Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

CommonGo

Aplicación Android que determina rutas similares entre usuarios para poder compartir vehículo y gastos

Alumno: Jorge Plaza Jiménez

Tutores: Miguel Ángel Martínez Prieto
Aníbal Bregón Bregón

"No siempre que pude, hice,

Pero cuando hice, di lo mejor de mí"

"Dime y lo olvido,

enséñame y lo recuerdo,

involúcrame y lo aprendo"

BENJAMIN FRANKLIN

"La mejor forma de predecir el futuro,

es implementarlo"

DAVID HEINEMEIER HANSSON

Gracias a todos los que han compartido conmigo estos años de carrera, a los que habéis formado parte de esta transición de mi vida. A esta transición también, que me ha dado la oportunidad de conocer gente excepcional que no podría haber conocido de ninguna otra forma.

Gracias a mis amigos, a su apoyo, a todos los que me habéis ayudado, aunque sólo sea un poco, a seguir adelante con el proyecto, yo siempre estaré ahí para vosotros.

Gracias a mi padre, y al *"¿Y qué te queda para acabar el tfgtfc, el trabajo ese?"* de mi madre, qué grandes sois.

Y en especial, quiero dar las gracias a mis dos tutores de mi TFG, Miguel Ángel Martínez Prieto y Aníbal Bregón Bregón. Gracias a los dos por tutorizarme, por ayudarme cuando lo necesitaba, por simplificarme la vida cuando yo solo me la complicaba, y sobre todo por aguantarme.

Resumen

CommonGo es un proyecto que persigue reducir las grandes concentraciones de vehículos que se aglutinan en las entradas, salidas y calles de las urbes con un gran número de habitantes y afluencia. A su vez, y derivado del principal objetivo, permitirá reducir los niveles de contaminación, ruido, estrés, problemas de aparcamiento y un largo etcétera de problemas que existen a día de hoy en las grandes metrópolis de todo el mundo.

Este es el motivo principal por el cual se ha desarrollado una herramienta capaz de poner en contacto a aquellas personas que tienen, en su trayecto rutinario al trabajo, universidad, etc., una ruta común con otras personas en su misma situación. Esta aplicación permite que estas personas tomen conciencia de que pueden compartir coche para ahorrar gastos, tiempo de conducción, incluso conocer gente nueva.

En conclusión, CommonGo busca reducir el tráfico en las carreteras. Actuando de medio con el cual las personas que tienen trayectos en común sepan unas de otras y puedan realizar sus trayectos diarios ahorrando vehículo, costes y reduciendo gastos e impactos medioambientales.

Palabras clave: compartir coche, reducir emisiones, viajes, ahorrar combustible, comunidad, medio ambiente.

Abstract

CommonGo is a project witch purpose is to reduce the large concentrations of vehicles that usually collapse entrance, exits and streets of cities with large number of inhabitants and affluence. In addition, and derived from the main objective, this application will reduce the levels of pollution, noise, stress, parking problems and many other issues existing nowadays in large cities around the world.

This is the main reason why a tool like this has been developed, a tool capable to join those people who have in their route to work, university, etc., a *common* route with other people in the same situation. This tool allows these people to become aware that they can share a car to save expenses, manage time, even meet new people.

In conclusion, CommonGo seeks to reduce traffic issues, and to be the way to connect those people who have common routes, and make them know each other, so they can realise their daily trips avoiding using their cars, saving costs and reducing environmental costs and impacts.

Keywords: car sharing, reduce emissions, travel, trip, save fuel, community, environment.

ÍNDICE DE CONTENIDOS

1 ÍNDICES 11

1.1 ÍNDICE DE IMÁGENES 11
1.2 ÍNDICE DE TABLAS 12

2 INTRODUCCIÓN 15

2.1 MOTIVACIÓN DEL PROYECTO 16
2.2 OBJETIVOS DEL PROYECTO 16
2.3 ALCANCE DEL PROYECTO 17
2.3.1 REGLAS DE NEGOCIO 18
2.4 CONTENIDO DEL CD-ROM 18
2.5 ORGANIZACIÓN DEL DOCUMENTO 19

3 ESTADO DEL ARTE 21

4 GESTIÓN DEL PROYECTO 27

4.1 METODOLOGÍA 27
4.2 PLANIFICACIÓN 27
4.3 ESTIMACIÓN DE LOS COSTES ECONÓMICOS 31
4.3.1 PUNTOS DE FUNCIÓN (PF) 31
4.3.2 COCOMO 37
4.3.3 PRESUPUESTO 39
4.3.4 COSTE ECONÓMICO REAL 41

5 ANÁLISIS DEL SISTEMA 47

5.1 DESCRIPCIÓN DE ACTORES 47
5.2 REQUISITOS DE USUARIO 48
5.2.1 DIAGRAMA DE CASOS DE USO 48
5.2.2 ESPECIFICACIÓN DE CASOS DE USO 49
5.3 REQUISITOS NO FUNCIONALES 59
5.4 REQUISITOS DE INFORMACIÓN 59
5.5 DICCIONARIO DE DATOS 60
5.6 DISEÑO CONCEPTUAL 62

6 DISEÑO SOFTWARE 65

6.1 ARQUITECTURA LÓGICA 65
6.1.1 ARQUITECTURA LÓGICA DEL SISTEMA 65

6.1.2 Arquitectura Lógica de la aplicación 66
6.2 Arquitectura Física 68
6.3 Diagrama de Clases (Diseño) 68
6.3.1 Servidor 69
6.3.2 Aplicación 70
6.4 Diagrama(s) de secuencia 71
6.5 Diseño Lógico 74
6.6 Diseño de interfaz 75

7 IMPLEMENTACIÓN 87

7.1 Descripción Técnica 87
7.2 Requisitos de Hardware y Software 88
7.3 Herramientas Empleadas 88
7.4 GIS 90
7.4.1 ¿Por qué utilizar PostGIS? 90
7.5 Creación de la base de datos en PostGIS 91
7.6 Cálculo de distancias en PostGIS 92
7.7 Estructura del proyecto 93
7.7.1 Servidor 93
7.7.2 Aplicación 95
7.8 Detalles de implementación 96
7.8.1 Menús 96
7.8.2 Verificación de permisos 98
7.8.3 Comunicación con Servicios Rest 98
7.8.4 Procesamiento por lotes 100
7.8.5 Comunicación con Google Geocode API 103

8 PRUEBAS 105

8.1 Pruebas de caja blanca 105
8.2 Pruebas de caja negra 105

9 MANUALES 117

9.1 Manual de instalación del servidor 117
9.1.1 Instalación de NetBeans 117
9.1.2 Instalación de la Base de Datos 119
9.2 Manual de instalación de la aplicación móvil 121
9.3 Manual de usuario 122

10 CONCLUSIONES 125

10.1 Trabajos futuros 126

11 BIBLIOGRAFÍA 127

1 ÍNDICES

1.1 ÍNDICE DE IMÁGENES

Imagen 1 - Diagrama de Características 17
Imagen 2 - Interfaz de BlaBlaCar 21
Imagen 3 - Interfaz de Amovens 22
Imagen 4 - Interfaz de Carpooling 23
Imagen 5 - Interfaz de Waze 23
Imagen 6 - Interfaz Bluemove 24
Imagen 7 - Captura de pantalla de Carpling 25
Imagen 8 - Diagrama de Gantt - Estimación temporal del proyecto 30
Imagen 9 - Diagrama de Gantt de la duración real del proyecto 43
Imagen 10 - Diagrama de casos de uso 49
Imagen 11 - Diseño conceptual 62
Imagen 12 - Arquitectura lógica del sistema 65
Imagen 13 - Arquitectura lógica de la aplicación 66
Imagen 14 - Arquitectura física 68
Imagen 15 - Diagrama de Clases del servidor 69
Imagen 16 - Diagrama de Clases de la aplicación 70
Imagen 17 - CU-01 - Diagrama de secuencia 71
Imagen 18 - CU-02 - Diagrama de secuencia 72
Imagen 19 - CU-12 - Diagrama de secuencia 73
Imagen 20 - Diseño Lógico 74
Imagen 21 - Captura final login 82
Imagen 22 - Captura final registro 82
Imagen 23 - Captura final Tablón 83
Imagen 24 - Captura final Conversaciones 83
Imagen 25 - Captura final Perfil 84
Imagen 26 - Captura final Viajes 84
Imagen 27 - Captura final Ajustes 85
Imagen 28 - Captura final Mapas 85
Imagen 29 - Descarga SDK 1 117
Imagen 30 - Descarga JDK 2 118
Imagen 31 - Descarga Netbeans 118
Imagen 32 - Descarga PostgreSQL 119
Imagen 33 - Instalación PostgreSQL 1 119
Imagen 34 - Instalación PostgreSQL 2 120
Imagen 35 - Instalación PostGIS 1 120
Imagen 36 - Instalación PostGIS 2 121
Imagen 37 - Instalación CommonGo 121
Imagen 38 - Manual de usuario CommonGo 1 122
Imagen 39 - Manual de usuario CommonGo 2 122
Imagen 40 - Manual de usuario CommonGo 3 123
Imagen 41 - Manual de usuario CommonGo 4 123

1.2 ÍNDICE DE TABLAS

Tabla 1 - Comparativa entre CommonGo y otras herramientas del mercado 26
Tabla 2 - Estimación de las iteraciones del desarrollo 29
Tabla 3 - Estimación de la distribución de horas por rol en cada iteración 30
Tabla 4 - Número de entradas 33
Tabla 5 - Número de salidas 33
Tabla 6 - Número de consultas externas 34
Tabla 7 - Número de ficheros lógicos internos 34
Tabla 8 - Número de ficheros lógicos externos 34
Tabla 9 - Pesos de los dominios de información según complejidad 34
Tabla 10 - Cálculo de los Puntos de Función No Ajustados 35
Tabla 11 - Cálculo del Factor de Ajuste 35
Tabla 12 - Estimación de las líneas de código por lenguaje de programación 36
Tabla 13 - Constantes para cada modelo COCOMO 38
Tabla 14 - Factores de coste del esfuerzo 38
Tabla 15 - Presupuesto de la mano de obra 40
Tabla 16 - Presupuesto del Hardware 40
Tabla 17 - Presupuesto del Software 41
Tabla 18 - Desviaciones en la duración del proyecto real 42
Tabla 19 - Tiempo real por rol en cada iteración en el desarrollo del proyecto 43
Tabla 20 - Coste real de la mano de obra 44
Tabla 21 - Coste real del Hardware 44
Tabla 22 - Coste real del Software 45
Tabla 23 - ACT-01 - Usuario general 47
Tabla 24 - ACT-02 - Usuario identificado 47
Tabla 25 - CU-01 - Registrarse 50
Tabla 26 - CU-02 – Identificarse 51
Tabla 27 - CU-03 - Cerrar sesión 51
Tabla 28 - CU-04 - Ver perfil 52
Tabla 29 - CU-05 - Modificar perfil 52
Tabla 30 - CU-06 - Añadir/Cambiar foto de perfil 53
Tabla 31- CU-07 - Ver viajes 53
Tabla 32 - CU-08 - Ver ruta 54
Tabla 33 - CU-09 - Modificar viaje 54
Tabla 34 - CU-10 - Eliminar viaje 55
Tabla 35 - CU-11 - Grabar ruta 55
Tabla 36 - CU-12 - Guardar Ruta 56
Tabla 37 - CU-13 - Solicitar conversación 56
Tabla 38 - CU-14 - Decidir conversación 57
Tabla 39 - CU-15 - Ver conversaciones 57
Tabla 40 - CU-16 - Ver conversación 58
Tabla 41 - CU-17 - Escribir mensaje 58
Tabla 42 - Entidad places 60
Tabla 43 - Entidad users 60
Tabla 44 - Entidad trips 61
Tabla 45 - Entidad routes 61
Tabla 46 - Entidad conversations 61
Tabla 47 - Entidad messages 61
Tabla 48 - Entidad commons 62
Tabla 49 - Entidad notifications 62
Tabla 50 - Diseño de la pantalla de login 75
Tabla 51 - Diseño de la pantalla de registro 76

Tabla 52 - Diseño de la pantalla principal ____ **77**
Tabla 53 - Diseño de la pantalla de mensajes ____ **78**
Tabla 54 - Diseño de la pantalla de perfil ____ **79**
Tabla 55 - Diseño de la pantalla de mapas ____ **80**
Tabla 56 - Diseño de la pantalla de ajustes ____ **81**
Tabla 57 - Requisitos Hardware y Software ____ **88**
Tabla 58 - Prueba 01 – Registro ____ **106**
Tabla 59 - Prueba 02 - Registro fallido ____ **107**
Tabla 60 - Prueba 03 - Identificación ____ **107**
Tabla 61 - Prueba 04 - Identificación incorrecta ____ **108**
Tabla 62 - Prueba 05 - Recibir nuevas notificaciones ____ **108**
Tabla 63 - Prueba 06 - Visualizar tablón ____ **109**
Tabla 64 - Prueba 07 - Visualizar viajes ____ **109**
Tabla 65 - Prueba 08 - Visualizar conversaciones ____ **110**
Tabla 66 - Prueba 09 - Visualizar mensajes ____ **110**
Tabla 67 - Prueba 10 - Enviar mensaje ____ **111**
Tabla 68 - Prueba 11 - Visualizar perfil ____ **111**
Tabla 69 - Prueba 12 - Editar perfil ____ **112**
Tabla 70 - Prueba 13 - Editar foto de perfil ____ **112**
Tabla 71 - Prueba 14 - Ver ajustes ____ **113**
Tabla 72 - Prueba 15 - Grabar ruta ____ **113**
Tabla 73 - - Prueba 16 - Mostrar mapas ____ **114**
Tabla 74 - Prueba 17 - Ver ruta ____ **114**
Tabla 75 - Prueba 18 - Guardar ruta ____ **115**
Tabla 76 - Prueba 19 - Editar ruta ____ **115**
Tabla 77 - Prueba 20 - Solicitar conversación ____ **116**
Tabla 78 - Prueba 21 - Decidir conversación ____ **116**

2 INTRODUCCIÓN

La necesidad de desplazamiento está a la orden del día, actualmente desplazarse a diario es prácticamente obligatorio, tanto que hay personas que necesitan desplazarse grandes distancias para llegar a su puesto de trabajo, universidad, etc. Parece razonable pensar que algo tan simple como el transporte público pueda cubrir esta necesidad, y así es en la mayoría de los casos. Pero a la hora de la verdad muchas de estas personas necesitan hacer uso de su vehículo personal ya sea por la incompatibilidad de horarios con el transporte público, la incompatibilidad de la ruta, la gran cantidad de transbordos, el alcance o el tiempo de viaje del mismo.

Para estas personas es frecuente encontrarse, sobre todo en grandes ciudades, con grandes retenciones en el trayecto de ida y/o de vuelta, con problemas de aparcamiento, dolores de cabeza, estrés, tener que salir de casa mucho antes para llegar a la misma hora e incluso tarde. Sin contar el aumento de polución y gastos extra que esto genera: combustible, recursos administrativos y activos del gobierno, entre otros. Como intento de solventar este problema, en algunas ciudades de habilita un carril adicional denominado para Vehículos de Alta Ocupación (VAO) que por lo general es para 2 o más ocupantes, pero aun así los problemas de tráfico no se ven reducidos. Y es que si echamos un vistazo a los vehículos que forman parte del atasco, nos daremos cuenta de que, en prácticamente todos los casos, hay una única persona por vehículo y que por tanto no pueden hacer uso del carril adicional.

Para estas personas, resultaría muy cómodo viajar con un acompañante que les permitiese cumplir el requisito de tener 2 o más ocupantes en el vehículo. Si pensamos en que muchos de esos conductores hacen la misma ruta prácticamente todos los días, resulta lógico pensar que muchos de ellos no sólo tengan esa ruta en común sino también un destino similar, e incluso una zona de origen de trayecto cercana. Esto hace posible que dichos conductores tengan la posibilidad de compartir vehículo para beneficiarse no sólo del uso del carril VAO (si lo hubiera) sino también de otras ventajas como tener a alguien con quien poder entablar una conversación, descansar de conducir de vez en cuando, ahorrar combustible, ahorrarse problemas de aparcamiento, ahorrarían tiempo de viaje, etc. El problema reside en que, casi con seguridad, estos conductores no se han parado a pensarlo, no se han planteado que esto pudiera ser posible o simplemente no conocen a nadie con quien poder llevarlo a cabo de manera periódica.

La idea de este proyecto nace de este vacío: una aplicación que consiga que todos estos conductores puedan unirse en su viaje rutinario y beneficiarse de todas las ventajas mencionadas. Esta aplicación trataría de ponerles en contacto haciéndoles saber que tienen a su alcance esa posibilidad.

2.1 MOTIVACIÓN DEL PROYECTO

A día de hoy las aplicaciones que se podrían considerar similares existentes en el mercado se centran principalmente en:

- **Viajes esporádicos** donde un usuario que quiere viajar busca viajes publicados por un conductor y contacta con él, si el conductor acepta entonces viajan juntos y el viajante abona el precio acordado por el viaje al conductor. Ej. Blablacar, Amovens, Carpooling.
- **Viajes rutinarios** donde el usuario puede informarse de alertas de tráfico, bloqueos, incidencias medioambientales, entre otros, para reducir su tiempo de viaje. Ej. Waze.
- **Viajes de corta distancia** una vez se está dentro de la ciudad mediante el uso de una red de coches dispuestos para ese fin. Ej. Bluemove
- **Viajes compartiendo taxi** entre los usuarios de la aplicación. Ej. Carpling.

En ninguno de los casos las herramientas se centran en reducir la cantidad de vehículos que ya realizan un determinado trayecto cada día. Y es que si sólo la mitad de los conductores compartiesen el coche con otro conductor, el tráfico ya se vería reducido en una cuarta parte.

Aprovechando el vacío existente en cuanto a herramientas de este tipo se refiere, se propone esta idea, que sin duda ayudaría a reducir tanto los atascos, como las emisiones, el gasto público y privado, etc.

2.2 OBJETIVOS DEL PROYECTO

Como objetivo principal, el software que se quiere desarrollar consiste en un automatismo que permita conseguir una aplicación Android capaz de determinar, a través de los algoritmos necesarios, qué usuarios realizan rutas parecidas para poder brindarles la posibilidad de compartir trayecto y beneficiarse de las ventajas que eso conlleva.

Este es el objetivo prioritario, aunque antes se han de conseguir algunos objetivos que le preceden. Para conseguir implementar satisfactoriamente esta herramienta, se tendrá que alcanzar unos sub-objetivos asociados al proyecto, tales como:

- Obtener rutas frecuentes de un usuario.
- Comparar total y parcialmente rutas de usuarios.
- Sugerir potenciales compañeros de viaje.

Una vez conseguido el objetivo principal, la aplicación tendrá, a su vez, una serie de efectos positivos derivados del mismo que aportan gran valor tanto ecológico como económico a la herramienta. Entre otros serían:

- Descongestionar el tráfico producido por la aglomeración de vehículos. Principalmente en las grandes ciudades y en horas punta.
- Reducir la contaminación y emisiones de CO_2.
- Reducir problemas de aparcamiento.

2.3 ALCANCE DEL PROYECTO

Esta herramienta está pensada principalmente para cualquier persona que realice de manera frecuente un trayecto, sufra o no el problema de aglomeración de vehículos en la carretera, ya que gracias a este software no sólo se puede beneficiar de este hecho, sino también ahorrarse los problemas que suponen buscar aparcamiento una vez se ha llegado al destino, disfrutar de un viaje rutinario en compañía, contaminar menos, etc. Con algunas restricciones, puesto que sólo podrían disfrutar de estas aplicaciones aquellas personas que necesiten desplazarse una distancia mínima.

La aplicación se centrará en un primer momento en comparar rutas entre usuarios, tarea que deberá realizar el servidor, aunque es esperable que en un futuro un usuario pueda buscar en un momento puntual un trayecto que necesite realizar y no disponga de transporte. Imitando en este caso la funcionalidad que ya tienen algunas aplicaciones actuales en el mercado y aumentando el valor de esta herramienta.

El siguiente diagrama refleja un resumen de las características detalladas que definen la aplicación:

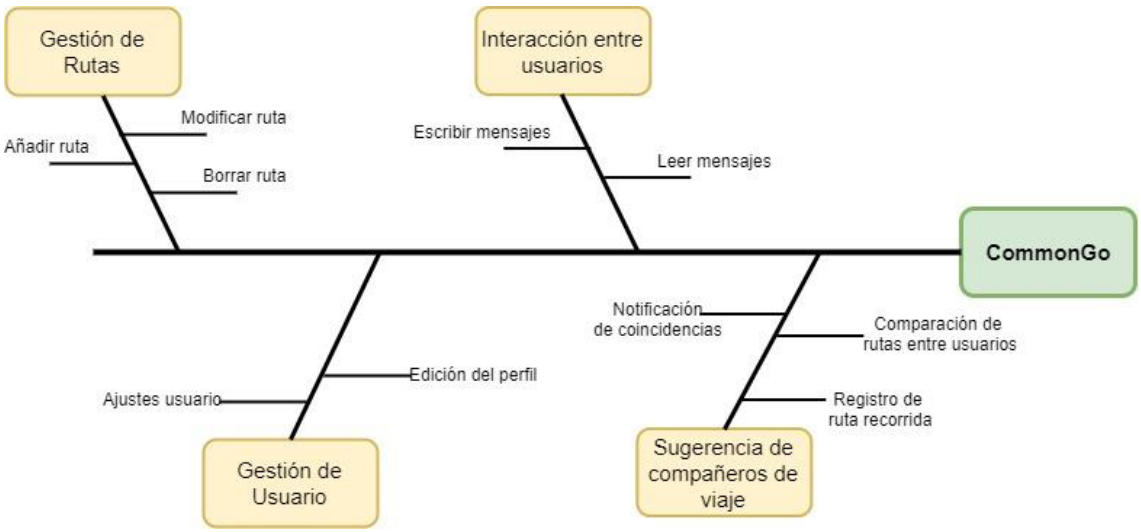


Imagen 1 - Diagrama de Características

Como se puede ver en la Figura, el sistema está dotado de 4 características principales:

- La característica encargada de la **gestión del perfil** del usuario. incluyendo ajustes parametrizados de la aplicación, cambio de datos personales o foto.

- La **interacción entre usuarios**, que le confiere al usuario la oportunidad de hablar con el usuario con el cual tiene una ruta en común.

- **Gestión de rutas**, característica encargada de añadir, modificar y borrar rutas.

- **Sugerencia de compañeros** de viaje, la característica clave de la aplicación, ya que es la encargada de realizar las aproximaciones entre rutas de usuarios y el sistema de recomendaciones.

2.3.1 REGLAS DE NEGOCIO

Las reglas de negocio son aquellas políticas o regulaciones que definen algún aspecto de la herramienta, se describen a continuación:

RN-1. El tratamiento y almacenamiento de los datos sensibles se realizará de acuerdo a la Ley Orgánica de Protección de Datos (LOPD).

RN-2. EL uso de la herramienta será gratuito para todos los usuarios de la misma.

2.4 CONTENIDO DEL CD-ROM

Se entregará un disco que contenga el código fuente de la aplicación, el código fuente del servidor y una copia de esta memoria. El contenido de dicho CD-ROM se ha estructurado de la siguiente manera:

- **Diagramas**: directorio donde se encuentran los diagramas utilizados en esta memoria, en alta resolución.
- **Memoria**: directorio con la copia en formato PDF de este documento.
- **Proyecto**: en este directorio se encuentran los archivos del proyecto:
  - **Código fuente**: directorio con los códigos fuentes de la herramienta
    - **Servidor**: directorio con el código fuente del servidor.
    - **Aplicación**: directorio con el código fuente de la aplicación.
  - **BBDD**: directorio con los ficheros necesarios para crear la base de datos y unos datos mínimos de prueba para su uso.

2.5 ORGANIZACIÓN DEL DOCUMENTO

En esta sección se va a resumir la restructura del documento de forma que sea de más fácil lectura para el lector:

- **Capítulo 1. Índices:** en esta sección encontraremos los índices para todas las tablas e ilustraciones que el usuario se puede encontrar a lo largo del documento.

- **Capítulo 2. Introducción:** en esta sección encontraremos una presentación al proyecto implementado. Junto con una introducción a la problemática de la cual ha surgido la idea y el nido de mercado que pretende cubrir CommonGo.

- **Capítulo 3. Estado del Arte:** en esta sección se hace una recopilación de las herramientas existentes en el mercado similares a la presentada y se comparan con ella.

- **Capítulo 4. Gestión del proyecto:** en esta sección se aborda el presupuesto del proyecto y el coste real del mismo.

- **Capítulo 5. Análisis del sistema:** esta sección detalla el dominio del proyecto, requisitos y casos de uso.

- **Capítulo 6. Diseño software:** en esta sección encontraremos la arquitectura lógica y física del servidor de la herramienta y de la aplicación Android.

- **Capítulo 7. Implementación:** en esta sección del documento se describe la implementación que se ha llevado a cabo para la elaboración de CommonGo

- **Capítulo 8. Pruebas:** en esta sección se detallan las pruebas que se han realizado para asegurarse de que la herramienta tiene un buen funcionamiento.

- **Capítulo 9. Manuales:** en esta sección se incluyen instrucciones de instalación, configuración y uso de todos los componentes necesarios para poner en funcionamiento la herramienta.

- **Capítulo 10. Conclusiones:** en esta sección se hace una valoración del trabajo realizado, impresiones una vez acabado, sensaciones y posibles ampliaciones del proyecto.

3 ESTADO DEL ARTE

En la actualidad existen una serie de herramientas software similares a la que se va a desarrollar, cuyo estado actual es el siguiente:

- **BlaBlaCar[1]:** herramienta destinada a compartir coche de manera puntual. El usuario introduce los datos del viaje que desea realizar y la aplicación le informa si hay otro usuario (conductor) que ha publicado el viaje y admite pasajeros a cambio de una retribución fijada. En esta herramienta el usuario registrado como conductor tiene que introducir los datos de su vehículo, así como sus datos personales. Al ser conductor, lo siguiente que tiene que hacer es publicar uno o varios viajes. Publicar un viaje consiste en establecer el origen del viaje, el destino, la fecha, hora del trayecto y cuánto dinero quiere recibir por cada pasajero. De esta manera cualquier otro usuario registrado (como pasajero o conductor) puede buscar trayecto filtrando por alguno de esos parámetros. Una vez encuentre uno que se ajusta a sus necesidades puede contactar con el conductor para más información, o simplemente reservar una plaza en el coche para el trayecto. En ese momento el conductor puede aceptar o rechazar al pasajero. Si acepta, la herramienta le concede los datos de contacto a ambas partes. Más tarde del viaje tendrán que confirmar el viaje ya que los pagos se hacen a través de la propia herramienta.

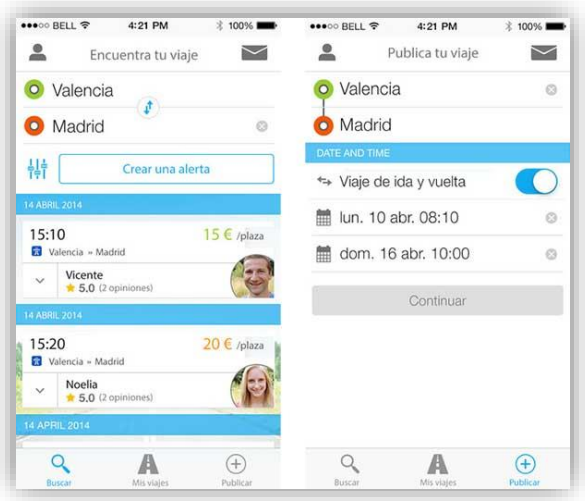


Imagen 2 - Interfaz de BlaBlaCar

En la *Imagen 2* vemos una captura de pantalla de la aplicación, donde vemos que se puede buscar viajes por origen y destino, y muestra las personas que van a hacer esa ruta, indicando también lo que cobra cada persona por el mismo viaje y otros datos de interés para el usuario. La radical diferencia que se pretende que CommonGo tenga con BlaBlaCar, es que los viajes estén destinados a algo más

[1] https://www.blablacar.es

rutinario, donde ningún usuario tiene que pagar nada a nadie porque se podrían turnar para llevar cada día el coche de uno de ellos.

