quedó prácticamente terminado.

#### 4.4.2.5 Octavo sprint

El octavo sprint recoge gran parte del conocimiento adquirido en el proceso de desarrollo de la presente tesis, utiliza los datos recopilados mediante los servicios web, hace uso de los mapas de Google, inserta íconos en ellos, se programan eventos; controla los mismos, y es capaz de mostrar en el mapa la ubicación del usuario haciendo uso del GPS del móvil. A continuación se detalla los requerimientos desarrollados en el octavo sprint dentro de la tabla 19.

Tabla 19

*Lista de requerimientos del octavo sprint*

| **Requerimientos** |
|---|
| Diseño de las interfaces para la elección del punto de origen y destino del viaje |
| Desarrollo de las interfaces para la elección del punto de origen y destino del viaje |
| Inserción de mapas de Google en ambas interfaces (origen y destino) |
| Inserción de íconos de paraderos en mapas de Google |
| Programación de eventos en mapas de Google |
| Control de coordenadas ingresadas en referencia a los paraderos registrados |
| Programación de los intents |
| Pruebas del octavo sprint |

Fuente propia.

##### 4.4.2.5.1 Diseño de las interfaces para la elección del punto de origen y destino del viaje

Los diseños de las mencionadas interfaces son importantes ya que deben facilitar al usuario la ubicación del punto que él o ella considera que es el punto de origen de su viaje, y también el de destino, para ello, una vez se pulsa el botón “¡A Viajar!” del menú de la aplicación (véase 4.4.2.4.1), se muestra un pequeño texto de

instrucción y un mapa donde el usuario puede escoger el punto de origen de su viaje, además se incluye un botón de siguiente para que una vez elegido el punto de origen, y pulsado el botón “siguiente”, se proceda a al siguiente vista donde elige la ubicación de su destino, para finalmente volver a pulsar el botón “siguiente”.

Las interfaces para elegir el punto de origen y destino son prácticamente las mismas, con la diferencia que en la interfaz donde se elige el destino (como previamente ya se eligió el punto de origen), el punto de origen aparece como un marcador de color verde, y, al elegir el punto de destino, aparece como un marcador de color rojo. También se muestra en ambas interfaces (origen y destino) los paraderos registrados en el sistema mediante los íconos de paradero.

Las interfaces para elegir el origen y el destino del viaje obedecen al diseño básico de interfaces mostrados en el punto 4.3.7, e integran diferentes elementos entre ellos un botón para ir al siguiente paso, un Label donde se visualiza las coordenadas de los puntos elegidos.

Estas interfaces también integran mapas de Google donde se muestran los íconos y se eligen los puntos. En la figura 71 se puede pareciar la interfaz de destino (paso 2) con el punto de origen previamente elegido

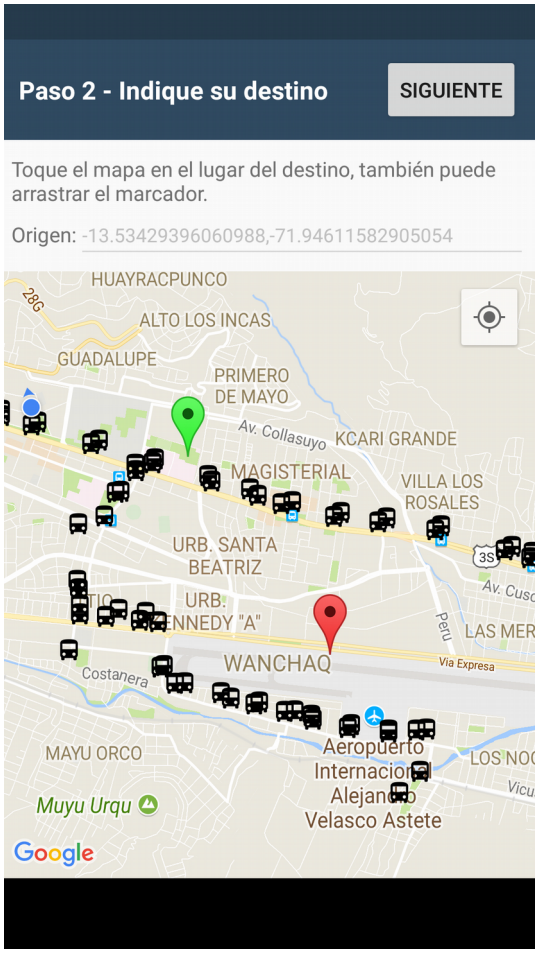



*Figura 71.* Interfaz de elección de punto de destino.

#### 4.4.2.5.2 Desarrollo de las interfaces de elección del punto de origen y destino de viaje

El proceso de desarrollo de ambas interfaces agrupa gran parte del conocimiento obtenido en cuanto al uso de mapas de Google, primeramente se debe declarar los “Layouts”, que si bien son similares, son ambos igual de importantes.

Se debe realizar consultas a la base de datos SQLite creada en el dispositivo móvil, de allí se extrae las coordenadas de los paraderos registrados y se insertan en el mapa haciendo uso de íconos. Además, se programó “listeners” sobre los mismos para que una vez sean tocados en la pantalla, muestren sus nombres.

Para controlar que el usuario no pueda elegir cualquier punto del mapa como punto inicial o final, el sistema analiza el punto elegido y verifica que este se encuentre a como máximo 500 metros en línea recta a cualquiera de los paraderos registrados. Esto se logra programando un “listener” al mapa para controlar el evento “click” que se hace sobre el mismo, y si cumple con la condición de los 500 metros, se inserta un ícono de ubicación.

Finalmente, se programa “listeners” sobre los marcadores que indican los puntos de origen y destino, para que si se pulsa sobre otro lugar del mapa, estos se muevan, y también el “listener” para que estos puedan ser arrastrados y reubicados.

#### 4.4.2.5.3 Inserción de mapas de Google en ambas interfaces (origen y destino)

Para insertar los mapas de Google en las vistas, es necesario declarar un fragmento en el “layout” donde se requiere el mapa, y posteriormente indicar los parametros que se necesite en la vista que hace uso del mismo.

#### 4.4.2.5.4 Inserción de íconos de paraderos en mapas de Google

El proceso de inserción de paraderos en los mapas de Google obedece a un par de reglas básicas al momento de programar en Android. La primera es ingresar los íconos a utilizar en las distintas resoluciones para que de esa manera se vean bien tanto en dispositivos con pantallas de baja como de alta resolución. Y la segunda es que se debe tratar de evitar que estos se aglomeren dentro del mapa, y para ello es necesario mostrarlos sólo cuando el usuario acerque lo suficiente el mapa.

Tras consultar la base de datos y obtener las coordenadas de los paraderos, estos se ingresan al mapa y se programa “listeners” sobre

ellos para que muestren su nombre una vez sean pulsados por el usuario.

#### 4.4.2.5.5 Programación de eventos en mapas de Google

La programación de eventos en los mapas de Google para Android es muy similar al usado en Javascript, entre ellos podemos destacar los siguientes.

- Evento “click” sobre el mapa.
- Evento “dragend” sobre el mapa.
- Evento “onChangeCamera”. (el cual se activa al momento de acercar o alejar el mapa)

#### 4.4.2.5.6 Control de coordenadas ingresadas en referencia a los paraderos registrados

Este es un punto importantísimo para evitar que el usuario pueda ingresar un punto de origen o destino muy alejado de la provincia del Cusco. Para ello, como se mencionó anteriormente, se controla que este no supere los 520 metros en línea recta, para lo cual se utiliza la función mostrada en la figura 72.

```
//comparar todas las coordenadas de los paraderos con el ptoElegido
//en caso haya al menos 1 paradero a una distancia correcta del ptoElegido
//el sistema aceptará el punto de origen como válido
public boolean validarPuntoOrigenIngresado(LatLng ptoElegido){
    Object[] listP = this.getAll();
    CParaderos listPdos = new CParaderos();
    boolean flag = false;


    Location ptoE = new Location("ptoElegido"); //ptoElegido por el usuario
    ptoE.setLatitude(ptoElegido.latitude);
    ptoE.setLongitude(ptoElegido.longitude);

    Location ptoComp = new Location("ptoComparativo"); //ptoComparativo para medir la distancia y verificar que se cumpla la lógica
    for(int k = 0; k<listP.length; k++){
        listPdos = (CParaderos)listP[k];
        ptoComp.setLatitude(listPdos.getLatitud());
        ptoComp.setLongitude(listPdos.getLongitud());

        flag = flag || (ptoE.distanceTo(ptoComp) <= distanciaMax);
    }


    return flag;
}
```

*Figura 72.* Captura de la función que valida el punto de origen o destino ingresado.

#### 4.4.2.5.7 Programación de intents

Los “intents” llevan al usuario a las distintas vistas, y también llevan consigo los datos necesarios para que estas funcionen correctamente. El proceso de programación de los intents cumple una función muy importante para mantener el flujo correcto dentro de la aplicación, tanto al momento de ser cargados con datos y mostrar la siguiente vista, como al momento de ser recibidos.

En la figura 73 se aprecia cómo se programa un intent, y cómo se carga datos sobre el mismo:

```
//Intent intent = new Intent(DestinoActivity.this,SugerenciaEmpresasTest0.class);
Intent intent = new Intent(DestinoActivity.this,SugerenciaEmpresas.class);
intent.putExtra("origen_lat",latOrigen);
intent.putExtra("origen_long",longOrigen);
intent.putExtra("destino_lat",coordsSplit[0]);
intent.putExtra("destino_long",coordsSplit[1]);

startActivity(intent);
```

*Figura 73.* Captura del código para programación de intents.

