Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de una app basada en Android para la gestión de viajes compartidos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Sancho García, Juan

Tutor: Conejero Casares, José Alberto

Cotutor: Poza Luján, José Luis

2016 - 2017

Resumen

En los últimos años la sociedad está experimentando un auge de la economía colaborativa en distintos sectores. Estos modelos están respaldados por plataformas tecnológicas que facilitan la interconectividad de los usuarios, la gestión de los procesos y pagos, así como intentar dar a conocer la reputación de los usuarios y agentes participantes. En lo que respecta al sector del transporte, en la actualidad existe una demanda creciente por parte de los ciudadanos de compartir sus propios vehículos con terceros en viajes. Esto no sólo permite ahorrar dinero a los usuarios, sino que algunos de ellos están encontrando una fuente de recursos. Este auge no sólo viene determinado por cuestiones económicas, sino que en muchos casos permite tener mayor facilidad a la hora de combinar medios de transporte, y de paso poder contribuir a la mejora del medio ambiente. Iniciativas como Uber, Cabify, o BlaBlaCar hacen uso de esta idea general, con ciertos matices.

En este trabajo, realizaremos la planificación, análisis, diseño e implementación de una aplicación para Android consistente en la implementación genérica de este tipo de modelo de negocio, que permita a los usuarios publicar ofertas de trayectos que vayan a realizar, así como consultar las ofertas disponibles para incorporarse al trayecto que les interese.

Palabras clave: Android, TFG, transporte, proyecto, usuarios, trayectos, negocio, aplicación, viajes.

Tabla de contenidos

1. Introducción

1.1. Motivación

En pleno siglo XXI, es innegable que nuestro día a día está marcado de manera notable por la tecnología. Multitud de servicios han ido apareciendo en los últimos años gracias al desarrollo tecnológico, y la informática ha sido un área clave en este desarrollo.

El campo de las **aplicaciones móviles** ha sido uno de los elementos principales en la modernización social. La aparición de smartphones supuso el nacimiento de una industria que en pocos años se ha convertido en un imprescindible para la vida de las personas.

Dentro de esta industria hay aplicaciones de todo tipo, las cuales podemos catalogar en múltiples categorías. Algunas de ellas son:

- Arte y diseño.
- Automoción.
- Belleza.
- Citas.
- Compras.
- Deportes.
- Educación.
- Entretenimiento.
- Finanzas.
- Fotografía.
- Medicina.
- Música.
- Salud y bienestar.
- Tiempo.
- **Transportes**.
- Videojuegos.

Introduciéndonos un poco en la materia de la que esta memoria tratará, nos centraremos en la categoría de aplicaciones móviles orientada a los servicios de transportes.

En el sector de los transportes ha aparecido un número importante de aplicaciones orientadas a ofrecer una alternativa a los servicios tradicionales de transporte. Aplicaciones tan conocidas como BlaBlaCar, Uber o Cabify permiten a sus usuarios desplazarse a otros puntos de su ciudad, país o incluso a otros países, sin tener que utilizar un avión, tren, autobús, etc.

En algunos casos, estas alternativas consisten en compartir los vehículos privados de los propios usuarios, para minimizar costes a la hora de desplazarse a destinos acordados desde un punto de partida común. La creciente demanda de este tipo de servicio da pie a un proyecto consistente en desarrollar un estándar de aplicación que cubra las características básicas de los usuarios con este tipo de necesidades.

El producto final de este proyecto dará pie a que, con sencillas modificaciones, se pueda orientar la aplicación a un grupo más específico de usuarios (ej.: viajes de negocios, trayectos a universidades, desplazamientos de aeropuertos al centro de la ciudad, desplazamientos a festivales de música, ...).

1.2. Objetivos

Los objetivos del proyecto son los siguientes:

- Desarrollar una aplicación en Android que represente un estándar de las aplicaciones donde los usuarios comparten sus vehículos particulares para desplazarse.
- Permitir la adaptación a un grupo más específico de usuarios a través de restricciones o funcionalidades fáciles de incluir en el código fuente.
- Asegurar la actualización de los datos en tiempo real, para que toda la información ofrecida esté actualizada en todo momento en cualquier dispositivo.
- Implementar una organización óptima en la base de datos que contenga la información de usuarios y trayectos.

1.3. Descripción del documento

Capítulo 1: Introducción del proyecto, definición de objetivos y descripción breve de los capítulos que componen este documento.

Capítulo 2: Elección del entorno de desarrollo, justificando la elección frente a otras opciones.

Capítulo 3: Análisis de requisitos, especificación de casos de uso y definición de requisitos específicos.

Capítulo 4: Diseño de la aplicación. Diseño de vistas y de base de datos, así como de la arquitectura que se va a llevar a cabo.

Capítulo 5: Implementación de la aplicación. Muestra del código PHP más importante y capturas de pantalla del resultado final.

Capítulo 6: La capacidad de escalabilidad y crecimiento de la aplicación. Posibles ampliaciones futuras para el proyecto.

Capítulo 7: Conclusiones.

Capítulo 8: Bibliografía.

2. Entorno de desarrollo

2.1. Introducción

El producto final de este proyecto es una aplicación para el sistema operativo Android. Debido a la popularidad de este SO en smartphones, tabletas, etc., hay una gran diversidad de entornos de desarrollo (IDE) que permiten desarrollar apps para su posterior publicación en las tiendas de aplicaciones. A continuación, exploraremos las principales posibilidades que se nos presentan y justificaremos la elección del IDE elegido para desarrollar esta app.

2.2. Sistemas similares

A continuación, presentamos una breve descripción de los principales IDE para el desarrollo en Android:

ECLIPSE

Plataforma de integración basada en plugins. Es compatible con una gran variedad de lenguajes. Es un IDE genérico. Fue elegido durante mucho tiempo por Google como el entorno ideal para iniciarse en el desarrollo de aplicaciones.

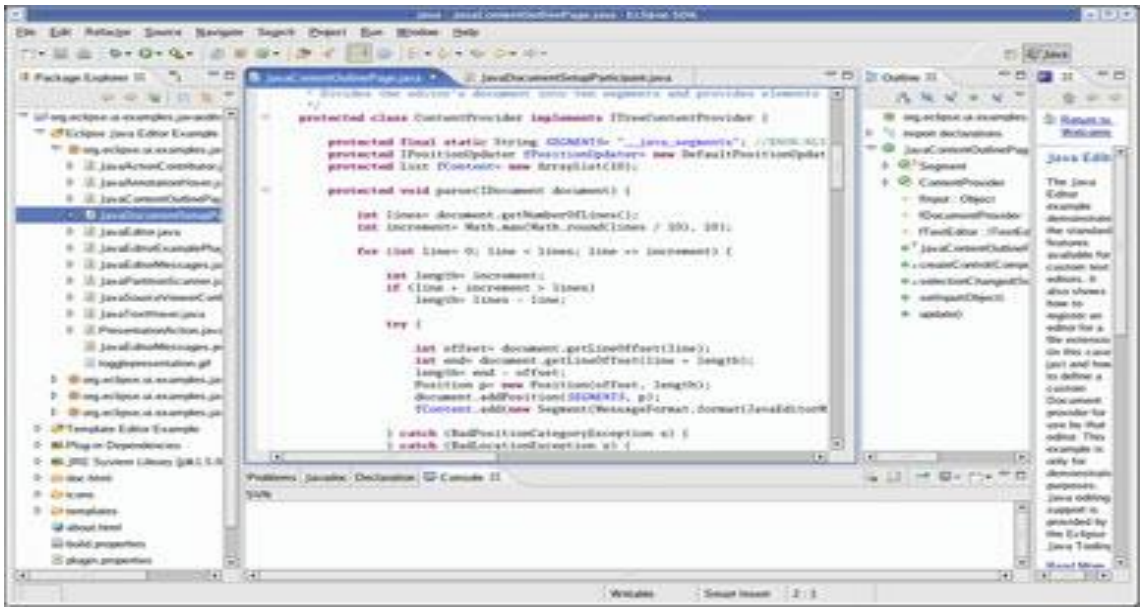

Ilustración 1: Pantalla de Eclipse

BASIC4ANDROID

Programa con el lenguaje VisualBasic. Es un entorno más gráfico y menos abstracto que el de Eclipse, y los avances a la hora de programar se ven más rápido.

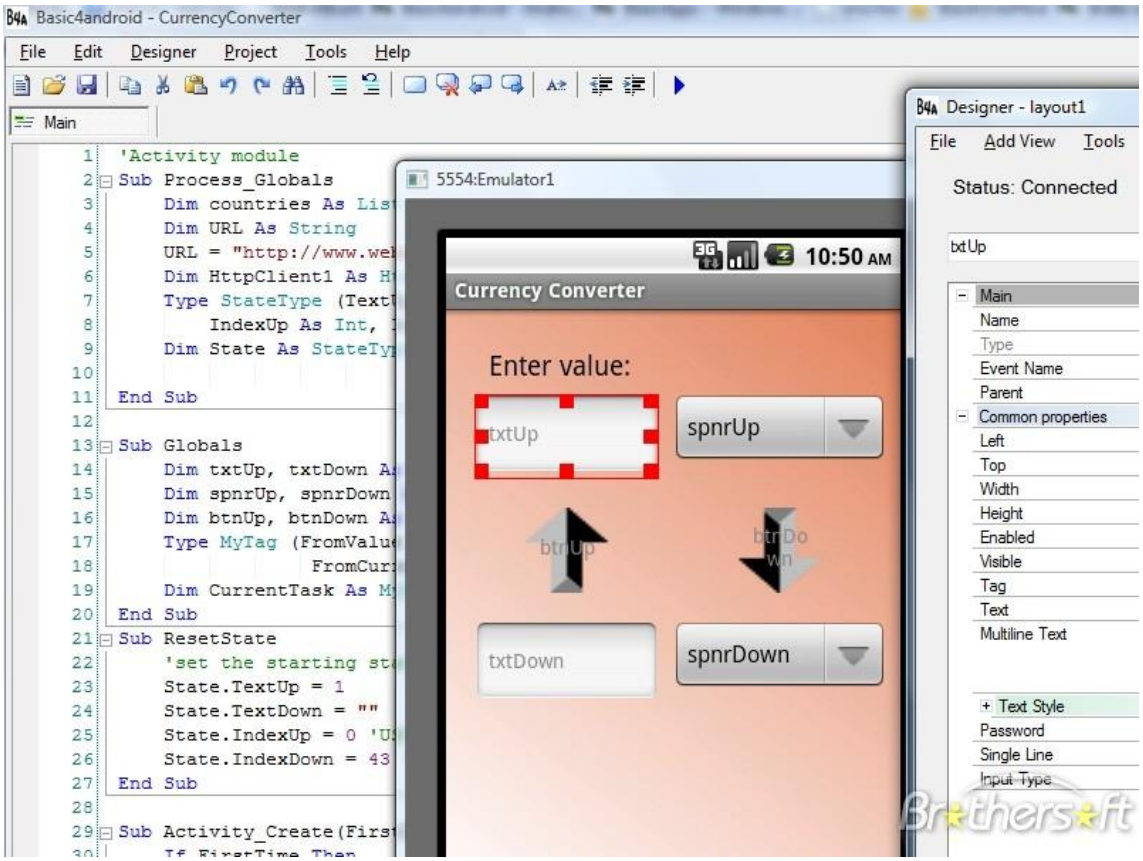

Ilustración 2: Pantalla de Basic4Android

APP INVENTOR

IDE desarrollado por Google Labs para la gente que no sabe programar, pero desea crear aplicaciones. No hace falta escribir código.