- **Amovens[2]:** similar a la anterior. En este caso la empresa, de momento, no se lleva comisión por viaje, excepto un porcentaje de un seguro a todo a riesgo que es obligatorio suscribir. Con la diferencia de que un usuario también puede alquilar un coche para realizar un trayecto, o poner en alquiler su vehículo para que alguien se lo alquile, es decir, ofrece un servicio de alquiler entre particulares. Es el particular el que decide el precio del alquiler. También ofrece otro servicio, el de renting de coches FreeCar, que permite a los usuarios alquilar un coche totalmente nuevo durante varios meses. Además, ofrece un sistema de coches compartidos dirigido a empresas, administraciones públicas, etc.

Imagen 3 - Interfaz de Amovens

En la *Imagen 3* se muestra la similitud entre esta aplicación con la anterior, se puede seleccionar el origen y destino, el tipo de viaje y la aplicación muestra los usuarios que van a realizar ese trayecto y muestra las tarifas de cada uno.

- **Carpooling[3]:** del estilo a las dos anteriores. Con una búsqueda por filtros donde se puede seleccionar viajar sólo entre mujeres, con no fumadores, etc. o un radio de salida cercano al usuario. En este caso el usuario únicamente quiere buscar viaje, no será necesario que se registre. El tipo de usuarios que hacen uso de esta plataforma son jóvenes universitarios. Existe la opción de dar de alta un viaje, escogiendo entre ofrecer un vehículo o buscar uno, para un viaje puntual o para una ruta que se pretende repetir varias veces. Hacer uso de esta aplicación es gratuito, y además ofrece una calculadora a través de la cual la propia plataforma realiza un cálculo de los costes que puede suponer cada viaje.

[2] https://amovens.com
[3] https://play.google.com/store/apps/details?id=com.carpooling.android.es&hl=es

Imagen 4 - Interfaz de Carpooling

En la *Imagen 4* se muestra una captura de la aplicación en cuestión, donde muestra diferentes filtros como el radio de distancia entre el punto de salida y llegada del usuario con respecto al del conductor.

- **Waze**[4]**:** herramienta desarrollada para reducir el tiempo de viaje y beneficiarse de los conocimientos de la comunidad. Los propios usuarios informan en tiempo real sobre policía, estado del tráfico, accidentes, peligros de la carretera, precio del combustible en las gasolineras, etc. No está destinada a compartir coche y gastos en sí, aunque sí a reducir las emisiones y atascos.

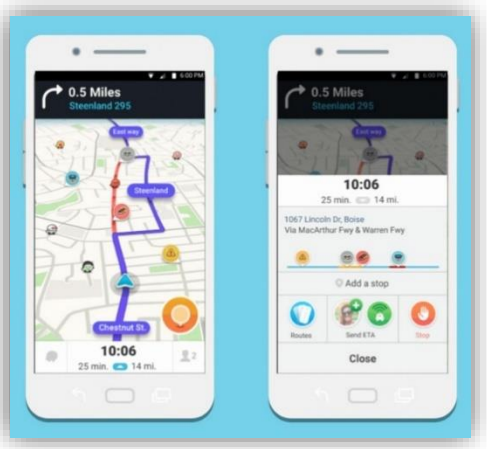


Imagen 5 - Interfaz de Waze

[4] https://www.waze.com/es/

Los propios usuarios de la herramienta actualizan los mapas constantemente en base a evitar cualquier obstáculo para la conducción.

La *Imagen 5* muestra los eventos que muestra esta aplicación, desvíos en tiempo real para evitar atascos, o planificados por el usuario, localización de atascos, de gasolineras, etc.

- **Bluemove[5]:** pone vehículos a disposición del usuario el tiempo que lo necesite, gracias a una red de coches (también eléctricos) dispuestos por la ciudad 24h al día y pagando sólo por lo que los usan. El usuario no tiene que preocuparse del mantenimiento, parking, combustible, etc. ya que los gastos se limitan a la tarifa.

  Por el momento está limitado a solamente unas pocas ciudades (grandes) y sin poder salirse de ellas. También ha añadido tarifas para viajes de larga distancia (más de cuatrocientos kilómetros) y por días, enfocados a viajes principalmente, no a trayectos rutinarios.

  En Madrid se ha implementado un período de prueba de un año sin compromiso alguno ni tampoco gastos fijos, incluyendo únicamente el coste del servicio escogido.

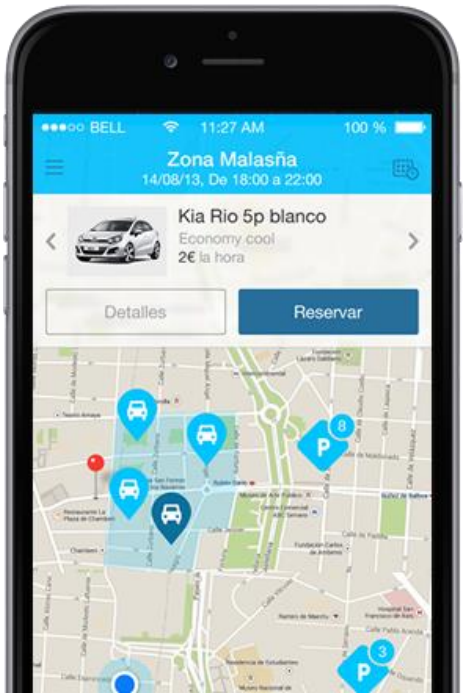


Imagen 6 - Interfaz Bluemove

En la *Imagen 6* se muestra una captura de la aplicación donde muestra información de los puntos donde se encuentran estos coches, donde marcando uno se puede reservar en tiempo real.

- **Carpling[6]:** herramienta destinada a la compartición de taxi, tren e incluso parking entre usuarios. Tiene el objetivo de reducir el número de coches en las calles, pero el inconveniente de depender que el taxi se llene para que empiece a salir realmente rentable añadido a que en trayectos largos el precio se dispare. Algo similar pasa al compartir tren, para que realmente salga rentable económicamente hay que reservar una mesa completa. Este portal de internet busca un mayor sostenimiento del medioambiente reduciendo las emisiones mediante la reducción de tráfico. Busca coincidencias tanto en origen y destino como a lo largo del

[5] https://bluemove.es/es
[6] https://www.carpling.com/es/

trayecto, para que surja la posibilidad de encontrar compañeros de viaje durante éste.

Los márgenes temporales que establece esta página se calculan en base a unos parámetros de tráfico fluido y de una velocidad establecida legalmente por el Código de Circulación, pero si el conductor prevé mayor tráfico o que circulará a otra velocidad distinta a la establecida podrá modificar estos márgenes.

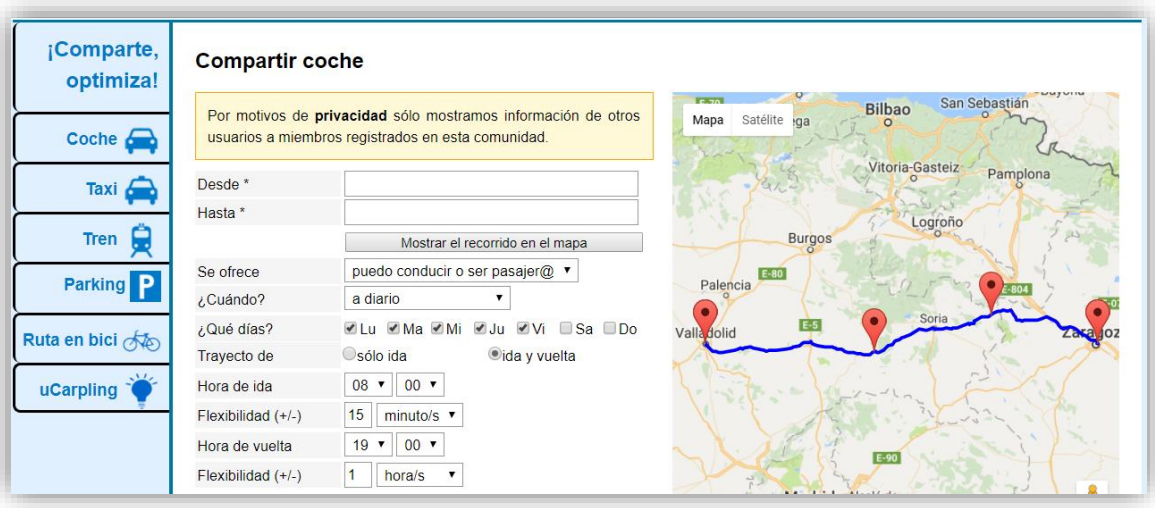


Imagen 7 - Captura de pantalla de Carpling

En la *Imagen 7* se muestra un pantallazo de la herramienta, donde ofrece diferentes filtros según el tipo de viaje, ya se quiera hace en coche, taxi, tren, en bici o simplemente compartir parking. En el caso de la imagen los filtros pasan por origen, destino, día y hora, hasta flexibilidad, tipo de trayecto, etc.

La funcionalidad que más valor añade a CommonGo es la que permite generar rutas automáticamente a partir de los desplazamientos del usuario, y a partir de ahí sugerir usuarios que realizan la misma ruta o similar. Además, no hace falta realizar ningún pago ya que se trata de dos usuarios que van a compartir coche, por lo que puede llevarlo un día cada uno o realizar un acuerdo por separado a la herramienta, lo que resulta mucho más cómodo para el usuario de la aplicación.

Además, se trata de una herramienta capaz de adaptarse al usuario, puesto que es válida tanto en trayectos largos como cortos, permite compartir coche a partir de trayectos ya realizados o buscar trayectos ya realizados por usuarios por lo que vale para trayectos rutinarios como para desplazamientos puntuales.

A modo de ilustración, se presenta la siguiente tabla comparativa entre las herramientas actuales y la que se propone:

4 GESTIÓN DEL PROYECTO

Este apartado está destinado a la realización de la planificación del proyecto, justo antes de realizar el análisis del mismo. Aquí, se describe la metodología que se va a seguir, la planificación temporal en la que se va a dividir el proyecto, una estimación presupuestaria y, finalmente, el coste real de la realización del proyecto.

4.1 METODOLOGÍA

Para el desarrollo del software se hará uso del modelo incremental, el cual nos proporciona un feedback constante durante el proceso. Además, no disponemos de la especificación completa del sistema que vamos a desarrollar, por lo que nos ayuda afrontar esta incertidumbre inicial.

El modelo incremental hace uso de ciclos o iteraciones que se espera que sean de un mes de duración cada uno. Para aplicar este modelo, se realizar un análisis inicial del problema y un diseño preliminar y por tanto la primera iteración se espera que sea más larga y tenga una mayor carga de análisis que el resto. A medida que se vayan sucediendo las iteraciones se irá reduciendo el peso del análisis e irán aumentando las cargas de desarrollo e implementación.

En cada iteración se llevan a cabo una serie de procesos, a saber:

- Análisis: modificaciones en el análisis del software.
- Diseño: cambios en el diseño.
- Implementación: desarrollo de los cambios y modificaciones.
- Pruebas: testeo de las implementaciones realizadas.

4.2 PLANIFICACIÓN

Como ya se ha expuesto, el desarrollo de esta herramienta está planteado siguiendo un proceso incremental, ajustado al tiempo disponible para realizar el proyecto. Dividiéndolo en 5 iteraciones de, aproximadamente, un mes cada una. Y, en cada una de ellas se llevarán a cabo 4 ó 5 etapas:

- **Análisis**: en esta etapa se intenta modelar lo que debería hacer la herramienta para satisfacer aquellas necesidades de los usuarios y finalidades para las cuales se está llevando a cabo el desarrollo de la herramienta. Para ello se lleva a cabo un proceso de especificación de requisitos, que se determinan en gran medida teniendo en cuenta las necesidades del usuario que la aplicación debe satisfacer.

- **Diseño**: durante esta etapa se especifica la construcción gráfica de la herramienta enfocada a las funcionalidades que la aplicación que se han determinado en la fase anterior. De manera bastante general y únicamente especificando las interacciones entre los componentes a diseñar.

- **Implementación**: es la fase en la que las dos etapas anteriores toman forma y se comienza a desarrollar el proyecto.

- **Pruebas**: esta etapa consiste en comprobar que los componentes desarrollados en la fase anterior funcionan correctamente y según a lo especificado previamente.

- **Documentación**: consiste en dejar constancia de lo que se ha llevado a cabo en las anteriores etapas. En este momento se genera la documentación del desarrollo del software, con el objetivo de tener una información de lo que hace el sistema y poder realizar modificaciones en futuras versiones de la manera más fácil posible.

Las iteraciones que se han llevado a cabo para este proyecto son las siguientes:

- **Iteración 1 (1 feb – 25 feb):** en la primera iteración la etapa de análisis ocupa casi toda la iteración y es donde el analista tendrá mayor carga de trabajo, y las etapas de implementación y pruebas son muy reducidas o nulas:
    - Análisis: del 1 al 20 de feb.
    - Diseño: del 21 al 24 de feb.
    - Implementación y pruebas: 25 de feb.

- **Iteración 2 (26 feb – 25 mar):** la etapa de análisis es más liviana pero todavía supera la mitad del tiempo estimado:
    - Análisis: del 26 de feb al 12 de mar.
    - Diseño: del 13 al 19 de mar.
    - Implementación: del 20 al 23 de mar.
    - Pruebas y documentación: del 24 al 25 de mar.

- **Iteración 3 (26 mar – 22 abr):** la etapa de diseño cobra fuerza y se empieza a dedicar más tiempo a la implementación:
    - Análisis: del 26 al 30 de mar.
    - Diseño: del 31 de mar al 8 de abr.
    - Implementación: del 9 al 16 de abr.
    - Pruebas: del 17 al 19 de abr.
    - Documentación: del 20 al 22 de abr.

- **Iteración 4 (23 abr – 26 may):** la etapa de análisis quedará relegada prácticamente a un segundo plano junto con la de diseño mientras que la etapa de implementación se hace más pesada:
    - Análisis: del 23 al 24 de abr.
    - Diseño: del 25 al 27 de abr.
    - Implementación: del 28 de abr al 5 de may.
    - Pruebas: del 6 al 12 de may.
    - Documentación: del 13 al 26 de may.

- **Iteración 5 (27 may – 14 jul):** en esta última etapa pierde importancia la fase de Análisis y cobran protagonismo las de pruebas y sobre todo documentación:
    - Diseño: del 27 al 31 de may.
    - Implementación: del 1 al 9 de jun.
    - Pruebas: del 10 al 17 de jun.
    - Documentación: del 18 de jun al 14 de jul.

En la tabla siguiente se reflejan las anteriores iteraciones, mostrando también la duración estimada de cada iteración, y cada fase dentro de estas iteraciones:

Nombre	Duración	Inicio	Fin
Iteración 1	18d	**01/02/2017**	**24/02/2017**
Análisis	14d	01/02/2017	20/02/2017
Diseño	3d	21/02/2017	23/02/2017
Implementación y pruebas	1d	24/02/2017	24/02/2017
Iteración 2	20d	**27/02/2017**	**24/03/2017**
Análisis	11d	27/02/2017	13/03/2017
Diseño	4d	14/03/2017	17/03/2017
Implementación	3d	21/03/2017	23/03/2017
Pruebas y documentación	1d	24/03/2017	24/03/2017
Iteración 3	20d	**27/03/2017**	**21/04/2017**
Análisis	4d	27/03/2017	30/03/2017
Diseño	5d	03/04/2017	07/04/2017
Implementación	6d	10/04/2017	17/04/2017
Prueba	3d	18/04/2017	20/04/2017
Documentación	1d	21/04/2017	21/04/2017
Iteración 4	25d	**24/04/2017**	**26/05/2017**
Análisis	1d	24/04/2017	24/04/2017
Diseño	3d	25/04/2017	27/04/2017
Implementación	6d	28/04/2017	05/05/2017
Prueba	5d	08/05/2017	12/05/2017
Documentación	10d	15/05/2017	26/05/2017
Iteración 5	34d	**29/05/2017**	**13/07/2017**
Diseño	4d	29/05/2017	01/06/2017
Implementación	6d	02/06/2017	09/06/2017
Prueba	5d	12/06/2017	16/06/2017
Documentación	19d	19/06/2017	13/07/2017

Tabla 2 - Estimación de las iteraciones del desarrollo

La forma gráfica de ilustrar la anterior tabla es mediante un Diagrama de Gantt, que nos ayuda a comprender de un solo vistazo, cómo se van a desarrollar las iteraciones.

Podemos observar la ausencia de tareas paralelas, lo que claramente reduce la eficiencia del ciclo. Esto es debido a que el equipo de trabajo está formado por una sola persona, y tanto Analista, como Programador y Documentalista son la misma, y esto imposibilita el

desarrollo paralelo de tareas que en una empresa al uso serían fácilmente desempeñables de manera concurrente.

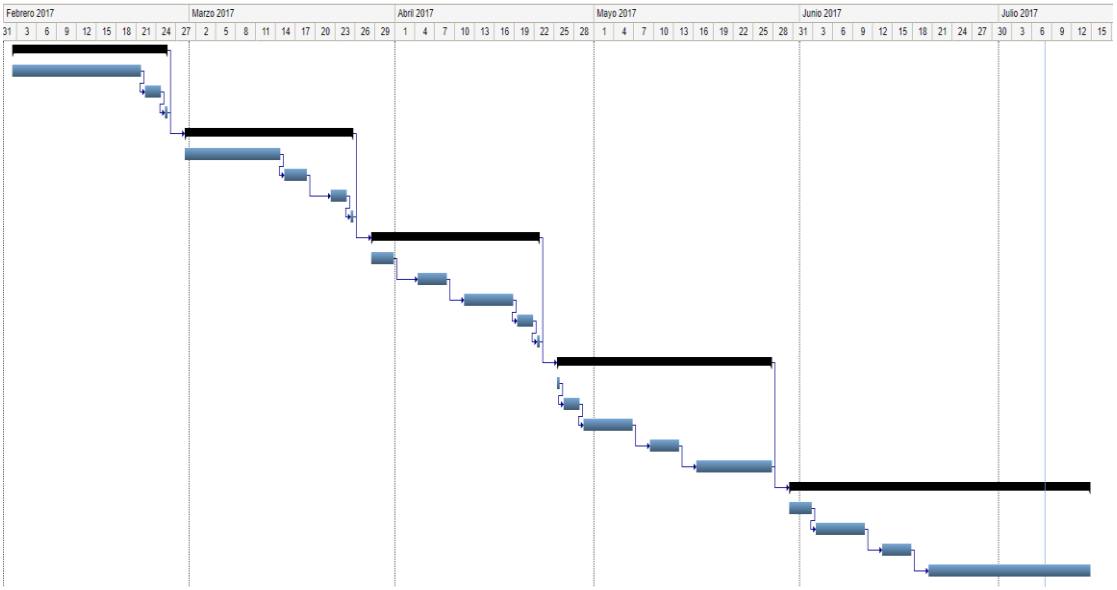


Imagen 8 - Diagrama de Gantt - Estimación temporal del proyecto

Podemos resumir el tiempo (en horas, tomando 8 laborables por día) estimado de cada miembro del personal para desarrollar la herramienta a partir de la anterior tabla:

Se tendrá en cuenta que:

- El **analista** realizará la fase que le corresponde, es decir, la de análisis.
- El rol de **programador** tendrá asignadas las fases de diseño, implementación y pruebas. La decisión de asignarle al programador también la fase de diseño es debida a la poca carga de trabajo que, en principio, puede tener la interfaz de la aplicación móvil y la herramienta en el servidor.
- El rol de **documentalista** tendrá asignada la fase de documentación.
- Los días en los que coinciden dos fases (por ejemplo *pruebas y documentación* en la iteración nº2) se reparten las horas de esos días en proporciones iguales).

Rol	***Iteración***					***Total (h)***
	1	***2***	***3***	***4***	***5***	
Analista	112	88	32	8	0	**240**
Programador	32	60	112	112	120	**436**
Documentalista	0	4	8	80	152	**244**

Tabla 3 - Estimación de la distribución de horas por rol en cada iteración

El total de horas previstas para la finalización del proyecto son 920 horas, (en total 23 semanas) de las cuales 436 son para el programador, que tiene una carga de trabajo significativamente mayor a la de los otros dos roles.

4.3 ESTIMACIÓN DE LOS COSTES ECONÓMICOS

Para llevar a cabo el cálculo del coste económico del desarrollo de la herramienta, primero se han de realizar una serie de estimaciones. En este caso las estimaciones se van a realizar calculando las líneas de código necesarias para la realización del proyecto mediante el cálculo de los puntos de función. Después, nos ayudaremos de las líneas de código obtenidas para, mediante el método de COCOMO, estimar el presupuesto económico del desarrollo total e íntegro del proyecto y, a continuación de esta aproximación, se hará un cálculo del coste real del proyecto.

4.3.1 Puntos de Función (PF)

El método de puntos de función basa sus estimaciones en la asignación de una puntuación a cada componente del proyecto software en función del grado de complejidad de su implementación. Sumando estas puntuaciones obtendremos un número de puntos de función que se multiplicará por un determinado factor que dependerá de las características del proyecto, obteniendo así un valor ajustado de puntos de función finales.

A partir de dicha puntuación es posible calcular los costes que conlleva la realización del proyecto. Bien sea en coste temporal, bien en número de personas necesarias para llevarlo a cabo, bien en número de líneas de código. Esta última también estará en función de la complejidad del lenguaje de programación en la que se implemente el software.

Para llevar a cabo el método de puntos de función se debe de seguir un determinado procedimiento:

1. Obtención de los Puntos de Función No Ajustados (PFNA). Esto se hace a partir de una suma ponderada de la cantidad de dominios de información y su complejidad. Los dominios de información están definidos por:

    - Número de entradas: datos que el usuario aporta al sistema. Es decir, procesos en los que se introducen datos y se actualiza cualquier fichero.

    - Número de salidas: datos que el sistema aporta al usuario. Procesos en los que se leen datos o se envían datos al exterior de la aplicación.

    - Número de consultas externas: entrada de datos externos al sistema que requieran una repuesta por parte del mismo. Es decir, procesos que son una combinación de entrada y salida.

    - Número de ficheros lógicos internos: grupos de datos relacionados entre sí internos del sistema (ficheros, bases de datos de la aplicación …)

- Número de ficheros lógicos externos: grupos de datos que se mantienen externamente (ficheros, base de datos del servidor…)

Estos dominios de información tienen un peso con el cual tendrán que ponderarse, acorde a su complejidad, que detallaremos más adelante.

2. Ajuste de los PFNA mediante el Factor de Ajuste (FA), que depende de las características del proyecto. Este valor se obtiene a partir de la suma de 14 factores, denominados de complejidad (FC), que su valor oscila entre 0 (menor peso) y 5 (mayor peso). Estos son:

    - Comunicación de datos: más alto cuantas más facilidades de comunicación haya disponibles para ayudar en el intercambio de información con la aplicación.

    - Procesamiento distribuido de datos: un valor alto tiene que ver con cómo se manejan los datos y las funciones de procesamiento distribuido.

    - Rendimiento: más alto cuantos más requerimientos de velocidad o tiempo de respuesta existan.

    - Gran carga de trabajo: más alto cuanto más intensa es la carga de trabajo de la(s) plataforma(s) donde se ejecuta el sistema.

    - Frecuencia de transacciones: directamente proporcional a la frecuencia con la que se realizan transacciones. Mensualmente, semanalmente, diariamente…

    - Entradas online de datos: porcentaje de la información que se registra online.

    - Manejo del usuario final: más alto cuanta más preparación se requiera tener por parte del usuario para manejar la herramienta.

    - Actualizaciones online: más alto cuantos más ficheros se actualicen a raíz de una operación o consulta online.

    - Procesos complejos: cantidad y dificultad de los procesos complejos en la herramienta.

    - Utilización de otros sistemas: cantidad de sistemas de los que depende la aplicación.

    - Facilidad de mantenimiento: cómo e efectivos son los procesos de arranque, parada, copia de seguridad y restauración de copia.

- Facilidad de operación: cómo de fácil es la instalación y conversión a la herramienta.

- Instalación en múltiples lugares: más alto si se puede instalar en múltiples sitios, dispositivos, organizaciones…

- Facilidad de cambio: más alto cuanto más fácilmente ampliable sea la aplicación.

3. Cálculo de los Puntos de Función Ajustados (PFA) a partir de los PFNA y el Factor de Ajuste.

4. Estimación de las líneas de código, que se obtendrán a partir de una tabla de equivalencias entre los PF y los lenguajes de programación utilizados en el desarrollo de la herramienta. En nuestro caso J2EE y Android OS.

4.3.1.1 Puntos de Función no ajustados

- Número de entradas:

Nombre	**Complejidad**
Formulario de registro	Media
Formulario de login	Baja
Formulario de edición de perfil	Media
Formulario de edición de ruta	Baja
Conversaciones entre usuarios	Media
Pantalla de ajustes	Baja

Tabla 4 - Número de entradas

- Número de salidas:

Nombre	**Complejidad**
Pantalla de notificaciones	Baja
Pantalla de viajes	Baja
Pantalla de mensajes	Baja
Pantalla del perfil	Media
Pantalla de ajustes	Baja
Mapas	Alta

Tabla 5 - Número de salidas

- Número de consultas externas:

Nombre	Complejidad
Identificación del usuario	Media
Viajes del usuario	Media
Mensajes del usuario	Baja

Tabla 6 - Número de consultas externas

- Número de ficheros lógicos internos:

Nombre	Complejidad
Base de datos de la aplicación	Baja
Base de datos del servidor	Alta

Tabla 7 - Número de ficheros lógicos internos

- Número de ficheros lógicos externos:

Nombre	Complejidad
Google Maps API	Alta
Google Geocode API	Media

Tabla 8 - Número de ficheros lógicos externos

- Pesos de los dominios de información según complejidad:

Dominio de Información	*Complejidad baja*	*Complejidad media*	*Complejidad alta*
Entradas	x3	x4	x6
Salidas	x4	x5	x7
Consultas externas	x3	x4	x6
Fich. Lógicos internos	x7	x10	x15
Fich. Lógicos externos	x5	x7	x10

Tabla 9 - Pesos de los dominios de información según complejidad

- Cálculo de los PFNA:

Dominio de información	nº x complejidad Baja	nº x complejidad media	nº x complejidad alta	Total
Entradas	3x3	3x4	0x6	21
Salidas	4x4	1x5	1x7	28
Consultas externas	1x3	2x4	0x6	11
Fich. Lógicos internos	1x7	0x10	1x15	22
Fich. Lógicos externos	0x5	1x7	1x10	17
Total				**99**

Tabla 10 - Cálculo de los Puntos de Función No Ajustados

4.3.1.2 Cálculo del Factor de Ajuste

Factores de complejidad:

Factor de complejidad	Complejidad / Peso
Comunicación de datos	4
Funciones distribuidas	3
Rendimiento	4
Gran carga de trabajo	2
Frecuencia de transacciones	3
Entradas online de datos	4
Requisitos de manejo del usuario final	2
Actualizaciones online	3
Procesos complejos	4
Utilización de otros sistemas	2
Facilidad de mantenimiento	3
Facilidad de operación	3
Instalación en múltiples lugares	3
Facilidad de cambio	3
TOTAL	**43**

Tabla 11 - Cálculo del Factor de Ajuste

El factor de ajuste viene dado por el resultado de la siguiente ecuación:

$$\boldsymbol{FA = 0.65 + \left(0.1\, x \sum FC\right)} = 0.65 + 0.01\, x\, 43 = \mathbf{1.08}$$

4.3.1.3 Obtención de los Puntos de Función ajustados

La obtención de los PFA se hace mediante el producto de los Puntos de Función y el Factor de Ajuste:

$$\boldsymbol{PFA = PF\ x\ FA} = 99\ x\ 1.08 = \mathbf{106.92}$$

4.3.1.4 Estimación de las líneas de código

Para calcular las líneas de código se hace uso de una media de las equivalencias de los lenguajes de programación, ya que no todos los lenguajes tienen la misma complejidad, en nuestro caso son Java EE y Android OS. Las líneas de código por PF para estos lenguajes se muestran en la siguiente tabla:

Lenguaje	**LDC/ PF**
J2EE	46
Android SO (Java)	53

Tabla 12 - Estimación de las líneas de código por lenguaje de programación

Pese a que no sabemos la proporción exacta de carga de trabajo de cada uno en el proyecto, sí que sabemos que la carga que conlleva el servidor suele ser más alta. Sin embargo, debido a las conexiones con APIs por parte de la aplicación móvil y la modelación a mano de datos de carácter espacial, podemos decir que aproximadamente la carga ha sido similar en ambos casos y, por tanto, nos quedaremos con que las líneas de código medias por punto de función es aproximadamente 50.