#### 4.4.2.5.8 Pruebas del octavo sprint

Las pruebas en el octavo sprint fueron (como en todos los casos) exhaustivas, se probaron las vistas creadas de manera integral, cada pequeño avance era pasado por una o varias pruebas hasta que no haya duda que funcione correctamente.

### 4.4.2.6 Noveno sprint

El noveno sprint concluye lo que el octavo sprint empieza. Esto quiere decir, que además de ser la continuación del octavo sprint en cuanto al flujo de la aplicación, también termina de integrar todo lo aprendido e investigado para realizar el presente proyecto. En la tabla 20 se listan los requerimientos del noveno sprint y posteriormente se detalla el proceso realizado para su cumplimiento.

Tabla 20

*Lista de requerimientos del noveno sprint*

| **Requerimientos** |
|---|
| Diseño de la interfaz de sugerencias de empresas en las que viajar |
| Desarrollo de interfaz de sugerencia de empresas en las que viajar |
| Programación del proceso para generar sugerencias de empresas en las que viajar |
| Diseño de la interfaz de guiado de viaje |
| Desarrollo del proceso de guiado de viaje |
| Gráfico de ruta a seguir |
| Seguimiento en tiempo real mediante el sistema de GPS |

Fuente propia.

#### 4.4.2.6.1 Diseño de la interfaz de sugerencias de empresas en las que viajar

La interfaz mencionada tiene como objetivo el indicar qué empresas son capaces de llevar al usuario a su destino una vez este haya elegido los puntos de origen y destino de su viaje. Para ello se hace uso una vez más de la herramienta “ListView”. Lógicamente para mostrar la lista de empresas recomendadas se realiza un proceso de búsqueda y comparación, tras el cual se muestra la lista de sugerencias,

El proceso de diseño de esta interfaz es similar al del proceso de diseño usado para mostrar empresas y rutas (ver 4.4.2.4.2), por lo tanto llevarlo a cabo no fue mayor reto. El resultado final se puede apreciar en la figura 74.

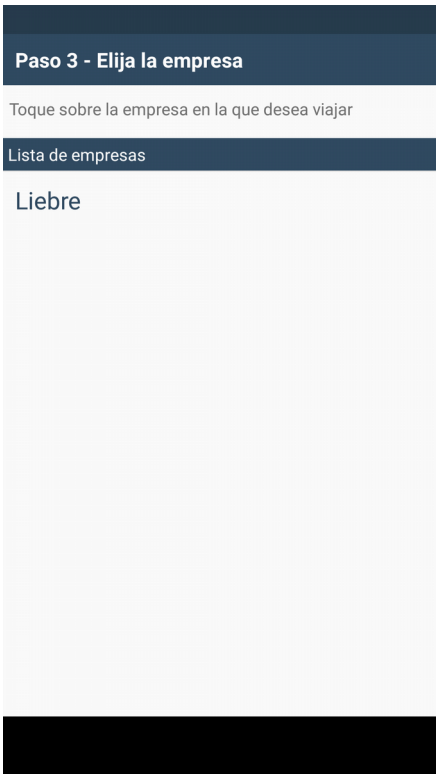



*Figura 74.* Interfaz de sugerencia de empresas en las que viajar.

#### 4.4.2.6.2 Desarrollo de la interfaz de sugerencias de empresas en las que viajar

El desarrollo de la interfaz es similar a las otras interfaces usadas, especialmente en las que se muestra el listado de rutas y empresas. Se diseña el “layout” de la misma, se declara en el archivo “AndroidManifest.xml”, y en la clase de la vista se infla y declara todas las variables necesarias para que esta funcione correctamente. En los items del “ListView” se agrega listeners para que una vez se pulse sobre la empresa elegida para viajar, el usuario visualice la siguiente interfaz donde se le indica de qué paradero debe tomar el bus y dónde debe bajar, además de la ruta que va a seguir.

#### 4.4.2.6.3 Programación del proceso para generar sugerencias de empresas en las que viajar

Este proceso es uno de los más importantes e interesantes del aplicativo. Gracias a él, la app es capaz de indicar al usuario qué empresa o empresas pueden llevarlo desde el punto de origen al destino que este ha ingresado.

Para poder sugerir rutas al usuario es necesario entender lo siguiente:

- Calcular la distancia entre los puntos ingresados.
- Listar qué paraderos cercanos tiene el punto de origen.
- Listar qué paraderos cercanos tiene el punto de destino.
- Listar las rutas que pasan por los paraderos cercanos al punto de origen.
- Listar las rutas que pasan por los paraderos cercanos al punto de destino.
- Definir el sentido (ida o vuelta) en la que pasan las rutas por los puntos elegidos.

Una vez definido estos cinco puntos, ya es posible generar sugerencias de rutas. Para ello se siguió el siguiente proceso:

Primeramente se verifica la distancia entre los dos puntos ingresados (origen y destino), y se procede a medir la distancia entre ambos. Si la distancia es menor o igual a 1040 metros en línea recta, la aplicación muestra un mensaje indicando que la distancia es menor a 1040 metros y se sugiere ir caminando. (figura 75)

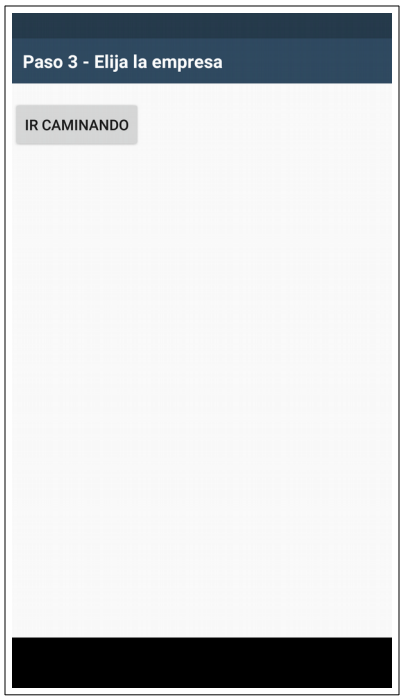



*Figura 75.* Interfaz que sugiere ir caminando.

En caso la distancia sea mayor a 1040 metros, se procede a obtener la lista de paraderos cercanos al punto de origen; y al punto de destino. Esto significa crear una lista con los paraderos que se encuentran dentro de un radio de 520 metros del punto de origen, y de manera similar, con el punto de destino. Esto sirve para obtener el listado de paraderos “útiles” tanto para subir como bajar del bus, además de obtener datos para poder comparar rutas y sentidos de rutas. Esto se explica mejor más adelante.

El código utilizado para obtener la lista de paraderos cercanos es bastante sencillo. Se compara los puntos de origen y destino al listado de paraderos ingresados al sistema, y si la distancia es menor o igual a 520 metros, estos se consideran como “útiles” y son agregados a los arreglos. Esto se muestra en la figura 76.

```
public List<CParaderos> getParaderosCercanos(LatLng punto){
    Object listP[] = this.getAll();
    List<CParaderos> result = new ArrayList<~>();

    //pto elegido por usuario
    Location ptoElg = new Location("ptoElegido");
    ptoElg.setLatitude(punto.latitude);
    ptoElg.setLongitude(punto.longitude);

    //pto a comparar distancia y verificar que se cumpla la regla
    Location ptoComp = new Location("ptoComparativo");


    CParaderos prdTemp = new CParaderos();
    for(int i=0;i<listP.length;i++){
        prdTemp = (CParaderos)listP[i];
        ptoComp.setLatitude(((CParaderos)listP[i]).getLatitud());
        ptoComp.setLongitude(((CParaderos)listP[i]).getLongitud());

        float distanc = distanciaMax+20;
        if(ptoElg.distanceTo(ptoComp) <= distanc){
            result.add((CParaderos)listP[i]);
        }
    }

    return result;
}
```

*Figura 76.* Captura de función que obtiene paraderos cercanos.

Una vez que se obtiene la lista de paraderos cercanos, se debe obtener las listas de empresas que pasan por los paraderos cercanos al punto de origen (figura 77), y la lista de empresas que pasan por los paraderos cercanos al punto de destino (figura 78), para ello se hace uso de dos funciones muy similares pero a la vez distintas en un punto determinante, el cual es la forma en que se ordenan los resultados en las consultas que se realizan a la base de datos.