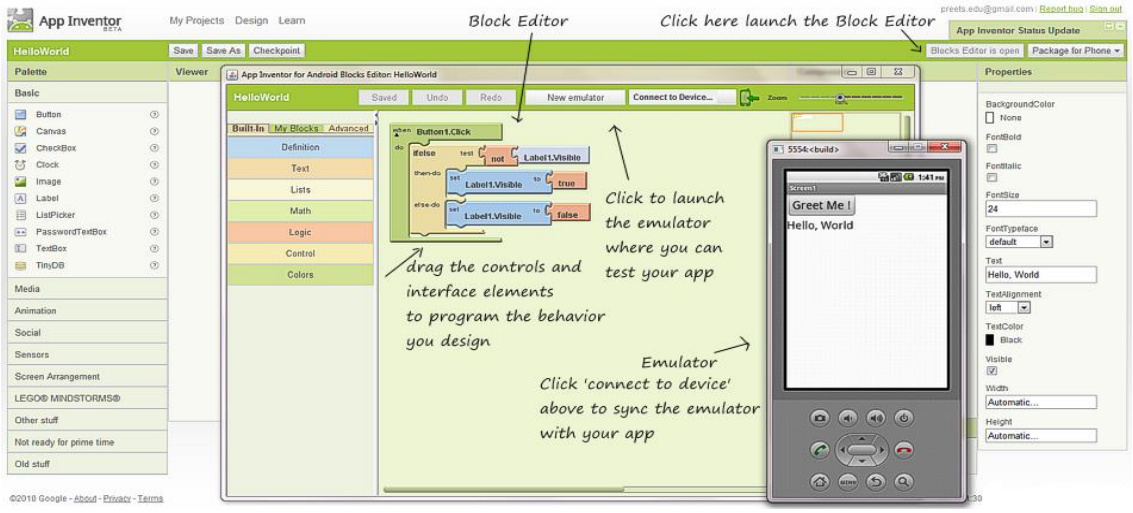

Ilustración 3: Pantalla de AppInventor

LIVECODE

Programación tanto para Android como para iOS, aplicaciones de escritorio, etc. IDE multiplataforma.

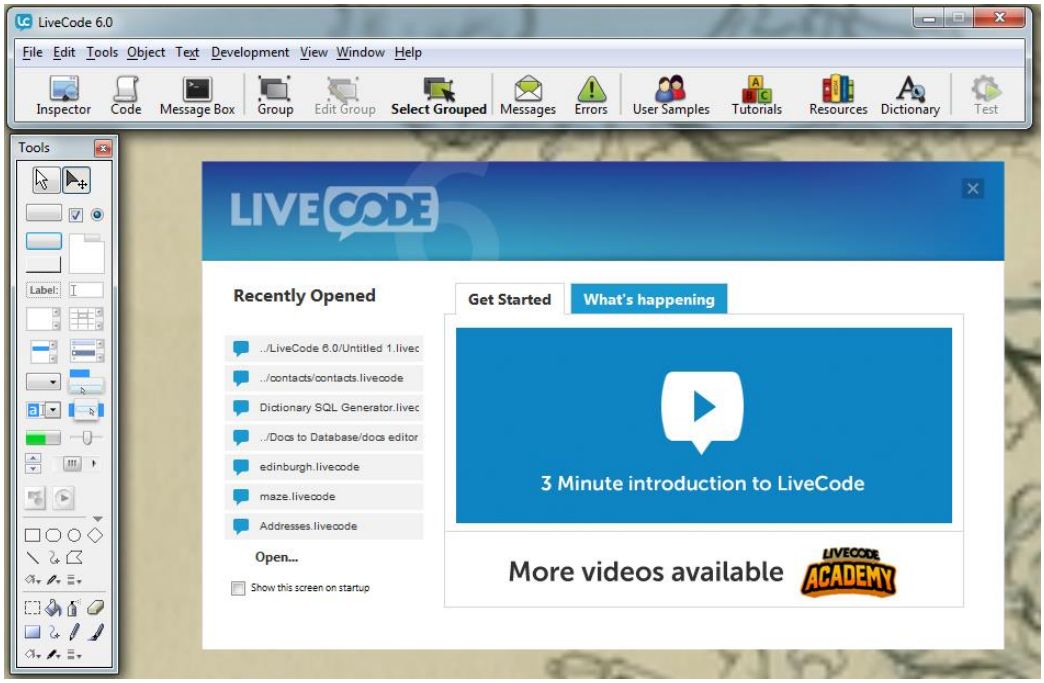

Ilustración 4: Pantalla de Livecode

ANDROID STUDIO

IDE oficial para la programación de aplicaciones en Android. Basado en el software de JetBrains, sustituyó a Eclipse como entorno de desarrollo integrado oficial.

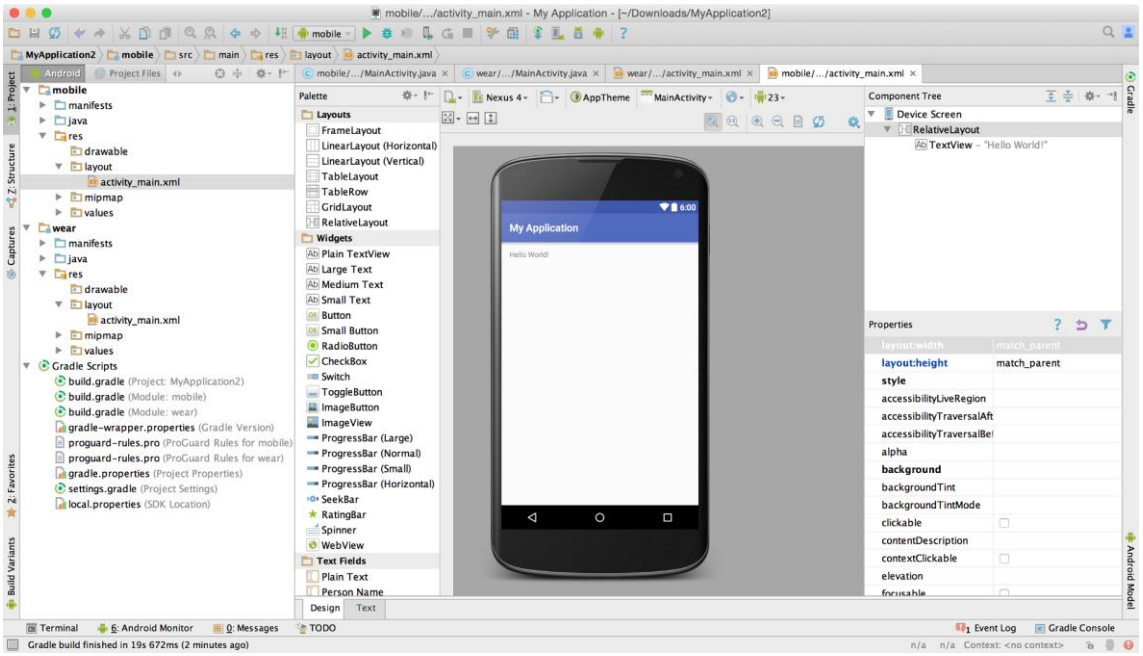

Ilustración 5: Pantalla de Android Studio

2.3. Tecnología a utilizar

En este proyecto utilizaremos el Android Studio por una serie de razones muy claras:

- La experiencia que tengo en el desarrollo de aplicaciones de Android está vinculada al uso de este IDE.
- Al ser el IDE oficial ofrecido por Google, dispone de una serie de herramientas altamente útiles:
    - Depuración USB, para hacer pruebas desde un smartphone real.
    - Simulador, en caso de no poder utilizar la depuración USB (principalmente se usa para la revisión de la interfaz en smartphones de distintos tamaños).
    - Renderizado en tiempo real.
    - Refactorización específica de Android.
    - Consola de desarrollador con consejos de optimización, estadísticas de uso, y muchas otras herramientas.
    - ...

- La generalidad tecnológica en IDEs como Eclipse o LiveCode tiene ventajas enormes en muchos casos, pero para este proyecto lo mejor es utilizar una herramienta lo más especializada posible, y por eso descartamos ambos IDEs.
- El IDE AppInventor también queda descartado por estar orientado a gente que no sabe programar.
- Finalmente, Basic4Android también era una opción interesante, pero como programador me siento más cómodo trabajando en Java, en vez de hacerlo en VisualBasic, y por tanto también descartamos esta posibilidad.

Además, el hecho de hacerla para Android y no para iOS viene dado por tres razones principales:

- 1) Para desarrollar para iOS debes hacerlo sobre un Macintosh o un iMac. El desarrollo de Android se puede hacer desde cualquier SO.
- 2) Para poder subir una app a la tienda de aplicaciones de iOS debes pagar 100$ anuales. Para hacerlo en la tienda de Android debes pagar 25$ y la cuenta es tuya de por vida.
- 3) El nº de usuarios de Android es incomparable al nº de usuarios de iPhones y iPads.