En este caso, el número de líneas de código resultante será:

$$\boldsymbol{LDC} = 50\ x\ 106.92\ \approx \mathbf{5346}$$

Por tanto, usando una estimación utilizando el método de Puntos de Función, obtenemos que nuestro proyecto tendrá aproximadamente 5300 líneas de código.

4.3.2 COCOMO

Abreviatura de *Constructive Cost Model*, es un modelo de estimación de coste del software, hablando tanto en términos monetarios como temporales. COCOMO necesita basarse en una estimación previa del tamaño del software en líneas de código, para lo cual nos valdremos de la aproximación obtenida en el punto 3.2.1.4. Estimación de las líneas de código.

COCOMO tiene tres métodos, cada uno más detallado que el anterior:

1. El primer método, denominado COCOMO básico, es bueno para proyectos simples, con código poco depurado y de desarrollo simple y poco maduro, aunque su precisión está limitada sobre todo debido a la falta de factores que tiene en cuenta.

2. El segundo método, COCOMO intermedio, está enfocado a aquellos al menos medianamente bien caracterizados, donde hace falta tener en cuenta los factores que el primer método obviaba.

3. El tercer nivel, COCOMO avanzado, tiene en cuenta también la influencia de las fases individuales del proyecto.

En nuestro caso utilizaremos el método intermedio, ya que nuestro proyecto ya se encuentra bien definido y en las últimas fases de desarrollo. Este método comprende tres modelos, dependiendo del tamaño y complejidad:

- Orgánico: pequeños equipos con buena experiencia trabajando con poca rigidez en los requisitos.

- Semi-acoplado: equipos medianos con una experiencia media, con proyectos de tamaño y dificultad intermedia y rigidez de los requisitos todavía baja.

- Integrado: desarrollos con fuertes restricciones y gran exigencia técnica. Es una combinación de proyectos orgánicos y semi-acoplados y con poca previsión de cambios.

Para nuestra herramienta, el cual consideramos un equipo de desarrollo pequeño pero cuyos requisitos están casi determinados. Elegiremos el modelo semi-acoplado, ya que mi experiencia en este tipo de sistemas también es media si tenemos en cuenta la experiencia en ambas partes del sistema.

A la hora de realizar los cálculos, debemos de tener en cuenta las siguientes constantes según el modelo de proyecto:

Tipo de proyecto	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semi-acoplado	3.0	1.12	2.5	0.35
Integrado	3.6	1.20	2.5	0.32

Tabla 13 - Constantes para cada modelo COCOMO

Y, además, los factores de coste del esfuerzo, que dependerán de las características del proyecto:

Factores de coste	**Valores**					
	Muy bajo	Bajo	Medio	Alto	Muy alto	Extra alto
Software						
Fiabilidad del software	0.75	0.88	1.00	*1.15*	1.40	
Tamaño de la base de datos		0.94	*1.00*	1.08	1.16	
Complejidad del producto	0.70	0.85	1.00	*1.15*	1.30	1.65
Hardware						
Restricciones de rendimiento en ejecución			*1.00*	1.11	1.30	1.66
Restricciones de memoria			1.00	*1.06*	1.21	1.56
Volatilidad del entorno de la máquina virtual		*0.87*	1.00	1.15	1.30	
Tiempo de respuesta requerido		0.87	1.00	*1.07*	1.15	
Personal						
Capacidad de los analistas	1.46	1.19	*1.00*	0.86	0.71	
Experiencia con el tipo de aplicación	1.29	1.13	*1.00*	0.91	0.82	
Experiencia con el hardware	1.42	1.17	1.00	*0.86*	0.70	
Experiencia con el lenguaje de programación	1.21	1.10	*1.00*	0.90		
Capacidad de los programadores	1.14	1.07	*1.00*	0.95		
Proyecto						
Técnicas modernas de programación	1.24	1.10	1.00	*0.91*	0.82	
Utilización de herramientas software	1.24	1.10	1.00	*0.91*	0.83	
Restricciones en la planificación temporal del desarrollo	1.23	1.08	*1.00*	1.04	1.10	

Tabla 14 - Factores de coste del esfuerzo

A partir de los valores *remarcados* en la tabla, obtendremos peso de nuestro factor de coste (en el cálculo omitimos los valores iguales a 1):

$$\prod \mathbf{m}(\mathbf{x_i}) = 1.15 * 1.15 * 1.06 * 0.87 * 1.07 * 0.86 * 0.91 * 0.91 = \mathbf{0.92}$$

A partir de aquí, las ecuaciones que se utilizan para llevar a cabo la estimación por COCOMO son:

- Esfuerzo nominal: $MM = a * \left(Kl^b\right) \quad (personas\ por\ mes)$
- Esfuerzo: $E = MM * \prod m(x_i) \quad (personas\ por\ mes)$
- Tiempo de desarrollo: $TDEV = c * \left(E^d\right) \quad (meses)$
- N.º medio de personas: $CosteH = {}^{E}/_{TDEV} \quad (personas)$

Haciendo uso de estas ecuaciones y el factor de coste previamente calculado, obtenemos:

$$E = a * \left(Kl^b\right) * \prod m(x_i) = 3 * 5.35^{1.12} * 0.92 = \mathbf{18.06\ personas\ por\ mes}$$

Para acabar este proyecto en aproximadamente un mes, necesitaríamos de 18 personas trabajando conjuntamente. Sin embargo, el tiempo que se tarda en realizar este proyecto es de:

$$TDEV = 2.5 * 18.06^{0.35} = \mathbf{6.88\ meses}$$

Que es lo que tardaría una única persona en acabar el proyecto entero. Con último dato, y haciendo de divisor del dato anterior, obtenemos las personas que harían falta para finalizar con éxito el proyecto.

$$\boldsymbol{CosteH} = \frac{18.06}{6.88} = \mathbf{2.62} \approx \mathbf{3\ personas}$$

De acuerdo a este resultado, necesitaríamos tres personas durante algo más de (6.88/3 = 2.293) dos meses para finalizar el proyecto correctamente. En nuestro caso, como sólo contamos con una persona, tiempo de desarrollo será de 6 meses y 26 días.

4.3.3 Presupuesto

Para calcular el presupuesto se hará uso de los resultados obtenidos en el apartado 3.2 Planificación. La suma total del presupuesto vendrá dada por el coste de personal asignado al proyecto, el coste del software y el coste del hardware necesarios para llevar a cabo el desarrollo de la herramienta.

- **Presupuesto de la mano de obra:**
  Nuestra planificación inicial estimaba las horas del personal en 240, 436 y 244 horas para el analista, programador y documentalista, respectivamente. Podemos realizar una estimación del presupuesto en base a estos datos. Teniendo en cuenta estos datos, para una jornada de 8 horas al día de 5 días a la semana, el presupuesto inicial para el personal es el siguiente:

	Tiempo (h)	*Coste (€/h)*[7]	*Total (€)*
Analista	240	13.80	3312
Programador	436	13.22	5763.92
Documentalista	244	10.66	2601.04

Tabla 15 - Presupuesto de la mano de obra

Total: 11676.96€

- **Presupuesto del Hardware:**
  Se tendrán en cuenta los dispositivos físicos necesarios para llevar a cabo el proyecto, y teniendo en cuenta que sólo se usarán mientras dure la jornada laboral, es decir 5 días a la semana por 8 horas al día, unas 40 horas semanales.
  - **Ordenador portátil**: se estima que el tiempo de vida útil son 4 años (208 semanas), por lo que su utilización es del 11.06% (23 semanas).
  - **Servidor:** se estima que el tiempo de vida útil son 5 años (260 semanas), su utilización ha sido del 8.85%.
  - **Internet**: cuota mensual de 30€+IVA. Durante 23 semanas (5.75meses) con un uso aproximado del 35%.
  - **Smartphone**: tiempo de vida estimado en 2 años, utilización del 5.53%

HW	Coste	Uso (%)	Total (€)
Ordenador portátil	1000	11.06	110.60
Servidor	3000	8.85	265.50
Internet	36.11*5.75	35	72.67
Smartphone	300	5.53	16.59

Tabla 16 - Presupuesto del Hardware

Total: 514.86€

[7] «Tusalario.es - Calcula y Compara tu Sueldo en España. Conoce el Salario Neto y Bruto de tus Colegas». [En línea]. Disponible en: http://www.tusalario.es/main/salario/comparatusalario?job-id=2512010000000. [Accedido: 13-jul-2017].

- **Presupuesto del Software:**
  Se tendrán en cuenta las herramientas utilizadas para llevar a cabo el desarrollo del software y teniendo en cuenta que sólo se usarán mientras dure la jornada laboral, es decir 5 días a la semana por 8 horas al día, unas 40 horas semanales.

    - **Windows 10 Home[8]:** licencia por uso de 135€, necesitamos dos licencias, una para la aplicación y otra para el servidor. Con doble utilización equivalente a la del ordenador personal, ya que calculamos en porcentaje al uso de dos licencias.

    - **Android Studio:** gratuito.

    - **Google Play Developer**: licencia de uso $25 (22€). La licencia de uso es vitalicia y no se cobra comisión por cada aplicación que se sube. Debido a esto, se considera amortizada y no repercute en el presupuesto.

    - **Netbeans IDE:** gratuito.

    - **Glassfish Server**: licencia de uso $100 (88€) anual. Con un uso equivalente a la del ordenador personal.

SW	Coste	Uso (%)	Total (€)
Windows 10 (2 Uds)	135*2	11.06	29.86
Glassfish Server	88	11.06	9.73

Tabla 17 - Presupuesto del Software

Total: 39.59€

Sumando los presupuestos individuales, obtenemos que el presupuesto total estimado es de:

Presupuesto = 11676.96 + 514.86 + 39.59 = **12231.41 €**

4.3.4 Coste económico real

El presupuesto arroja un resultado que no deja de ser una estimación que no debe considerarse definitiva para un proyecto real. Para calcular el coste económico real del proyecto se tendrá en cuenta el tiempo y recursos necesitados durante todo el proceso de desarrollo. Para ello, se tendrán en cuentan las desviaciones ocurridas con respecto a la planificación inicial. La suma total del coste vendrá igualmente dada por el coste de personal, el coste del software y el coste del hardware necesarios para llevar a cabo el desarrollo de la herramienta:

[8] Microsoft.com. (2017). *Comprar Windows 10* Home: https://www.microsoft.com/es-es/store/d/windows-10-home/d76qx4bznwk4/1NT3 [Accedido: 13-jul-2017]

- **Desviaciones:** Una vez realizado el proyecto, podemos ilustrar el tiempo y recursos utilizados para llevarlo a cabo, lo haremos de la misma manera que se hizo en el apartado 3.2 Planificación:

Nombre	Duración	Inicio	Fin
Iteración 1	20d	**30/01/2017**	**24/02/2017**
Análisis	14d	30/01/2017	16/02/2017
Diseño	4d	17/02/2017	22/02/2017
Implementación y pruebas	2d	23/02/2017	24/02/2017
Iteración 2	20d	**27/02/2017**	**24/03/2017**
Análisis	11d	27/02/2017	13/03/2017
Diseño	6d	14/03/2017	21/03/2017
Implementación	3d	22/03/2017	24/03/2017
Iteración 3	20d	**27/03/2017**	**21/04/2017**
Análisis	4d	27/03/2017	30/03/2017
Diseño	5d	03/04/2017	07/04/2017
Implementación	6d	10/04/2017	17/04/2017
Prueba	3d	18/04/2017	20/04/2017
Documentación	1d	21/04/2017	21/04/2017
Iteración 4	32d	**24/04/2017**	**06/06/2017**
Análisis	1d	24/04/2017	24/04/2017
Diseño	6d	25/04/2017	02/05/2017
Implementación	11d	03/05/2017	17/05/2017
Prueba	4d	18/05/2017	23/05/2017
Documentación	10d	24/05/2017	06/06/2017
Iteración 5	28d	**07/06/2017**	**14/07/2017**
Diseño	5d	07/06/2017	13/06/2017
Implementación	11d	14/06/2017	28/06/2017
Prueba	5d	29/06/2017	05/07/2017
Documentación	7d	06/07/2017	14/07/2017

Tabla 18 - Desviaciones en la duración del proyecto real

En la *Tabla 18* anterior se reflejan la duración final de las iteraciones, mostrando también la duración ocurrida de cada iteración, y cada fase dentro de estas iteraciones. La forma gráfica de ilustrar la anterior tabla es mediante un Diagrama de Gantt que se añade más abajo, que nos ayuda a comprender de un solo vistazo, cómo se han desarrollado estas desviaciones.

Podemos observar la ausencia de tareas paralelas, lo que claramente reduce la eficiencia del ciclo. Esto es debido a que el equipo de trabajo está formado por una sola persona, y tanto Analista, como Programador y Documentalista son la misma, y esto imposibilita el desarrollo paralelo de tareas que en una empresa al uso serían fácilmente desempeñables de manera concurrente.

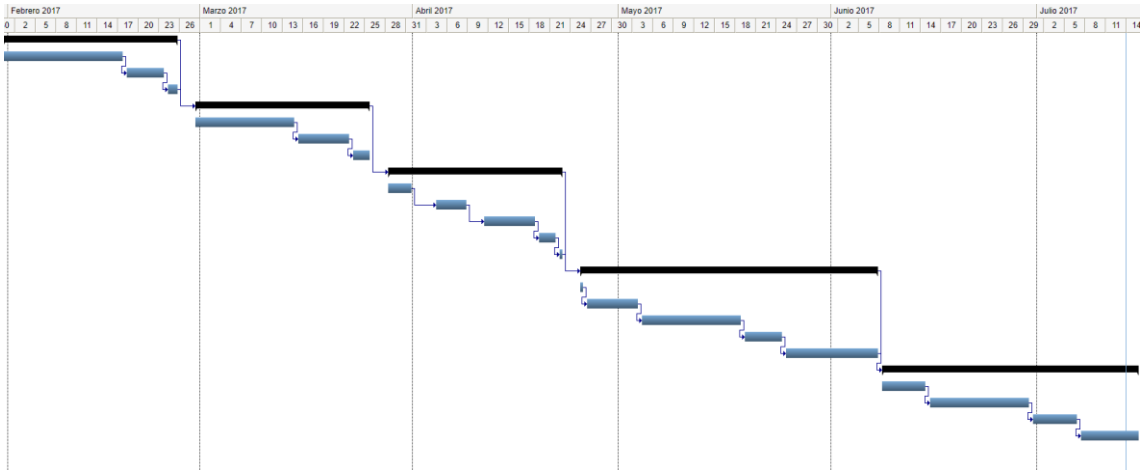


Imagen 9 - Diagrama de Gantt de la duración real del proyecto

Las desviaciones principales con respecto a la planificación radican en un cambio de la duración de las iteraciones, las primeras son más cortas que las últimas. También que el fin de desarrollo del proyecto ha tenido lugar un día más tarde. Además, la fase de documentación ha perdido el peso que tenía en la planificación y, por el contrario, las fases de análisis, pruebas y sobre todo la de implementación han cobrado relevancia:

Rol	***Iteración***					***Total (h)***
	1	***2***	***3***	***4***	***5***	
Analista	112	88	32	8	40	**280**
Programador	48	72	112	168	168	**568**
Documentalista	0	0	8	80	56	**144**

Tabla 19 - Tiempo real por rol en cada iteración en el desarrollo del proyecto

El total de horas que han hecho falta para la finalización del proyecto son 992 horas, (72 horas más que en la planificación). En total han sido 24.8 semanas, es decir, 24 semanas y 4 días laborables.

- **Coste real de la mano de obra:**
  Nuestro proyecto ha requerido de 280, 568, 144 horas para el analista, programador y documentalista, respectivamente. Podemos realizar un cálculo del coste real del desarrollo en base a estos datos. Teniendo en cuenta estos datos, para una jornada de 8 horas al día de 5 días a la semana, el presupuesto inicial para el personal es el siguiente:

	Tiempo (h)	*Coste (€/h)*[9]	*Total (€)*
Analista	280	13.80	3312
Programador	568	13.22	7508.96
Documentalista	144	10.66	2601.04

Tabla 20 - Coste real de la mano de obra

Total: 13422€

- **Cote real del Hardware:**
  Se han tenido en cuenta los dispositivos físicos necesarios para llevar a cabo el proyecto, y teniendo en cuenta que sólo se han usado mientras ha durado la jornada laboral, es decir 5 días a la semana por 8 horas al día, unas 40 horas semanales.

  - **Ordenador portátil**: Su utilización es del 11.92% (24.8 semanas de 208).
  - **Servidor**: se estima que el tiempo de vida útil son 5 años (260 semanas), su utilización ha sido del 9.53%.
  - **Internet**: cuota mensual de 30€+IVA durante 24.8 semanas (6.2 meses) con un uso aproximado del 35%.
  - **Smartphone**: tiempo de vida estimado en 2 años, utilización del 5.96%

HW	Coste	Uso (%)	Total (€)
Ordenador portátil	1000	11.92	119.20
Servidor	3000	9.53	285.90
Internet	36.11*6.2	35	78.36
Smartphone	300	5.96	17.88

Tabla 21 - Coste real del Hardware

Total: 501.34€

- **Coste real del Software:**
  Se han tenido en cuenta las herramientas utilizadas para llevar a cabo el desarrollo del software y teniendo en cuenta que sólo se han usado mientras dure la jornada laboral, es decir 5 días a la semana por 8 horas al día, unas 40 horas semanales.

  - **Windows 10 Home**: licencia por uso de 135€, necesitamos dos licencias, una para la aplicación y otra para el servidor. Con doble utilización equivalente a la

[9] «Tusalario.es - Calcula y Compara tu Sueldo en España. Conoce el Salario Neto y Bruto de tus Colegas». [En línea]. Disponible en: http://www.tusalario.es/main/salario/comparatusalario?job-id=2512010000000. [Accedido: 13-jul-2017].

del ordenador personal, ya que calculamos en porcentaje al uso de dos licencias.

- **Android Studio**: gratuito.
- **Google Play Developer**: licencia de uso $25 (22€). La licencia de uso es vitalicia y no se cobra comisión por cada aplicación que se sube. Debido a esto, se considera amortizada y no repercute en el presupuesto.
- **Netbeans IDE**: gratuito.
- **Glassfish Server**: licencia de uso $100 (88€) anual. Con un uso equivalente a la del ordenador personal.

SW	Coste	Uso (%)	Total (€)
Windows 10 (2 Uds)	135*2	11.92	32.18
Glassfish Server	88	11.92	10.49

Tabla 22 - Coste real del Software

Total: 42.67€

Sumando los presupuestos individuales, obtenemos que el presupuesto total estimado es de:

Coste real = 13422 + 501.34 + 42.67 = **13966.01 €**

El coste real ha sido **1736.60 €** más caro de lo que se presupuestó (13966.01 - 12231.41), esto ha sido principalmente al aumento de coste temporal del presupuesto, y al aumento de horas que ha necesitado el programador para llevar a cabo la herramienta con respecto al analista y al documentalista, ya que tiene un sueldo mayor a los otros dos.

5 ANÁLISIS DEL SISTEMA

Llegados a este punto del documento, es el momento de especificar los tipos de usuarios que tendrá la aplicación, requisitos que debe cumplir la herramienta y el diagrama de casos de uso.

En todo momento se ha querido desarrollar una herramienta simple y amigable al usuario. Es por eso que, en cuanto a la caracterización de los usuarios del sistema tenemos que, en nuestro caso, la aplicación sólo contará un tipo de usuario capaz de interactuar con la aplicación, más concretamente:

- **Usuario genérico**: este usuario podrá ver y editar toda la información relativa a su perfil, empezando por las preferencias dentro de la propia aplicación, como datos personales y rutas. También podrá contactar libremente con cualquier otro usuario que le haya recomendado el sistema.

5.1 DESCRIPCIÓN DE ACTORES

Únicamente se contemplan dos tipos de actores que puedan hacer interacción con la aplicación, de tal manera que la actividad del sistema queda limitada a estos dos tipos de actores: logueados y sin loguear.

ACT-01	**Usuario general**
Versión	1.0 (16 mayo 2017)
Autor	Jorge Plaza Jiménez
Descripción	Este actor representa a un actor cualquiera en el sistema que aún esté sin loguear
Comentarios	Solamente puede visualizar las pantallas de registro e inicio de sesión.

Tabla 23 - ACT-01 - Usuario general

ACT-02	**Usuario identificado**
Versión	1.0 (16 mayo 2017)
Autor	Jorge Plaza Jiménez
Descripción	Este actor representa a un actor cualquiera en el sistema que ya esté identificado
Comentarios	No puede visualizar las pantallas de registro y logueo, pero tiene acceso al resto de pantallas: perfil, ajustes, mapas, mensajes y notificaciones.

Tabla 24 - ACT-02 - Usuario identificado

5.2 REQUISITOS DE USUARIO

En esta parte del documento se pasa a especificar los requisitos que tanto aplicación como servidor deben cumplir.:

RU-1. Un usuario general podrá registrarse haciendo uso de su nombre, apellidos, email válido, número de teléfono y contraseña.

RU-2. Un usuario general podrá iniciar sesión mediante su email y contraseña.

RU-3. Un usuario identificado podrá cerrar sesión.

RU-4. El usuario identificado podrá ver la información personal en su perfil.

RU-5. El usuario identificado podrá modificar su información personal en su perfil, incluyendo su foto.

RU-6. El usuario identificado podrá consultar su última ruta.

RU-7. El usuario identificado podrá visualizar sus rutas frecuentes.

RU-8. El usuario identificado podrá guardar y compartir su última ruta grabada.

RU-9. El usuario identificado podrá modificar una ruta frecuente propia.

RU-10. El usuario identificado podrá borrar una ruta frecuente propia si así lo desea.

RU-11. El usuario identificado podrá visualizar qué rutas propias tienen coincidencia con las rutas de otros usuarios, cuando el sistema se lo permita.

RU-12. El usuario identificado podrá iniciar una conversación con otro usuario que realice una ruta en la cual esté interesado.

RU-13. El usuario identificado podrá elegir con qué usuarios sugeridos quiere aceptar o denegar una solicitud de conversación.

RU-14. El usuario identificado podrá visualizar las conversaciones con otros usuarios.

5.2.1 Diagrama de casos de uso

Una vez materializadas las necesidades de los usuarios en el apartado anterior, se pueden listar los Casos de Uso a partir de ellas. Estos requisitos definen y delimitan la funcionalidad que ofrece el sistema, y se modelan en los casos de uso. Los actores caracterizados en el apartado 4.1 Descripción de Actores realizan los casos de uso listados en el apartado anterior. Para representar la interacción entre los Casos de Uso y los actores se ha elaborado el siguiente diagrama:

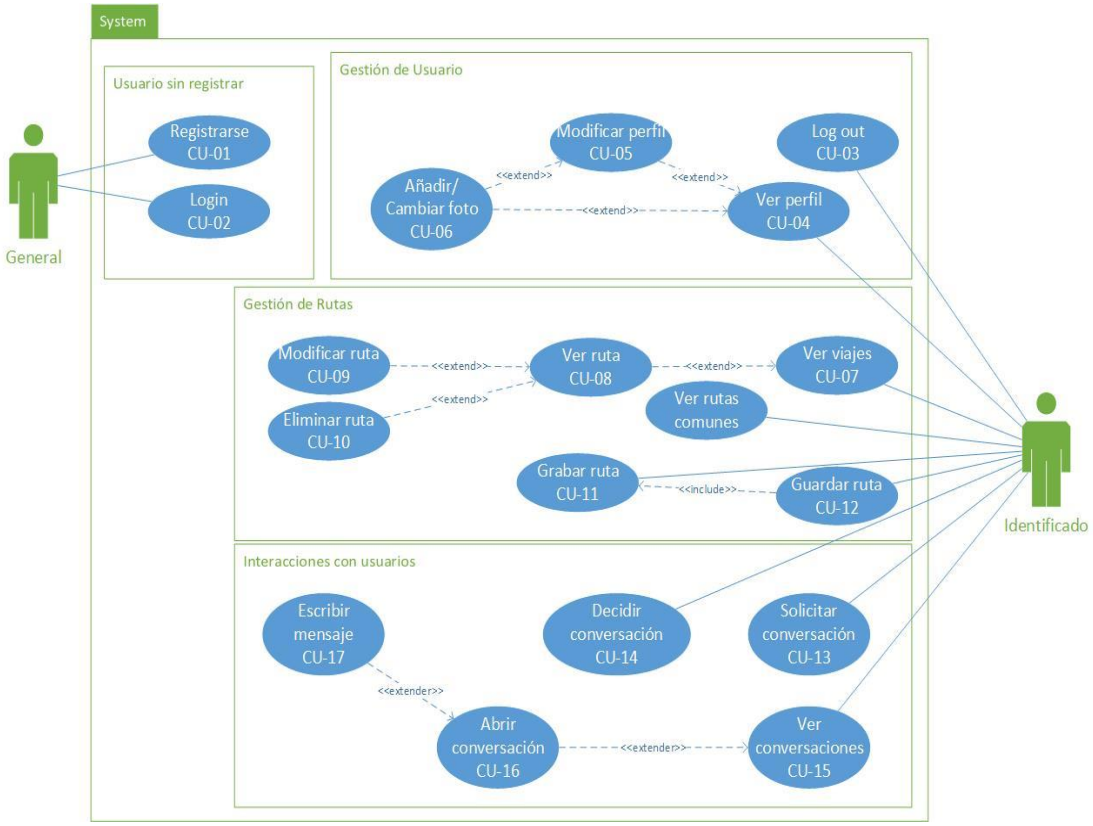


Imagen 10 - Diagrama de casos de uso

5.2.2 Especificación de casos de uso

En este apartado especificaremos los casos de uso ilustrados en el anterior diagrama.

CU-1. Registro.

CU-2. Identificación.

CU-3. Cerrar sesión.

CU-4. Ver perfil.

CU-5. Modificar perfil.

CU-6. Añadir/cambiar foto de perfil.

CU-7. Ver viajes

CU-8. Ver ruta.

CU-9. Modificar ruta.

CU-10. Eliminar ruta.

CU-11. Grabar ruta.

CU-12. Guardar ruta.

CU-13. Solicitar conversación.

CU-14. Decidir conversación.