Este ordenamiento se realiza de esa manera para que al momento de comparar ambos arreglos, se pueda extraer el listado de rutas y sentidos de las mismas que cubren ambos puntos.

```
//si la consulta no genera ninguna fila de respuesta, retornará NULL
//public CPrdEnRutas[] getAllFromPdosIdsOrderByPdoId(int []idsParaderos){
public CPrdEnRutas[] getAllFromPdosIdsOrderByPdoId(int []idsParaderos){

    String res = "";
    CPrdEnRutas[] resultado = null;
    CommonQueries cm = new CommonQueries(c);
    //construyendo la query
    String query = "select * from TPrdEnRutas where ";
    for(int i =0;i<idsParaderos.length;i++){
        if(i!=idsParaderos.length-1){
            query += "TParaderos_id="+idsParaderos[i]+" OR ";
        }else{
            query += "TParaderos_id="+idsParaderos[i]+" order by TParaderos_id,id;";
        }
    }

    Cursor cursor = cm.executeSelection(query);
    if(cursor.getCount() > 0){
        resultado = new CPrdEnRutas[cursor.getCount()];
        int i=0;
        if(cursor.moveToFirst()){
            do{
                CPrdEnRutas prdEnR = new CPrdEnRutas();

                prdEnR.setId(cursor.getInt(0));
                prdEnR.setTRutas_id(cursor.getInt(1));
                prdEnR.setTParaderos_id(cursor.getInt(2));
                prdEnR.setIda(cursor.getInt(3));

                resultado[i] = prdEnR;
                i++;
            }while(cursor.moveToNext());
        }
    }


    return resultado;

}
```

*Figura 77.* Captura de código que obtiene la lista de empresas que usan paraderos cercanos al punto de origen.

```
public CPrdEnRutas[] getAllFromPdosIdsOrderByIdaId(int []idsParaderos){
    CPrdEnRutas[] resultado = null;
    CommonQueries cm = new CommonQueries(c);
    //construyendo la query
    String query = "select * from TPrdEnRutas where ";
    for(int i =0;i<idsParaderos.length;i++){
        if(i!=idsParaderos.length-1){
            query += "TParaderos_id="+idsParaderos[i]+" OR ";
        }else{
            query += "TParaderos_id="+idsParaderos[i]+" order by ida,id;";
        }

    }

    Log.d("query2",query);

    Cursor cursor = cm.executeSelection(query);
    if(cursor.getCount() > 0){
        resultado = new CPrdEnRutas[cursor.getCount()];
        int i=0;
        if(cursor.moveToFirst()){
            do{
                CPrdEnRutas prdEnR = new CPrdEnRutas();

                prdEnR.setId(cursor.getInt(0));
                prdEnR.setTRutas_id(cursor.getInt(1));
                prdEnR.setTParaderos_id(cursor.getInt(2));
                prdEnR.setIda(cursor.getInt(3));

                resultado[i] = prdEnR;
                i++;
            }while(cursor.moveToNext());
        }
    }


    return resultado;
}
```

*Figura 78.* Captura de función que obtiene la lista de empresas que usan paraderos cercanos al punto de destino.

Luego se comparan ambos resultados y se extrae la lista de posibles rutas y el sentido de las mismas mediante el algoritmo mostrado en la figura 79.

```
//empiezan comparativas y entramos a extraer rutas útiles
//final List<int[][]> posiblesRutas = new ArrayList<int[][]>();
List<int[][]> posiblesRutas = new ArrayList<int[][]>();


for(int i=0;i<ACompleto.length;i++){
    for(int k=0; k<BCompleto.length; k++){
        if(ACompleto[i].getId() < BCompleto[k].getId()){//comparamos idA < idB
            if(ACompleto[i].getTRutas_id() == BCompleto[k].getTRutas_id()){ //
                if(ACompleto[i].getIda() == BCompleto[k].getIda()){ //si el se
                    //posiblesRutas.add(String.valueOf(ACompleto[i].getTRutas_
                    //cargando posibles rutas con data obtenida

                    int[][] arrTmp = new int[1][2];
                    arrTmp[0][0] = ACompleto[i].getTRutas_id(); //rutaID
                    arrTmp[0][1] = ACompleto[i].getIda(); //ida
                    posiblesRutas.add(arrTmp);
                }
            }
        }
    }
}
```

*Figura 79.* Captura de código que obtiene posibles rutas.

Una vez hecho esto, se eliminan los resultados repetidos y se obtiene el resultado de rutas y lo sentidos de las mismas mediante el algoritmo mostrado en la figura 80.

```
//este array se conforma por el id de la ruta que se sugiere y el sentido de la misma (ida),
//es por ello que es bidimensional
//es final debido a que es un requerimiento del listview
final List<int[][]> posiblesRutasFinal = new ArrayList<int[][]>();


for(int i=0;i<posiblesRutas.size();i++){
    //ff += String.valueOf(posiblesRutas.get(i)[0][0]) + "-" + String.valueOf(posiblesRutas.get(i)[0][1]) + "\n";
    int fll = 1;
    if(posiblesRutasFinal.size() > 0){
        for(int k=0;k<posiblesRutasFinal.size();k++){
            if((posiblesRutas.get(i)[0][0] == posiblesRutasFinal.get(k)[0][0]) && (posiblesRutas.get(i)[0][1] == posiblesRutasFinal.get(k)[0][1])) {
                fll = 0;
            }
        }
        if(fll == 1){
            posiblesRutasFinal.add(posiblesRutas.get(i));
        }

    }else{
        posiblesRutasFinal.add(posiblesRutas.get(i));
    }
}
```

*Figura 80.* Captura de código que elimina posibles rutas repetidas.

Una vez se tienen las rutas útiles y los sentidos de las mismas, se procede a buscar las empresas a las que se les asignó dicha ruta, y estas son cargadas en la vista del ListView. En caso no haya coincidencias, la aplicación mostrará un botón para ir caminando.

#### 4.4.2.6.4 Diseño de la interfaz de guiado de viaje

La interfaz de guiado de viaje, obedece a lo generado en la interfaz de sugerencia de rutas y a la empresa elegida por el usuario. El diseño es similar a las interfaces de origen y destino del viaje, con la diferencia que muestra los paraderos donde el usuario debe subir y bajar para llegar a su destino, además de la ruta que va a seguir. (fig. 81) En caso se deba ir caminando, se muestra los puntos de origen y destino, y la ruta que se debe seguir a pie. (fig. 82)

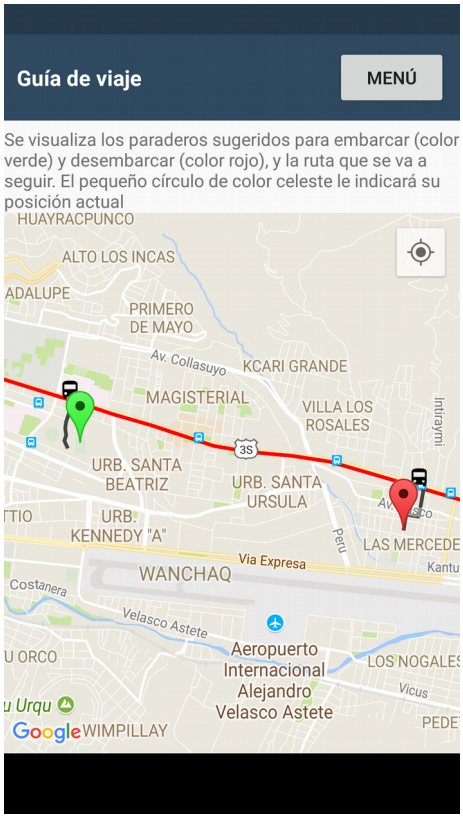



*Figura 81.* Guía de viaje en bus.

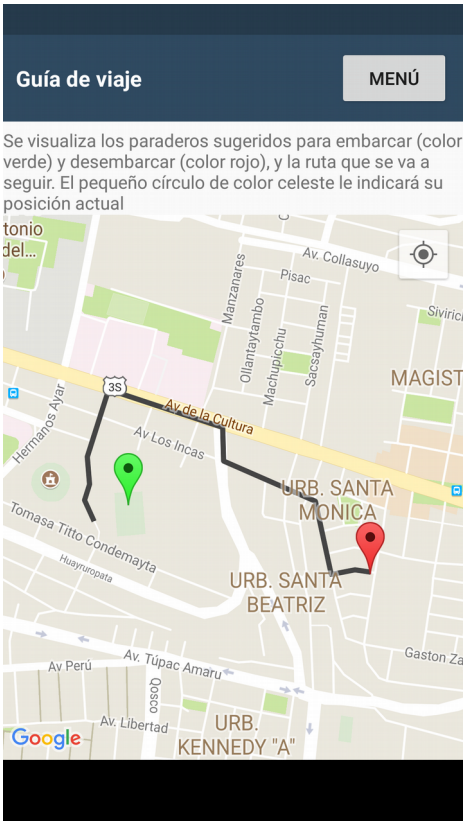



*Figura 82.* Guía de viaje a pie.

#### 4.4.2.6.5 Desarrollo del proceso de guiado de viaje

Para guiar al usuario en el viaje que va a emprender, es necesario haber recopilado todos los datos necesarios de ubicación de el origen y destino del viaje, además de la empresa que desea usar para llegar a su destino.

**Gráfico de ruta a seguir y paraderos a usar**

El gráfico de la ruta obedece a los “polilynes” (que son básicamente la herramienta que se utiliza para graficar dentro de los mapas de Google; siguiendo las rutas que estos contienen), y a las coordenadas de ruta ingresadas en el backend para cada ruta registrada. Haciendo uso de las rutas y los datos obtenidos previamente; se muetra los paraderos que el usuario deberá usar para subir y bajar, y se hace seguimiento del mismo mediante el GPS del móvil en tiempo real.

En caso se vaya caminando, se grafica sólo un “polyline” de color plomo. La figura 83 muestra el código usado para graficar la ruta a seguir en caso se sugiera ir a pie.

```
mapM.addMarker(new MarkerOptions().position(puntoOrigen).title("Origen")
        .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)));
mapM.addMarker(new MarkerOptions().position(puntoDestino).title("Destino")
        .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED)));


String url = getUrl(puntoOrigen,puntoDestino,empEl.getTRutas_id(),ida);
FetchUrlGrey FetchUrl = new FetchUrlGrey();
FetchUrl.execute(url);
```

*Figura 83.* Captura de código que grafica ruta a pie.