Por tanto, la aplicación se desarrollará para móviles y tabletas que trabajen con el SO de Android, y para ello utilizaremos el entorno de desarrollo integrado oficial facilitado por Google, Android Studio.

3. Análisis de requisitos

3.1. Introducción

Una vez realizada la presentación de la realidad que rodea a este proyecto, y habiendo definido el entorno desde el que desarrollaremos la aplicación, vamos a definir el comportamiento de la misma mediante la especificación de requisitos del software, basándonos en las referencias del estándar IEEE 830.

3.2. Terminología

En este apartado vamos a definir las palabras técnicas a las cuales nos referiremos en los siguientes puntos de este apartado, y que es necesario explicar brevemente.

Las palabras son las siguientes:

- Aplicación → producto final de este proyecto. Cuando hablemos de la aplicación, nos referiremos a la aplicación de Android que vamos a desarrollar.
- Registro → creación de un nuevo perfil de usuario en la aplicación.
- Login → inicio de sesión de un usuario en la aplicación.
- BBDD → base de datos vinculada a la aplicación.
- Scripts → ficheros PHP utilizados para la comunicación entre app y BBDD.

3.3. Descripción general

3.3.1. Funcionalidad del producto

Este proyecto es una aplicación dirigida a usuarios que quieren compartir vehículos particulares para desplazarse a destinos comunes. Por tanto, nuestros usuarios tendrán la opción, para cada trayecto, de actuar como conductores de sus vehículos, o como pasajeros de otros usuarios. Puntualizar que un usuario puede ser conductor en algunos trayectos, y pasajero en otros, la elección del rol puede variar bajo demanda.

Debido a la distinción de estos dos roles, vamos a separar los casos de uso para pasajeros y conductores, ya que, aunque un usuario pueda utilizar ambas opciones, en función de la elegida los casos de uso serán diferentes. Los mostramos en la siguiente ilustración:

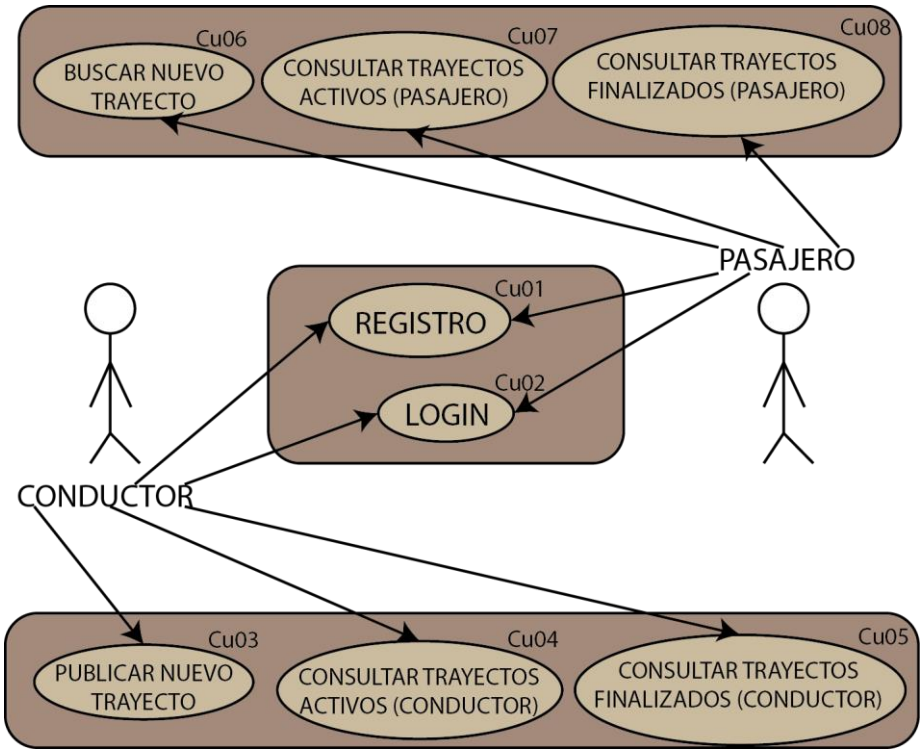

Ilustración 6: Diagrama de casos de uso

3.3.2. Características de los usuarios

Las características de los usuarios mostrados en la sección 3.3.1 se explican con mayor detalle a continuación:

En nuestro producto, los usuarios que no tengan un perfil creado deberán crearse una cuenta para poder acceder al contenido de la aplicación (**Cu01**). Una vez hecho esto, podrán identificarse (**Cu02**) y realizar cualquiera de las funciones ofrecidas.

Los usuarios podrán escoger su rol (pasajero o conductor), y en función de la opción escogida tendrán un menú u otro.

En el menú de los conductores, las diferentes opciones que aparecerán serán:

- Publicar un nuevo trayecto (**Cu03**) → rellenar un formulario con todos los datos necesarios de un trayecto, y publicarlo para que los pasajeros interesados puedan apuntarse.

- Consultar los trayectos activos (**Cu04**) → se podrán visualizar los detalles de todos los trayectos publicados que todavía no se han realizado o no han terminado. Uno de los datos de mayor interés en este apartado es poder ver el número de pasajeros que se han inscrito al trayecto. También se podrá indicar que un trayecto ya ha finalizado (o cancelarlo).

- Consultar los trayectos finalizados (**Cu05**) → se guardará un historial de los trayectos realizados, para poder consultarlos en cualquier momento.

En el menú de los pasajeros, las diferentes opciones que aparecerán serán:

- Buscar un nuevo trayecto (**Cu06**) → un usuario desea inscribirse a un trayecto publicado por un conductor. Aquí podrá ver los trayectos publicados, y en caso de encontrar uno que se adapte a sus necesidades, inscribirse.

- Consultar los trayectos activos (**Cu07**) → como pasajero, los datos de los trayectos pendientes son algo importante a lo que se debe poder acceder fácilmente. En esta opción aparecerán los detalles de los trayectos en los que el pasajero se ha inscrito.

- Consultar los trayectos finalizados (**Cu08**) → aquí aparecerán los trayectos en los que el pasajero se ha inscrito y el conductor ha dado por finalizados/cancelados. Esta opción permite la consulta de trayectos anteriores en cualquier momento.

3.4. Requisitos específicos

A continuación, se expone la definición de requisitos y las acciones a realizar dentro de la aplicación. Para ello, vamos a describir las filas que compondrán las tablas de descripción:

- Nombre → nombre del requisito.
- Actor → usuario que realiza la acción.
- Objetivo → objetivo a cumplir por parte del requisito.
- Precondición → condición o condiciones a cumplir para llegar a este requisito.
- Postcondición → acción a realizar después de realizar esta acción.
- Descripción → explicación de la acción explicada.
- Caso de uso → caso de uso al que hace referencia.

Nombre	Creación de un nuevo usuario
Actor	Conductor/Pasajero
Objetivo	Crear una cuenta con la que poder acceder a la aplicación
Precondición	Ninguna
Postcondición	Ninguna
Descripción	Es requisito indispensable para utilizar la aplicación, ser un usuario registrado. Por tanto, todos los usuarios activos deberán haber creado una cuenta previamente.
Caso de uso	Cu01

Nombre	Inicio de sesión (Login)
Actor	Conductor/Pasajero
Objetivo	Acceder al contenido de la aplicación
Precondición	Registro del usuario
Postcondición	Ninguna
Descripción	Introducción de los campos email y password especificados en el registro.
Caso de uso	Cu02

Nombre	Selección del rol
Actor	Conductor/Pasajero
Objetivo	Definir el rol que el usuario va a utilizar en la sesión actual de la aplicación
Precondición	Inicio de sesión
Postcondición	Ninguna
Descripción	Cada vez que un usuario acceda a la app, podrá elegir si para esa sesión desea el menú de pasajero o de conductor.
Caso de uso	Cu03 – Cu08

Nombre	Publicar trayecto
Actor	Conductor
Objetivo	Publicar un nuevo trayecto para que pasajeros interesados puedan unirse
Precondición	Selección de rol conductor
Postcondición	Publicación del trayecto para que sea visible para los pasajeros
Descripción	El conductor rellena un formulario con los datos clave del trayecto. Estos datos podrán ser consultados por los pasajeros, para inscribirse a la oferta en caso de estar interesados.
Caso de uso	Cu03

Nombre	Consultar trayectos activos (conductor)
Actor	Conductor
Objetivo	Consultar los datos de los trayectos publicados, pero no realizados o terminados
Precondición	Selección de rol conductor, publicar trayecto
Postcondición	Ninguna
Descripción	El conductor podrá consultar todos los datos del trayecto que ha publicado y siguen vigentes, varios si es el caso, así como dar por finalizado o cancelado un trayecto
Caso de uso	Cu04

Nombre	Consultar trayectos finalizados (conductor)
Actor	Conductor
Objetivo	Consultar los datos de los trayectos finalizados
Precondición	Publicar trayecto, cerrar un trayecto vigente
Postcondición	Ninguna
Descripción	El conductor podrá consultar todos los datos de los trayectos finalizados o cancelados, por si la información le resulta de interés para futuras publicaciones
Caso de uso	Cu05

Nombre	Buscar un nuevo trayecto
Actor	Pasajero
Objetivo	Conseguir un trayecto para el desplazamiento que debe realizar el usuario
Precondición	Selección de rol pasajero
Postcondición	Mostrar los trayectos publicados que están disponibles
Descripción	El usuario podrá ver una lista de los trayectos publicados, así como apuntarse al que le interesa, en caso de encontrar alguno que se adapte a sus necesidades
Caso de uso	Cu06

Nombre	Consultar trayectos activos (pasajero)
Actor	Pasajero
Objetivo	Consultar los datos de los trayectos en los que el usuario está inscrito, pero no han terminado ni han sido cancelados por el conductor
Precondición	Selección de rol pasajero, buscar trayecto
Postcondición	Ninguna
Descripción	El pasajero podrá consultar todos los datos del trayecto al que se ha suscrito, varios si es el caso, y siguen vigentes.
Caso de uso	Cu07

Nombre	Consultar trayectos finalizados (pasajero)
Actor	Pasajero
Objetivo	Consultar los datos de los trayectos finalizados
Precondición	Buscar trayecto
Postcondición	Ninguna
Descripción	El pasajero podrá consultar todos los datos de los trayectos finalizados o cancelados por el conductor, por si la información le resulta de interés para futuras búsquedas de trayectos
Caso de uso	Cu08

3.5. Conclusiones

En este capítulo hemos definido algunos conceptos que requerían una breve explicación para su mejor comprensión, y que se utilizan en el resto del documento.