CU-15. Ver conversaciones

CU-16. Abrir conversación

CU-17. Escribir mensajes

CU-01	**Registrarse**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá a un usuario no logueado crear una cuenta en la aplicación, previa introducción de los datos solicitados.	
Precondiciones		
Secuencia normal	**Paso**	**Acción**
	1	El usuario no logueado indica que desea registrarse en la plataforma.
	2	El sistema presenta los campos necesarios para realizar el registro.
	3	El usuario no logueado rellena los campos solicitados.
	4	El sistema valida los campos introducidos por el usuario envía los datos al servidor.
	5	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema crea un nuevo usuario.	
Excepción 1	**Paso**	**Acción**
	4	Si falta algún campo por rellenar
	5	El sistema notifica al usuario qué campos faltan por rellenar
	6	El sistema vuelve al paso 3
Excepción 2	4	El email introducido por el usuario ya existe
	5	El sistema notifica al usuario que el *usuario* ya existe
	6	El sistema vuelve al paso 3
Excepción 3	3	El usuario decide no registrarse
	4	El sistema vuelve al paso 1
Rendimiento		
Frecuencia	Media	
Importancia	Alta	
Comentarios		

Tabla 25 - CU-01 - Registrarse

CU-02	**Identificación**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá a un usuario no logueado acceder a la parte privada de la aplicación, siempre y cuando tenga una cuenta creada en la misma.	
Precondiciones	El usuario debe estar registrado en la plataforma	
Secuencia normal	**Paso**	**Acción**
	1	El usuario no logueado indica que desea acceder al sistema.
	2	El sistema presenta los campos necesarios para loguearse en la aplicación.
	3	El usuario no logueado rellena los campos solicitados.
	4	El sistema comprueba si existe el usuario en la base de datos e inicia la sesión del usuario.
	5	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El usuario inicia sesión	
Excepciones	**Paso**	**Acción**
	4	Si los datos introducidos no son correctos.
	5	El sistema indica que el usuario no existe
	6	El sistema vuelve al paso 3
Frecuencia	Alta	
Importancia	Alta	

Tabla 26 - CU-02 – Identificarse

CU-03	**Cerrar sesión**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá cerrar sesión a un usuario identificado	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *perfil*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *cerrar sesión*
	2	El usuario confirma la acción
	3	El sistema saca al usuario del sistema y borra sus datos
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones		
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Media	
Importancia	Alta	
Comentarios		

Tabla 27 - CU-03 - Cerrar sesión

CU-04	**Ver perfil**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema mostrará al usuario la pantalla de su perfil cuando lo solicite	
Precondiciones	Usuario identificado	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *perfil*
	2	El sistema muestra la pantalla de *perfil* con los datos del usuario
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones		
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Alta	
Importancia	Alta	
Comentarios		

Tabla 28 - CU-04 - Ver perfil

CU-05	**Modificar perfil**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá al usuario modificar sus datos personales	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *perfil*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *editar perfil*
	2	El sistema muestra al usuario sus datos para editar
	3	El usuario modifica los datos que quiera y confirma
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema modifica los datos personales del usuario	
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Baja	
Importancia	Media	
Comentarios		

Tabla 29 - CU-05 - Modificar perfil

CU-06	**Añadir/Cambiar foto de perfil**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá a un usuario identificado cambiar su foto de perfil o añadir una nueva si no la ha añadido todavía	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *perfil*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *editar foto*
	2	El usuario selecciona la foto deseada en su galería
	3	El usuario confirma la acción
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema guarda la nueva imagen de perfil del usuario	
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Baja	
Importancia	Baja	
Comentarios		

Tabla 30 - CU-06 - Añadir/Cambiar foto de perfil

CU-07	**Ver viajes**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema mostrará al usuario sus viajes realizados cuando lo solicite	
Precondiciones	Usuario identificado	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *viajes*
	3	El sistema muestra al usuario la pantalla de sus viajes realizados
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones		
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Media	
Importancia	Alta	
Comentarios		

Tabla 31- CU-07 - Ver viajes

CU-08	**Ver ruta**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema mostrará al usuario la ruta completa que haya realizado en un viaje	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *tablón*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *mapas*
	2	El sistema muestra al usuario la pantalla de *mapas*
	3	El usuario pulsa sobre una ruta concreta
	4	El sistema dibuja en el mapa la ruta realizada.
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema crea un nuevo usuario.	
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Media	
Importancia	Alta	
Comentarios		

Tabla 32 - CU-08 - Ver ruta

CU-09	**Modificar viaje**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá al usuario identificado modificar el nombre y los días habituales de la ruta	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *viajes*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *editar viaje*
	2	El sistema muestra al usuario un cuadro de diálogo con los datos editables de la ruta: nombre y días
	3	El usuario modifica los datos que quiera y confirma
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema modifica los datos del viaje	
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Baja	
Importancia	Media	
Comentarios		

Tabla 33 - CU-09 - Modificar viaje

CU-10	**Eliminar viaje**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá borrar una ruta guardada a un usuario identificado	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *viajes*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *editar viaje*
	2	El sistema muestra el cuadro de diálogo del CU-09
	3	El usuario pulsa sobre el botón de *eliminar viaje*
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema elimina el viaje del usuario y su ruta correspondiente	
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Baja	
Importancia	Baja	
Comentarios		

Tabla 34 - CU-10 - Eliminar viaje

CU-11	**Grabar ruta**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá grabar una nueva ruta a un usuario identificado	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *tablón* El sistema borrará una ruta grabada anterior (y no guardada)	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de grabar ruta (**+**) y confirma la acción
	2	El sistema comienza a registrar el geoposicionamiento del usuario
	3	Cuando el usuario desee, pulsa el botón de *parar de grabar ruta* (STOP) y confirma la acción
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema graba la ruta	
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Media	
Importancia	Muy Alta	
Comentarios		

Tabla 35 - CU-11 - Grabar ruta

CU-12	**Guardar Ruta**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá a un usuario general crear rutas.	
Precondiciones	1: El usuario debe estar logueado en la plataforma	
	2: El usuario debe tener una ruta grabada	
	3: El usuario debe tener abierta la pantalla *mapas*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario selecciona la última ruta
	2	El usuario pulsar el botón de guardar y confirma la acción
	3	El sistema almacena la ruta en la base de datos local.
	4	El sistema envía la ruta al servidor y la almacena en la BBDD
	5	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema guarda una nueva ruta.	
Excepción 1	**Paso**	**Acción**
	3	El usuario indica que no desea guardar la ruta
	4	El sistema vuelve al paso 2
Frecuencia	Baja	
Importancia	Alta	

Tabla 36 - CU-12 - Guardar Ruta

CU-13	**Solicitar conversación**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá cerrar sesión a un usuario identificado	
Precondiciones	Usuario identificado El usuario debe tener una ruta en común con el usuario al que quiere enviar la solicitud El usuario debe tener abierta la pantalla de *perfil*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre la tarjeta de la notificación
	2	El usuario confirma la acción
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones	El sistema envía al otro usuario la solicitud	
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Media	
Importancia	Alta	
Comentarios		

Tabla 37 - CU-13 - Solicitar conversación

CU-14	**Decidir conversación**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá decidir si quiere hablar o no con el usuario que le ha enviado una solicitud de conversación	
Precondiciones	Usuario identificado El otro usuario debe haber mandado una solicitud de conversación al usuario identificado El usuario debe tener abierta la pantalla de *perfil*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre la tarjeta de la notificación
	2	El usuario confirma la acción
	3	El sistema muestra al usuario la pantalla de mensajes (chat) para hablar con el otro usuario
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones		
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Media	
Importancia	Alta	
Comentarios		

Tabla 38 - CU-14 - Decidir conversación

CU-15	**Ver conversaciones**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá ver sus conversaciones activas con otros usuarios a un usuario identificado	
Precondiciones	Usuario identificado	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de *conversaciones*
	3	El sistema muestra al usuario las conversaciones activas con otros usuarios
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones		
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Alta	
Importancia	Alta	
Comentarios		

Tabla 39 - CU-15 - Ver conversaciones

CU-16	**Ver conversación**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá ver los mensajes de una conversación activa a un usuario identificado	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *conversaciones*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre la tarjeta de la conversación que quiere ver
	3	El sistema muestra al usuario el chat de esa conversación
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones		
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Media	
Importancia	Alta	
Comentarios		

Tabla 40 - CU-16 - Ver conversación

CU-17	**Escribir mensaje**	
Versión	*1.0*	
Autores	*Jorge Plaza Jiménez*	
Descripción	El sistema permitirá escribir mensajes en el chat de una conversación a un usuario identificado	
Precondiciones	Usuario identificado El usuario debe tener abierta la pantalla de *conversación (chat)*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario escribe el mensaje que quiere enviar en el cuadro de texto
	2	El usuario pulsa sobre el botón enviar
	3	El sistema manda el mensaje al otro usuario
Secuencia alternativa	**Paso**	**Acción**
Postcondiciones		
Excepción	**Paso**	**Acción**
Rendimiento		
Frecuencia	Media	
Importancia	Alta	
Comentarios		

Tabla 41 - CU-17 - Escribir mensaje

5.3 Requisitos no Funcionales

RNF-1. (Rendimiento) El sistema debe soportar la modificación concurrente de datos de, al menos, 100 usuarios.

RNF-2. (Seguridad) El sistema almacenará las contraseñas cifradas con hash MD5.

RNF-3. (Rendimiento) La aplicación deberá localizar una nueva posición del usuario en tiempo real si este se está moviendo.

RNF-4. (Rendimiento) Cada nueva posición del usuario deberá distar de la anterior un mínimo de 500 metros.

RNF-5. (Rendimiento) La respuesta del sistema ante la interacción del usuario no deberá ser superior a 4 segundos.

RNF-6. (Usabilidad) El sistema no podrá mostrar más de 1 formulario para realizar el registro en la plataforma.

RNF-7. (Usabilidad/Seguridad) La autenticación de los usuarios en la aplicación se hará mediante su email y contraseña.

RNF-8. (Seguridad) Las contraseñas de los usuarios deberán ser hasheadas en la base de datos mediante MD5.

RNF-9. (Disponibilidad) El sistema deberá estar disponible 24 horas al día, 7 días a la semana.

5.4 Requisitos de Información

RI-1. El sistema almacenará los datos personales del usuario como nombre, apellidos, correo electrónico verificado, contraseña cifrada, número de teléfono, fecha de nacimiento y foto.

RI-2. La foto de perfil no será de carácter obligatorio.

RI-3. Un usuario sólo podrá tener una foto de perfil, ya sea una propia o por defecto.

RI-4. La contraseña del usuario deberá tener un mínimo de 8 caracteres, y un máximo de 15.

RI-5. El email del usuario deberá tener un patrón ‘xxx@xxx.xxx’.

RI-6. El sistema almacenará datos relevantes del usuario como preferencias y mensajes con otros usuarios.

RI-7. El sistema almacenará las rutas de cada usuario de forma permanente.

RI-8. El sistema almacenará última ruta de cada usuario hasta que el usuario grabe otra.

RI-9. El sistema almacenará los mensajes del usuario con otros usuarios.

RI-10. Un usuario podrá tener un número ilimitado de rutas frecuentes.

5.5 DICCIONARIO DE DATOS

A continuación, se presenta el diccionario de datos lógico basado en el diseño relacional, especificando los atributos que aparecen en cada una de las tablas y añadiendo una breve descripción de cada uno: tipo (tipo de datos), null (si puede llegar a ser nulo o no), clave (tipo de clave, default (valor por defecto) y extra (si tiene alguna cualidad que no aparezca antes).

El diccionario de datos es el siguiente:

- Entidad **places**

Campo	Tipo	Null	Clave	Default	Extra
id	SERIAL	NO	PK		Auto_increment
Google_id	Varchar	NO			UNIQUE
Country	Varchar	NO			
Administrative_area_level_2	Varchar	NO			
Locality	Varchar	NO			
Place	Geometry	NO			POINT

Tabla 42 - Entidad places

- Entidad **users**

Campo	Tipo	Null	Clave	Default	Extra
id	SERIAL	NO	PK		Auto_increment
Name	Varchar	NO			
Lname	Varchar	NO			
Birth	Date	NO			
Phone	Integer	NO			UNIQUE
Email	Varchar	NO			UNIQUE
Passwd	Varchar	NO			MD5
Photo	Byte[]	YES			

Tabla 43 - Entidad users

- Entidad **trips**

Campo	Tipo	Null	Clave	Default	Extra
id	SERIAL	NO	PK		Auto_increment
Origin	Integer	NO	FK		Places(id)
Destination	Integer	NO	FK		Places(id)

Tabla 44 - Entidad trips

- Entidad **routes**

Campo	Tipo	Null	Clave	Default	Extra
id	SERIAL	NO	PK		Auto_increment
User_id	Integer	NO	FK		Users(id)
Trip_id	Integer	NO	FK		Trips(id)
Days	Boolean[]	YES			
StartTime	Time	NO			
Endtime	Time	NO			
Route	Geometry	NO			LINESTRING

Tabla 45 - Entidad routes

- Entidad **conversations**

Campo	Tipo	Null	Clave	Default	Extra
id	SERIAL	NO	PK		Auto_increment
User1_id	Integer	NO	FK		Users(id)
User2_id	Integer	NO	FK		Users(id)

Tabla 46 - Entidad conversations

- Entidad **messages**

Campo	Tipo	Null	Clave	Default	Extra
id	SERIAL	NO	PK		Auto_increment
Conversation_id	Integer	NO	FK		Conversations(id)
Date	Date	NO			
Text	Varchar	NO			

Tabla 47 - Entidad messages

- Entidad **commons**

Campo	Tipo	Null	Clave	Default	Extra
id	SERIAL	NO	PK		Auto_increment
User1_id	Integer	NO	FK		Users(id)
User2_id	Integer	NO	FK		Users(id)
Trip_id	Integer	NO	FK		Trips(id)
Abname	Varchar	NO			
Distance	Integer	NO			

Tabla 48 - Entidad commons

- Entidad **notifications**

Campo	Tipo	Null	Clave	Default	Extra
id	SERIAL	NO	PK		Auto_increment
User1_id	Integer	NO	FK		Users(id)
User2_id	Integer	NO	FK		Users(id)

Tabla 49 - Entidad notifications

5.6 DISEÑO CONCEPTUAL

A partir de los Requisitos de Información vistos anteriormente (4.4 Requisitos de Información), surgen una serie de relaciones y entidades que se pueden modelar en el diseño conceptual del sistema, también llamado modelo Entidad-Relación:

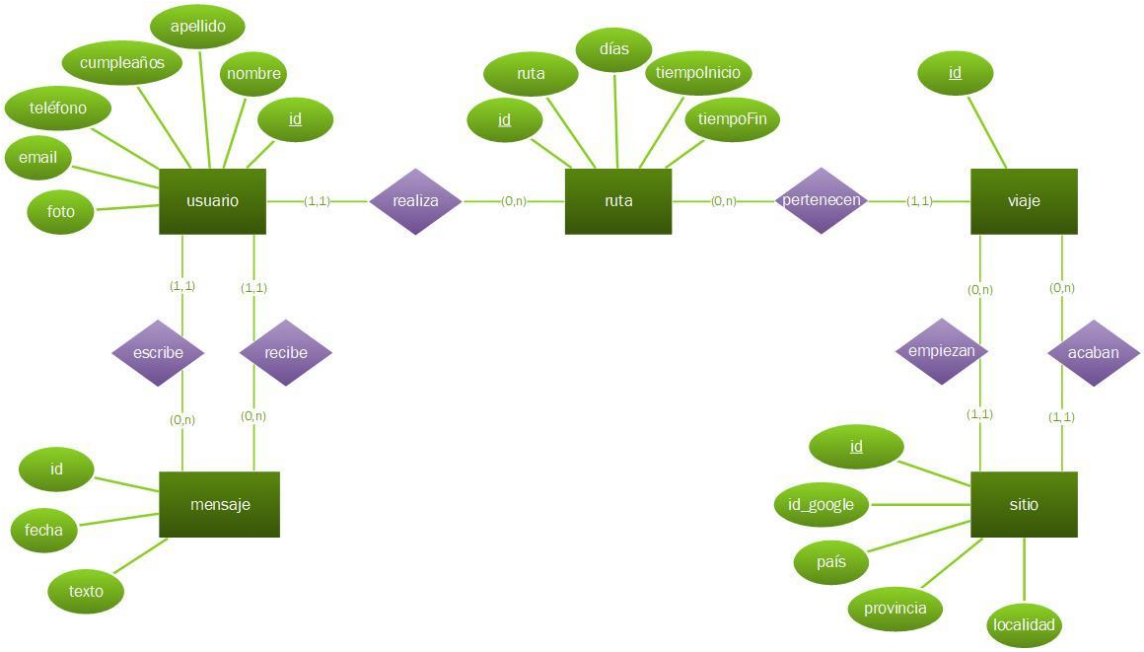


Imagen 11 - Diseño conceptual

Para empezar, la entidad que podríamos llamar la principal del sistema es el *Usuario*, este tendrá una serie de atributos (identificador, nombre, apellido(s), fecha de nacimiento, teléfono de contacto, correo electrónico y foto) que formarán parte de su información personal.

Esta entidad se relacionará con otras dos. Por una parte, se encuentra la entidad de *Mensajes*, con la cual se relacionará de manera dual, ya que un usuario puede tanto escribir como recibir mensajes. Para identificar y ordenar estos mensajes se hará uso de un identificador único pero también se registrará la fecha y hora de la emisión del mensaje, así como el texto del mensaje en sí.

Por otro lado, la entidad de usuarios también tiene relación con la entidad *Rutas*, las rutas podrían tratarse como información personal del usuario, ya que una ruta pertenece a un único usuario, aunque un usuario puede tener un número indeterminado de rutas. Esta entidad estará definida por un identificador único, el registro de puntos geográficos o coordenadas que componen la ruta en sí, los días en los que el usuario realiza la ruta y el momento de inicio y fin de la misma.

Esta entidad tiene relación directa, y podríamos decir que está contenida, con la entidad *Viajes*, ya que en la práctica cada ruta forma parte de un viaje. Por ejemplo, un viaje podría ser el que va de Madrid → Segovia, aunque para ir desde Madrid hasta Segovia podemos ir por la AP-6, por el Puerto de Guadarrama, por el de Navacerrada, o por otro sinfín de rutas.

La última entidad del diseño conceptual es con la que está relacionada dualmente (al igual que *Usuarios* lo está con *Mensajes*) la anterior entidad. La entidad de *Sitios* (también podríamos llamarla *lugares*) está caracterizada por un identificador único, un identificador único del lugar proporcionado por la Geocode API de Google, un nombre del país, nombre de la ciudad y de la localidad. El nombre utilizado para estos atributos viene reutilizado de la misma API mencionada.

6 DISEÑO SOFTWARE

En este apartado del documento, vamos a describir e ilustrar los componentes y arquitectura lógica y física que presentan tanto el servidor como la aplicación móvil.

6.1 ARQUITECTURA LÓGICA

La arquitectura lógica refleja cuáles son los componentes lógicos que forman el sistema, así como la relación entre ellos y cómo están conectados entre sí para dar funcionalidad a la herramienta. A continuación, veremos primero cuál es la arquitectura lógica del sistema completo, y después mostraremos más concretamente la lógica de la aplicación.

6.1.1 Arquitectura Lógica del Sistema

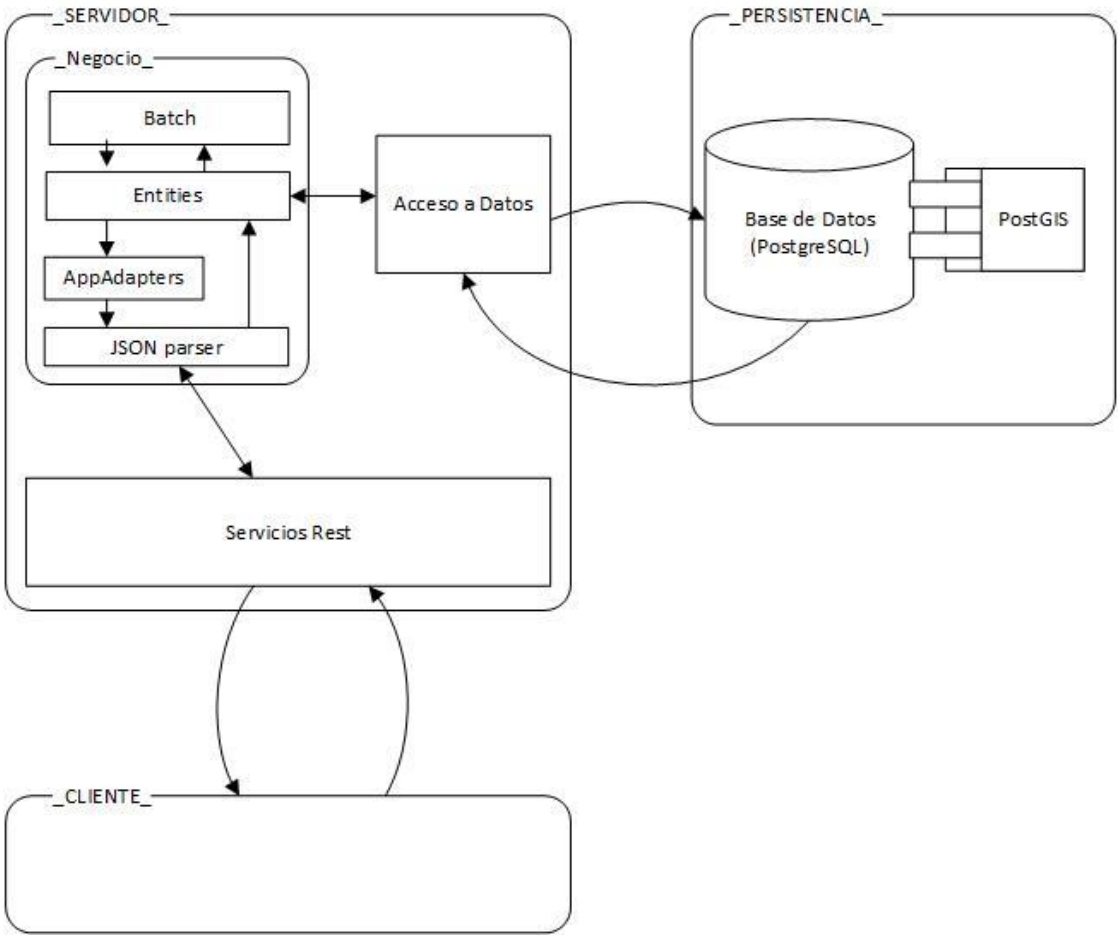


Imagen 12 - Arquitectura lógica del sistema

A la hora de diseñar la arquitectura lógica del conjunto del sistema, se ha optado por un modelo sencillo de arquitectura en tres capas. Tenemos una primera capa de cliente, que representa la aplicación que verá el usuario final, una capa intermedia que será la encargada de realizar todas las operaciones transparentes al usuario, y una tercera capa encargada de almacenar los datos:

- **Capa de persistencia:** está formada por la base de datos, en nuestro caso es un Sistema Gestor de Bases de Datos llamado PostgreSQL con una extensión de datos espaciales PostGIS.

- **Capa de negocio:** está encargada de realizar toda la lógica y operaciones transparentes al usuario tanto la comunicación con la aplicación, como la comunicación con la base de datos, operaciones de parseo, transformación, mapeo y tratamiento de datos y procesos en segundo plano para la comparación de rutas.

- **Capa cliente:** formada por la aplicación móvil que se detallará en el siguiente apartado.

6.1.2 Arquitectura Lógica de la aplicación

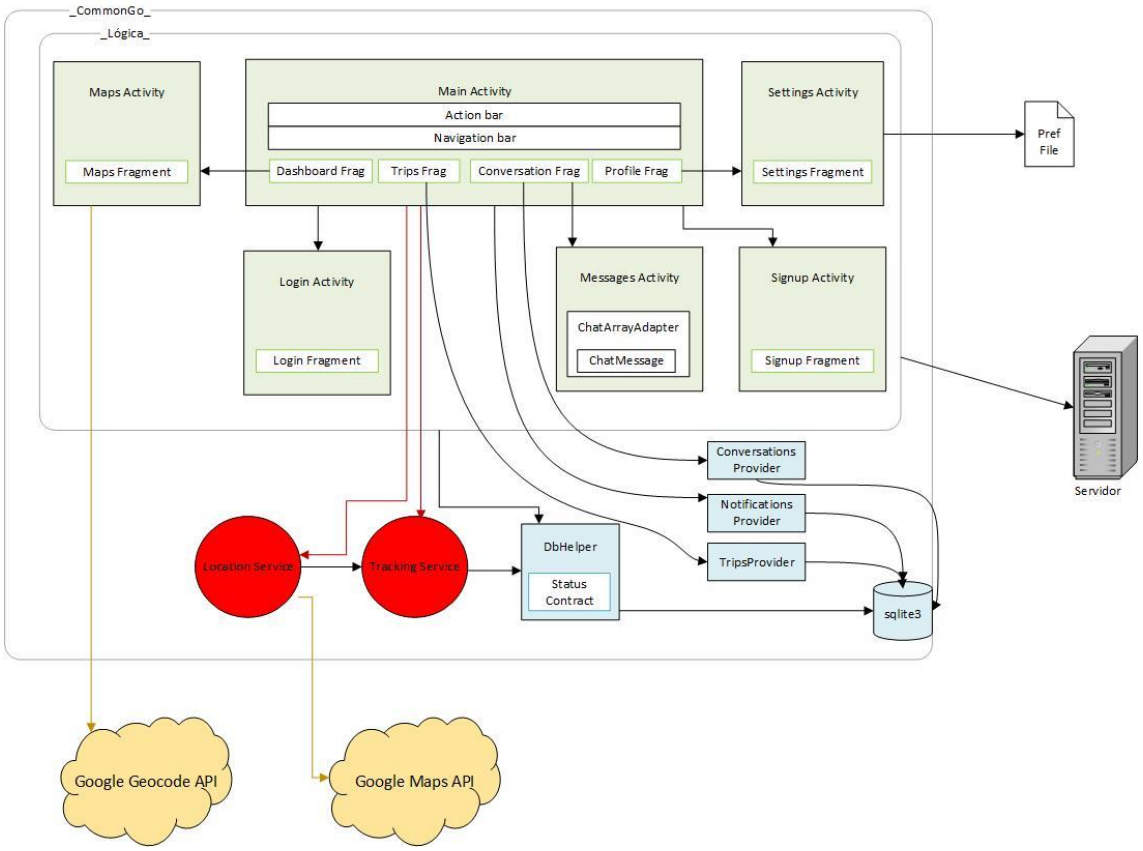


Imagen 13 - Arquitectura lógica de la aplicación

La estructura de la aplicación se resume en lo siguiente:

- **Aplicación**: componentes de la aplicación
    - **Lógica**: componentes lógicos de la aplicación, definen el comportamiento de la aplicación y las conexiones entre los demás componentes.
        - **MapsActivity**: pantalla de mapas, define la lógica de esta pantalla.
            - **MapsFragment**: fragmento que contiene las vistas de la pantalla de mapas
        - **MainActivity**: pantalla principal de la aplicación
            - **DashboardFragment**: fragmento que contiene las vistas de la pantalla *tablón*
            - **TripsFragment**: fragmento que contiene las vistas de la pantalla *viajes*
            - **ConversationFragment**: fragmento que contiene las vistas de la pantalla *conversaciones*
            - **ProfileFragment**: fragmento que contiene las vistas de la pantalla *perfil*
        - **SettingsActivity**: pantalla de ajustes, contiene la lógica de esta pantalla
            - **SettingsFragment**: contiene las vistas de la pantalla de *ajustes*
        - **LoginActivity**: pantalla de inicio de sesión, contiene la lógica de esta pantalla
            - **LoginFragment**: contiene las vistas de la pantalla de *inicio de sesión*
        - **SignupActivity**: pantalla de registro, contiene la lógica de esta pantalla
            - **SignupFragment**: contiene las vistas de la pantalla de *registro*
        - **MessagesActivity**: pantalla de mensajes, contiene la lógica de esta pantalla
            - **ChatArrayAdapter**: contiene el conjunto de los **ChatMessage**
    - **Servicios**: servicios que se ejecutan en segundo plano
        - **LocationService**: se encarga de geolocalizar continuamente la posición del usuario
        - **TrackingService**: encargado de registrar la posición del usuario
    - **DbHelper**: conexión con la base de datos local
        - **StatusContract**: define los campos, tablas, atributos de la base de datos local

- **ConversationsProvider**: provider que se comunica con la base de datos local y carga los datos en el ConversationsFragment
- **TripsProvider**: provider que se comunica con la base de datos local y carga los datos en el TripsFragment
- **NotificationsProvider**: provider que se comunica con la base de datos local y carga los datos en el DashboardFragment
- **BBDD**: base de datos sqlite3 local

- **APIs**: lógica de las interfaces de programación de aplicaciones de Google utilizadas
  - **Google Maps API**: API para cargar y manejar mapas, así como la geolocalización
  - **Google Geocode API**: API para convertir ubicaciones en direcciones físicas

- **Servidor:** definido en el punto 5.1.3 Arquitectura Lógica del Sistema

6.2 ARQUITECTURA FÍSICA

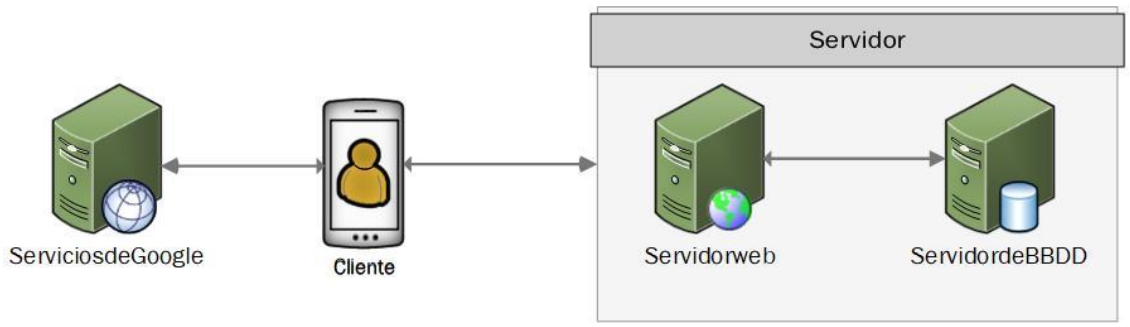


Imagen 14 - Arquitectura física

La arquitectura física de la aplicación es un modelo Cliente-Servidor clásico en tres capas como se ha explicado anteriormente, que utiliza como añadido los servicios de Google.