En caso se utilice alguna empresa de transporte se grafican tres polylines. La ruta a seguir (rojo), el camino para llegar al paradero donde se abordará al bus de la empresa elegida (plomo), y el camino para llegar desde el paradero donde se debe bajar hasta el punto elegido como destino ( también de color plomo). Para ello se ejecuta las líneas de código de la figura 84.

```
//GRAFICAR LA RUTA ELEGIDA, DESDE EL PTO INICIAL AL FINAL EN EL SENTIDO QUE SE INDICÓ
CRutas rut = new CRutas(getApplicationContext());
LatLng[] coo = rut.coordenadasInicFinDeRuta(idEmpresaElegida);

ptoInicialRuta = new LatLng(coo[0].latitude,coo[0].longitude);
ptoFinalRuta = new LatLng(coo[1].latitude,coo[1].longitude);


//Grafico de ruta en caso sea ida
if(ida==1){
    String url1 = getUrl(ptoInicialRuta,ptoFinalRuta,empEl.getTRutas_id(),ida);
    FetchUrl FetchUrl1 = new FetchUrl();
    FetchUrl1.execute(url1);
}else{
//Grafico de ruta en caso sea vuelta
    String url2 = getUrl(ptoFinalRuta,ptoInicialRuta,empEl.getTRutas_id(),ida);
    FetchUrl FetchUrl2 = new FetchUrl();
    FetchUrl2.execute(url2);
}

//INGRESAR LOS PARADEROS DONDE DEBE SUBIR Y BAJAR
CParaderos pr = new CParaderos(getApplicationContext());
CParaderos prdUtSub= new CParaderos();
prdUtSub = pr.getParaderoUtilMasCercano(puntoOrigen,empEl.getTRutas_id(),ida);
LatLng ubicParSub = new LatLng(prdUtSub.getLatitud(),prdUtSub.getLongitud());

CParaderos prdUtBaj = new CParaderos();
prdUtBaj = pr.getParaderoUtilMasCercano(puntoDestino,empEl.getTRutas_id(),ida);
LatLng ubicParBaj = new LatLng(prdUtBaj.getLatitud(),prdUtBaj.getLongitud());

mapM.addMarker(new MarkerOptions().position(ubicParSub).title(prdUtSub.getNombre())
        .icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_directions_bus_black_18dp)));
mapM.addMarker(new MarkerOptions().position(ubicParBaj).title(prdUtBaj.getNombre())
        .icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_directions_bus_black_18dp)));
//mapM.addMarker(new MarkerOptions().position(puntoDestino).title("Destino").icon(BitmapDescriptorF


//FINALMENTE GRAFICAR LAS RUTAS A CAMINAR ENTRE LOS PARADEROS Y LOS PUNTOS ELEGIDOS

String urlSub = getUrl(puntoOrigen,ubicParSub,0,0);
FetchUrlGrey FetchUrlSub = new FetchUrlGrey();
FetchUrlSub.execute(urlSub);

String urlBaj = getUrl(puntoDestino,ubicParBaj,0,0);
FetchUrlGrey FetchUrlBaj = new FetchUrlGrey();
FetchUrlBaj.execute(urlBaj);
```

*Figura 84.* Captura de código que grafica ruta en bus.

**Seguimiento en tiempo real usando el sistema de GPS**

El proceso de seguimiento del usuario en tiempo real es producto de haber obtenido previamente el permiso de ubicación y que este tenga el GPS activado (requisitos para que la aplicación funcione, y que si no se tienen activos, no se puede ingresar a la misma). Esto se hace mediante el servicio de ubicación que ya tienen programado los mapas de Google, y se hace evidente mediante el punto celeste en el mapa que avanza a la vez que el usuario también lo hace.

## 4.5 Fase de Transición

Durante la fase de transición se realizan distintas actividades para generar la versión final del producto. En este caso particular, el producto que llega a la fase de transición es el desarrollado durante el proceso de construcción, y solamente se ha mostrado al público elegido la app.

Las fases del proceso de transición que se han tomado en cuenta en el presente proyecto, teniendo en cuenta al manifiesto ágil, han sido las siguientes;

### 4.5.1 Preparación de la versión beta a partir de la versión con capacidad operativa inicial

La versión operativa de la app es la que se construyó hasta la parte final de la fase de construcción, y es esta la que se muestra al público elegido.

### 4.5.2 Instalación y muestra de la versión beta

El proceso de instalación del aplicativo ha sido mediante la activación del modo desarrollador de los dispositivos de los usuarios, luego se conecta mediante un cable USB la laptop al móvil, y mediante Android Studio se procede a correr el aplicativo; proceso que a su vez instala la aplicación en el dispositivo móvil, sin embargo, se debe dejar en claro que solo 4 usuarios recibieron la instalación del aplicativo, debido a que el servidor Apache se encuentra en la laptop donde se ha desarrollado la app, por lo tanto la base

de datos y los servicios web también se encuentran allí, y es necesario estar en la misma red que la laptop para poder acceder a los servicios web.

Al resto de entrevistados se les mostró el aplicativo, se les explicó sus funciones y en base a eso dieron lo probaron y expresaron sus opiniones e impresiones.

### 4.5.3 Recolección y toma de decisiones a partir de la información procedente de los usuarios

Mediante el uso del aplicativo, los usuarios dejaron en claro sus impresiones las cuales fueron satisfactorias, y a su vez; indicaron ideas que de acuerdo a su experiencia y criterio eran necesarias para mejorar la app.

Luego de un proceso de filtrado, en el cual se analizó la complejidad de desarrollo de las distintas ideas, se tomó la decisión de implementar las siguientes:

- El botón “Menu” que se ubica en las vistas donde se ve los datos de las empresas, los datos de las rutas y donde se realiza la guía del viaje, debe funcionar y enviar al usuario al menú principal.
- Al momento de elegir el punto de origen del viaje (paso 1), también se debería poder elegir el mismo mediante la pulsación del del botón de GPS de los mapas de Google.
- En la vista donde se visualiza las rutas de las empresas, también se debería poder ver los paraderos por donde pasan.
- Sería útil poder visualizar el tiempo estimado de viaje.
- En la vista de “Guía de viaje”, se debería graficar solo el fragmento de ruta que se va a seguir en lugar de toda la ruta de la empresa.

### 4.5.4 Implementación de nuevas necesidades

El proceso de implementación de los requerimientos elegidos en esta fase; pone punto final al proceso de desarrollo de la presente versión del

producto, a continuación se detalla el proceso de desarrollo y los resultados que se lograron mediante la implementación de los mismos.

#### 4.5.4.1 Corrección del botón “Menú” en las interfaces de vista de empresas, rutas y guía de viaje

En las figuras 67, 69, 81 y 82; se aprecian en la parte superior derecha dichos botones, hasta la fase de transición no se había notado ese pequeño error, sin embargo ahora esos botones ya envían al usuario a la vista del Menú.

#### 4.5.4.2 Agregar el marcador del punto de origen de viaje al presionar el botón de GPS del mapa de Google para Android

Para realizar esta modificación en la vista donde se escoge el punto de origen, fue necesario comprender un poco más los mapas de Google para Android. Primeramente fue necesario agregar un nuevo “listener” al mapa que se tiene en dicha interfaz, luego se implementó la función que recibe este evento, que en este caso se llama “onMyLocationButtonClick”, y una vez dentro; se debe procesar el dato recibido mediante una variable de tipo “Location”, a partir de la cual se puede obtener las coordenadas. En la figura 85 se puede apreciar el código implementado.

```
mapM.setOnMyLocationButtonClickListener(new GoogleMap.OnMyLocationButtonClickListener() {
    @Override
    public boolean onMyLocationButtonClick() {

        LocationManager locatMan = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        if (ActivityCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.ACCESS_
                PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(getApplicat

        }
        Location location = locatMan.getLastKnownLocation(LocationManager.GPS_PROVIDER);

        if(location != null){
            double lati = location.getLatitude();
            double longi = location.getLongitude();

            LatLng latLng = new LatLng(lati,longi);
```

*Figura 85.* Captura del código usado para obtener la ubicación del usuario al presionar el botón del GPS del mapa de Google para Android.

Finalmente, el resultado se muestra en la figura 86

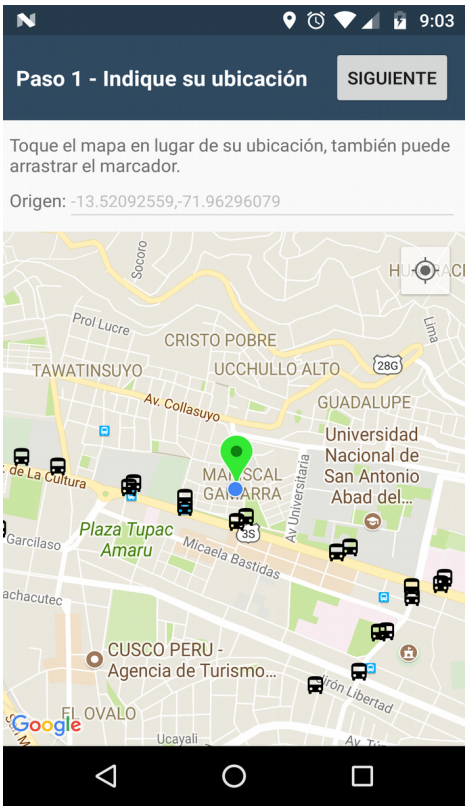



*Figura 86.* Al pulsar al botón del GPS, aparece el marcador encima de la ubicación del usuario.