A través del uso de la IEEE 830, hemos especificado los requisitos del software y definido las características y acciones de la aplicación.

El primer punto del capítulo es una introducción a éste. El segundo punto explica la terminología utilizada para la definición del proyecto. Después, tratamos la funcionalidad de la aplicación, mediante la definición de los casos de uso, así como la definición de requisitos y acciones, vinculando cada una de ellas al caso de uso (o casos de uso) correspondiente/s.

4. Diseño

4.1. Introducción

En este apartado de diseño vamos a describir con más detalle cómo la aplicación va a satisfacer los requisitos, a todos los niveles. Vamos a describir más detalladamente el funcionamiento de cada módulo que compone la aplicación.

4.2. Especificación conceptual

El funcionamiento general de la aplicación es el siguiente:

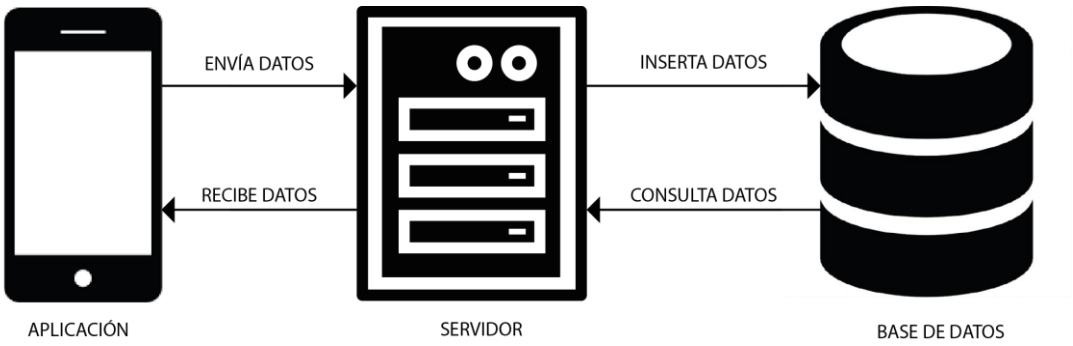

Ilustración 7: Esquema de comunicaciones

Todos los datos de usuarios y trayectos deben almacenarse en una base de datos remota, para mantener la información actualizada en todos los dispositivos.

Para ello, tanto al crear usuarios nuevos como al publicar trayectos, la aplicación debe recoger los datos introducidos en los formularios correspondientes, y enviarlos al servidor, que se ocupará de almacenar correctamente la información en la base de datos.

Asimismo, cuando un usuario quiera iniciar sesión, necesitaremos leer de la base de datos, para realizar la comprobación de existencia del usuario y concordancia email/pasword. Para esto, la aplicación enviará al servidor los datos del login, y éste hará la consulta a base de datos en función de los parámetros introducidos.

Además, toda la información que queramos recoger de trayectos publicados/cancelados/finalizados, la obtendremos consultando la base de datos.

4.3. Capa de persistencia

La base de datos es una base de datos relacional MySQL.

La elección de este tipo de base de datos dota de una serie de ventajas que detallamos a continuación:

- Se facilita poder garantizar que no habrá registros duplicados.
- Resuelve automáticamente los problemas de integridad referencial.
- El uso de tablas para la organización de los datos, y la realización de consultas mediante SQL, facilita su uso.

Por tanto, utilizaremos una base de datos relacional MySQL alojada en un servidor remoto para almacenar los datos que nuestra aplicación desee tener disponibles en todo momento, desde cualquier dispositivo.

A continuación, mostramos el diagrama de la base de datos:

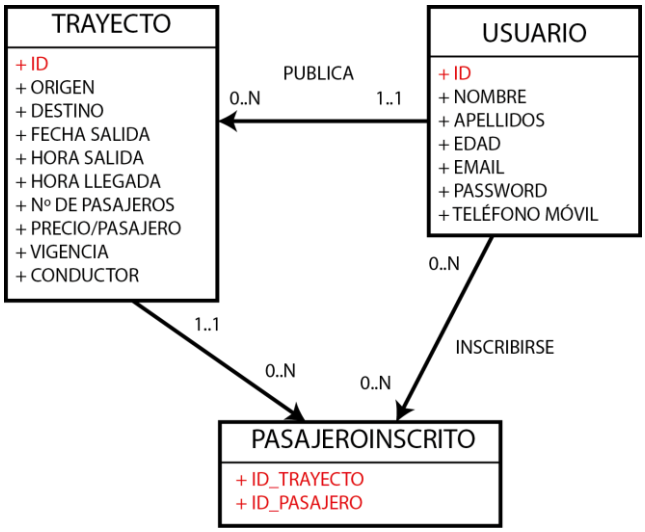

Ilustración 8: Diagrama de la base de datos

Con estas tres tablas podemos recoger toda la información que necesitamos:

- La información de un nuevo usuario se guarda como una nueva fila en la tabla Usuarios.
- La información de un nuevo trayecto, publicado por un usuario de rol conductor, se guarda como una nueva fila en la tabla Trayectos.
- Cuando un usuario de rol pasajero quiere inscribirse en un trayecto, en la tabla PasajeroInscrito guardaremos el id de dicho usuario, y el id del trayecto al que se quiere inscribir.

La aplicación se conectará al servidor para pasarle la información, o para solicitar la que necesite, y desde el servidor se realizará una conexión a la base de datos a través de scripts PHP, los cuales almacenarán los nuevos datos en sus respectivas tablas, o devolverán la información que la aplicación ha solicitado.

4.4. Capa de presentación

En este apartado del capítulo se explica, mediante bocetos digitales, cómo se ha realizado el diseño de la aplicación. Al tratarse de una aplicación móvil, el sistema de vistas que se utiliza es el de ventanas, por lo que se va a mostrar el esquema de cada ventana junto a una breve descripción.

4.4.1. Diseño de la aplicación móvil

Ilustración 9: Esquema login

Esta es la primera ventana que aparece al abrir la aplicación, la ventana de Login.

Aquí podremos iniciar sesión siempre que hayamos creado una cuenta previamente.

En caso de no hacerlo, podremos crear un nuevo perfil a través del botón de registro, que nos llevará a la ventana de Registro.

Ilustración 10: Esquema registro

Esta es la ventana de Registro. Aquí es donde los nuevos usuarios podrán crearse una cuenta para poder acceder al contenido de la aplicación.

Todos los usuarios, tanto pasajeros como conductores, pasarán por esta ventana antes de empezar a utilizar la aplicación.

Todos los campos son obligatorios para poder crear el perfil.

Ilustración 11: Esquema elección de rol

Una vez el usuario inicia sesión en la aplicación, lo primero que deberá hacer es elegir el rol que desea utilizar durante esa sesión, ya que según las necesidades que tenga en cada momento, este rol puede variar.

Dependiendo de la opción que escoja, la aplicación le llevará a un menú de opciones más adaptado a sus necesidades.

A continuación, mostraremos primero las ventanas que podrán ver los conductores, y después pasaremos a las que pueden ver los pasajeros, para no perder el hilo del flujo de la aplicación.

VENTANAS DISPONIBLES SÓLO PARA CONDUCTORES

Ilustración 12: Esquema menú conductor

Esta es la ventana de menú de los conductores. Aquí tendrán tres opciones posibles, tal y como se muestra en la imagen.

La primera, los llevará a la ventana que permite publicar un nuevo trayecto, para que los pasajeros interesados puedan inscribirse.

La segunda permite consultar los trayectos publicados y que todavía están vigentes.

La tercera permite consultar los datos de todos los trayectos publicados por ese usuario hasta la fecha.

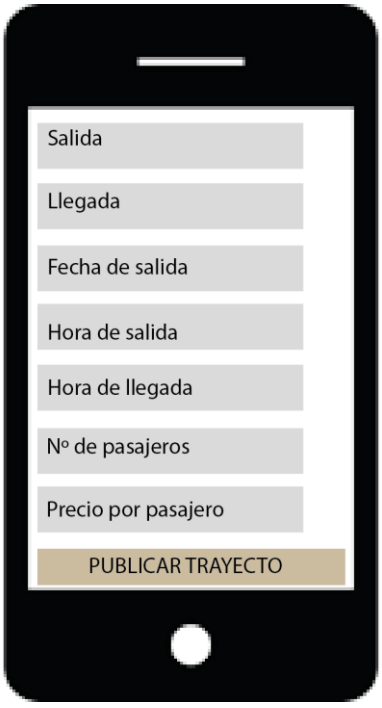

Ilustración 13: Esquema publicar trayecto

Esta es la ventana desde la que los conductores pueden publicar un nuevo trayecto.

Deberán rellenar todos los campos que aparecen en la imagen.

Una vez hecho esto, podrán publicar el trayecto y éste será visible para todos los pasajeros.

Ilustración 14: Esquema listados

Esta es la ventana de trayectos activos. Lo que se ve es una lista con todos los trayectos publicados por el conductor y que siguen vigentes.

Al pulsar en uno de ellos pasaremos a la ventana de detalles del trayecto.

NOTA IMPORTANTE → este listado de trayectos es igual para trayectos tanto activos como finalizados, de conductor y de pasajero, solo que estos últimos ven los trayectos a los que se han inscrito. Por tanto, más adelante no repetiremos la ventana, por no alargar más todavía este apartado.

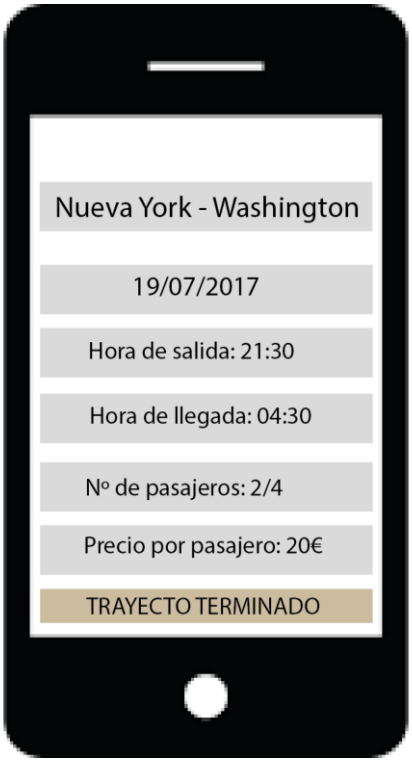

Ilustración 15: Detalle trayecto activo conductor

Esta es la ventana a la que nos lleva la aplicación pulsando en cualquiera de los trayectos disponibles (ventana anterior).