6.3 DIAGRAMA DE CLASES (DISEÑO)

Los diagramas de clases de diseño permiten conocer las principales clases que forman parte del sistema. En ellos se muestran las clases que conforman el sistema y las relaciones entre ellas.

6.3.1 Servidor

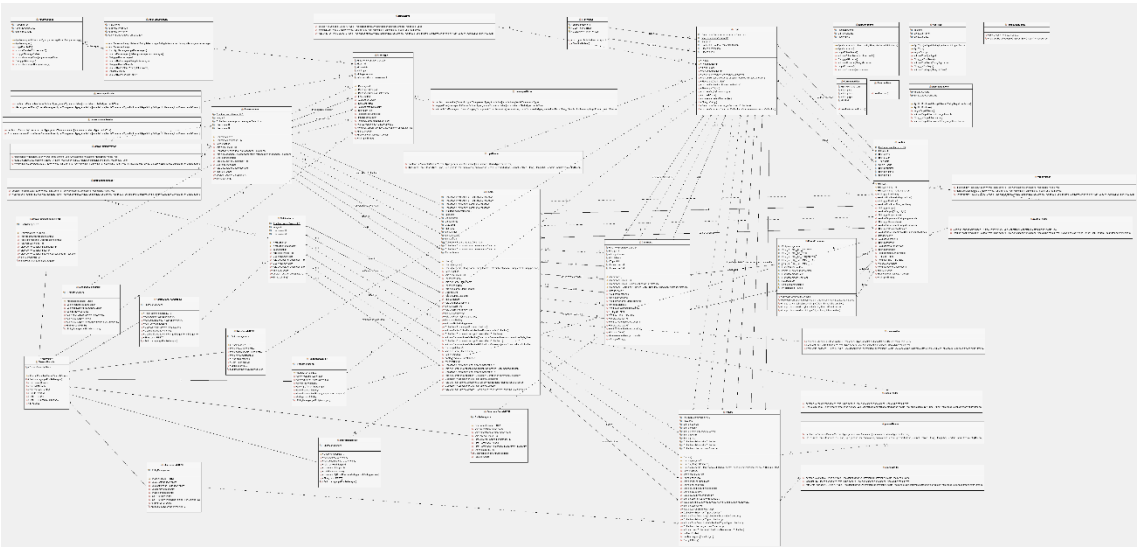


Imagen 15 - Diagrama de Clases del servidor

6.3.2 Aplicación

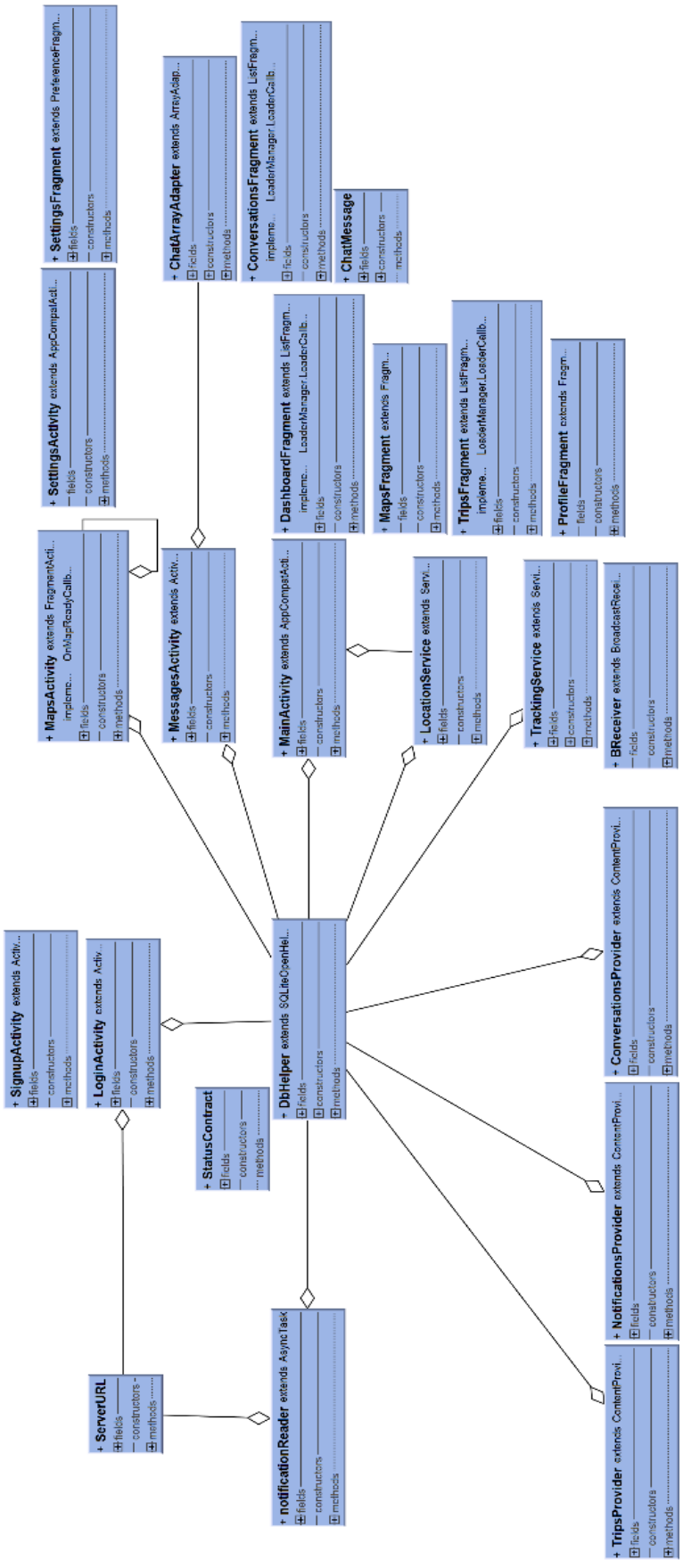


Imagen 16 - Diagrama de Clases de la aplicación

6.4 DIAGRAMA(S) DE SECUENCIA

Los diagramas de secuencia muestran la interacción entre los conjuntos de objetos que participan en una operación a través del tiempo:

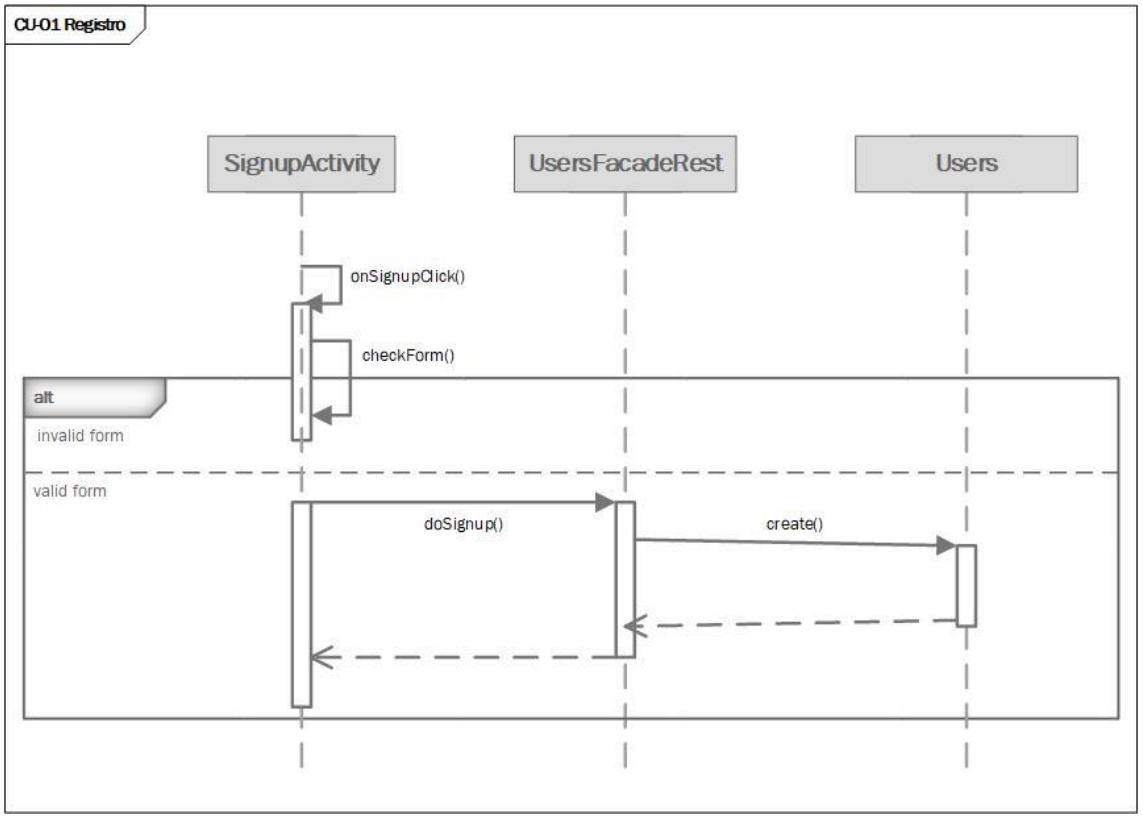


Imagen 17 - CU-01 - Diagrama de secuencia

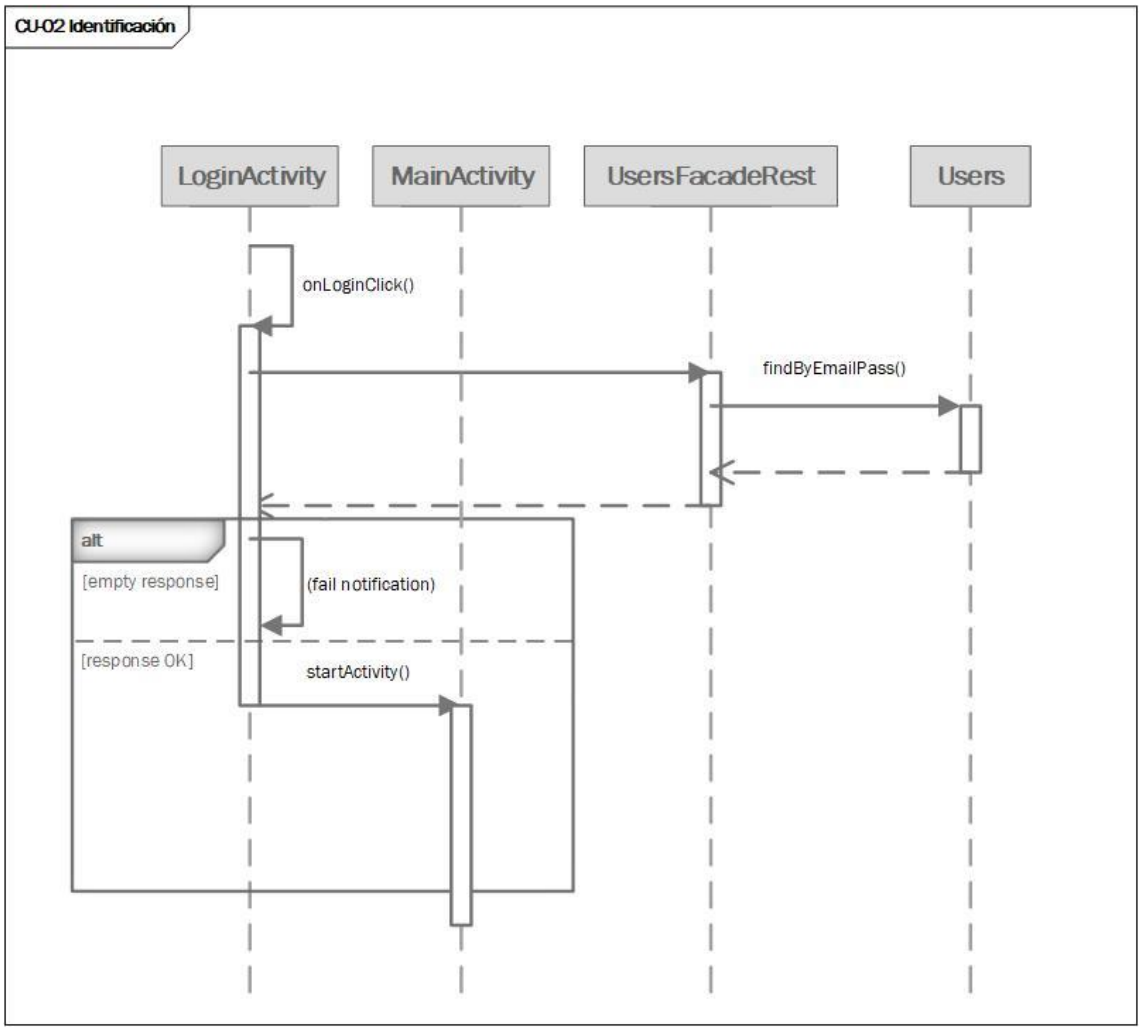


Imagen 18 - CU-02 - Diagrama de secuencia

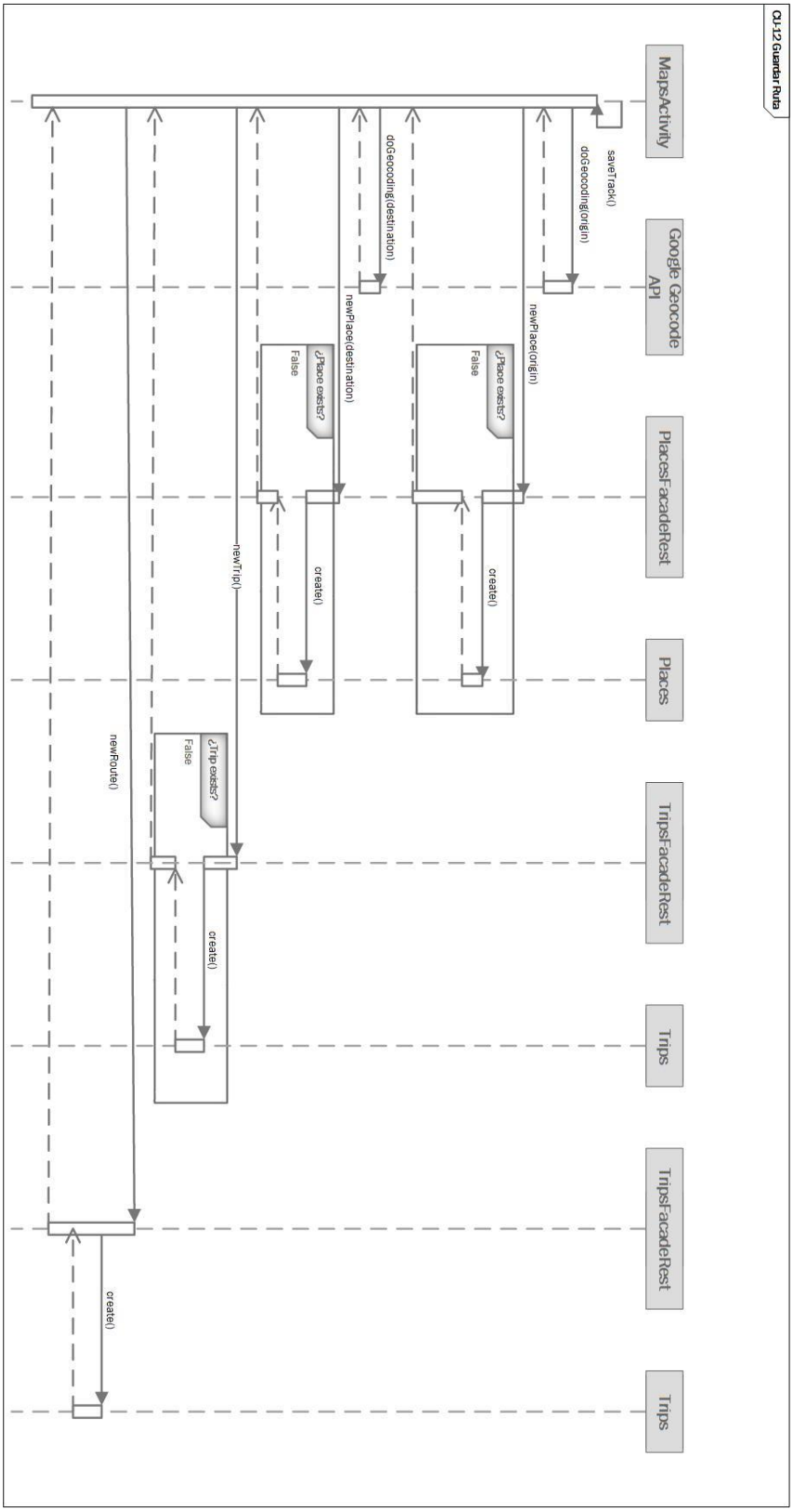


Imagen 19 - CU-12 - Diagrama de secuencia

6.5 DISEÑO LÓGICO

Imagen 20 - Diseño Lógico

La estructura está definida en 8 tablas. Por un lado, tenemos la de usuarios, que almacena los datos personales de cada usuario, nombre, apellidos, teléfono, fecha de nacimiento, foto de perfil.

La tabla notificaciones es la que lee la aplicación cuando un usuario quiere conversar con otro, se relaciona doblemente con la tabla usuarios, ya que tiene que almacenar quién quiere hablar con quién. Cuando los usuarios inician la conversación, se borran de aquí los datos y se almacenan en la tabla conversaciones, que tiene los mismos atributos y se relaciona también doblemente con la tabla de usuarios.

Esta tabla de conversaciones está referenciada por la de mensajes, que almacena los mensajes y la fecha y hora de cada uno.

La tabla commons es la encargada de almacenar qué rutas de qué usuarios coinciden con qué rutas de qué otros usuarios. Para ello se relaciona doblemente con la tabla usuarios y una vez con la tabla de viajes. Almacenando así los dos usuarios que tienen la ruta en común, el viaje que tienen en común, el nombre del viaje (origen-destino) y la distancia en la que difieren las rutas, en metros.

Después tenemos la tabla sitios (places) que únicamente almacena el id de Google para esa ubicación y el nombre de dicho lugar (país, provincia y municipio). A esta tabla la referencia doblemente la tabla viaje, almacenando el origen y el destino del viaje. Esta tabla de viajes está referenciada por la de rutas, almacenando así la clave del viaje del que forma parte, también hace referencia a la tabla de usuarios para saber de quién es la ruta, la ruta como un string de puntos geográficos, y los días en los que va el usuario.

6.6 DISEÑO DE INTERFAZ

	IDENTIFICACIÓN (LOGIN)
Descripción	Página de inicio de la aplicación, los elementos que contiene son: • Logotipo • Cajón de texto para el email • Cajón de texto para la contraseña • Botón de inicio de sesión • Referencia a la pantalla de registro
Activación	Automáticamente cuando un usuario abre la aplicación y no está identificado. Pulsando en *iniciar sesión (login)* desde la pantalla de registro
Boceto	
Eventos	• Identificación de un usuario ➢ Acceso a la página principal ➢ Resultado de la acción fallida de identificación (usuario no existe, no hay conexión a internet, etc) • Acceso a la pantalla de registro

Tabla 50 - Diseño de la pantalla de login

	REGISTRO (SIGN UP)
Descripción	Pantalla de registro en el CommonGo, en ella podremos encontrar el logo de la aplicación, el formulario que es necesario rellenar para el registro, un botón que hace efectivo el registro enviando los datos al servidor, y un enlace a la pantalla de identificación (login).
Activación	Pulsar en *registrarse* en la pantalla de identificación (login)
Boceto	
Eventos	• Registro en la aplicación ➢ Acceso a la pantalla de identificación (login) tras el registro ➢ Resultado fallido de registro (campo inválido, no hay conexión a internet, etc) • Vuelta a la pantalla de identificación (login)

Tabla 51 - Diseño de la pantalla de registro

	PANTALA PRINCIPAL (HOME)
Descripción	Pantalla de inicio de la aplicación cuando el usuario abre la aplicación y está registrado, o se identifica en la pantalla de login. La aplicación se abre por defecto aquí. En esta pantalla podremos ver los distintos botones a las pantallas de *mapas, mensajes y perfil,* así como las notificaciones que reciba el usuario.
Activación	Pulsar en el botón *inicio (home)* Abrir aplicación con usuario identificado Identificarse
Boceto	
Eventos	• Nueva notificación • Interactuar con la notificación • Acceso a la pantalla de mensajes • Acceso a la pantalla de perfil • Acceso a la pantalla de mapas

Tabla 52 - Diseño de la pantalla principal

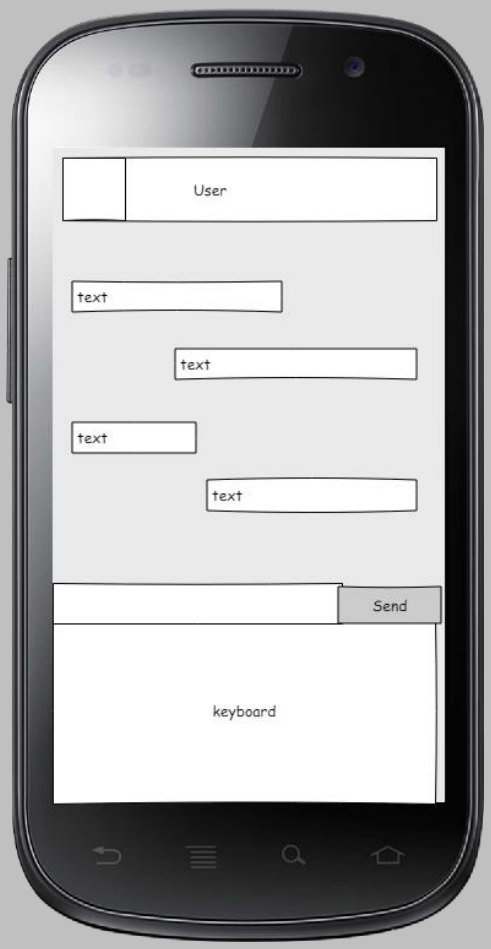
	PANTALLA MENSAJES (MESSAGES)
Descripción	Esta página es un chat, contiene una identificación del usuario con el que se está hablando, así como los mensajes enviados, y una ventana de texto desde donde se pueden mandar mensajes nuevos al usuario.
Activación	Pulsar en la ventana de mensajes Confirmar conversación
Boceto	
Eventos	• Enviar mensaje • Recibir mensaje

Tabla 53 - Diseño de la pantalla de mensajes

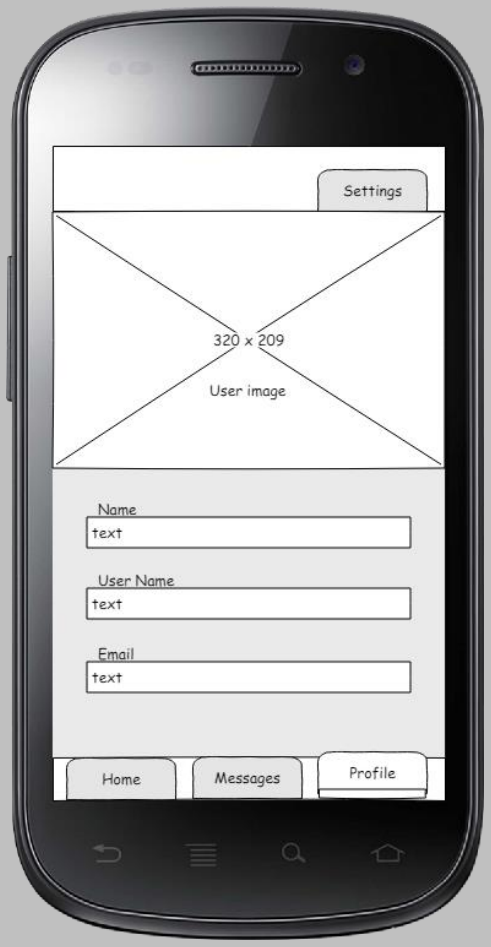
	PANTALLA PERFIL (PROFILE)
Descripción	Pantalla correspondiente al perfil, en esta vista podemos encontrar los datos del usuario identificado con los cuales se registró en CommonGo, la foto del mismo si la ha actualizado, y un acceso a la pantalla de ajustes.
Activación	Pulsar sobre el botón de *perfil (profile)* en la barra de navegación inferior.
Boceto	
Eventos	• Acceso a la pantalla de *ajustes (settings)* • Editar perfil • Editar foto • Acceso a la pantalla de *inicio (home)* • Acceso a la pantalla de *mensajes (messages)*

Tabla 54 - Diseño de la pantalla de perfil

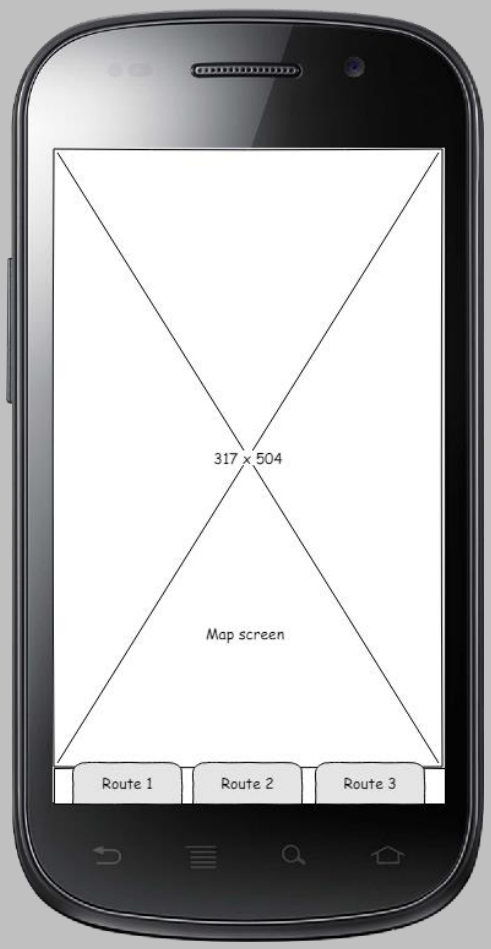
	PANTALLA MAPAS (MAPS)
Descripción	Pantalla donde se registran las rutas que realiza el usuario, en ella podremos ver, en la parte inferior, las distintas rutas que el usuario ha guardado y, en el resto de la pantalla, un mapa donde se ilustrará la ruta que el usuario realice.
Activación	Pulsar sobre en el botón *mapas* en la pantalla de *inicio (home)*
Boceto	
Eventos	• Seleccionar una ruta • Guardar ruta • Volver a la pantalla de *inicio (home)*

Tabla 55 - Diseño de la pantalla de mapas

PANTALLA AJUSTES (SETTINGS)	
Descripción	Pantalla donde el usuario podrá parametrizar todos los ajustes de la aplicación que estén sujetos a cambios, como tiempo de rastreo de ruta, vibración en las notificaciones, etc.
Activación	Pulsar sobre el botón de *ajustes (settings)* en la pantalla del perfil
Boceto	
Eventos	• Cambiar ajuste • Volver a la pantalla de *perfil (profile)*

Tabla 56 - Diseño de la pantalla de ajustes

A continuación, se muestran capturas finales de estas páginas bocetadas, así como de las páginas que ha sido necesario crear para ofrecer una funcionalidad completa de la aplicación:

La CommonGo tiene una única interfaz principal, sin hacer distinción por tipo de usuario. Aunque un usuario no registrado no tendrá acceso a la totalidad de la misma, a excepción de las pantallas de registro (sign in) e identificación (login).

Un usuario sin identificar que abra CommonGo obtendrá una vista de la pantalla de ***inicio de sesión o login***, donde podrá identificarse mediante su email y contraseña para acceder al resto de la aplicación.