#### 4.5.4.3 Mostrar los paraderos que utilizan las distintas rutas

En la figura 69 se muestra el listado de rutas del sistema, una vez se pulsa en alguna de ellas, se visualiza la interfaz mostrada en la figura 70; donde se grafica el trazado de ida como la de vuelta de dicha ruta. Es allí donde el usuario sugirió agregar el listado de paraderos que cubre dicha ruta, y es exactamente lo que se hizo.

Para llevar a cabo este requerimiento, se crearon al inicio dos funciones, una para ingresar los paraderos de la ruta de ida, y otra para los de vuelta, sin embargo se optimizó y al final se utiliza solamente una función. En la figura 87 se muestra dicha función.

```
public void mostrarParaderosDeRuta(int sentido){
    CPrdEnRutas prdEnR = new CPrdEnRutas(getApplicationContext());
    Object[] paraderos = prdEnR.getListaParaderoDeRuta(idRuta,sentido);

    final Marker[] arrayParaderosMapIda;
    arrayParaderosMapIda = new Marker[paraderos.length];

    for (int i = 0; i < paraderos.length; i++) {
        //convertir elemento de arreglo (Object)paraderos a CParaderos
        prdEnR = (CPrdEnRutas) paraderos[i];
        MarkerOptions paraPoint = new MarkerOptions();

        CParaderos prdo = new CParaderos(getApplicationContext());
        CParaderos parUnico = prdo.returnParaderoById(prdEnR.getTParaderos_id());


        paraPoint.position(new LatLng(parUnico.getLatitud(), parUnico.getLongitud()));
        paraPoint.title(parUnico.getNombre());
        paraPoint.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_directions_bus_black_18dp));
        paraPoint.visible(true);
        //
        //mapM.addMarker(new MarkerOptions().position(new LatLng(prd.getLatitud(), prd.getLongitud())).
        arrayParaderosMapIda[i] = mapRuta.addMarker(paraPoint);
    }
}
```

*Figura 87*. Captura de la función que inserta en el mapa los paraderos que cubre la ruta elegida, tanto de ida como de vuelta.

Las figuras 88 y 89 muestran el resultado tras la implementación del presente requerimiento.

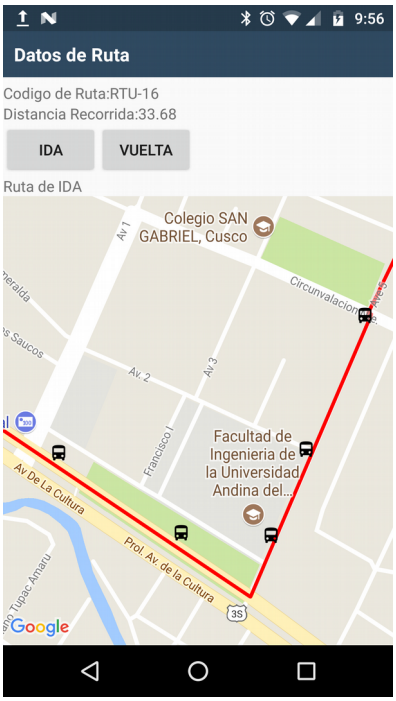



*Figura 88*. Se muestran los paraderos en la "ida" de la ruta RTU-16

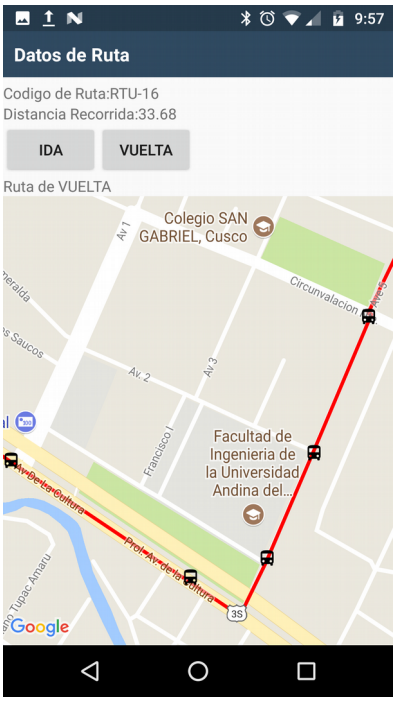



*Figura 89*. Se muestran los paraderos en la "vuelta" de la ruta RTU-16

#### 4.5.4.4 Implementar el tiempo estimado de viaje

Para implementar el tiempo estimado de viaje fue necesario realizar modificaciones en la base de datos, el backend y también en el aplicativo. Pero, ¿cómo se logró esto?, aumentando un campo más en la tabla TPrdEnRutas que almacena en orden el listado de paraderos que cubren las distintas rutas tanto de ida como de vuelta. Este campo almacena el tiempo estimado que se demora en viajar desde el paradero anterior hasta el paradero que se ingresa en ese momento, las figuras 14, 15, y 38 muestran cómo quedaron las bases de datos y la interfaz. Al ser un requerimiento especial del Ing. Emilio Palomino, que es uno de los dos dictaminantes del presente proyecto, se muestran los resultados en la fase de construcción.

Básicamente lo que se hace es, al momento de elegir la empresa que desea utilizar para realizar el viaje y obtener el paradero de subida y bajada, se obtiene el listado de paraderos entre el paradero de subida y bajada teniendo en cuenta el sentido y la ruta elegida, y se hace una sumatoria de los valores almacenados en el campo “tiempoEstimado” de la tabla TPrdEnRutas, obviando el valor del paradero de subida. El resultado final se puede apreciar en la figura 90 en el campo llamado “tiempo restante de viaje”.


*Figura 90.* Implementación del tiempo restante de viaje

#### 4.5.4.5 Graficar solamente el fragmento de ruta que se va a recorrer en la vista Guía de Viaje

Para poder llevar a cabo este requerimiento ha sido necesario modificar la base de datos, el backend y el frontend, sin embargo el resultado ha sido satisfactorio.

La base de datos fue modificada agregando el campo TParaderos_id en la tabla TCoordsRutas (ver figuras 14 y 15), este campo permite indicar al sistema si la coordenada ingresada tiene como elemento anterior una coordenada o un paradero, esto significa lo siguiente:

A medida que se van ingresando las coordenadas de una ruta; se va graficando la misma para poder visualizar como va quedando, en el mapa se puede apreciar los paraderos de la ciudad, y esto sirve para que al ingresar una coordenada y se grafique el “polilyne”, se pueda visualizar si el elemento anterior a la coordenada ingresada es otra coordenada u otro paradero. En caso no haya paraderos entre la nueva coordenada y la anterior, este valor quedara vacío, en caso haya un paradero, se debe elegir el paradero y el “id” del mismo automáticamente se agregará en el formulario. La figura 91 muestra este proceso, y reemplaza a la figura 32 en el backend.

Para poder modificar una coordenada de ruta, ha sido necesario modificar la interfaz que se muestra en la figura 35, se agrega el campo “IdParaderoAnterior” que se actualiza al dar click sobre algun paradero que se muestra en el mapa, y para borrar el campo se da click en “Limpiar”, en la figura 92 se muestra esta interfaz.

Ciudad del Cusco


*Figura 91.* Nueva interfaz para agregar puntos de rutas.


*Figura 92.* Nueva interfaz para modificar una coordenada de ruta

Una vez modificado el backend y la base de datos, se tuvo que modificar la app para aprovechar este nuevo dato. Para ello se crea una nueva función para genera “waypoints”, la cual lo que realiza es lo siguiente.

Primeramente extrae las coordenadas entre el paradero de subida y el de bajada, y es allí donde se aprovecha nuevo dato ingresado. Luego, en base a ello, se genera el listado de coordenadas entre el paradero de subida y bajada; que se deben ingresar en la gráfica de la ruta para que esta se realice de manera correcta. La figura 93 detalla este proceso.

```
public String generarWaypoints2(CParaderos pdoSubida, CParaderos pdoBajada, int idRuta, int sentido){
    String wayp = "";
    CRutas rut = new CRutas(getApplicationContext());
    CRutas ruElg = rut.getRutaById(idRuta);

    CCoordsRutas coor = new CCoordsRutas(getApplicationContext());
    //CCoordsRutas[] coords = coor.getCoordsDeRuta(idRuta,sentido);

    LatLng[] coords = coor.getCoordsEntreParaderosDeRuta(idRuta,sentido,pdoSubida,pdoBajada);

    if(coords==null){
        Log.i("coordNull","true");
    }else{
        Log.i("CoordsGuiaViaje",String.valueOf(coords.length));

        String cc = "";
        for(int i=0; i < coords.length; i++){
            cc = "cord"+(i+1);
            Log.i(cc,String.valueOf(coords[i]));
        }
    }

    //Log.i("coordsLength:",String.valueOf(coords.length));

    //generador de waypoints
    if(coords != null){
        for(int i=0;i<coords.length;i++){
            if(i<coords.length-1){
                wayp += coords[i].latitude+","+coords[i].longitude+"|";
            }else{
                wayp += coords[i].latitude+","+coords[i].longitude;
            }
        }
    }

    return wayp;
```

*Figura 93.* Generación de los “waypoints” para la gráfica de la ruta a seguir en la vista de “Guía de Viaje”.