Aquí aparecen los detalles del trayecto, donde también se puede ver el nº de pasajeros que se han inscrito.

Pulsando el botón de trayecto finalizado, éste dejará de estar en el apartado de trayectos activos, y pasará al apartado de finalizados (siguiente ventana).

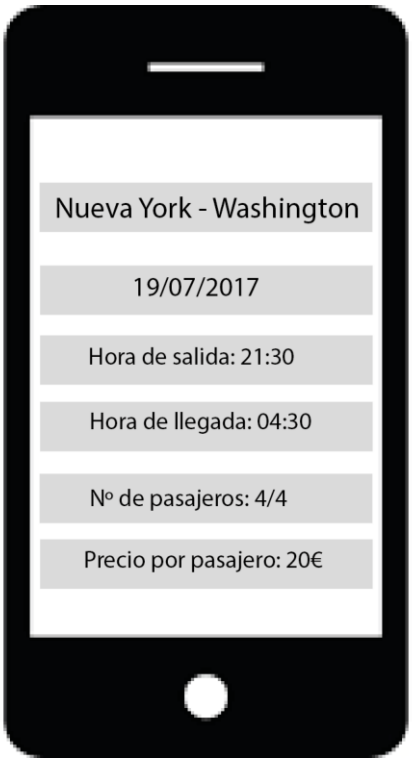

Ilustración 16: Detalle trayecto finalizado conductor

Y como última ventana de los pasajeros, podemos consultar los detalles de los trayectos finalizados, por si los usuarios quieren consultar la información del trayecto una vez acabado.

VENTANAS DISPONIBLES SÓLO PARA CONDUCTORES

Ilustración 17: Esquema menú pasajero

Esta es la ventana de menú de los pasajeros. Aquí tendrán tres opciones posibles, tal y como se muestra en la imagen.

La primera, los llevará a la ventana que permite buscar un nuevo trayecto, e inscribirse al que trayecto que les interese.

La segunda permite consultar los trayectos en los que se ha inscrito y que todavía están vigentes.

La tercera permite consultar los datos de todos los trayectos realizados por ese usuario hasta la fecha.

La ventana de “Ver trayectos disponibles” es simplemente un listado de los trayectos publicados, igual que la ventana de trayectos activos para conductores. Para no repetir imágenes no vamos a ponerla. La única diferencia funcional es que aquí los pasajeros verán todos los trayectos activos publicados en la aplicación, no solamente el de un conductor. Al pulsar en un trayecto en el que tengan interés, pasarán a la siguiente ventana:

Ilustración 18: Esquema suscripción a trayecto

Esta es la ventana de detalle del trayecto que ve el usuario de rol pasajero cuando pulsa en uno de los trayectos publicados, en “Ver trayectos disponibles”.

Desde esta ventana es desde la que el usuario podrá inscribirse a un trayecto.

Las vistas de los listados de “Mis trayectos activos” y “Mis trayectos finalizados”, como hemos dicho anteriormente, son iguales, cambiando únicamente el contenido de los listados, por lo que tampoco vamos a repetir aquí las imágenes. Solamente faltaría mostrar cómo se ve la ventana de detalle de trayectos activos y finalizados para los pasajeros, y lo hacemos a continuación:

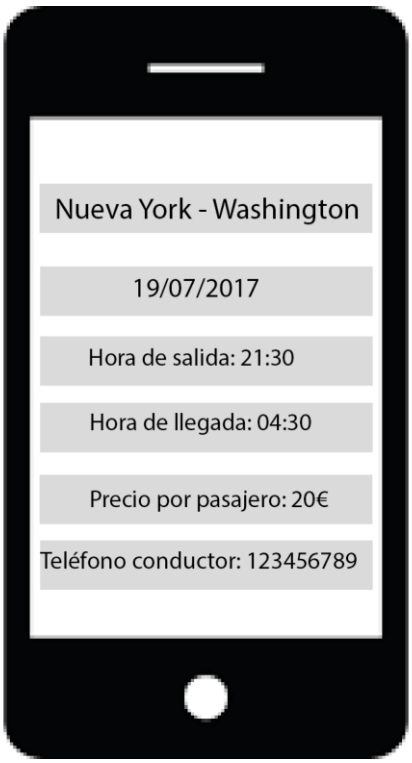

Ilustración 19: Esquema detalle pasajero

Esta es la ventana que verá el pasajero cuando apriete en alguno de los trayectos de los listados de “Mis trayectos activos” y “Mis trayectos finalizados”.

Destacar el dato del teléfono del conductor, esa es la manera en la que los pasajeros se pondrán en contacto con el conductor para acordar los puntos de recogida, y todo lo que necesiten acordar entre ellos.

Y hasta aquí la capa de presentación.

4.5. Capa lógica

Hemos creado unos diagramas de secuencia, con el fin de explicar de mejor manera el funcionamiento de la capa lógica de la aplicación.

En este caso se muestran los casos de uso del conductor y del pasajero, además de las que pueden hacer ambos, y todas las acciones que se pueden realizar por medio de ambos actores.

Común a pasajeros y conductores:

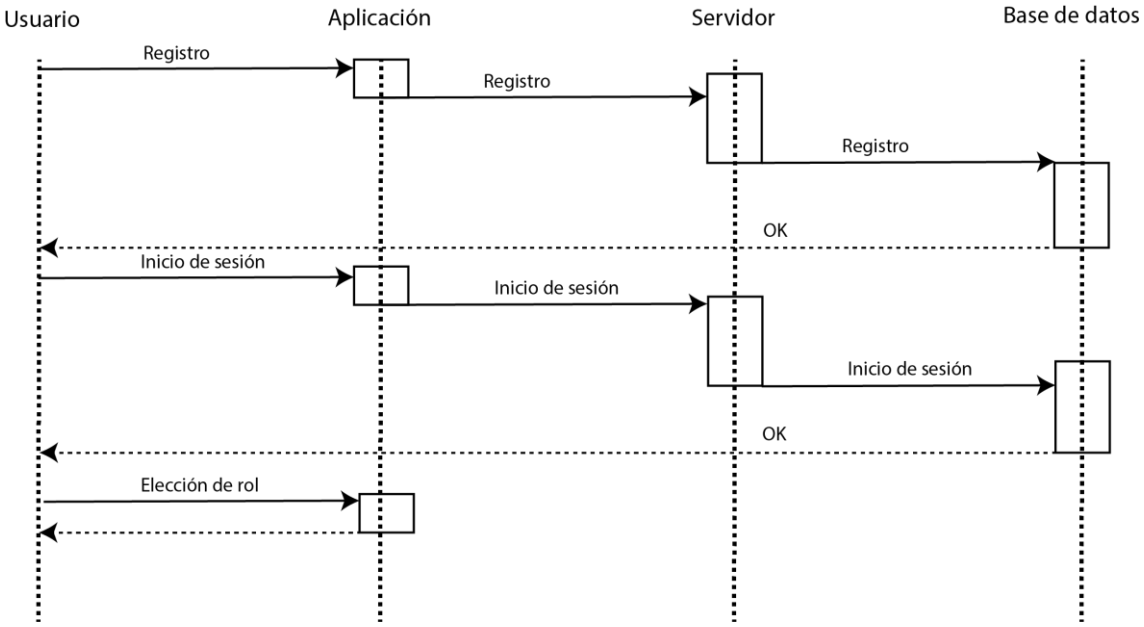

Ilustración 20: Diagrama de secuencia Cu01 y Cu02

Conductores:

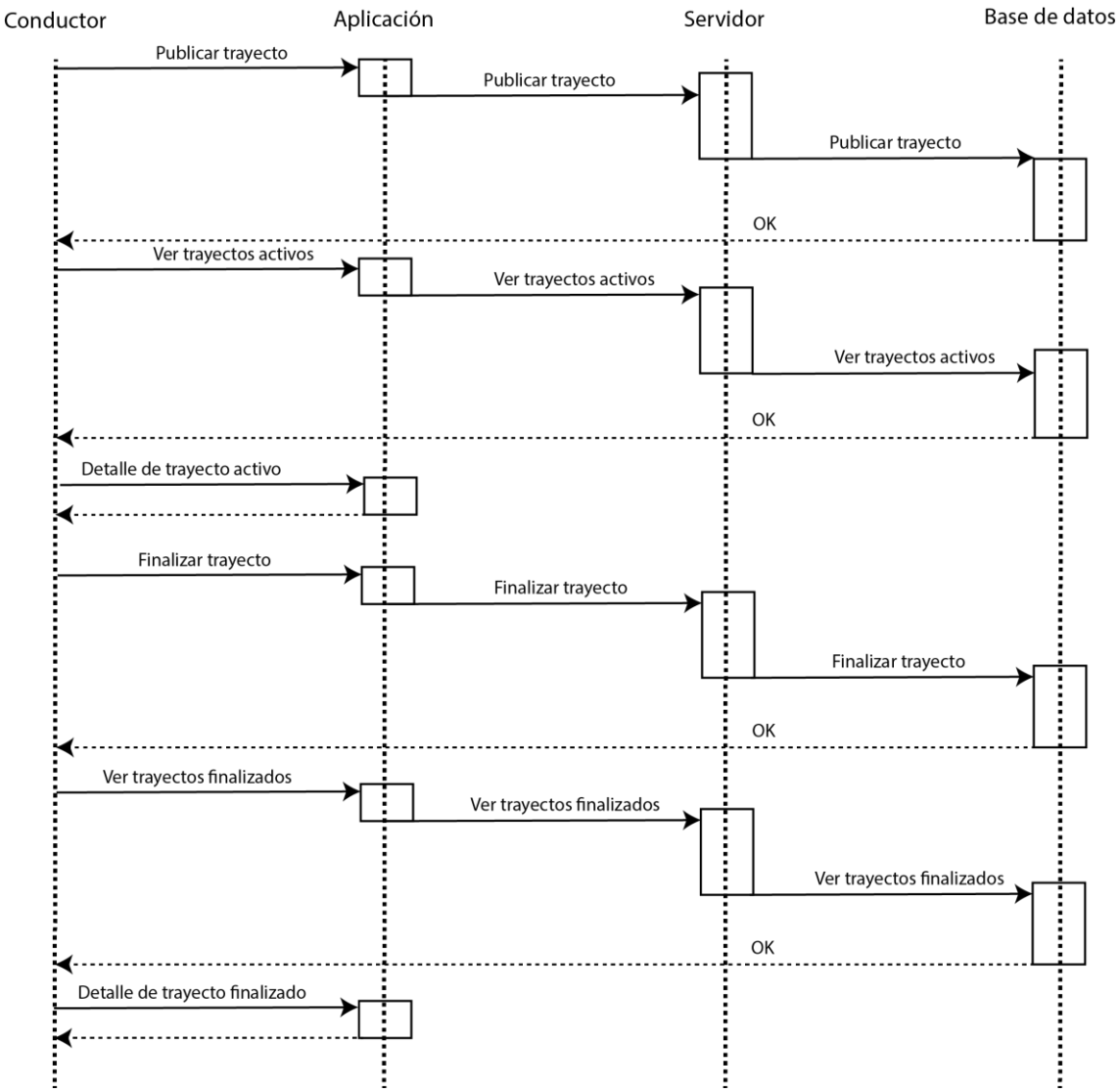

Ilustración 21: Diagrama de secuencia Cu03 - Cu05

Pasajeros:

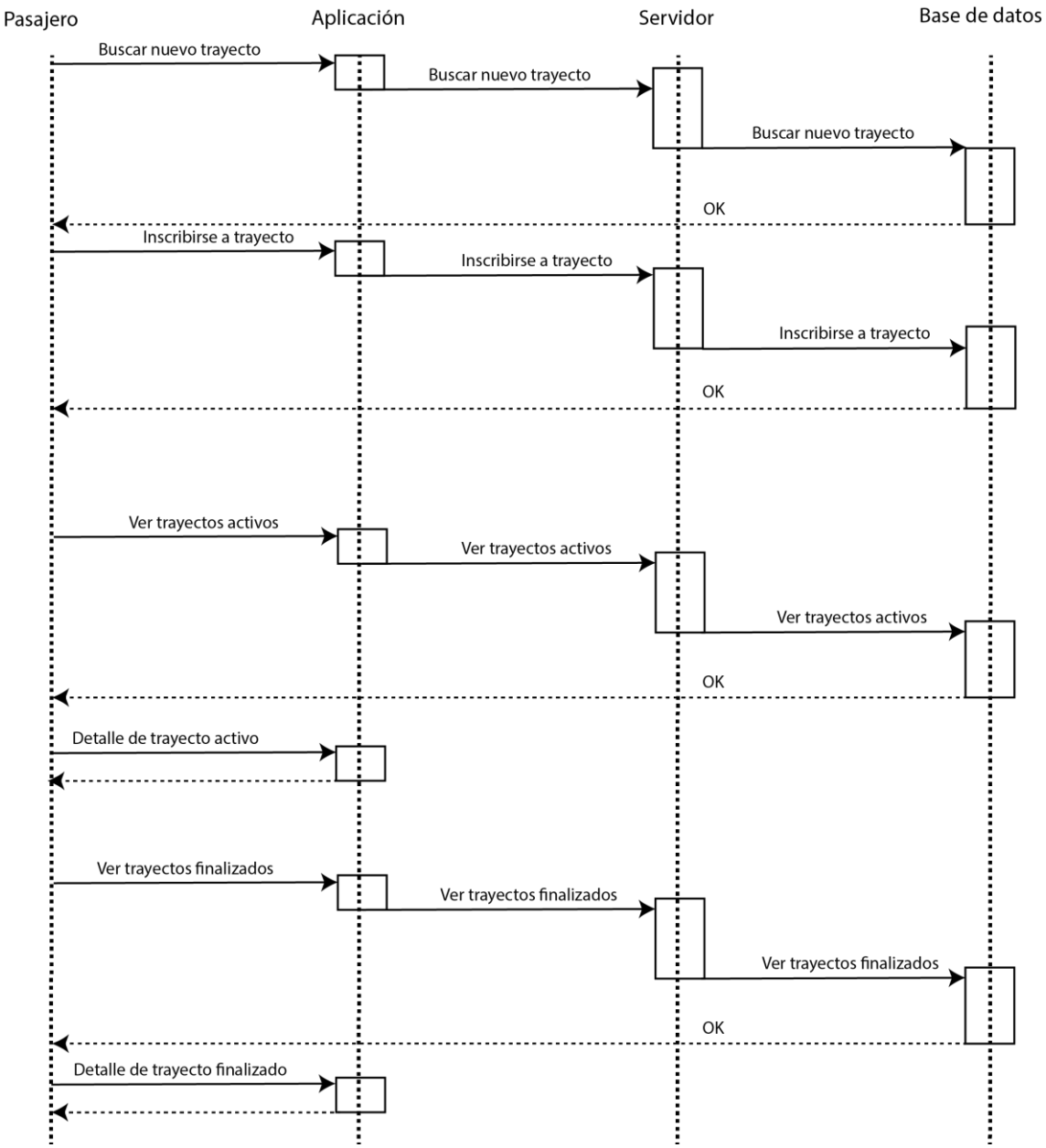

Ilustración 22: Diagrama de secuencia Cu06 - Cu08

4.6. Conclusiones

En este capítulo hemos mostrado el diseño de la aplicación. Para ello, hemos dividido en tres capas la estructura de la misma.

- En la capa de presentación hemos mostrado unos bocetos digitales de todas las ventanas de la aplicación, explicando brevemente el objetivo de cada una de ellas. Más adelante, en el capítulo de implementación, mostraremos el resultado final mediante capturas de pantalla a estas mismas ventanas.
- En la capa de lógica, hemos mostrado las acciones mencionadas anteriormente, en los casos de uso, utilizando para ello diagramas de secuencia para los distintos actores de la aplicación.
- En la capa de persistencia hemos explicado la manera que tiene la aplicación para conectarse a la base de datos remota, además de mostrar la estructura interna de la base de datos.

5. Implementación e implantación

En este capítulo, habiendo presentado toda la planificación del proyecto, se va a explicar cómo ha sido la implementación del proyecto basándonos en los capítulos anteriores.

Dicha implementación ha consistido en el desarrollo de una aplicación en Android, una base de datos remota, y scripts PHP alojados en el servidor para establecer la comunicación entre aplicación y base de datos.

A la aplicación se le ha puesto el nombre de "Unify", ya que pensamos que refleja el objetivo de este producto: una aplicación capaz de "heredar" (desde un punto de vista de la programación orientada a objetos) aplicaciones dirigidas a un grupo de usuarios más específico.

5.1. Herramientas utilizadas

- Aplicación Android → como ya hemos mencionado, la aplicación se ha desarrollado en su totalidad en el IDE Android Studio, por ser el presentado por Google como el IDE oficial para el desarrollo de aplicaciones. Ofrece un emulador muy útil para ejecutar las aplicaciones, además de permitir ejecutarlas en dispositivos físicos. La herramienta de Debug de este IDE ha sido muy utilizada para corregir fallos en el código, especialmente en los fragmentos de establecer comunicación con el servidor. La creación de interfaces es muy intuitiva, permitiendo su construcción vía XML o vía gráfica. En su conjunto, es uno de los IDE más completos en el desarrollo de aplicaciones.
- Base de datos → la base de datos es relacional, utiliza la tecnología MySQL, y está alojada en un servidor creado específicamente para este proyecto. Se puede acceder a ella mediante el panel de control de PHPMyAdmin. Esta herramienta de gestión de base de datos permite la creación/modificación/eliminación de tablas, filas, columnas, etc., también mediante dos vías: manualmente o mediante consultas SQL.
- Scripts PHP → estos scripts están alojados en el servidor donde se encuentra la base de datos. Gracias a ellos podemos comunicar la base de datos con la aplicación. Para desarrollar esta parte, se ha utilizado el IDE multilenguaje Sublime Text, ya que soporta el lenguaje PHP, entre otros cientos, y facilita el

desarrollo en este lenguaje (autocompletado de palabras, cierre automático de paréntesis y llaves, etc).

5.2. Implementación de la aplicación

En este apartado vamos a presentar cómo queda la aplicación final, tras haber realizado la implementación de la misma basándonos en el diseño descrito en el capítulo anterior.

Es importante destacar que no se van a mostrar capturas de pantalla de toda la aplicación, ya que en muchos casos las interfaces son similares, cambiando simplemente el contenido. Es el caso de los listados de trayectos, que aparecen en varios puntos de la aplicación, y que por lo tanto solamente mostraremos uno de ellos, con el fin de no extender este documento con información que no aporta nada nuevo.