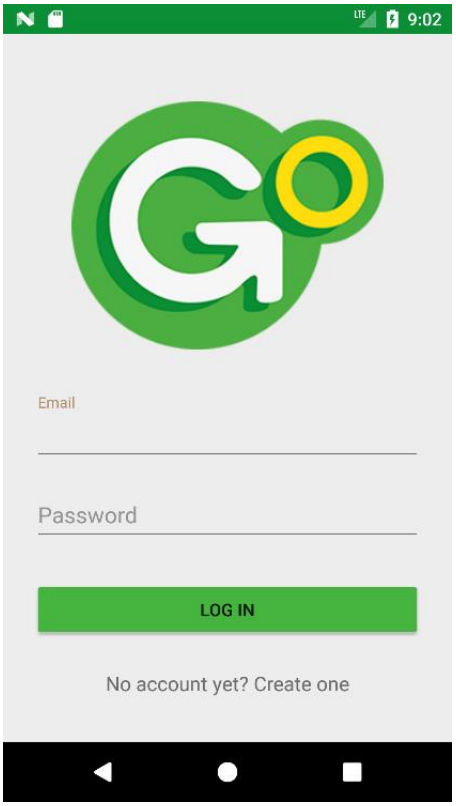


Imagen 21 - Captura final login

O bien, desde esa misma vista, puede acceder a la pantalla de ***registro (register)*** donde podrá introducir sus datos personales para poder identificarse más adelante.

Desde esta pantalla también se puede volver a la pantalla anterior si el usuario no quiere registrarse o ha pulsado sin querer.

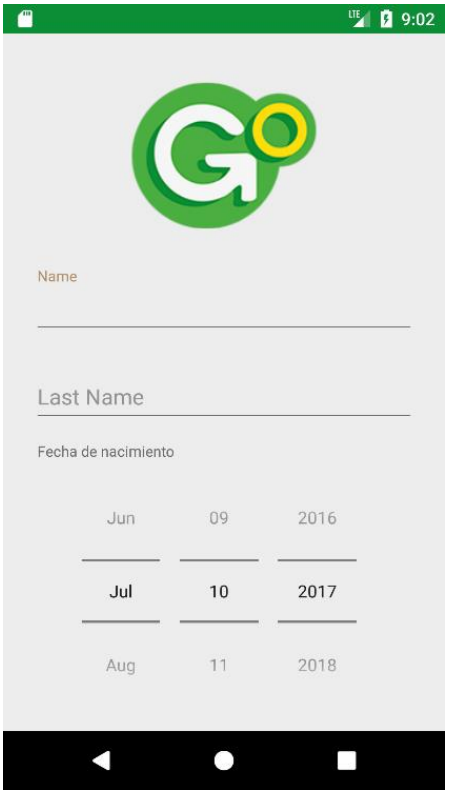


Imagen 22 - Captura final registro

Si un usuario sin identificar introduce sus datos (válidos) tendrá acceso al resto de la interfaz. Inicialmente, el usuario accede a la página de notificaciones (originalmente llamada *inicio (home)*), llamada ***tablón (dashboard)***, que hace las veces de página principal. En esta pantalla se ven las notificaciones principales de la aplicación, como una ruta nueva disponible, un usuario nuevo compatible, etc. Desde esta pantalla, el usuario podrá moverse a las pantallas de conversaciones (originalmente llamada mensajes), viajes, perfil y mapas. Además de disponer de la opción de empezar/parar a registrar una ruta.

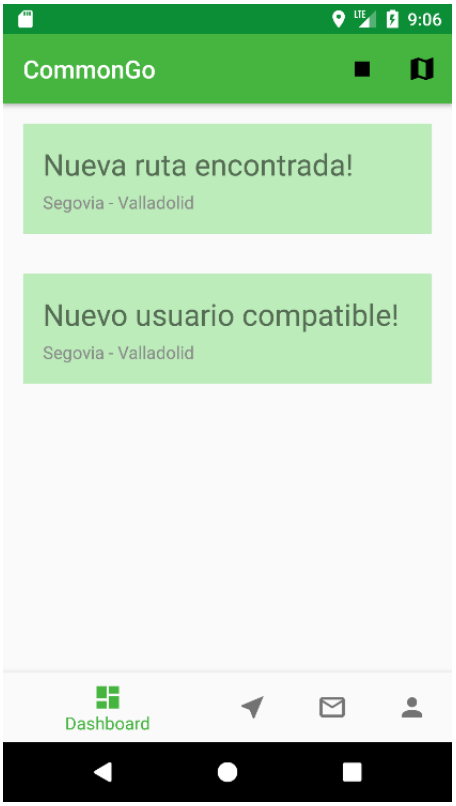


Imagen 23 - Captura final Tablón

En la pantalla de ***conversaciones (conversations)***, el usuario logueado podrá ver las conversaciones que tiene con otros usuarios, pulsando en alguna de ellas, accederá a la pantalla de mensajes o chat con ese usuario. Desde esta pantalla, el usuario sólo podrá volver a la pantalla de conversaciones. Desde la pantalla de conversaciones, el usuario podrá ir a las pantallas tablón, viajes y perfil.

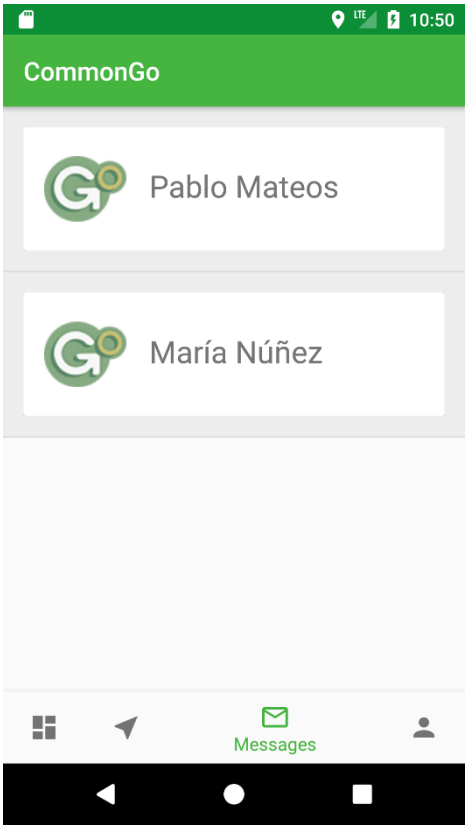


Imagen 24 - Captura final Conversaciones

En la pantalla de ***perfil (profile)***, el usuario podrá ver sus datos de perfil, así como añadir nuevos datos y modificar los ya existentes, incluída la foto. Desde esta pantalla, el usuario podrá moverse a las pantallas de tablón, conversaciones, viajes, y ajustes.

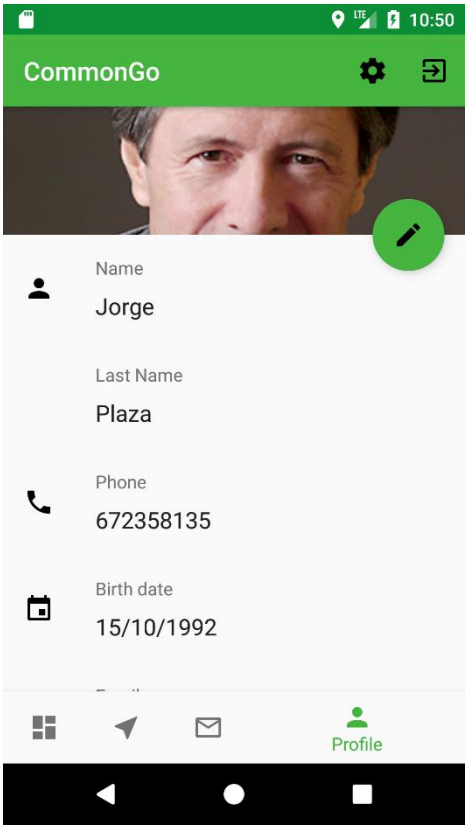


Imagen 25 - Captura final Perfil

En la pantalla de ***viajes (trips)***, el usuario verá los viajes que ha guardado, pudiendo cambiar el nombre y días que hace esa ruta. Desde esta pantalla, el usuario podrá moverse a las pantallas de tablón, mensajes y perfil.

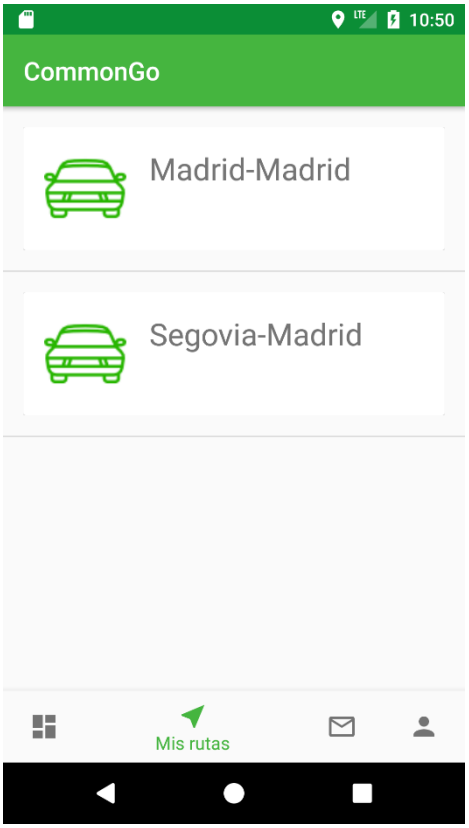


Imagen 26 - Captura final Viajes

En la pantalla de ***ajustes (settings)***, el usuario podrá modificar los ajustes parametrizables. Desde esta pantalla, el usuario sólo podrá moverse de vuelta a la pantalla del perfil.

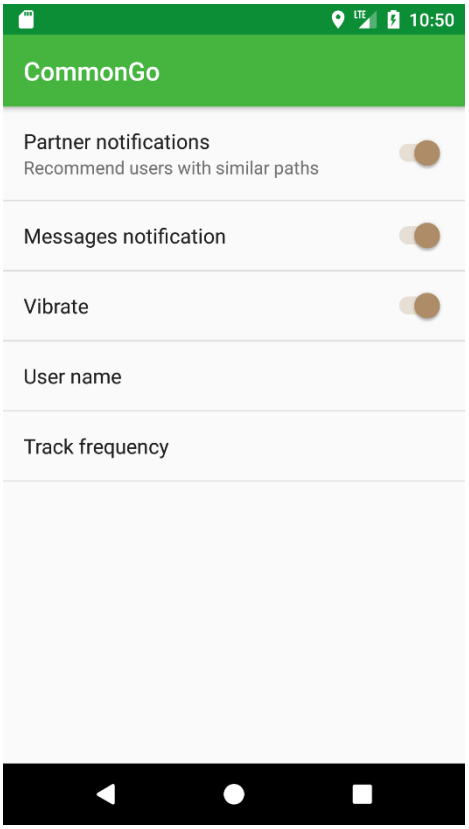


Imagen 27 - Captura final Ajustes

Cuando el usuario carga la pantalla de ***mapas (maps)***, inicialmente se cargará el mapa con la posición actual del dispositivo. En ella tendrá en la parte inferior una lista donde verá su última ruta grabada, y las rutas guardadas. Pulsando sobre cualquiera de ellas se dibujarán en el mapa. Además, el usuario podrá guardar su última ruta grabada y ver su posición actual de nuevo. Desde esta pantalla el usuario sólo podrá volver a la pantalla del tablón.

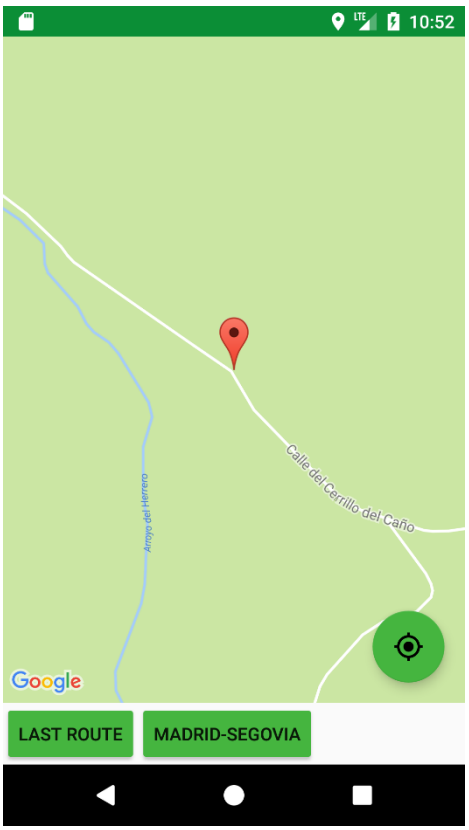


Imagen 28 - Captura final Mapas

7 IMPLEMENTACIÓN

En este capítulo pasamos a describir cómo se ha llevado a cabo la implementación del proyecto en sí. Para ello, se va a realizar primeramente una descripción técnica del funcionamiento, procesos dentro de la herramienta y aspectos innovadores de la misma; después de hablará de las herramientas y tecnologías utilizadas para sacar adelante el proyecto; a continuación se ilustrará cómo se puede crear una base de datos en PostGIS y uno de los aspectos más relevantes del proyecto: el cálculo de distancias; para acabar de detallará la estructura del proyecto y, por último, se añadirán los detalles clave o que se consideran de mayor importancia y relevancia del proyecto, en cuanto a la implementación se refiere.

7.1 DESCRIPCIÓN TÉCNICA

La herramienta desarrollada tiene como una de las funcionalidades principales la captura y registro de la ubicación del usuario. Esta funcionalidad se lleva a cabo primeramente gracias a la API de Google Maps, que es capaz de informarnos en tiempo real de la ubicación del usuario. Nuestra herramienta se comunica en todo con esta API gracias a un servicio propio que, cuando el usuario lo requiera, es capaz de comunicarse con un segundo servicio de la aplicación que se encarga de dejar registro en la base de datos local de la aplicación.

A través de la misma API de Google Maps, la aplicación también tiene la funcionalidad de mostrar un mapa con la ubicación actual del usuario en un activity, así como mostrar los registros de ubicaciones (que hemos llamado viajes/rutas) y el último trayecto (que hemos llamado última ruta).

Otra de las funcionalidades principales de la herramienta, es la que se encarga de la manipulación de las rutas de los usuarios en el servidor. Gracias al uso del Sistema Gestor de Bases de Datos elegidos con la extensión de PostGIS, esta tarea se hace mucho más sencilla, ya que nos permite realizar cálculos directamente en la base de datos y sobre las coordenadas geográficas de cada puto de una ruta.

Para apoyar esta funcionalidad, la aplicación cuenta con un sistema de mensajería, que permite que dos usuarios que tienen una ruta en común puedan ultimar los detalles para compartir trayecto una vez ambas partes han confirmado que desean hablar con la otra. La aplicación realiza una llamada al servidor según un tiempo parametrizable por el usuario. Esta parametrización se lleva a cabo mediante el uso de Preferencias Compartidas.

La comunicación con el servidor se hace a través de llamadas HTTP usando la clase Android HttpUrlConnection, que nos permite realizar llamadas asíncronas (GET, POST, PUT y DELETE) directamente a los servicios Rest implementados en el servidor, y recibir las respuestas dentro de la misma llamada.

Para la correcta comunicación, tanto aplicación como servidor envían y reciben los datos en formato JSON. Esto se consigue mediante el uso de parseadores o parsers para JSON, que convierten los datos que viajan en este formato al formato que más convenga (según nos encontremos en el servidor o en la aplicación) y viceversa.

La obtención de datos geográficos se hace a través de la API de Google Geocode, la cual nos permite obtener las coordenadas de una población o área concreta a partir de su nombre, código postal, etc. Aunque también nos permite obtener direcciones en lenguaje natural a partir de unas coordenadas en concreto, este proceso se conoce como geocodificación inversa.

Para el almacenamiento de los datos se utiliza:

- **SQLite** para los datos locales de la aplicación, que es una versión ligera para Android del popular SGBD MySQL. La comunicación con esta base de datos es muy rápida y sencilla, ya que es nativa del Sistema Operativo Android.
- **PostgreSQL (+PostGIS)** para os datos generales de la herramienta, que ya se ha explicado anteriormente (ver apartado 2.1 GIS) en qué consiste y porqué se utiliza en este proyecto. La comunicación entre el sistema y este SGBD se hace a través de unas entidades existentes en el proyecto del servidor que *mapean* los datos entre la base de datos y los servicios Rest.

7.2 Requisitos de Hardware y Software

En este punto hay que diferenciar dos partes, servidor y smartphone. A continuación, se presentan los requisitos mínimos que deberá tener cada parte para poder usar la herramienta.

		Servidor	**Smartphone**
HW	CPU	2.3GHz Quad-Core	1.2GHz Dual-Core
	RAM	6Gb	1Gb
	Otros	Conexión a internet	Conexión a internet, GPS
SW	SO	Windows 10	Android API 15 o superior
	Otros	-	-

Tabla 57 - Requisitos Hardware y Software

7.3 Herramientas Empleadas

En este apartado se va a listar y explicar, aunque brevemente, todas las herramientas y tecnologías que se han utilizado (bien sea en la parte cliente o en la parte servidor) del proyecto:

- **Android OS:** sistema operativo basado en Linux. Está diseñado principalmente para dispositivos móviles. Su coste temporal de desarrollo es rápido y el mantenimiento económico, además de tener la mayor cuota de mercado (en torno al 86.8% en el tercer cuarto de 2016 según International Data Corporation (IDC).

- **Android Studio (v2.3.3):** es el entorno de desarrollo intgrado (IDE) oficial para el desarrollo el SO Android. Entre muchas ventajas, destaca su absoluta compatibilidad con el Sistema Operativo, el renderizado o emulación de las aplicaciones en tiempo

real, el soporte oficial en su página para desarrollo (https://developer.android.com) y la gran cantidad de documentación y guías que ofrece la misma, etc.

- **SQLite (v3):** como ya se ha explicado brevemente en el apartado anterior, es un SGBD compatible con Android OS. Es relacional y compatible con ACID (Atomicity, Consistency, Isolation, Durability).

- **Google Maps API (v11.0.1):** Interfaz de Programación de Aplicaciones para incorporar las funcionalidades de Google Maps.

- **Google Geocode API:** Interfaz de Programación de Aplicaciones de Google que permite convertir direcciones en coordenadas geográficas, y viceversa.

- **JSON:** formato de texto ligero abreviado de JavaScript Object Notation. Es una clara alternativa a XML dado que su notación es efectiva para "la máquina" y de fácil lectura para el lenguaje humano.

- **Netbeans (v8.2):** IDE libre multiplataforma de código abierto, principalmente orientado al desarrollo con el lenguaje de programación Java.

- **JavaEE7:** plataforma de programación principalmente orientada a desarrollar y correr software de aplicaciones Java. Permite utilizar arquitecturas de n capas distribuidas y se ejecuta sobre un servidor de aplicaciones.

- **Maven:** software para la gestión y construcción de proyectos Java, con un modelo de construcción muy simple basado en un formato XML. Utiliza un POM (Project Object Model) para describir el proyecto a construir.

- **Glassfish (v4.1):** servidor de aplicaciones software libre. Permite ejecutar aplicaciones que siguen la especificación JavaEE.

- **PostgreSQL:** es un SGBD relacional, orientado a objetos y libre.

- **PostGIS:** extensión de PostgreSQL que lo convierte en una base de datos espacial manteniendo sus características de base de datos empresarial y dotándolo de soporte de objetos geográficos.

Dependencias en el servidor:

- **Batch (javax.batch-api-all-1.0-b17):** dependencia que nos permite desarrollar un proceso que se ejecuta por lotes (batch processing).

- **Postgresql (org.postgresql-42.1.1):** dependencia que nos permite operar con la BD de PostgreSQL.

- **Postgis (postgis-jdbc-2.1.1):** dependencia que funciona como una extensión o complemento de la anterior y da soporte a los tipos de datos espaciales de la base de datos en PostGIS.

7.4 GIS

Un GIS (Geographic Information System) es, literalmente, un Sistema de Información Geográfica. Un conjunto de herramientas que utiliza varios componentes (usuarios, hardware, software, procesos) con el fin de analizar, almacenar, manipular y modelar datos reales con una referencia espacial.

Los GIS nos permiten visualizar, analizar e interpretar datos para entender relaciones, patrones y tendencias de los mismos. Actualmente hay un creciente interés en el valor económico y estratégico de los GIS por parte de las empresas.

En CommonGo, para beneficiarnos de estas herramientas, haremos uso de un gestor de bases de datos espacial llamado PostGIS. El cual parte de un sistema de gestión de bases de datos relacional orientado a objetos y libre como es PostgreSQL y sobre él utiliza la tecnología de GIS, que añade básicamente tres características: tipos de datos espaciales, índices espaciales y funciones que puedan operar sobre ellos.

Al estar construido sobre PostgreSQL, nuestro gestor de BBDD espaciales hereda las características de las BBDD empresariales, así como los estándares abiertos que implementan un GIS dentro del motor de bases de datos.

7.4.1 ¿Por qué utilizar PostGIS?

PostGIS es una extensión de PostgreSQL que le confiere un carácter espacial, esta extensión ha cobrado especial importancia en los últimos años. Al combinar ambos, conseguimos una solución óptima para el almacenamiento, gestión y procesamiento de datos espaciales.

Además, PostGIS tiene una serie de características que hacen que sea único:

- Para empezar, PostGIS es software libre, tiene licencia GNU General Public License (GLP) y, por tanto, es totalmente gratuito.
- Es compatible con los estándares de OGC (Open Geoespatial Consortium), para facilitar el intercambio de datos de información geográfica.
- Soporta datos e índices espaciales, y tiene más de mil funciones para poder trabajar con ellos.
- Facilita la importación y exportación de datos a través de varias herramientas conversoras, como pueden ser *shp2pgsql*, *pgsql2shp*, *ogr2ogr*, *dxf2postgis*.
- Existe un gran número de herramientas de escritorio que pueden trabajar con datos de PostGIS, entre ellas *QGIS*, *gvSIG* y *GeoServer*.
- Al tener de carácter libre, es claramente una alternativa al software propietario.

- En la actualidad, es la base de datos espacial de código abierto más utilizada. Muchas organizaciones de todo el mundo utilizan PostGIS. Registrando más de 800 descargas mensuales de su código fuente.

7.5 CREACIÓN DE LA BASE DE DATOS EN POSTGIS

En este apartado se detallará el proceso que hay que seguir para crear una BBDD en PostgreSQL con la extensión espacial PostGIS, además de cómo poblarla.

El procedimiento empieza igual que si creamos una base de datos en PostgreSQL:

```
"CREATE DATABASE commongo
    WITH
    OWNER = postgres
    TEMPLATE = template1
    ENCODING = 'UTF8'
    LC_COLLATE = 'Spanish_Spain.1252'
    LC_CTYPE = 'Spanish_Spain.1252'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;"
```

Detallar todos los atributos no es obligatorio, únicamente la primera línea de la sentencia es obligatoria, pero se detallan todos los atributos a modo de ejemplo de una sentencia completa. Esta sentencia es la que se ha utilizado para crear la base de datos real del servidor, cuyos atributos definen lo siguiente:

- OWNER: usuario que tiene los permisos de administración del esquema.
- TEMPLATE: plantilla de creación de la base de datos, *template1* indica por defecto.
- ENCODING: formato de codificación de caracteres.
- LC_COLLATE: soporte local (caracteres según región, ordenación, formateo de fechas…)
- LC_CTYPE: ordenación local de caracteres.
- TABLESPACE = lugar de almacenamiento de la BBDD en el sistema.
- CONNECTION LIMIT: límite de conexiones concurrentes a la BBDD.

Una vez creada la base de datos, tenemos que crear la extensión para PostGIS con una simple sentencia:

```
"CREATE EXTENSION postgis;"
```

Llegados a este punto, sólo quedaría poblar la base de datos. El DDL o sentencias CREATE TABLE necesarias para hacer esto no son diferentes de una base de datos en PosgreSQL normal:

```
"CREATE TABLE places(
    id SERIAL PRIMARY KEY,
    google_id varchar NOT NULL UNIQUE,
    country varchar NOT NULL,
    administrative_area_level_2 varchar NOT NULL,
    locality varchar NOT NULL
);"
```

A diferencia de que las columnas que queramos que sean de tipo geográfico, tienen que ser añadidas a mayores:

```
"SELECT AddGeometryColumn('places','place', -1,'POINT', 2);"
```

Donde el primer elemento hace referencia a la tabla donde se quiere introducir la columna espacial, seguido del nombre de la columna, el SRID (un sistema de referencia de coordenadas), el tipo de GEOMETRY o dato geográfico que es y el número de dimensiones que tiene el atributo geofráfico.

Los tipos de datos geográficos que admite PostGIS son:

- **POINT**: coordenada o punto geográfico.
- **LINESTRING**: línea de POINT, conjunto ordenado de puntos.
- **POLYGON**: línea cerrada de un conjunto de POINT.
- **MULTIPOINT**: conjunto discreto de POINT.
- Etc

7.6 CÁLCULO DE DISTANCIAS EN POSTGIS

El cálculo de distancias en PostGIS entre dos *geometries* es posible gracias a las funciones que esta extensión de PostgreSQL incluye de manera nativa. Este tipo de cálculo se puede hacer entre cualquier tipo de figura geográfica o geométrica, ya que cualquiera de las siguientes funciones retornará la distancia mínima entre dos *geometries*, ya sean dos puntos, dos líneas, un cuadrado y un punto, etc:

- **ST_Distance:** devuelve la distancia (en metros) mínima bidimensional entre dos geometrías en formato float. Su sintaxis es la siguiente:
    - ST_Distance(geo gg1, geo gg2, boolean use_spheroid);

  Donde los dos primeros parámetros admiten tanto una geometría como un punto geográfico y son de carácter obligatorio, mientras que el tercero es opcional y determina si se usará un esferoide para el cálculo.

- **ST_DistanceSphere:** devuelve la distancia mínima (en metros) entre dos puntos geográficos en formato float. Para realizar los cálculos usa la curvatura y radio de la tierra según el SRID proporcionado en los puntos.
    - ST_DistanceSphere(geometry lonlat1, geometry lonlat2);

  Los dos puntos introducidos en los parámetros deben contener el SRID.

- **ST_DistanceSpheroid:** devuelve la distancia mínima (en metros) entre dos puntos geográficos en formato float a partir de un esferoide determinado:
    - ST_DistanceSpheroid(geo lonlatA, geo lonlatB, spheroid measurement_spheroid);

En nuestro caso hemos hecho uso de la segunda opción, ya que nos calcula distancias entre dos puntos teniendo en cuenta la curvatura de la tierra y, aunque en un viaje de Madrid a Segovia no es relevante, la distancia calculada difiere de la recorrida cuanto más largo ha sido el viaje. La sentencia completa que hemos construido puede verse en el punto Cálculo de distancias en PostGIS.

7.7 ESTRUCTURA DEL PROYECTO

En esta sección se detallará cómo se organizan los proyectos (aplicación y servidor) a nivel de ficheros.