Y una vez realizadas estas modificaciones, se puede apreciar el resultado en la figura 94.


*Figura 94.* Gráfico correcto de las rutas. Ya no se grafica toda la ruta y los paraderos de subida y bajada, sino solo el trazado necesario.

El PUDS también sugiere generar un planeamiento para la siguiente o siguientes versiónes, indicando los requerimientos que se deberían implementar en la misma. A continuación se indican algunos requerimientos que se podrían implementar a futuro:

- Mejorar el diseño de interfaces.
- Generar un ícono de “launcher” personalizado (el ícono de la app).
- Implementar alertas que indiquen al usuario que ya se encuentra próximo al paradero donde debe bajar.
- Actualizar los valores de tiempo de viaje en base a; datos recogidos de los viajes de otros usuarios, o tiempos que cambien dependiendo de la hora (el tiempo de viaje es diferente en hora punta que en otras horas).

- Recolectar datos de los usuarios para generar estadísticas, como por ejemplo; qué empresas son las más usadas, qué paraderos son los más usados, etc.
- Brindar a los usuarios la capacidad de almacenar sus lugares favoritos, para que no tengan que indicar en el mapa el origen y el destino cada vez que utilicen la app, sino puedan escoger estos puntos a partir de un listado guardado con los nombre que deseen ponerles.
- Brindar a los usuarios la capacidad de calificar el viaje que han realizado, para así obtener datos sobre qué empresas son las mejor y peor calificadas.
- Mejorar las interfaces para poder dar mantenimiento a las tablas de la base de datos en el backend.
- Agregar al sistema backend y frontend funciones para que los dueños de las empresas, puedan hacer seguimiento a sus buses y conductores, para que (valga la redundancia), entre otras cosas, los usuarios puedan calificar no solo a la empresa, sino a un bus y conductor en particular, generando de esta manera data útil para mejorar el servicio brindado.

# CAPÍTULO 5 - RESULTADOS

1. La presente tesis ha logrado el Desarrollo de una Aplicación Móvil para la Consulta de Rutas del Transporte Público de la ciudad de Cusco. El aplicativo, haciendo uso de los mapas de Google, es capaz de:
   - Indicar la ubicación del usuario en tiempo real.
   - Obtener las coordenadas desde dónde y hasta dónde desea viajar el usuario.
   - Sugerir empresas de transporte que pueden llevarlo(a) hacia su destino.
   - En caso no haya sugerencias o la distancia sea corta; es capaz de graficar en los mapas de Google la ruta que debe seguir a pie.
   - Graficar toda la ruta a seguir una vez el usuario elija una empresa, esto quiere decir; el camino a pie desde el punto de origen hasta el paradero en el que debe embarcarse, la ruta que seguirá en el bus, y el camino a pie que debe seguir desde el paradero de bajada hasta su destino.

2. Se desarrolló un sistema web (backend) para dar mantenimiento a la base de datos en línea mediante interfaces amigables. El backend es capaz de:
   - Dar mantenimiento a paraderos registrados.
   - Ubicar mediante interfaces amigables los distintos elementos en el mapa, como íconos, marcadores y líneas de gráfico de rutas.
   - Dar mantenimiento a empresas registradas.
   - Dar mantenimiento a rutas registradas, junto a todas las coordenadas que se utilizan para graficar rutas de ida y vuelta.
   - Realizar pruebas de gráficos de rutas.
   - Indicar qué paraderos son usados por qué rutas tanto de ida como de vuelta.

3. Se logró con éxito utilizar al Proceso Unificado de Desarrollo de Software junto al marco de trabajo Scrum para el desarrollo de la presente tesis.

4. El uso de Scrum sirvió para centrarse en el desarrollo del producto (backend y frontend) por sobre la documentación exhaustiva, además de integrar al cliente en

el proceso de desarrollo y tener una mayor respuesta a los cambios que se realizaron.

5. Se crearon y gestionaron dos bases de datos, una en MySql y la otra en SQLite. La primera es usada por el backend mientras que la segunda por el aplicativo móvil.

6. Se descargó los datos necesarios para el correcto funcionamiento de la aplicación desde la base de datos en línea hacia la base de datos del móvil; mediante el uso de Servicios Web.

7. Se programó el pedido del permiso de ubicación y uso del GPS del móvil en el aplicativo Android, además de explicar al usuario por qué estos son solicitados.

8. La investigación realizada permitió obtener la información necesaria para la utilización a medida de los mapas de Google tanto en Web como en Android.

9. El aplicativo para Android también sugiere el camino a pie a seguir desde el punto de origen hacia el paradero para embarcarse, y desde el paradero de bajada hacia el punto de destino.

10. El aplicativo es capaz de mostrar al usuario las distintas rutas existentes en la ciudad, los datos y el trazo de la misma.

11. El aplicativo es capaz de mostrar al usuario los datos y rutas de las distintas empresas de transporte de la ciudad.

12. Se logró desarrollar correctamente todos los requerimientos de la Lista de Producto.

13. Al crear una base de datos SQLite donde se almacenan casi todos los datos de la base de datos en línea (MySql), se ha logrado ahorrar los megas contratados

por los usuarios, depender menos del servidor, y evitar tiempos de espera innecesarios.

14. La manera en que se desarrolla y explica el proceso seguido para completar la siguiente tesis, permite a quien lo lea; entender de manera natural y ordenada todo lo realizado.

15. Se utilizaron tres lenguajes de programación (PHP, Javascript, Java), un lenguaje de Marcado (HTML), un lenguaje de etiquetado (XML), un lenguaje de estilos (CSS), dos gestores de bases de datos (SQLite y MySql), y un lenguaje manipulación, control y definición de datos (SQL), para el desarrollo de la presente tesis.

16. La manera en la que se desarrolla este proyecto (en la gestión de los textos que se muestran en la aplicación), brinda carta abierta a traducirlo en otros idiomas.

17. Se implementaron los requerimientos elegidos tras la fase de transición.

# CAPÍTULO 6 – DISCUSIÓN

El proceso de desarrollo en esta tesis, en comparación a la gran mayoría de tesis sobre Ingeniería de Sistemas donde la programación es parte fundamental, muestra de manera natural la manera en la que se desarrolla un sistema. Es común que primero se trate de definir todas las interfaces o las clases, pero esto no necesariamente se cumple al momento de programar. Lo que se hizo en este proyecto es cumplir con los requerimientos funcionales a nivel básico al momento de diseñar interfaces o programar clases en etapas tempranas de desarrollo, cubrir las necesidades básicas como los colores y la ubicación de los elementos, o tener en cuenta todas las variables que se van a necesitar en base a las columnas creadas en la base de datos; generando de esta manera los constructores básicos y lo mismo con los “getters” y “setters”.

Las interfaces han sido creadas de manera que no representen una complicación al momento de diseñar y programar, mantener la simpleza ha sido la clave. Si bien todo se puede mejorar (como en cualquier sistema), el objetivo principal no ha sido crear interfaces visualmente atractivas, sino interfaces que cumplan con su función y ubiquen a los elementos de modo que la navegación sea sencilla.

La tesis del bachiller Francesco Galiano Abanto utiliza SCRUM como medotología de desarrollo de software, y concuerdo con él en el hecho que Scrum se adecúa mejor que el PUDS en proyectos pequeños. Documentar exhaustivamente se aplica eficientemente a grandes proyectos donde es necesario tener todo perfectamente documentado, sin embargo en este tipo de proyectos puede agregar mucho tiempo de manera innecesaria tomando el cuenta la envergadura del proyecto. Por otro lado a comparación de la tesis del bachiller Abanto, no se utilizan procedimientos almacenados debido a que la complejidad de la base de datos en línea y móvil no es grande, una vez los datos en el backend son ingresados y probados; estos quedan prácticamente congelados, lo mismo sucede con los datos de la base de datos móvil.

La tesis del bachiller Jorge Fabrisio Cornejo Aramayo, gestiona su proyecto mediante una metodología basada en el PMBook y utiliza RUP como metodología, lo cual se evidencia en la documentación que realiza. No concuerdo en utilizar RUP para documentar a menos que necesario u obligatorio. Los casos de uso de entienden muy bien mediante historias de usuario, incluso hasta mejor, ya que en lugar de seguir una estructura rígida, siguen una estructura más natural, donde el usuario puede escribir con sus propias palabras sus necesidades, lógicamente en caso el proyecto sea muy grande, es mejor documentar de manera exhaustiva. Por otro lado, concuerdo con el bachiller Cornejo en el uso de JSON, si bien cada persona podría generar el formato que desee, es mejor seguir los parámetros pre-establecidos para que si el sistema en algún momento es mantenido por otro desarrollador, esto sea mucho más sencillo de hacer.

La tesis de los bachilleres Mariela A. Alfonso R. y Jesus E. Segnini R. utiliza PUDS como metodología de desarrollo de software. En ella muestran claramente la fase de inicio, elaboración, construcción y transición, sin embargo la cantidad de páginas que tienen su tesis supera las 300. La documentación que realizan es bastante significativa, lo cual; teniendo en cuenta que son dos personas, no llega a ser un problema tan grande, sin embargo, no concuerdo con documentar tan exhaustivamente un producto, (a menos que sea necesario u obligatorio). Gran parte de la documentación que se realiza puede ir en comentarios dentro del código, es más, comentando lo necesario es posible entender cómo funciona el sistema, obviamente se debe complementar la lectura de comentarios con el entendimiento de las líneas de código, pero si surge un cambio, especialmente si es importante, generará mucho trabajo al desarrollarlo y documentarlo.