LOGIN Y REGISTRO

Ilustración 23: Login

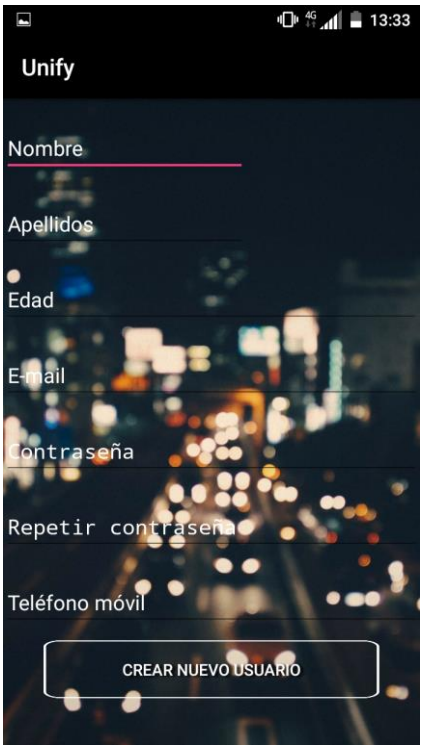

Ilustración 24: Registro

PANTALLA DE SELECCIÓN DE ROL

Ilustración 25: Elección de rol

MENÚS CONDUCTOR Y PASAJERO

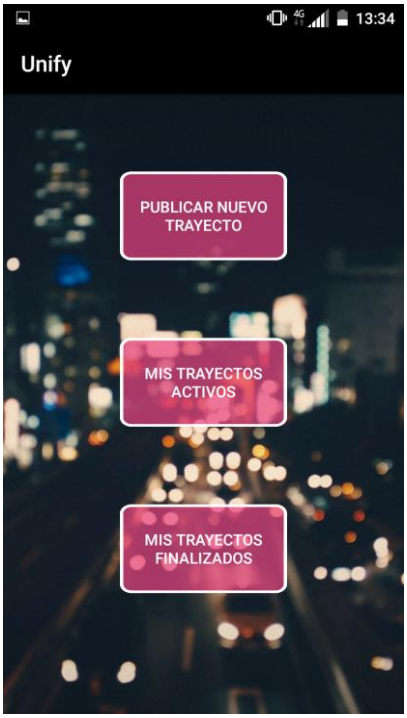

Ilustración 26: Menú conductor

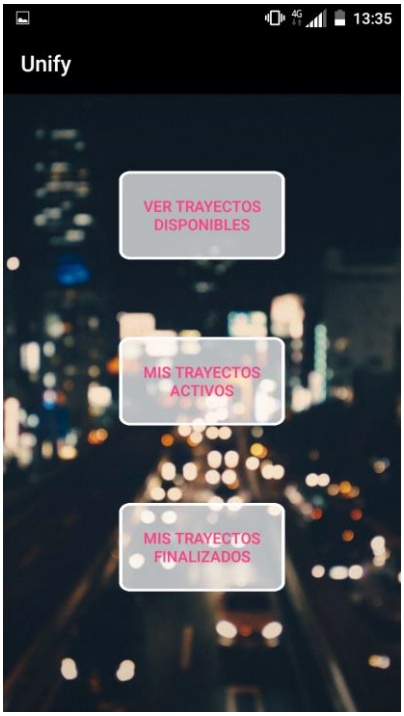

Ilustración 27: Menú pasajero

PUBLICAR TRAYECTO (CONDUCTOR)

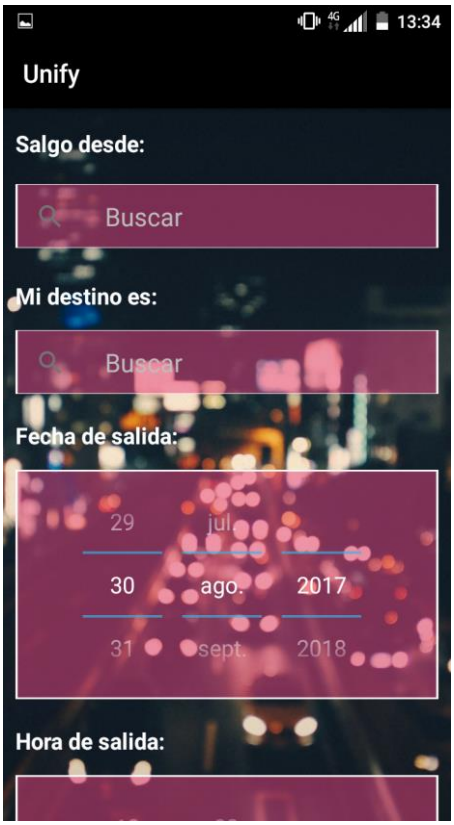

Ilustración 28: Publicar trayecto (1)

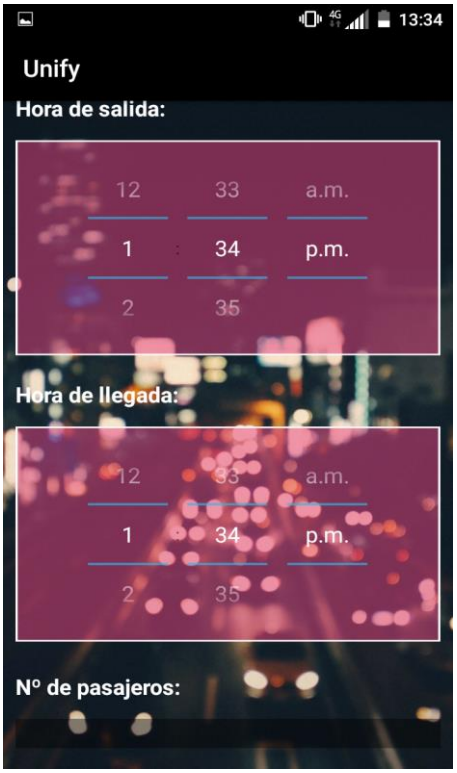

Ilustración 29: Publicar trayecto (2)

Ilustración 30: Publicar trayecto (3)

LISTADOS DE TRAYECTOS Y DETALLES DEL TRAYECTO

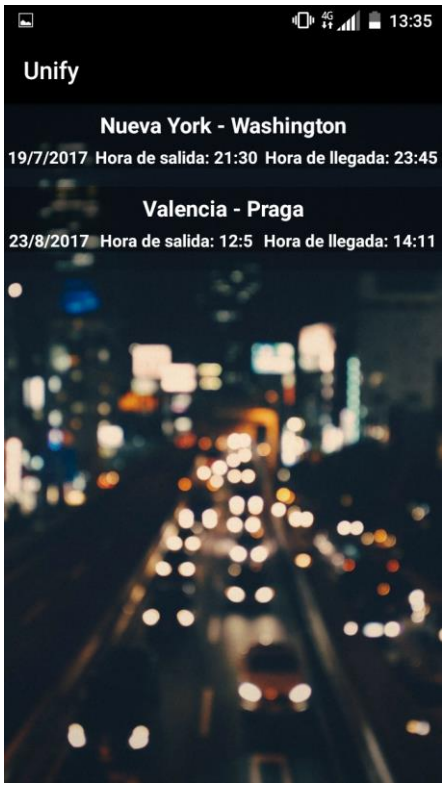

Ilustración 31: Listado trayectos

Ilustración 32: Detalle trayecto activo pasajero

5.3. Implementación del servidor

A continuación, mostraremos algunas capturas del código PHP, que se encarga de conectar la aplicación con la base de datos.

- En primer lugar, mostramos el código que realiza el servidor para hacer Login. La aplicación realiza una petición HTTP de tipo POST, adjuntando los valores del email y la contraseña introducidos por el usuario en la aplicación, y realiza una consulta en la base de datos. En función del resultado que este script devuelve, la aplicación dará por bueno el inicio de sesión, o no.

```php
    if (!empty($_POST)) {

        try {
            $stmt = $conexion->prepare("SELECT email,password FROM Usuarios WHERE email=? AND password=?");
            $stmt->bind_param("ss", $email, $password);

            $email = $_POST["email"];
            $password = $_POST["password"];
            $stmt->execute();
        }
        catch (PDOException $ex) {

            $response["success"] = 0;
            $response["message"] = "Problema con la base de datos, vuelve a intentarlo";
            die(json_encode($response));

        }

        $row = $stmt->fetch();
        if ($row) {

            $response["success"] = 1;
            $response["message"] = "Login correcto!";
            die(json_encode($response));
        }


            $response["success"] = 0;
            $response["message"] = "Login INCORRECTO";
            die(json_encode($response));

    }

?>
```

Ilustración 33: Código PHP Login

- En segundo lugar, mostraremos el código que realiza el servidor en la creación de un nuevo usuario. Lo primero que hace es comprobar que el email introducido no se está utilizando ya en otra cuenta existente, ya que se restringe una única cuenta por cada correo electrónico. Después, recoge los datos que la aplicación adjunta con la petición HTTP de tipo POST, y los inserta en una nueva fila de la tabla Usuarios. A partir de ahora, el nuevo usuario ya podrá acceder al resto del contenido de la aplicación.

```php
    if (!empty($_POST)) {

        try {

            $comprobacion = $conexion->prepare("SELECT * FROM Usuarios WHERE email = ?");
            $comprobacion->bind_param("s", $emailComprobacion);
            $emailComprobacion = $_POST["email"];
            $comprobacion->execute();
            $row = $comprobacion->fetch();
            if($row){
                $response["success"] = 0;
                $response["message"] = "El email ya existe";
                die(json_encode($response));
            }
            else{
                $stmt = $conexion->prepare("INSERT INTO Usuarios (name,surname,age,email,password,phone) VALUES
                (?,?,?,?,?,?)");
            $stmt->bind_param("ssssss", $name, $surname, $age, $email, $password, $phone);

            $name = $_POST["name"];
            $surname = $_POST["surname"];
            $age = $_POST["age"];
            $email = $_POST["email"];
            $password = $_POST["password"];
            $phone = $_POST["phone"];

            $stmt->execute();
            }
        }
        catch (PDOException $ex) {

            $response["success"] = 0;
            $response["message"] = "Problema con la base de datos, vuelve a intentarlo";
            die(json_encode($response));

        }
```

Ilustración 34: Codigo PHP registro

- Por último, mostraremos el script que gestiona las peticiones de la aplicación cuando un conductor desea consultar sus trayectos activos. Este código resulta de interés porque es el primero que devuelve, junto a su respuesta, información que debe mostrarse en la aplicación.

```php

if (mysqli_num_rows($result) > 0) {
        // looping through all results
        // products node
        $response["Trayectos"] = array();

        while ($row = mysqli_fetch_array($result)) {
            // temp user array
            $product = array();
            $product["origen"] = $row["origen"];
            $product["destino"] = $row["destino"];
             $product["fechaSalida"] = $row["fechaSalida"];
            $product["horaSalida"] = $row["horaSalida"];
             $product["horaLlegada"] = $row["horaLlegada"];
             $product["numPasajeros"] = $row["numPasajeros"];
             $product["precioPorPasajero"] = $row["precioPorPasajero"];
            // push single product into final response array
            array_push($response["Trayectos"], $product);
        }
        // success
        $response["success"] = 1;

        // echoing JSON response
        echo json_encode($response);
    } else {
        // no products found
        $response["success"] = 0;
        $response["message"] = "No products found";

        // echo no users JSON
        echo json_encode($response);
    }
```

Ilustración 35: Código PHP detalle trayecto

5.4. Conclusiones

En este capítulo se ha mostrado de manera detallada el resultado final del proyecto tras la implementación, mediante la muestra de las capturas de pantallas de la aplicación funcionando, así como de algunos ejemplos del código que gestiona las peticiones de la aplicación cuando solicita información a la base de datos (o quiere introducir nueva información). También se ha detallado el software utilizado para la implementación del proyecto.