7.7.1 Servidor

Todos los ficheros se encuentran en el directorio:

- **~/Documents/NetBeansProjects/commongo2/src/main/**

A partir del cual, el proyecto se organiza de la siguiente manera, sólo se detallan los archivos principales o relevantes del proyecto:

- **java**/
    - **batch**: comprende las clases que, trabajando juntas, ejecutan el proceso en segundo plano por lotes encargado de comparar las rutas de los usuarios y obtener los resultados. El proceso se inicia con el CommonBean, que define el xml (más abajo se explica) que a su vez define la secuencia con la que se ejecuta: Reader → Processor → Writer
        - CommonBean.java

        - CommonWriter.java
        - RouteProcessor.java
        - TripReader.java

    - **entities**: comprende todas las entidades que mapean los datos con la base de datos:
        - Commons.java
        - Conversations.java
        - Messages.java
        - Places.java
        - Routes.java
        - Trips.java
        - Users.java

    - **json**: aquí se encuentran las clases que se encargan de la comunicación entre la aplicación y el servidor a través de JSON. Los *readers* traducen los JSON provenientes de la aplicación a los servicios Rest, mientras que los *writers* hacen el proceso contrario:
        - placesReader.java
        - placesWriter.java
        - routesReader.java
        - routesWriter.java
        - tripsReader.java
        - tripsWriter.java
        - usersReader.java
        - usersWriter.java

    - **rest**: los servicios Rest son la conexión directa con la aplicación móvil, se encargan de recibir todas las peticiones HTTP provenientes de la misma, y traducir las peticiones a transacciones con la BBDD, procesos internos, y mandar una respuesta de vuelta:
        - AbstractFacade.java
        - CommonsFacadeREST.java
        - ConversationsFacadeREST.java
        - MessagesFacadeREST.java
        - PlacesFacadeREST.java
        - RoutesFacadeREST.java
        - TripsFacadeREST.java
        - UsersFacadeREST.java

- **Resources/**
    - **META-INF.batch-jobs**: aquí se encuentra el fichero xml que define el proceso por lotes contenido en el directorio de batch
        - eod-commons.xml

7.7.2 Aplicación

Todos los ficheros se encuentran en la ruta:

- **~/AndroidStudioProjects/Commongo/app/src/main/**

A partir de la cual, la aplicación se estructura de la siguiente manera (sólo se explican los directorios y ficheros más importantes para el funcionamiento de la misma):

- **Java/com/tfg/jorge/commongo/**
    - **Activities**: contiene las clases que realizan la mayor parte de la lógica de la aplicación y, a su vez, son contenedoras de la UI con la que interactúa el usuario:
        - LoginActivity
        - MainActivity
        - MapsActivity
        - SettingsActivity.
        - SignupActivity

    - **DB**: se encarga de la conexión con la base de datos local. Contiene tanto la definición de la misma (en el StatusContract) como la forma de interactuar con ella (en el DbHelper):
        - DbHelper
        - StatusContract

    - **Fragments**: con los contenedores de los layouts que, a su vez, están contenidos en los activities. La ventaja de usar estos fragments es que se puede tener varios en un solo activity y mostrarlos independientemente como es el caso del dashboard, message, profile y trips fragments:
        - DashboardFragment
        - MapsFragment
        - MessagesFragment
        - ProfileFragment
        - SettingsFragment
        - TripsFragment

    - **Receivers**: los receivers capturan eventos del sistema, como batería baja, notificaciones entrantes, encendido del dispositivo, etc. En nuestro receiver capturamos cuando el dispositivo se enciende, de manera que podamos iniciar los servicios y mantenerlos activos todo el tiempo que el dispositivo está encendido:
        - BReceiver

    - **Server**: aquí está contenida la clase ServerURL, que se encarga de suplir las direcciones correctas al servicio Rest del servidor que corresponda en cada momento:
        - ServerURL

- **Services**: los servicios son procesos en segundo plano que capturan eventos del sistema o realizan acciones sobre el mismo:
  - LocationService
  - TrackingService

- **Res/**
  - **drawable**: en esta carpeta se encuentran todos los recursos gráficos de la aplicación, entre ellas el logo de la app, los iconos que se utilizan dentro de ella, imágenes por defecto, etc.

  - **layout**: aquí se encuentran los layouts correspondientes a las vistas que están contenidas en los fragments y algunos activities, tarjetas, etc.

  - **menu**: similar a la carpeta anterior, pero en este caso las vistas se corresponden con los menús de la aplicación: el action bar (o menú superior) de algunas vistas y el navigation bar (menú inferior) del activity principal.

  - **values**: como su propio nombre indica, almacena los "valores" o datos que usa la aplicación para renderizar vistas, colores, textos, dimensiones, claves de API y estilos de la misma.

  - **values-en**: similar a la carpeta anterior, pero en este caso sólo para el idioma inglés.

  - **xml**: contiene el fichero settings.xml, el cual es un layout o vista específica para la pantalla de ajustes de la aplicación.

- **AndroidManifest.xml**: en este fichero se definen todos los permisos necesarios para ejecutar la aplicación, así como la declaración de todas las activities, receivers, y servicios.

7.8 DETALLES DE IMPLEMENTACIÓN

En este apartado se explicarán los procesos más relevantes de la herramienta en conjunto y, aquellas implementaciones que se consideran que son claves y distintivas de la misma:

7.8.1 Menús

En la aplicación disponemos de dos menús distintos, el primero de ellos, que llamaremos menú de navegación, es el que se encarga de cambiar entre fragments del MainActivity o activity principial, se crea en el método onCreate() del método principal y se instancia de la siguiente manera:

```
private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
        = new BottomNavigationView.OnNavigationItemSelectedListener() {

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()) {
            case R.id.navigation_dashboard:
                DashboardFragment D = new DashboardFragment();
                getFragmentManager()
                        .beginTransaction()
                        .replace(R.id.content, D,
                                D.getClass().getSimpleName())
                        .commit();
                setMenu(R.id.menu_dashboard);
                return true;
            case R.id.navigation_trips:
                TripsFragment T = new TripsFragment();
                getFragmentManager()
                        .beginTransaction()
                        .replace(R.id.content, T,
                                T.getClass().getSimpleName())
                        .commit();
                setMenu(R.id.menu_trips);
                return true;
            case R.id.navigation_messages:
                MessagesFragment M = new MessagesFragment();
                getFragmentManager()
                        .beginTransaction()
                        .replace(R.id.content, M,
                                M.getClass().getSimpleName())
                        .commit();
                setMenu(R.id.menu_messages);
                return true;
            case R.id.navigation_profile:
                ProfileFragment PF = new ProfileFragment();
                getFragmentManager()
                        .beginTransaction()
                        .replace(R.id.content, PF,
                                PF.getClass().getSimpleName())
                        .commit();
                setMenu(R.id.menu_profile);
                return true;
        }
        return false;
    }

private void setMenu(int enable) {
        menu.setGroupVisible(R.id.menu_dashboard, false);
        menu.setGroupVisible(R.id.menu_profile, false);
        menu.setGroupVisible(R.id.menu_trips, false);
        menu.setGroupVisible(R.id.menu_messages, false);
        menu.setGroupVisible(enable, true);
    }

};
```

El comportamiento de este menú es básico: según el botón inferior que pulse el usuario, se carga un fragment u otro en el activity principal y, además, se modifica el menú superior (o Action Bar) para que sólo muestre las acciones pertenecientes a dicha vista.

7.8.2 Verificación de permisos

Cuando nos hace falta saber si disponemos de algún permiso en la aplicación, como puede ser de ubicación, de acceso a internet, etc, se hace de la siguiente manera (ejemplo para el permiso de ubicación):

```
//Checking for Location Access Permission
public void checkLocationAccessPermission() {
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                PERMISSIONS_REQUEST_FINE_LOCATION);
    } else {
        Intent LocationIntent = new Intent(this, LocationService.class);
        this.startService(LocationIntent);
    }
}
```

Si disponemos del permiso, ejecutamos lo que queríamos hacer, si no, llamamos *requestPermissions* y solicitamos el permiso en cuestión que el usuario podrá permitir o denegar mediante un menú emergente. El resultado que marca el usuario se recoge así:

```
@Override
//For use when any permission has been requested
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    switch (requestCode) {
        case PERMISSIONS_REQUEST_FINE_LOCATION: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                // permission was granted
                Intent LocationIntent = new Intent(this, LocationService.class);
                this.startService(LocationIntent);
                Log.i(TAG, "Permission granted");
            } else {
                // permission denied
                Log.e(TAG, "Permission denied");
            }
            return;
        }
    }
}
```

De tal manera que si el usuario marca positivo al permiso, podemos seguir con la ejecución, y si no, paramos el proceso y podremos volver a preguntar más tarde.

7.8.3 Comunicación con Servicios Rest

La comunicación con los servicios Rest del servidor debe hacerse con un método en segundo plano, ya que en Android no está permitido realizar estas peticiones en primer plano. La manera de hacerlo es la siguiente (ejemplo usando el método POST de registro):

```
private class doSignup extends AsyncTask<String, Void, Boolean> {

    @Override
    protected Boolean doInBackground(String... params) {
        try {
            HttpURLConnection connection = null;
            URL url = new URL(params[0]);
            String data = "[" + params[1] + "]";
            String responseContent = "";

            //Obtain a new HttpURLConnection, cast the result to HttpURLConnection.
            connection = (HttpURLConnection) url.openConnection();

            //Prepare the request           //include metadata
            connection.setRequestMethod("POST");
            connection.setDoOutput(true);
            connection.setDoInput(true);
            connection.setChunkedStreamingMode(0);
            connection.setRequestProperty("Content-Type", "application/json");


            DataOutputStream dos = new
DataOutputStream(connection.getOutputStream());
            byte[] bs = data.getBytes();
            dos.write(bs);
            dos.flush();
            dos.close();

            Log.d(TAG, "POST method sent to: " + url.toString());
            Log.d(TAG, "Data ___sent: [" + params[1] + "]");

            //Read the response...
            respCode = connection.getResponseCode();

            //Disconnect
            connection.disconnect();

            return respCode == HttpURLConnection.HTTP_NO_CONTENT;

        } catch (MalformedURLException e) {
            Log.e(TAG, "ERROR: 'new URL(params[0])' failed");
            e.printStackTrace();
        } catch (IOException e) {
            Log.e(TAG, "ERROR: could not open connection");
            e.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPostExecute(Boolean res) {
        if (res) {
            Toast.makeText(context, R.string.form_submited,
Toast.LENGTH_LONG).show();
            //Finish activity
            finish();
        } else
            Toast.makeText(context, R.string.server_error, Toast.LENGTH_LONG).show();
    }
}
```

Mediante un proceso AsyncTask se realiza la petición, se hace uso de HttpUrlRequest para formar y enviar la petición. La forma es sencilla:

- Se crea la url y se abre la conexión con ella
- Se prepara la petición (en este caso se incluye cuerpo porque es un POST)
- Se envía la request
- Se lee la respuesta
- Se cierra la conexión
- Se procesa la respuesta (en el ejemplo se implementa en el onPostExecute)

7.8.4 Procesamiento por lotes

El batch es un proceso totalmente transparente para el usuario, ya que se realiza en el servidor, la manera en la que se ejecuta es la siguiente. El proceso empieza en la clase CommonBean:

```
@Named
@RequestScoped
public class CommonBean {


    public void runJob() throws JobSecurityException, JobStartException {
        try {
            JobOperator jo = BatchRuntime.getJobOperator();
            long jobId = jo.start("eod-commons", new Properties());
            System.out.println("Started job: with id: " + jobId);
        } catch (JobStartException ex) {
            ex.printStackTrace();
        }
    }
}
```

La cual ejecuta un proceso en segundo plano, declarando en qué fichero se encuentra concretado qué clase hace qué trabajo, dicho fichero es el eod-commons.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<job id="endOfDayCommons"
     xmlns="http://xmlns.jcp.org/xml/ns/javaee"
     version="1.0">
    <step id="calculateCommons" >
        <chunk item-count="3" skip-limit="5">
            <reader ref="TripReader"/>
            <processor ref="RouteProcessor"/>
            <writer ref="CommonWriter"/>
            <skippable-exception-classes>
                <include class="java.lang.NumberFormatException"/>
            </skippable-exception-classes>
        </chunk>
    </step>
</job>
```

El cual especifica que el proceso se empezará con el *reader*, luego el *processor* y, finalmente, el *writer*. La clase TripReader ejecuta lo siguiente:

```
@Named("TripReader")
@Dependent
public class TripReader extends AbstractItemReader {


    @PersistenceContext(unitName = "tfgseg01_commongo2_war_1.0-SNAPSHOTPU")
    private EntityManager em;


    List<Trips> allTrips;
    ListIterator<Trips> iterator;


    @Override
    public void open(Serializable checkpoint) {
        allTrips = em.createNamedQuery("Trips.findAll", Trips.class).getResultList();
        //allTrips = em.createQuery(QUERY_SELECT_TRIPS).getResultList();
        iterator = allTrips.listIterator();
    }


    @Override
    public Object readItem() throws Exception {
        if (iterator.hasNext()) {
            return iterator.next();
        } else {
            return null;
        }
    }
}
```

Principalmente lo que hace es recoger todas las rutas existentes en la base de datos e ir pasándolas una por una al RouteProcessor, cuya ejecución se resume en procesar, por cada Trip, todas las rutas una a una, descartando las que difieren en origen, destino, horario y o son del mismo usuario. Después de eso se calcula la distancia entre las rutas de la siguiente manera (se ha omitido todo el código porque la clase es demasiado extensa para ponerla aquí, pero se añade el código clave):

Sentencias de consulta a la BBDD:

```
 private final String QUERY_DISTANCE_START = "SELECT ST_Distance_sphere((SELECT ST_AsText(ST_StartPoint(route)) FROM routes WHERE id=IDORIGIN), (select ST_AsText(ST_StartPoint(route)) FROM routes WHERE id=IDDESTINATION))";

   private final String QUERY_DISTANCE_END = "SELECT ST_Distance_sphere((SELECT ST_AsText(ST_EndPoint(route)) FROM routes WHERE id=IDORIGIN), (select ST_AsText(ST_EndPoint(route)) FROM routes WHERE id=IDDESTINATION))";
```

Cálculo de distancias:

```
private Integer calculateDistance(String idOrigin, String idDestination) {
        return distanceInStart(idOrigin, idDestination) + distanceInEnd(idOrigin, idDestination);
    }

    private Integer distanceInStart(String idOrigin, String idDestination) {
        String query = QUERY_DISTANCE_START.replace("IDORIGIN", idOrigin);
        query = query.replace("IDDESTINATION", idDestination);
        Query nativeQ = em.createNativeQuery(query);
        Double result = (Double) nativeQ.getSingleResult();
        return result.intValue();
    }

    private Integer distanceInEnd(String idOrigin, String idDestination) {
        String query = QUERY_DISTANCE_END.replace("IDORIGIN", idOrigin);
        query = query.replace("IDDESTINATION", idDestination);
        Query nativeQ = em.createNativeQuery(query);
        Double result = (Double) nativeQ.getSingleResult();
        return result.intValue();
    }
```

El resultado de la ejecución de estas funciones resulta en los metros distantes entre las dos rutas, que posteriormente se ordenarán por proximidad con la ruta del usuario. Una vez ha terminado el *processor*, se manda cada relación entre rutas coincidentes a CommonWriter, cuyo código se resume en:

```
@Named("CommonWriter")
@Dependent
public class CommonWriter extends AbstractItemWriter {

    @PersistenceContext(unitName = "tfgseg01_commongo2_war_1.0-SNAPSHOTPU")
    private EntityManager em;

    @Override
    public void writeItems(List list) throws Exception {
        List<Commons> newList = (List<Commons>) list.get(0);

        for (Commons c :  newList) {
            System.out.println("CommonWriter.writeItems: " + c.toString());
            em.persist(c);
        }
    }
}
```

7.8.5 Comunicación con Google Geocode API

La respuesta de la geocodificación inversa de la API Geocode de Google es algo engorrosa, para procesar toda la información que nos devuelve cuando le preguntamos por unas coordenadas, se ha creado el siguiente código que parsea el JSON que devuelve la consulta:

```
private JSONObject readGeocode(InputStream s) throws IOException, JSONException {
    JSONObject response = new JSONObject();
    JsonReader reader = new JsonReader(new InputStreamReader(s));
    int count = 0;