A comparación de las tesis consultadas (no solamente las listadas en esta tesis), no se utilizó Windows como sistema operativo. El uso de Ubunbu (16.04) fue un reto al inicio, sin embargo luego se vuelve algo natural y además brinda mejor entendimiento de cómo funciona un sistema operativo; a medida que se va dando solución a los distintos requerimientos que aparecen. Por otro lado, el aplicativo para Android y el sistema web basado en PHP no genera problemas al momento de desarrollarlo en Linux.

En cuanto a Moovit, en la presente tesis no se logra la complejidad con la que cuenta este aplicativo, sin embargo el avance es significativo en cuanto a la investigación sobre uso de mapas de Google; especialmente al momento de graficar, utilizar servicios como el GPS, y la adquisición de conocimiento para poder incluir esta tecnología en un aplicativo móvil.

# CONCLUSIONES

1. La creación de un backend para dar mantenimiento a las diferentes tablas; ha sido extremadamente útil especialmente en etapas tempranas de desarrollo, ya que hubiera sido bastante lento mediante código SQL.

2. Mostrar en el backend los gráficos de las rutas y la ubicación de paraderos; para poder probar el ingreso de datos, ha sido determinante para el control de los datos ingresados.

3. Brindar al usuario la opción de ingresar los puntos de origen y destino del viaje, ha permitido al usuario tener control sobre el aplicativo, además de abrir las puertas a excelentes ideas de mejoras para el futuro.

4. Mostrar en la app el listado de empresas y rutas, el gráfico de ida y vuelta de las rutas, y la la ubicación del usuario, ha sido de gran utilidad para los usuarios que probaron y vieron el funcionamiento de la app.

5. Sugerir empresas a los usuarios para que estos puedan llegar a su destino; ha sido un gran punto a favor de la app, ya que en la ciudad no existe otra aplicación en la actualidad que tenga esa función.

6. Graficar la ruta que el usuario va a viajar, y el camino que debe tomar tanto desde su ubicación inicial al paradero de subida como del paradero de bajada a su destino, ha sido una excelente idea, ya que brindó al usuario la comodidad de saber qué paraderos debe utilizar con exactitud y dónde estos se encuentran, y de saber la ruta a seguir durante todo su viaje.

7. Mostrar al usuario su ubicación durante todo el viaje ha sido crucial para la aceptación de la app por parte de los usuarios.

8. Es posible utilizar PUDS y Scrum como metodología de desarrollo y marco de trabajo al momento de realizar un proyecto de este tamaño.

9. Integrar correctamente a los clientes en el proceso de desarrollo es una práctica que debería realizarse constantemente, para guiar el proceso de desarrollo hacia donde se desea que vaya, recibir constante retroalimentación, y cambiar lo que sea necesario en el momento adecuado.

# RECOMENDACIONES

Se recomienda investigar otras opciones aparte de los mapas de Google para desarrollar este tipo de aplicativos, como por ejemplo los Open Street Maps.

Se recomienda utilizar una metodología ágil para el desarrollo de proyectos de similar envergadura, para tener mayor respuesta a los cambios y no sentirse agobiados por la documentación.

Se recomienda Libre Office como herramienta ofimática. Es gratuito y ofrece muy buenas características como reemplazo a Microsoft Office.

Se recomienda conocer el PUDS antes de utilizar una metodología ágil. Esto permite entender a mayor profundidad el proceso de desarrollo de software, para que luego (una vez se haya comprendido) se utilice una metodología ágil conociendo los pormenores del proceso de desarrollo.

Se recomienda explicar a los usuarios la razón por la que se piden los distintos permisos al programar un aplicativo para Android, para que de esa manera sepan por qué se solicitan, y denieguen permisos que consideren innecesarios.

Se recomienda continuar con el proceso de desarrollo de este tipo de aplicativos móviles, para cuidar el medio ambiente fomentando el uso del transporte público por sobre el particular, y obtener suficiente data con la cual poder solicitar a las autoridades encargadas y a las empresas de transporte sistemas de transporte más eficientes, seguras y agradables para los usuarios.

Se recomienda desarrollar sistemas backend para el mantenimiento de las bases de datos mediante interfaces amigables.

Se recomienda migrar el sistema a multiplataforma, además de estilizar mejor los íconos utilizados y colores, para evitar confusiones.

## GLOSARIO

**AsyncTasks:** Se refiere a tareas asíncronas.

**Click (orientado a eventos y listeners):** Es el evento que ocurre cuando damos click al mouse o ratón.

**Dpi:** Dots per inch. En español significa puntos por pulgada. Es la cantidad de puntos disponibles dentro de una pantalla por pulgada.

**Dragend:** Es un evento que se activa al momento de dar click sobre un elemento, arrastrarlo a una nueva posición y soltarlo.

**EditText:** Son cuadros de texto que se pueden editar. Son utilizados en Android Studio.

**Evento (orientado a listeners):** Es determinada actividad que ocurre en algún momento dado, por ejemplo arrastrar un elemento, dar click, apretar alguna tecla, etc.

**Hash**: Son algoritmos que permiten crear, a partir de una palabra clave como por ejemplo una contraseña, cadenas de caracteres alfanuméricas que posteriormente pueden ser usados para almacenar contraseñas, firmar documentos digitales, etc.

**Hilo (thread):** Es el hilo que crea una aplicación al ser utilizada. En caso se programe paralelamente o exista concurrencia, es posible crear más de un hilo.

**Intents:** Son los llamados que realiza Android para dejar de usar la interfaz actual y pasar a usar una nueva.

**Labels:** Son etiquetas que se utilizan dentro de un programa para dar algún de tipo de información.

**Launcher (icono de lanzamiento de apps):** Es el ícono que se utiliza en Android para abrir una aplicación.

**Listener:** Dentro de la programación, se encargan de escuchar distintos eventos y reaccionar si estos se activan. Un listener se activa después de un evento predeterminado.

**Loggeo:** Actividad que se realiza al momento de ingresar a un sistema informático mediante un usuario y contraseña. Por ejemplo; al momento de validar el usuario y contraseña en la página de Facebook.

**Polilyne:** Es la línea de color que se grafica dentro de un mapa de Google para indicar una ruta.

**Pop up:** Se refiere a las ventanas emergentes que se abren mientras se hace uso de un navegador web.

**Rooteado**: Se refiere a un dispositivo Android que ha sido alterado para que el usuario tenga acceso como “root” (raiz). Esto significa que puede utilizar el móvil con privilegios que usualmente el usuario común no tiene, como por ejemplo poder acceder libremente a las base de datos guardadas en el teléfono.

**TextView:** Son cuadros de texto que el usuario no puede editar. Son utilizados en Android Studio.

**Timer**: Se refiere a un contador, que una vez cumple su ciclo permite que se ejecuten instrucciones.

**URL:** Es la ruta que se utiliza para acceder a cualquier página web desde un navegador, o en algunos casos, la ruta para consumir servicios web.

## REFERENCIAS

About My Sql s.f. Visto el 27 de Mayo del 2018 de: https://www.mysql.com/about/

About SQLite. Visto el 27 de Mayo del 2018 de: https://www.sqlite.org/about.html

Afonso R., Mariela A. & Jesus E., Segnini R.. (2009). Desarrollo de un sistema automatizado bajo entorno web para el control de la programación académica en la Universidad de Oriente núcleo de Anzoátegui. (Tesis) Universidad de Oriente núcleo de Anzoátegui. Disponible en: http://ri.bib.udo.edu.ve/bitstream/123456789/1098/1/Tesis.DESARROLLO%20DE%20UN%20SISTEMA%20AUTOMATIZADO%20BAJO%20ENTORNO%20WEB.pdf Último día de visita: 27 de Mayo del 2018

Agregar un mapa de Google con un marcador a tu web. Visto el 27 de Mayo del 2018 de: https://developers.google.com/maps/documentation/javascript/adding-a-google-map

Android Studio. s.f. Página oficial. Visto el 27 de Mayo del 2018 de: https://developer.android.com/studio/index.html?hl=es-419

Android 4.4 KitKat. Visto el 27 de Mayo del 2018 de: https://www.android.com/history/#/kitkat

Android 5.0 Lollipop. Visto el 27 de Mayo del 2018 de: https://www.android.com/history/#/lollipop

Android 6.0 Marshmallow. Visto 27 de Mayo del 2018 de: https://www.android.com/history/#/marshmallow

Android 7.0 Nougat. Visto el 27 de Mayo del 2018 de: https://www.android.com/versions/nougat-7-0/

API. 2007. API, Interfaz de Programación de Aplicaciones. Visto el 27 de Mayo del 2018 de: http://www.elwebmaster.com/referencia/api-interface-de-programacion-de-aplicaciones

Bahit E. (s.f). POO y MVC en PHP. Visto 27 de Mayo del 2018 de: http://www.duea.umss.edu.bo/documentos/Plan_car_184799.pdf

Bases de datos relacionales. Visto el 27 de Mayo del 2018 de: https://www.ibm.com/support/knowledgecenter/es/SSEPGG_8.2.0/com.ibm.db2.udb.doc/admin/c0004099.htm

Bell C. (2012) Expert MySql. Estados Unidos de América. Apress.