6. Proyecto escalable

Una vez realizado el proyecto, queda por ver su posible crecimiento en el futuro. El tipo de aplicación que este proyecto ha desarrollado permite un crecimiento en cuanto a funcionalidades que optimizarían la calidad del servicio ofrecido.

Estas optimizaciones podrían centrarse en diversas áreas, y en función del modelo de negocio que se deseara llevar a cabo, habría que valorar detenidamente el orden de inclusión de las funcionalidades dentro de la aplicación.

Antes de entrar en detalles concretos, es importante puntualizar que estas mejoras no se han llevado a cabo por dos sencillas razones: inversión temporal, e inversión económica. Para poder optar a hacer la competencia, en el mercado, a las aplicaciones que actualmente ofrecen estos servicios, habría que ofrecer un producto tremendamente complejo, ya que estas aplicaciones (Blablacar, Uber, Cabify…) son todas casos de éxito reconocidos en el mundo empresarial actual, y detrás de ellas reside un equipo amplio de personas altamente cualificadas, además de inversiones, en prácticamente todos los casos, multimillonarias.

Para poder llegar a estar al nivel de estas aplicaciones, y centrándonos únicamente en la parte tecnológica (no entraremos en detalles de modelos de negocio, marketing, diseño, etc.), habría que implementar una serie de funcionalidades dentro y fuera de la aplicación. A continuación, detallamos algunas de ellas:

1) Confirmación de registro vía email:

   Ahora mismo, la aplicación únicamente comprueba que el email introducido a la hora de crear la cuenta de un nuevo usuario no exista ya en la base de datos. En caso de existir, rechaza la creación del nuevo perfil, pero no se comprueba de ninguna manera que el email introducido realmente sea del usuario, o que exista. Para poder solucionarlo, simplemente deberíamos implementar una confirmación de registro vía email, como se hace en la mayoría de aplicaciones y páginas web.

   Esto se podría llevar a cabo de muchas maneras. Una de ellas podría ser tener los usuarios pendientes de confirmación en una tabla temporal, y para poder

activar el perfil, el servidor inicializaría una rutina escrita en PHP que enviaría un email a la cuenta indicada, con un link de confirmación, que a su vez apuntaría a otro script del servidor. Si éste fuese activado, y el emisor del correo fuese el email pendiente, entonces nuestro usuario pasaría de estar en la tabla temporal a estar ya en la de usuarios activados, y con la certeza de haberse registrado con un email existente y de su propiedad.

2) Valoración del trayecto por parte de los pasajeros:

   En la vista de detalle de trayecto, en los trayectos finalizados del pasajero, podría haber un campo de valoración del conductor de cero a cinco estrellas, para que dicho conductor vaya acumulando una puntuación media, y así los pasajeros saber más acerca de la persona con la que van a subirse al vehículo.
   Por supuesto todos los conductores podrían revisar la valoración que reciben en cada trayecto en particular, y la valoración media de todos sus trayectos.

   Esto se podría implementar añadiendo un nuevo campo en la vista de detalles de trayectos finalizados, pudiendo ser modificado únicamente por los pasajeros. Además, en la vista de trayectos finalizados del conductor, éste tendría en algún lugar de la pantalla la valoración media que sus pasajeros le han ido asignando a lo largo de sus trayectos. Habría que añadir un campo nuevo en la base de datos tanto para los trayectos (valoración del trayecto) como para los usuarios (valoración media total).

3) Pago a través de la aplicación:

   En el proyecto que hemos desarrollado, está pensado que los pasajeros paguen en mano al conductor cuando hagan el trayecto (antes o después, según acuerden los usuarios). Una mejora notable, que además significaría la posible entrada de ingresos para los propietarios de la aplicación, sería establecer una pasarela de pago a través de la cual los pasajeros que quieran inscribirse a un trayecto deban pasar para poder confirmar la inscripción. El conductor recibiría sus ingresos también a través de la aplicación, pudiendo ser posible que la propia empresa se quedara con una comisión previamente establecida en el modelo de negocio.

Para llevar esta implementación a cabo, habría que introducir una pasarela de pago dentro de la aplicación (TPV virtual). Esto se podría llevar a cabo utilizando cualquier API proporcionada por las entidades financieras para cubrir estos fines. Estas pasarelas de pago ofrecidas por los bancos ya incluyen todos los aspectos de seguridad necesarios para garantizar la seguridad de los datos y todas las normas incluidas en la LOPD (Ley Orgánica de Protección de Datos).

4) Comunicación entre conductor y pasajeros vía chat interno:

En el proyecto que hemos desarrollado, está pensado que los pasajeros se comuniquen con el conductor vía teléfono móvil, facilitando el teléfono del conductor cuando un pasajero se inscribe a alguno de sus trayectos.
Una mejora muy interesante sería que no hiciera falta facilitar este tipo de datos de contacto, ya que la comunicación se puede llevar a cabo a través de la propia aplicación, generando un chat privado entre el conductor y los pasajeros que se inscriban a su trayecto, para cada uno de los trayectos.

Esto se podría llevar a cabo a través de un sistema de sockets e hilos en Java, ya que Android utiliza este lenguaje para el desarrollo de aplicaciones.

5) Sistema de denuncias de incidencias:

También se podría implementar dentro de la aplicación un sistema que permita a los usuarios (tanto pasajeros como conductores), notificar de algún tipo de incidencia (ej.: persona peligrosa, agresiva, maleducada, algún pasajero se niega a pagar, el conductor es un peligro público y no debería poder publicar trayectos, un conductor publica un trayecto y después no aparece, o ídem con un pasajero…).
Este tipo de servicios mejoraría la atención al cliente, y aseguraría una mayor calidad en el producto, atendiendo a los usuarios que no se han sentido cómodos con cualquier aspecto del servicio, y permitiendo a la empresa dueña de la aplicación, recibir este tipo de notificaciones para aplicar las medidas que considerasen oportunas.

Para implementar esta funcionalidad, se podría añadir una vista que permita enviar un correo al email de soporte de la empresa desde dentro de la aplicación. Un aspecto muy positivo de este tipo de funcionalidades podría ser que el propio usuario recibiera una llamada telefónica de atención al cliente de la empresa, para comprobar la incidencia y asegurarse de la veracidad de la denuncia.

6) Incorporación de usuarios a mitad trayecto

Una funcionalidad que también resultaría muy útil para los usuarios sería poder incorporarse a un trayecto que tiene como destino un destino que le interesa, pero que no salga desde un punto de partida común.

Esto se podría llevar a cabo si todos los trayectos incluyeran una visualización en Google Maps del recorrido del trayecto, para que así, los pasajeros que quieren compartir destino, pero están a mitad camino del trayecto, pudieran incorporarse en algún punto acordado con el conductor.

Implementando esta visualización con la API proporcionada por Google para sus mapas, podríamos añadir esta funcionalidad.

7) Integración con redes sociales

El uso de las redes sociales para compartir la publicación de trayectos facilitaría que los conductores encontraran pasajeros más rápidamente, además de la publicidad que se genera indirectamente con este tipo de integración.

Para llevar a cabo esto, habría que disponer de botones capaces de facilitar al usuario compartir los datos del trayecto publicado (si son conductores) o al que se han inscrito (si son pasajeros), mediante la utilización de las API que facilitan las principales redes sociales para este tipo de proyectos.

Como puede observarse, con suficiente capital para contratar personal, y con una planificación temporal bien estructurada, se podrían añadir cientos de funcionalidades a una aplicación como la que hemos desarrollado en este proyecto. Por tanto, podemos decir sin lugar a dudas que el proyecto elegido tiene capacidad para crecer, y que con una inversión fuerte podría llegar a crecer lo suficiente como para llegar a ser un posible competidor de la oferta actual del mercado de aplicaciones de este sector.

7. Conclusiones

Este último capítulo del documento tiene por objeto presentar el trabajo realizado a lo largo de todo el proyecto, los problemas encontrados durante el camino y las soluciones utilizadas para seguir adelante.

7.1. Trabajo realizado

El primer paso realizado para el desarrollo de este proyecto ha sido la elección del mismo, encontrando una oportunidad real de negocio, y tratando de cubrir un vacío en la oferta que actualmente existe en el mercado de aplicaciones.

El siguiente paso ha sido la planificación del mismo, definiendo los objetivos que se iban a llevar a cabo, así como decidir hasta dónde iba a llegar este proyecto, ya que, como hemos expuesto en el capítulo anterior, podría haberse extendido cuanto se hubiese querido.

A partir de aquí se empezó a construir la aplicación, comenzando por entender cómo crear una base de datos remota, y lo más importante, cómo conectarla con nuestra aplicación. El contacto con Android Studio existía, y ya se había desarrollado alguna aplicación que requería de almacenamiento local, pero nunca en remoto. Cómo crear usuarios, iniciar sesión, publicar trayectos, consultar los activos, inscribir a un usuario en un trayecto publicado... Una vez implementadas todas estas funcionalidades se pudo dar por completada la implementación de la aplicación (servidor y base de datos incluidas).

7.2. Problemas y soluciones

Los principales problemas han venido con la necesidad de establecer conexión con un servidor remoto, y que a su vez conectara con una base de datos para insertar o extraer información de ella. Se añade a esta dificultad el hecho de que en la carrera no se imparte ninguna asignatura que introduzca al lenguaje PHP, por lo que al principio los primeros intentos de conexión no se llevaron a cabo con éxito, y conseguir establecer

estas conexiones fue algo muy costoso de conseguir, básicamente por la barrera del lenguaje, y del funcionamiento del backend en general.

La solución fue, a grandes rasgos, el uso de la herramienta de depuración de Android Studio. A través de puntos de parada se iba revisando que los valores en cada momento fueran correctos, y se pudo detectar los lugares donde algo fallaba. Poco a poco, corrigiendo los fallos que se presentaban en cada punto, se consiguió crear una versión de la aplicación estable y operativa.

8. Bibliografía

- https://developer.android.com
- https://www.php.net/manual
- http://www.json.org/json-es.html
- *Gironés, Jesús Tomás. El gran libro de Android. S.A. Marcombo, 2016.*