    reader.beginObject();
    while (reader.hasNext()) {
        String object1 = reader.nextName();
        if (object1.equalsIgnoreCase("results")) {
            reader.beginArray();
            while (reader.hasNext()) {
                if (count < 1) { reader.beginObject();
                    while (reader.hasNext()) {
                        String name = reader.nextName();
                        if (name.equalsIgnoreCase("address_components"))
                            addressCompReader(reader, response);
                        else if (name.equalsIgnoreCase("place_id"))
                            response.put("googleId", reader.nextString());
                        else reader.skipValue();
                    } reader.endObject();
                    count++;
                } else { reader.close();
                    return response;
                } } reader.endArray();
        } else reader.skipValue();
    } reader.endObject();
    return response;
}
```

8 PRUEBAS

En este apartado se reflejan las pruebas realizadas durante la vida del desarrollo del proyecto, estas pruebas han sido de dos tipos:

- Pruebas de caja blanca: realizadas en la fase de *Diseño* e *Implementación.*
- Pruebas de caja negra: realizadas en la fase de *Pruebas.*

8.1 PRUEBAS DE CAJA BLANCA

Este tipo de pruebas están enfocadas a analizar cada función o método por separado. Es decir, se prueban de forma individual todos los elementos que se van desarrollando en la aplicación, en muchos casos conociendo la entrada y sabiendo la salida esperada del método. Con el fin de asegurar su funcionamiento cuando se realicen las pruebas de caja negra.

Estos test se han realizado de manera simultánea a la implementación de la aplicación y del servidor, comprobando cada elemento por separado y en consonancia con otros métodos con los que trabaja en cadena o en paralelo. Se han basado en:

- Comprobación de la salida de los formularios, correcto formato y campos requeridos.
- Comunicación correcta entre el servidor y la aplicación.
- Cerrar conexiones con el servidor y la base de datos local cuando se han utilizado.
- Utilización de logs para comprobar la secuencia de los procesos, salida y entrada de métodos que se utilizan para la comunicación entre aplicación y servidor, entre aplicación y base de datos local, y entre la aplicación y las APIs que utiliza.
- Comprobación del visionado de mensajes, guardado y envío de los mismos.
- Comprobación del correcto funcionamiento de hilos y servicios.
- Comprobación del correcto registro y guardado de las rutas que realiza el usuario en tiempo real.
- Verificación de la lectura y escritura de los ajustes parametrizables.
- Comprobación de la correcta ejecución de las sesiones, al iniciar sesión y al cerrarla.

8.2 PRUEBAS DE CAJA NEGRA

Estos test están enfocados a verificar el funcionamiento de tareas completas tanto en la aplicación como en el servidor, de tal forma que satisfagan los requisitos que se han establecido en el análisis. Este tipo de pruebas se podrían considerar como complementarias a las mismas, ya que de igual manera se esperan unos resultados a partir de unos datos de entrada, pero a más alto nivel.

Las pruebas que se han realizado en esta fase están detalladas a continuación:

Prueba 01 – Registro		
Objetivo	Registrar los datos del usuario en el sistema correctamente	
Precondiciones	Ninguna	
Datos de entrada	Nombre: María Apellido: Núñez Fecha de nacimiento: 15/04/1995 Teléfono: 681686461 Email: maria@mail.com Contraseña: 12341234 Repetir contraseña: 12341234	
Acción esperada	Se crea el usuario en el servidor Se notifica el registro correcto Se abre la pantalla de Login	
Secuencia	Paso	Acción
	1	Rellenar datos
	2	Comprobar datos
	3	Subir datos al servidor
	4	Guardar datos en la BBDD
	5	Recibir OK del servidor
	6	Mostar notificación
	7	Abrir Login
Resultado	Correcto	

Tabla 58 - Prueba 01 – Registro

Prueba 02 – Registro fallido		
Objetivo	Evitar que se registre en el sistema un usuario con datos inválidos	
Precondiciones	Ninguna	
Datos de entrada	Nombre: María Apellido: Núñez Fecha de nacimiento: 15/04/1995 Teléfono: 6816 Email: maria@mail.com Contraseña: 12341234 Repetir contraseña: 12341234	
Acción esperada	Se notifica el campo incorrecto Se bloquea el proceso	
Secuencia	Paso	Acción
	1	Rellenar datos
	2	Comprobar datos
	3	Notificar campo inválido (teléfono en este caso)
Resultado	Correcto	

Tabla 59 - Prueba 02 - Registro fallido

Prueba 03 – Identificación		
Objetivo	Ingresar al usuario en la aplicación con los datos correctos de login	
Precondiciones	Usuario registrado	
Datos de entrada	Email: maria@mail.com Contraseña: 12341234	
Acción esperada	Confirmar datos correctos Abrir pantalla principal Obtener datos del usuario del servidor Guardar datos en la BBDD local	
Secuencia	Paso	Acción
	1	Completar los campos del formulario
	2	Comprobar campos del formulario
	3	Enviar petición al servidor con los datos
	4	Comprobar en la BBDD que existe el usuario
	5	Devolver OK y datos del usuario
	6	Recibir OK y datos del usuario
	7	Guardar datos en la BBDD local
	8	Abrir pantalla principal
Resultado	Correcto	

Tabla 60 - Prueba 03 - Identificación

Prueba 04 – Identificación incorrecta		
Objetivo	Evitar que un usuario no registrado entre en el sistema	
Precondiciones	Usuario no registrado	
Datos de entrada	Email: pepe@mail.com Contraseña: 12341234	
Acción esperada	Notificar usuario inexistente Volver al estado inicial	
Secuencia	Paso	Acción
	1	Completar los campos del formulario
	2	Comprobar campos del formulario
	3	Notificar usuario inexistente
Resultado	Correcto	

Tabla 61 - Prueba 04 - Identificación incorrecta

Prueba 05 – Recibir nuevas notificaciones		
Objetivo	Descargar las nuevas notificaciones, coincidencias de rutas y mensajes para el usuario	
Precondiciones	Usuario identificado	
Datos de entrada	Ninguno	
Acción esperada	Recibir datos del servidor y guardarlos en BBDD local	
Secuencia	Paso	Acción
	1	Lanzar petición al servidor
	2	Mirar en BBDD del servidor nuevas coincidencias, notificaciones y mensajes
	3	Recibir datos del servidor
	4	Guardar en BBDD local
	5	Notificar las nuevas coincidencias, notificaciones y mensajes al usuario
Resultado	Correcto	

Tabla 62 - Prueba 05 - Recibir nuevas notificaciones

Prueba 06 – Visualizar Tablón (Dashboard)		
Objetivo	Ver la pantalla de tablón y las notificaciones para el usuario si las hubiera	
Precondiciones	Usuario identificado	
Datos de entrada	Ninguno	
Acción esperada	Mostrar pantalla de Tablón Mostrar notificaciones para el usuario si hay	
Secuencia	Paso	Acción
	1	Iniciar sesión/ pulsar sobre el botón de tablón
	2	Iniciar Fragment de la vista del tablón
	3	Llamar al provider de las notificaciones
	4	Leer las notificaciones de la BBDD local
	5	Mostrar las notificaciones en la vista
Resultado	Correcto	

Tabla 63 - Prueba 06 - Visualizar tablón

Prueba 07 – Visualizar Viajes (Trips)		
Objetivo	Ver la pantalla de viajes y las rutas para el usuario si las hubiera	
Precondiciones	Usuario identificado	
Datos de entrada	Ninguno	
Acción esperada	Mostrar pantalla de viajes Mostrar rutas para el usuario si hay	
Secuencia	Paso	Acción
	1	Pulsar sobre el botón de viajes
	2	Iniciar Fragment de la vista de los viajes
	3	Llamar al provider de los viajes
	4	Leer los viajes de la BBDD local
	5	Mostrar los viajes en la vista
Resultado	Correcto	

Tabla 64 - Prueba 07 - Visualizar viajes

Prueba 08 – Visualizar Conversaciones (Conversations)		
Objetivo	Ver la pantalla de conversaciones y las conversaciones para el usuario si las hubiera	
Precondiciones	Usuario identificado	
Datos de entrada	Ninguno	
Acción esperada	Mostrar pantalla de Conversaciones Mostrar conversaciones para el usuario si hay	
Secuencia	Paso	Acción
	1	Pulsar sobre el botón de conversaciones
	2	Iniciar Fragment de la vista de las conversaciones
	3	Llamar al provider de las conversaciones
	4	Leer las conversaciones de la BBDD local
	5	Mostrar las conversaciones en la vista
Resultado	Correcto	

Tabla 65 - Prueba 08 - Visualizar conversaciones

Prueba 09 – Visualizar mensajes		
Objetivo	Ver la pantalla de mensajes y los mensajes existentes en esa conversación	
Precondiciones	Usuario identificado Pulsar en el botón de conversaciones	
Datos de entrada	Ninguno	
Acción esperada	Mostrar pantalla de Mensajes Mostrar los mensajes en esa conversación	
Secuencia	Paso	Acción
	1	Pulsar sobre la tarjeta de una conversación
	2	Iniciar Activity de mensajes
	3	Cargar Fragment y vistas de los mensajes
	4	Leer los mensajes de la BBDD local
	5	Mostrar los mensajes en la nueva vista
Resultado	Correcto	

Tabla 66 - Prueba 09 - Visualizar mensajes

Prueba 10 – Enviar mensaje		
Objetivo	Enviar un mensaje a otro usuario	
Precondiciones	Usuario identificado Pulsar en conversaciones Pulsar en la tarjeta de una conversación	
Datos de entrada	Texto del mensaje	
Acción esperada	El mensaje se muestra en la pantalla y se guarda en el servidor y en local	
Secuencia	Paso	Acción
	1	Escribir un mensaje
	2	Pulsar en *enviar*
	3	Mostar el mensaje en pantalla
	4	Enviar mensaje al servidor
	5	Guardar mensaje en la BBDD
	6	Guardar mensaje en la BBDD local
Resultado	Correcto	

Tabla 67 - Prueba 10 - Enviar mensaje

Prueba 11 – Visualizar Perfil (Profile)		
Objetivo	Mostrar correctamente la pantalla de perfil con los datos del usuario	
Precondiciones	Usuario identificado	
Datos de entrada	Ninguno	
Acción esperada	Se muestra la pantalla de perfil del usuario con los datos con los que ingresó en el sistema, a excepción de la contraseña. Y la foto si la ha cambiado, si no, muestra una predefinida.	
Secuencia	Paso	Acción
	1	Pulsar sobre el botón de perfil
	2	Iniciar Fragment de perfil
	3	Leer los datos del usuario de la BBDD
	4	Mostrar la vista con los datos inclusive
Resultado	Correcto	

Tabla 68 - Prueba 11 - Visualizar perfil

Prueba 12 – Editar perfil		
Objetivo	Modificar los datos personales del usuario	
Precondiciones	Usuario identificado Pantalla de perfil abierta	
Datos de entrada	Email: mery@mail.com	
Acción esperada	Enviar los nuevos datos al servidor, guardarlos en la BBDD y actualizarlos en la vista y en local	
Secuencia	Paso	Acción
	1	Pulsar en el botón de editar perfil
	2	Modificar datos
	3	Pulsar en el botón de modificar
	4	Comprobar datos válidos
	5	Modificar en BBDD local
	6	Enviar datos al servidor
	7	Guardar en BBDD del servidor
Resultado	Correcto	

Tabla 69 - Prueba 12 - Editar perfil

Prueba 13 – Editar foto de perfil		
Objetivo	Modificar la foto del usuario	
Precondiciones	Usuario identificado Pantalla de perfil abierta	
Datos de entrada	<Imagen>	
Acción esperada	Enviar la foto al servidor, guardarla en la BBDD y actualizarla en la vista y en local	
Secuencia	Paso	Acción
	1	Pulsar en la imagen
	2	Buscar una foto nueva en el explorador
	3	Seleccionar foto
	4	Formatear foto y guardar en local
	5	Enviar foto al servidor
	6	Guardar en BBDD del servidor
Resultado	Correcto	

Tabla 70 - Prueba 13 - Editar foto de perfil

Prueba 14 – Ver ajustes		
Objetivo	Visualizar la pantalla de ajustes	
Precondiciones	Usuario Identificado Pantalla de perfil abierta	
Datos de entrada	Ninguno	
Acción esperada	Abrir la pantalla de ajustes Mostrar parámetros con los valores actuales	
Secuencia	Paso	Acción
	1	Pulsar sobre el icono de ajustes
	2	Cargar Activity de ajustes
	3	Mostrar ajustes
Resultado	Correcto	

Tabla 71 - Prueba 14 - Ver ajustes

Prueba 15 – Grabar ruta		
Objetivo	Recordar una ruta y guardarla en la BBDD local	
Precondiciones	Usuario identificado Pantalla de tablón abierta Permisos de ubicación aceptados	
Datos de entrada	Ubicaciones definidas por GPS	
Acción esperada	La ruta se graba correctamente y se almacena en la BBDD local	
Secuencia	Paso	Acción
	0	Borrar anterior ruta grabada
	1	Pulsar en el icono **+**
	2	Pulsar en *Common! Go!* en el cuadro de diálogo
	3	Comenzar a recibir nuevas ubicaciones
	4	Guardar las ubicaciones en BBDD
	5	Pulsar en el icono de *stop* cuando se quiera parar de recordar la ruta
	6	Aceptar el cuadro de diálogo
	7	Notificar que la ruta está en la pantalla de mapas
Resultado	Correcto	

Tabla 72 - Prueba 15 - Grabar ruta

Prueba 16 – Mostar mapas		
Objetivo	Mostar correctamente la pantalla de mapas	
Precondiciones	Usuario identificado Pantalla de tablón abierta	
Datos de entrada	Ninguno	
Acción esperada	Abrir la pantalla de mapas Cargar todos los componentes en ella	
Secuencia	Paso	Acción
	1	Pulsar sobre el icono de mapas
	2	Cargar Activity de mapas
Resultado	Correcto	

Tabla 73 - - Prueba 16 - Mostrar mapas

Prueba 17 – Ver ruta		
Objetivo	Mostar correctamente la pantalla de mapas	
Precondiciones	Usuario identificado Pantalla de mapas abierta Ruta grabada	
Datos de entrada	Ninguno	
Acción esperada	Cargar la ruta y dibujarla en el mapa	
Secuencia	Paso	Acción
	1	Pulsar sobre la etiqueta de una ruta
	2	Leer la ruta de la BBDD local
	3	Dibujar la ruta en el mapa
Resultado	Correcto	

Tabla 74 - Prueba 17 - Ver ruta

Prueba 18 – Guardar ruta		
Objetivo	Guardar la ruta de manera permanente	
Precondiciones	Usuario identificado Pantalla de mapas abierta Ruta grabada	
Datos de entrada	Ninguno	
Acción esperada	Cargar la ruta y dibujarla en el mapa	
Secuencia	Paso	Acción
	1	Pulsar sobre la etiqueta de la última ruta
	2	Pulsar sobre el icono de guardar
	3	Guardar en BBDD local
	4	Enviar petición de geocodificación inversa a la API Geocode para el punto de origen y destino de la ruta
	5	Enviar los puntos de origen y destino al servidor
	6	Guardar los puntos en BBDD del servidor
	7	Enviar el viaje a servidor
	8	Guardar el viaje en BBDD del servidor
	9	Enviar ruta al servidor
	10	Guardar ruta en BBDD del servidor
Resultado	Correcto	

Tabla 75 - Prueba 18 - Guardar ruta

Prueba 19 – Editar ruta		
Objetivo	Editar el nombre por defecto de la ruta y/o los días en los que se realiza	
Precondiciones	Usuario identificado Pantalla de viajes abierta Ruta existente	
Datos de entrada	Nombre: Trabajo	
Acción esperada	Modificar los datos de la ruta	
Secuencia	Paso	Acción
	1	Pulsar sobre el icono de editar de la etiqueta de una ruta
	2	Modificar los datos en el menú emergente
	3	Aceptar los cambios
	4	Guardar los nuevos datos en BBDD local
Resultado	Correcto	

Tabla 76 - Prueba 19 - Editar ruta

Prueba 20 – Solicitar conversación		
Objetivo	Solicitar empezar una conversación con un usuario	
Precondiciones	Usuario identificado Pantalla de tablón abierta Ruta coincidente con ruta de otro usuario	
Datos de entrada	Ninguno	
Acción esperada	Enviar una notificación al otro usuario para empezar a hablar	
Secuencia	Paso	Acción
	1	Pulsar sobre la etiqueta de una notificación de ruta coincidente
	2	Aceptar el cuadro de diálogo
	3	Enviar solicitud al servidor
Resultado	Correcto	

Tabla 77 - Prueba 20 - Solicitar conversación

Prueba 21 – Decidir conversación		
Objetivo	Aceptar o rechazar iniciar una nueva conversación con un usuario	
Precondiciones	Usuario identificado Pantalla de tablón abierta Notificación de petición de conversación	
Datos de entrada	Ninguno	
Acción esperada	Abrir pantalla de mensajes	
Secuencia	Paso	Acción
	1	Pulsar sobre la etiqueta de una notificación de petición de conversación
	2	Aceptar el cuadro de diálogo
	3	Cargar el activity de mensajes para ese usuario
Resultado	Correcto	

Tabla 78 - Prueba 21 - Decidir conversación

9 MANUALES

En este punto se encuentran los manuales de instalación de los principales componentes que forman parte de la herramienta, así como los manuales de uso correspondientes a la aplicación.

9.1 MANUAL DE INSTALACIÓN DEL SERVIDOR

Primeramente se va a ilustrar la manera de instalar los componentes para el normal funcionamiento de la herramienta, estos son el IDE, el SGBD con la extensión espacial y la aplicación en sí.

9.1.1 Instalación de NetBeans

Necesitamos descargar el JDK, en nuestro caso se ha usado la versión 131 (Java SE Development Kit 8u131) en su versión para Windows x64 bits, lo podemos descargar desde la página de Oracle:

http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

Imagen 29 - Descarga SDK 1

Overview | Downloads | Documentation | Community | Technologies | Training

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 8u131 checksum

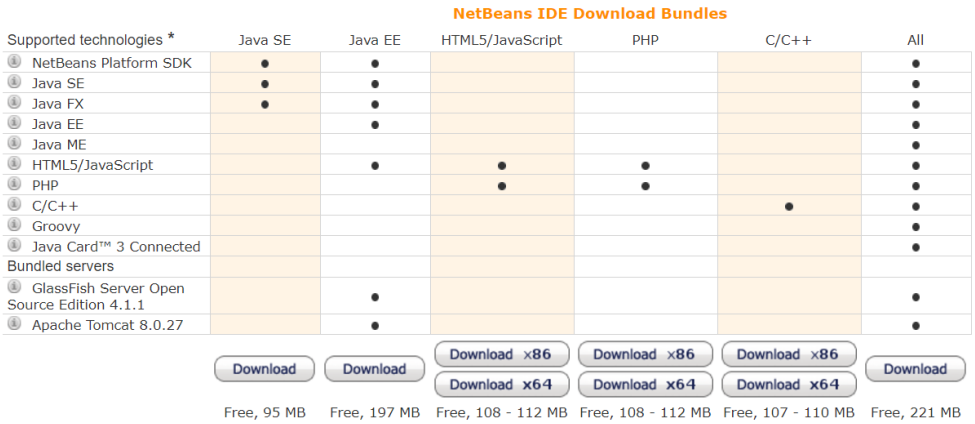
Java SE Development Kit 8u131

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.87 MB	jdk-8u131-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.81 MB	jdk-8u131-linux-arm64-vfp-hflt.tar.gz
Linux x86	164.66 MB	jdk-8u131-linux-i586.rpm
Linux x86	179.39 MB	jdk-8u131-linux-i586.tar.gz
Linux x64	162.11 MB	jdk-8u131-linux-x64.rpm
Linux x64	176.95 MB	jdk-8u131-linux-x64.tar.gz
Mac OS X	226.57 MB	jdk-8u131-macosx-x64.dmg
Solaris SPARC 64-bit	139.79 MB	jdk-8u131-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.13 MB	jdk-8u131-solaris-sparcv9.tar.gz
Solaris x64	140.51 MB	jdk-8u131-solaris-x64.tar.Z
Solaris x64	96.96 MB	jdk-8u131-solaris-x64.tar.gz
Windows x86	191.22 MB	jdk-8u131-windows-i586.exe
Windows x64	198.03 MB	jdk-8u131-windows-x64.exe

Imagen 30 - Descarga JDK 2

Una vez descargado e instalado, debemos descargar el IDE. En este caso es Netbeans en su versión Java EE 8.2. Está disponible también en su página oficial: https://netbeans.org/downloads/



NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						•
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						•
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•
	Download	Download	Download x86 / Download x64	Download x86 / Download x64	Download x86 / Download x64	Download
	Free, 95 MB	Free, 197 MB	Free, 108 - 112 MB	Free, 108 - 112 MB	Free, 107 - 110 MB	Free, 221 MB

Imagen 31 - Descarga Netbeans

Bastará con instalarlo y ya lo tendremos listo. El proceso de instalación es muy simple y no necesita de ilustraciones.

9.1.2 Instalación de la Base de Datos

Esta fase se hace en dos partes, primero necesitamos instalar el SGBD en cuestión y después añadir la extensión para que adquiera características espaciales.

9.1.2.1 PostgreSQL

La descarga se encuentra disponible en el enlace, únicamente debemos seleccionar la versión de PostgreSQL deseada y la versión de nuestro Sistema Operativo, en nuestro caso se ha usado la versión 9.6.3:

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

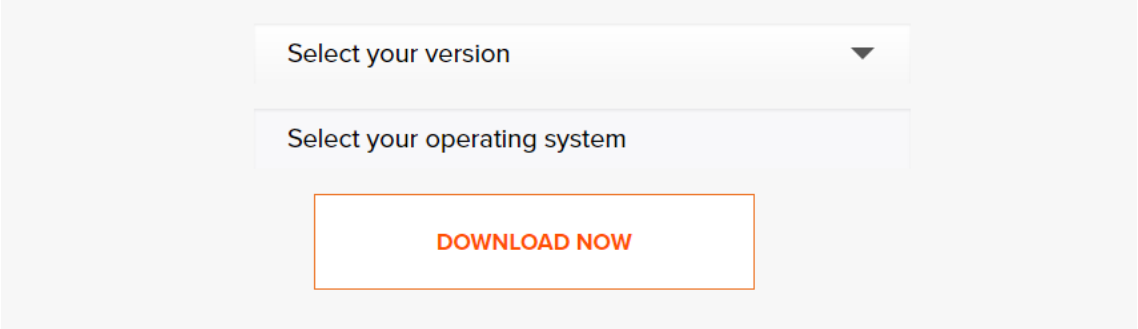


Imagen 32 - Descarga PostgreSQL

Una vez descargado, ejecutar e instalar:

Imagen 33 - Instalación PostgreSQL 1

Tomaremos todas las configuraciones por defecto, dado a siguiente en todos los pasos sin modificar nada. Cuando nos pregunte si queremos iniciar el Stack Builder al finalizar, marcamos que sí para poder continuar con la instalación de PostGIS y finalizamos.

Imagen 34 - Instalación PostgreSQL 2

9.1.2.2 PostGIS

Una vez finalizado el proceso anterior, continuamos con el StackBuilder. Una vez abierto seleccionamos nuestro SGBD recién instalado y clicamos en siguiente.

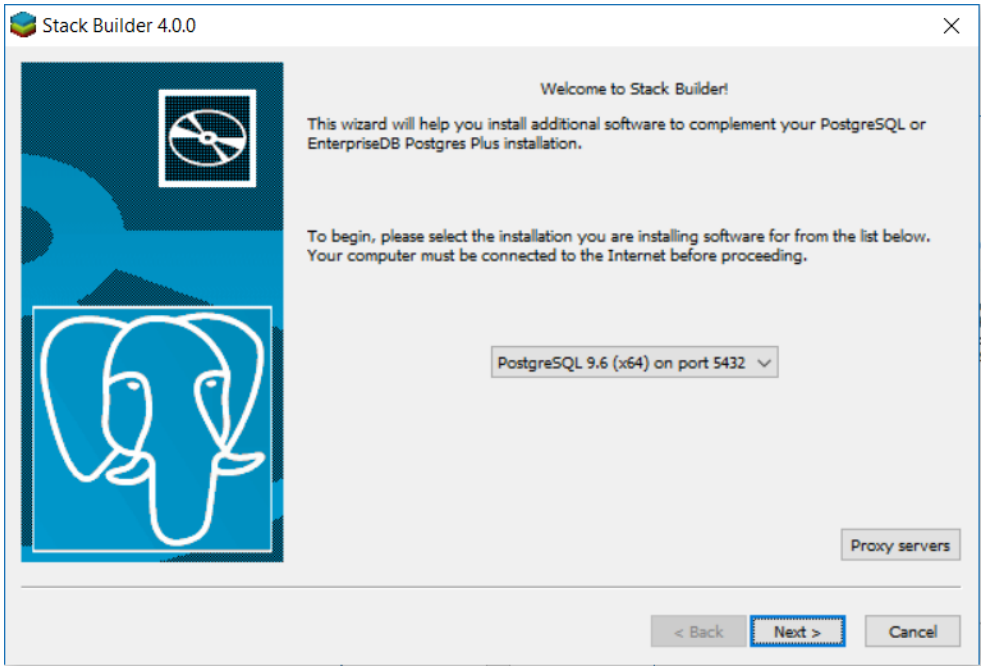


Imagen 35 - Instalación PostGIS 1

Se nos abrirá un pequeño menú de plugins donde deberemos marcar Categories – Spatial Extensions – PostGIS 2.3 Blundle for PostgreSQL 9.6 (64 bit) v2.3.2 (en nuestro caso ya está instalada)

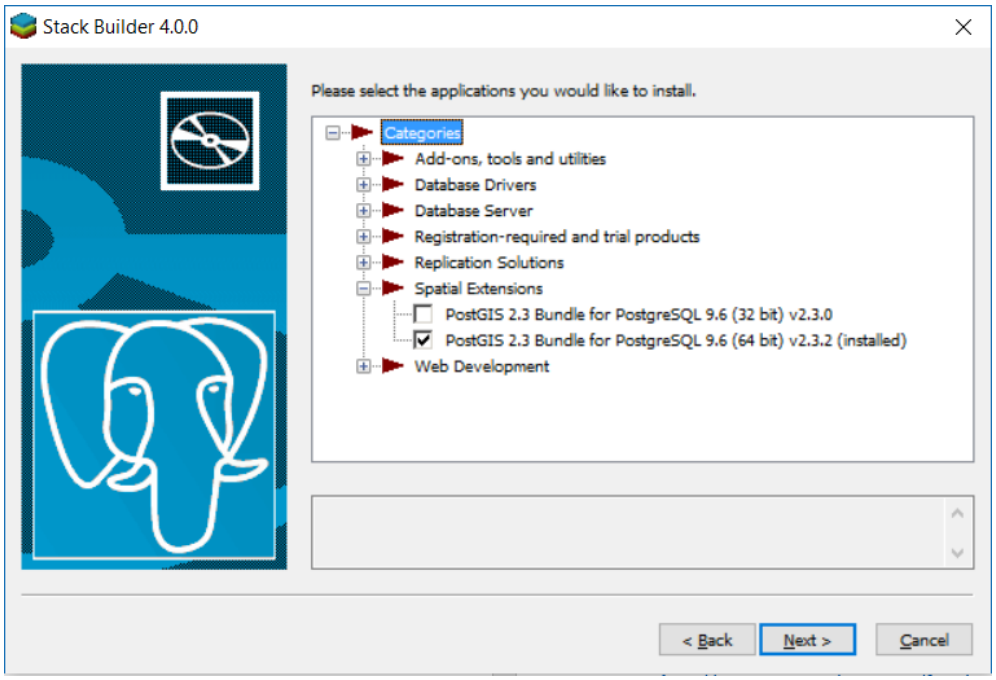


Imagen 36 - Instalación PostGIS 2

A partir de aquí, sólo tenemos que marcar siguiente hasta el final de la instalación, y aceptar la licencia de uso de PostGIS cuando así lo requiera.

9.2 MANUAL DE INSTALACIÓN DE LA APLICACIÓN MÓVIL

Para instalar la app, debemos disponer del fichero de instalación *.apk*, o bien descargar la app CommonGo desde la Google Play Store si se encontrar disponible en el momento de leer este manual. Si se descarga desde Google Play, su instalación es rápida y sencilla. Si se hace lo primero, el proceso, aunque también fácil, es el siguiente:

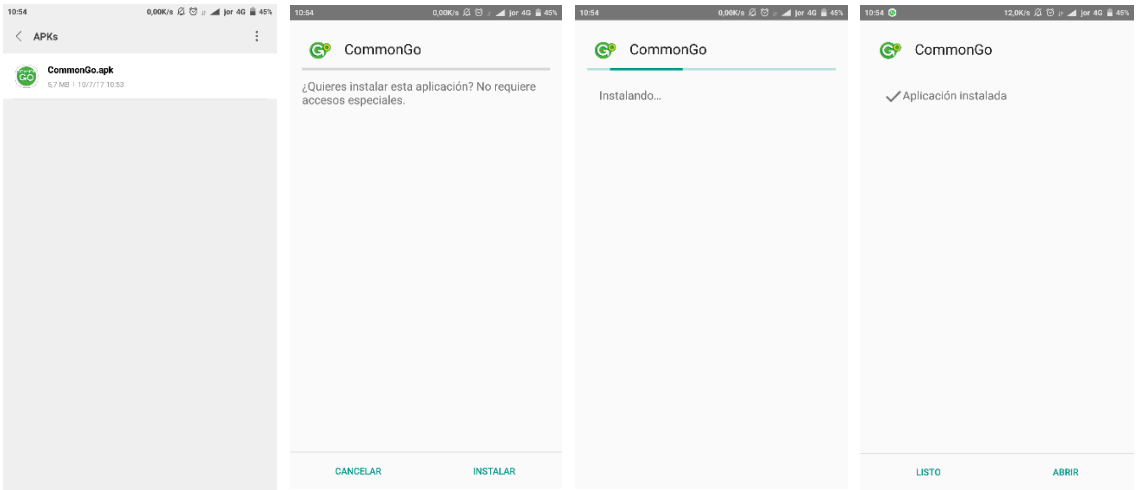


Imagen 37 - Instalación CommonGo

Ejecutamos el fichero .apk y clicamos en instalar, la aplicación se instalará y ya está lista para usarse.

9.3 MANUAL DE USUARIO

- Login: sólo será necesario introducir nuestro correo y contraseña de la aplicación para acceder correctamente. Si no estamos registrados, deberemos registrarnos primero clicando sobre *No account yet? Create one* o *¿No tienes cuenta? Crea una*, según el idioma que esté configurado en nuestro dispositivo.

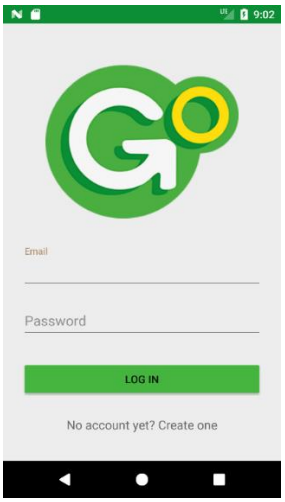


Imagen 38 - Manual de usuario CommonGo 1

- Registro: para registrarnos correctamente únicamente deberemos ingresar nuestros datos, esto incluye nombre, apellidos, fecha de nacimiento, número de teléfono, correo y contraseña.

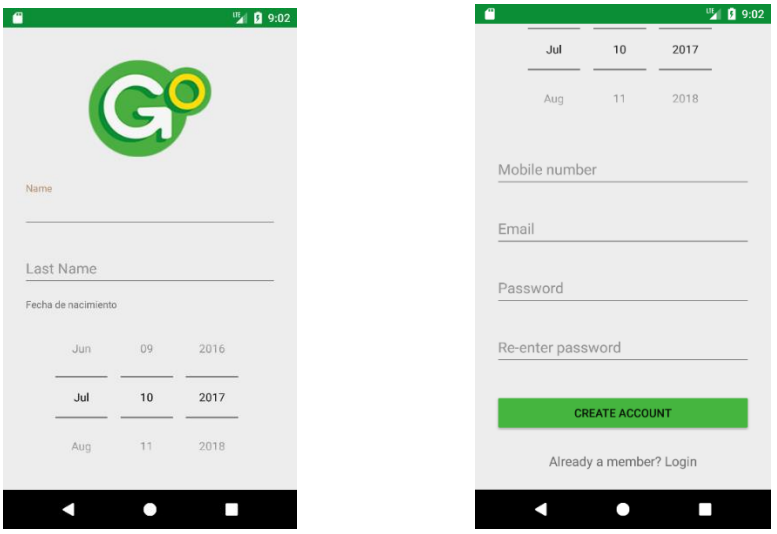


Imagen 39 - Manual de usuario CommonGo 2

- Capturar una ruta es algo más elaborado, primero debemos situarnos en la pantalla del *tablón* o *dashboard*, hacer clic en el icono de *más (+)*, que encontraremos en la esquina superior derecha y confirmar que queremos comenzar a capturar una ruta, cuando realicemos la ruta o queramos interrumpir este proceso, clicaremos sobre el icono de *stop* y confirmaremos.

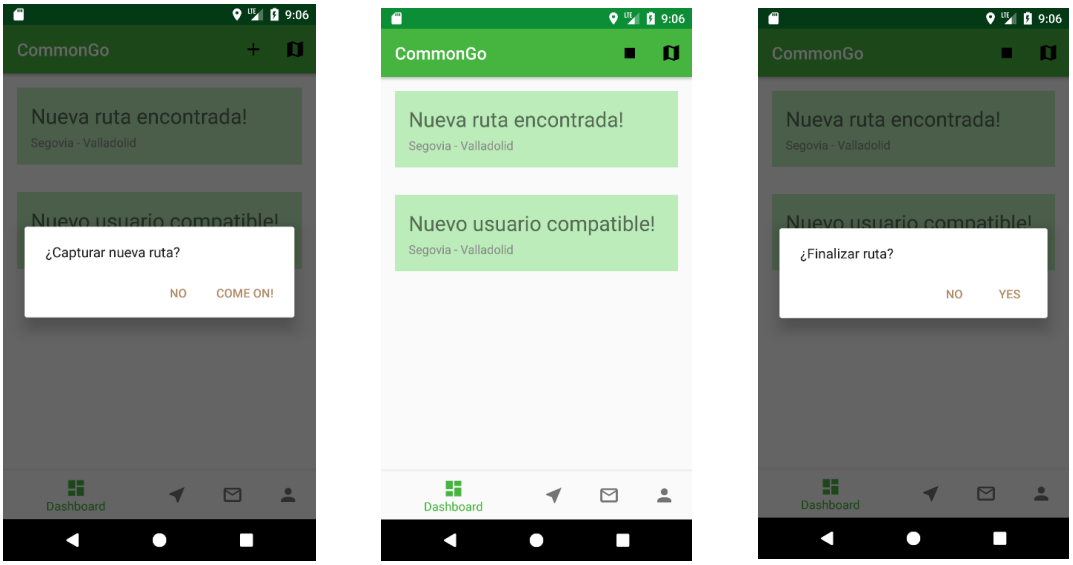


Imagen 40 - Manual de usuario CommonGo 3

- Para guardar una ruta, y para ello comenzar a recibir notificaciones sobre las coincidencias con otros usuaros, deberemos clicar sobre el icono de mapa de la esquina superior derecha de la pantalla del tablón. Una vez en la pantalla de mapas, haremos clic en *last route* o *última ruta* y seguidamente sobre el icono de guardado. Después confirmaremos que queremos guardar la ruta, y ésta se guardará en nuestra pantalla de *trips* o *viajes*, además de compartirla en el servidor.

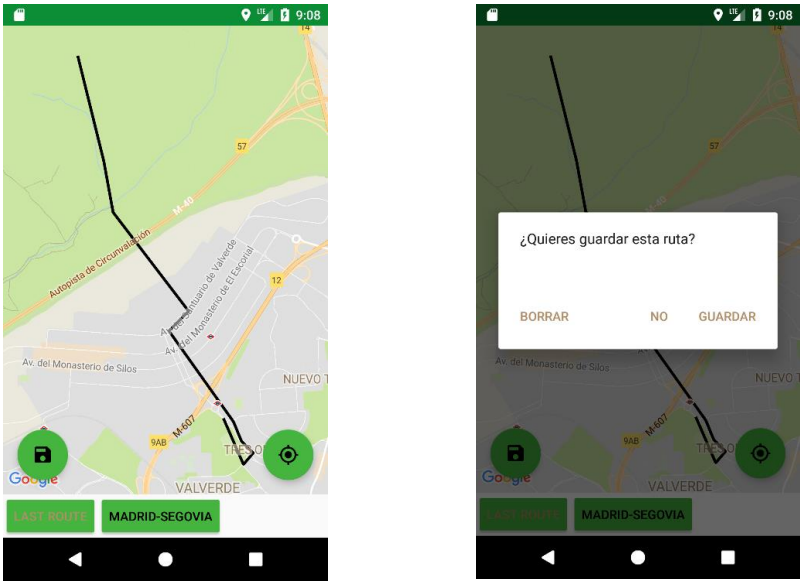


Imagen 41 - Manual de usuario CommonGo 4

10 CONCLUSIONES

CommonGo comenzó como una idea surgida de la propia experiencia, la idea de crear una herramienta capaz de solucionar el gran problema real de los inevitables atascos en los accesos a las grandes ciudades y también dentro de ellas. Pero también de los grandes beneficios que esta herramienta podría tener para la sociedad.

Para afrontar este proyecto, se pensó en unir a esas personas que realizan todos los días un viaje por el mismo trayecto al trabajo y que colaboran, de forma involuntaria, a engrosar las retenciones. El primer problema que se encontró para llevar el proyecto a cabo fue cómo comparar rutas: cálculo de distancia, tramos en común, origen y destino comunes, etc. Este problema se solucionó gracias a PostGIS, la base de datos espacial de la que se ha hablado durante todo el proyecto y que ofrece, de manera nativa, herramientas para el manejo de este tipo especial de datos.

A lo largo del desarrollo del proyecto se han tenido una serie de dificultades. Primero, los escasos conocimientos que tenía en el SO Android, los cuales quería incrementar y que, sin duda, contribuyeron a buscar un Trabajo de Fin de Grado en el que hubiera que realizar una aplicación en Android. Segundo, las bases de datos espaciales, nunca había trabajado con ellas y no sabía cómo funcionaban o si iba a saber arreglármelas con ellas. Esto sumado a que no existe un soporte/ plugin/ extensión que, de manera nativa, ayude a relacionar datos geográficos y geométricos de la BBDD PostGIS con las clases Java.

Como conocimientos adquiridos me llevo, por una parte, la satisfacción de por fin haber aprendido a manejarme con Android, la cual era mi asignatura personal “casi” pendiente. Por otra parte, haber trabajado todo este tiempo con este tipo de base de datos me ha parecido de lo más interesante, es cierto que la he sufrido (y mucho), pero por otra parte agradezco la gran oportunidad que he tenido de poder trastear con ella todo lo que he querido, y más, mucho más.

Lo que más me ha marcado en este proyecto, es la oportunidad de formar parte del proyecto Prometeo de la Universidad de Valladolid, del cual CommonGO es un proyecto premiado. Sin duda ha sido una experiencia inolvidable donde he aprendido muchas cosas, y lo que todavía queda, ya que todavía no está finalizado.

10.1 TRABAJOS FUTUROS

Una de las cosas que más he lamentado durante el proceso de desarrollo de CommonGo ha sido la falta de tiempo ya que, a medida que más trabajaba en el proyecto, más ideas se me ocurrían para añadirle funcionalidad:

- Añadir **ubicación de residencia** del usuario en los datos personales.
- Sugerir vuelta a casa.
- Añadir funcionalidad de **need a ride**, que indicaría al usuario cuando le hace falta ir a un destino en concreto, qué otro usuario va en esa dirección.
- Añadir valoraciones y reportes a usuarios.
- Añadir amigos.
- Compañeros recientes.
- Sugerir rutas alternativas a la frecuente.
- **Diferenciar** entre coche/ bici/ correr a la hora de grabar rutas, para poder formar grupos de rutas en bici, conocer gente también haciendo deporte, etc.

11 BIBLIOGRAFÍA

- Ableson, F., Collins, C. and Sen, R. (2011). *Android*. Madrid: Anaya Multimedia
- Ribas Lequerica, J. (2011). *Manual imprescindible de desarrollo de aplicaciones para Android*. Madrid: Anaya Multimedia.
- Gargenta, M. (2011). *Learning Android*. Sebastopol, Calif.: O'Reilly
- Developers, P. (2016). *PostGIS — Spatial and Geographic Objects for PostgreSQL*. [online] Postgis.net. Disponible en: http://postgis.net/ [Accedido 11 mar. 2017].
- Corti, P., Mather, S., Kraft, T. and Park, B. (2014). *PostGIS Cookbook*. Birmingham: Packt Publishing.
- Martínez Llario, J. (2013). *PostGIS 2*. Lexington, KY: Juan Carlos Martínez Llario.
- Obe, R. and Hsu, L. (2011). *PostGIS in action*. Greenwich, Conn.: Manning.
- Cita previa INEM. (2017). *¿Cuánto cuesta un trabajador?*. [online] Disponible en: http://www.citapreviainem.es/cuanto-cuesta-un-trabajador/ [Accedido 7 Jul. 2017].
- Google Developers. (2017). *Google Developers*. [online] Disponible en: https://developers.google.com [Accedido 11 Jul. 2017].
- Postgis.net. (2017). *PostGIS 2.3.4dev Manual*. [online] Disponible en: https://postgis.net/docs/ [Accedido 16 Jun. 2017].
- QSM SLIM-Estimate. (n.d.). *Function Point Languages Table*. [online] Disponible en: http://www.qsm.com/resources/function-point-languages-table [Accedido 5 Jul. 2017].
- www.idc.com. (2017). *IDC: Smartphone Vendor Market Share*. [online] Disponible en: http://www.idc.com/promo/smartphone-market-share/vendor [Accedido 24 Mar. 2017].