Berzal F. (s.f) Relaciones entre clases: Diagramas de Clases UML. Universidad de Granada. Visto el 27 de Mayo del 2018 de: http://elvex.ugr.es/decsai/java/pdf/3C-Relaciones.pdf

Boletín Estadístico de Turismo 2015. Dircetur. Visto el 27 de Mayo del 2018 de: http://www.dirceturcusco.gob.pe/wp-content/uploads/2017/07/BOLETIN-ESTADISTICO-2015-Final.pdf

Compatibilidad con diferentes pantallas. 2018. Visto el 27 de Mayo del 2018 de: https://developer.android.com/guide/practices/screens_support

Conozca más sobre la tecnología Java. s.f. Visto el 27 de Mayo del 2018 de: https://www.java.com/es/about/

Contratos de Concesión de Rutas. Visto el 27 de Mayo del 2018 de: http://www.cusco.gob.pe/gerencia-de-transito-vialidad-transporte/contratos-de-concesion-de-rutas/

Cornejo Aramayo, Jorge Fabrisio. (2013). Análisis, diseño e implementación de una Aplicación para administrar y consultar avisos clasificados para tabletas Android. (Tesis) PUCP. Disponible en:
http://tesis.pucp.edu.pe/repositorio/handle/123456789/4786
Último día de visita: 27 de Mayo del 2018.

Coronel E. (2010). PHP Profesional. Lima – Perú. Macro E.I.R.L

Creating Scrum. Visto el 27 de Mayo del 2018 de: https://www.scrum.org/about

Crypt. Visto el 27 de Mayo del 2018 de:
http://php.net/manual/es/function.crypt.php

Database Management Systems. s.f. Visto el 27 de Mayo del 2018 de: http://searchsqlserver.techtarget.com/definition/database-management-system

Datatypes In SQLite Version 3. Visto el 27 de Mayo del 2018 de:
https://sqlite.org/datatype3.html

Diagramas de Clases en UML (s.f.) Universidad San Martín de Porres. Visto el 27 de Mayo del 2018 de:
http://www.usmp.edu.pe/publicaciones/boletin/fia/info67/UML.pdf

Entorno de Desarrollo Integrado. 2011. Visto el 27 de Mayo del 2018 de: http://programacion-laura.blogspot.pe/2011/08/entorno-de-desarrollo-integrado-ide.html

Estado de la Población Peruana 2015. Visto el 27 de Mayo del 2018 de: https://www.inei.gob.pe/media/MenuRecursivo/publicaciones_digitales/Est/Lib1251/Libro.pdf

Ferré J. & Sánchez I. (s.f.) Desarrollo Orientado a Objetos con UML. Facultad de Informática – Universidad Politécnica de Madrid.

Galiano Abanto, Francesco Fabrizzio, (2015). Scrum como metodología de desarrollo de software para el control de caja en el restaurante Il Giardino del Cusco. (Tesis). Universidad Andina del Cusco.

Gimp. s.f. Página oficial. Visto el 27 de Mayo del 2018 de: https://www.gimp.org/

Google Maps API – Location Picker Example. Visto el 27 de Mayo del 2018 de: http://thisinterestsme.com/google-maps-api-location-picker-example/

Google Maps Directions API. Visto el 27 de Mayo del 2018 de: https://developers.google.com/maps/documentation/directions/usage-limits?hl=Es

Grant. A & Owens. M. (2010) The Definitive Guide to SQLite 2nd edition. Estados Unidos de América. Apress.

Guía breve de Tecnologías XML. s.f. Visto el 30 de Noviembre del 2017 de: https://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML

Guía para uso de citas y bibliografía (APA). Visto el 27 de Mayo del 2018 de: http://www.poli.edu.co/sites/default/files/guiausocitasbibliografiaapa.pdf

Hardware. RAE. Visto el 27 de Mayo del 2018 de: http://dle.rae.es/?id=K1Wwkf7

Hernández R., Fernández C., Baptista P. (2010) Metodología de la Investigación. Perú. Mc Graw Hill

Historia de PHP. s. f. Visto el 27 de Mayo del 2018 de: http://php.net/manual/es/history.php.php

Insercion de tablas y figuras en normas APA. Visto en 27 de Mayo del 2018 de: http://normasapa.com/insercion-de-tablas-y-figuras/

Introducción a SQL. s.f.. Visto el 27 de Mayo del 2018 de: https://www.w3schools.com/sql/sql_intro.asp

Lenguaje. RAE . Visto el 27 de Mayo del 2018 de: http://dle.rae.es/?id=N7BnIFO

Libre Office s.f. Página oficial. Visto el 30 de Noviembre del 2017 de: http://www.libreoffice.org/discover/libreoffice/

Metodología. RAE. Visto el 30 de Noviembre del 2017 de: http://dle.rae.es/?id=P7eTCPD

Metodologías de Desarrollo de Software. s.f. Universidad de Murcia. Visto el 27 de Mayo del 2018 en: http://www.um.es/docencia/barzana/IAGP/Iagp2.html

Moovit página oficial. s.f. Visto el 27 de Mayo del 2018 de: https://www.company.moovitapp.com/es

Moovit Play Store. s.f. Visto el 27 de Mayo del 2018 de: https://play.google.com/store/apps/details?id=com.tranzmate&hl=es_419

MySql Workbench (s.f.) MySql. Visto el 27 de Mayo del 2018 de: https://www.mysql.com/products/workbench/

Netbeans s.f. Página Oficial. Visto el 27 de Mayo del 2018 de: https://netbeans.org/community/releases/82/

Nolasco J. (2013) Desarrollo de Aplicaciones Móviles con Android. Lima – Perú. Macro EIRL.

Paradigma. RAE. Visto el 27 de Mayo del 2018 de: http://dle.rae.es/?id=RpXSRZJ

Paradigmas de Programación, 2011. Departamento de Informática de la Universidad de Valladolid. Visto el 27 de Mayo del 2018 de: https://www.infor.uva.es/~cvaca/asigs/docpar/intro.pdf

PHP 7.0 (and 5.6) on Ubuntu. 2016. Visto el 27 de Mayo del 2018 de: https://lornajane.net/posts/2016/php-7-0-and-5-6-on-ubuntu

¿Qué es el Kernel y para qué sirve?, 2014. Visto el 27 de Mayo del 2018 de: http://www.androidpit.es/que-es-kernel-para-que-sirve

¿Qué es GNU/Linux? Visto el 27 de Mayo del 2018 de: https://www.debian.org/releases/stable/mips/ch01s02.html.es

Qué es HTML. s.f. Visto el 27 de Mayo del 2018 de: https://codigofacilito.com/articulos/que-es-html

¿Qué es PHP?. s.f. Visto el 27 de Mayo del 2018 de: http://php.net/manual/es/intro-whatis.php

Qué es un navegador Web. s.f. Visto el 27 de Mayo del 2018 de: http://www.informatica-hoy.com.ar/aprender-informatica/Que-es-un-navegador-web.php

Rutas en PDF. Visto el 5 de mayo del 2017 http://sial.cusco.gob.pe/index.php?accion=verElemento&idElementoInformacion=2415&idformula=

Schwaber & Sutherland, 2016 La Guía de Scrum. Visto el 27 de Mayo del 2018 de: http://www.scrumguides.org/download.html

Servidores Web. Visto el 27 de Mayo del 2018 de: http://www.ub.edu/stat/docencia/bioinformatica/introbiocomputacio/ServidoresWeb/ServidoresWeb-Concepto_Configuracion_Uso.pdf

Sistemas Operativos. (s.f). Visto el 27 de Mayo del 2018 de: http://www.areatecnologia.com/sistemas-operativos.htm

Software. RAE. Visto el 27 de mayo del 2018 de: http://dle.rae.es/?id=YErlG2H

SQLite. Visto el 27 de Mayo del 2018de: https://www.sqlite.org/

StarUML s.f. Página oficial de StarUML, Visto el 27 de Mayo del 2018 de: http://staruml.io/

Takeuchi and Nonaka, The roots of Scrum. Visto el 27 de Mayo del 2018 de: https://www.scruminc.com/takeuchi-and-nonaka-roots-of-scrum/

Tarifas y planes. Visto el 27 de Mayo del 2018 de: https://developers.google.com/maps/pricing-and-plans/#details

The Android story. Visto el 27 de mayo del 2018 de: https://www.android.com/history/

The New New Product Development Game. Visto el 27 de Mayo del 2018 de: https://hbr.org/1986/01/the-new-new-product-development-game

Ubuntu, Instale y domine el sistema Linux mas veloz y fácil de Usar, 2012. Buenos Aires – Argentina. Sevagraf.

Stellman A., Greene J.. (2015) Learning Agile. Estados Unidos de América. O'Reilly.

Tam J., Vera G. y Oliveros R.. (2008) Tipos, métodos y estrategias de investigación científica. Visto el 27 de Mayo del 2018 de: http://www.imarpe.pe/imarpe/archivos/articulos/imarpe/oceonografia/adj_modela_pa-5-145-tam-2008-investig.pdf

Zakas, N. (2012) Professional Javascript for Web Developers, Third Edition. Indianapolis, Indiana – Estados Unidos de América. John Wiley & Sons, INC.