ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA INFORMÁTICA

APLICACIÓN LOYMERBUS

LOYMERBUS APPLICATION

Realizado por
Samuel Hindley López
Tutorizado por
Eduardo Guzmán De los Riscos
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, septiembre 2016

Fecha defensa:
El Secretario del Tribunal

Resumen: Hoy en día gracias a la tecnología móvil y el Internet podemos consultar todo tipo de información y realizar todo tipo de transacciones en cualquier momento. El transporte público se ha acabado uniendo a esta moda con páginas web y aplicaciones móviles donde consultar horarios, precios, etc. Loymerbus, una pequeña empresa de Málaga, con una flota de autobuses limitada se ha quedado atrás en este sentido. Con cientos de clientes todos los días de la Axarquía, siguen sin forma de ver cualquier tipo de información sobre las rutas de estos autobuses. Este proyecto tiene esto en cuenta y trata de solucionar este problema con el desarrollo de una aplicación móvil híbrida que puede ser ejecutada en Android, iOS, Windows Phone y web. Su objetivo es ofrecer a los usuarios una aplicación simple y fácil de usar donde pueden consultar, en todo momento, los horarios y los precios insertando la salida y el destino, un mapa y una imagen de las rutas que siguen los autobuses, e incluso el tiempo que se tendría que esperar si hubiera algún autobús en ruta hacia la parada en la que te encuentres. Para esto último es necesario no solo una aplicación para el usuario, sino también una aplicación secundaria que iría en el móvil del conductor del autobús para poder acceder a su GPS y tener en cuenta su posición.

Palabras claves: móvil, aplicación, Android, iOS, Windows, web, Ionic, AngularJS, horarios, geolocalización, autobús.

Abstract: Nowadays, thanks to mobile technology and the Internet we can get all kinds of information and make all kinds of transactions at any time. Public transport has ended up joining this trend with websites and mobile applications where you can check out timetables, prices, etc. Loymerbus, a small company in Málaga, with a limited fleet of buses has fallen behind in this sence. With hundreds of clients everyday in the Axarquía, they still have no way of seeing any kind of information about the routes of these buses. This project bears this in mind and tries to solve this problem with the development of a hybrid mobile application that can run on Android, iOS, Windows Phone and web. It's objective is to offer users a simple and easy to use application where you can get timetables and prices at any time inserting the departure and destination, a map and an image of the routes the buses follow, and even the amount of time that you would have to wait if there were a bus on route to the bus stop you are at. For this last element, it is necessary for not only one application for the user, but another secondary application which would be on the bus drivers mobile to access the GPS and take into account its position.

Keywords: mobile, application, Android, iOS, Windows, Ionic, AngularJS, timetables, geolocation, bus.

Índice general

Capítulo 1. Introducción

1.1 Motivación

Las tecnologías hoy en día avanzan a un ritmo increíble e incluso difícil de seguir. La aparición del Internet ha revolucionado las comunicaciones a tal punto que actualmente es el medio preferido de comunicación para nuestro día a día. En casi todo lo que hacemos usamos el Internet: hablar con un amigo, pedir comida, comprar ropa, compartir fotos, … Antes del uso extendido del Internet, si querías estar al tanto de las ultimas noticias tenías que acercarte a por un periódico, pero hoy, un par de clics es suficiente para leer cualquier tipo de noticia de fuentes variadas.

En el año 1990 con la aparición del primer cliente y servidor web, las capacidades del Internet comenzaron a notarse en los ámbitos informáticos de universidades y centros de investigación, y más tarde, en entidades públicas, instituciones o empresas privadas de todo el mundo.

En la figura 1.1 podemos ver la cantidad de usuarios en Internet hoy día comparado con hace unos 10-15 años para tener una mejor idea del crecimiento e importancia del que estamos hablando:

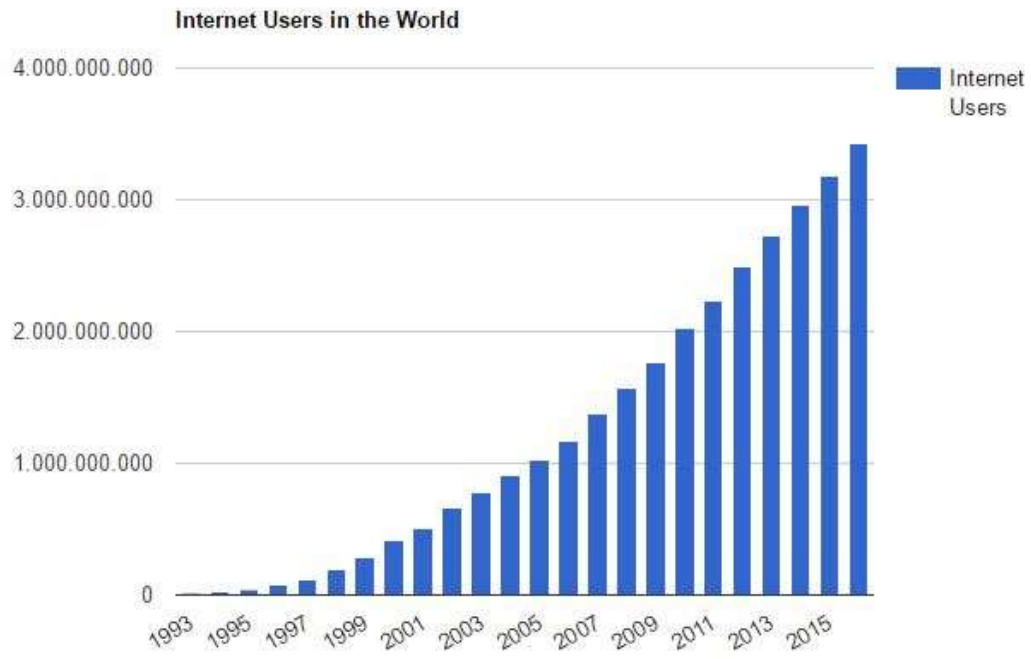

Fig. 1.1: Usuarios de Internet en el mundo (Fuente: http://www.internetlivestats.com/internet-users/)

Hoy día ya no es una herramienta de simple intercambio de información, sino una herramienta sofisticada para crear contenido, comunicar con otros y hasta escapar de la realidad. Podemos enviar datos de una parte del mundo a otra en cuestión de segundos, y ya no solo desde nuestro ordenador de escritorio o portátil, sino desde la palma de nuestra mano con el uso de un móvil.

El año 2014 fue un año importante para las telecomunicaciones ya que el uso del Internet en el móvil sobrepasó el uso en los ordenadores de sobremesa. Con redes como el 3G/4G se alcanzan velocidades similares o hasta superiores a las encontradas en las casas. Esto, junto a la comodidad de acceder fácilmente a la información del mundo exterior desde un dispositivo que llevamos siempre en nuestros bolsillos, hizo que la gran mayoría de la población mundial se pasara a esta tendencia.

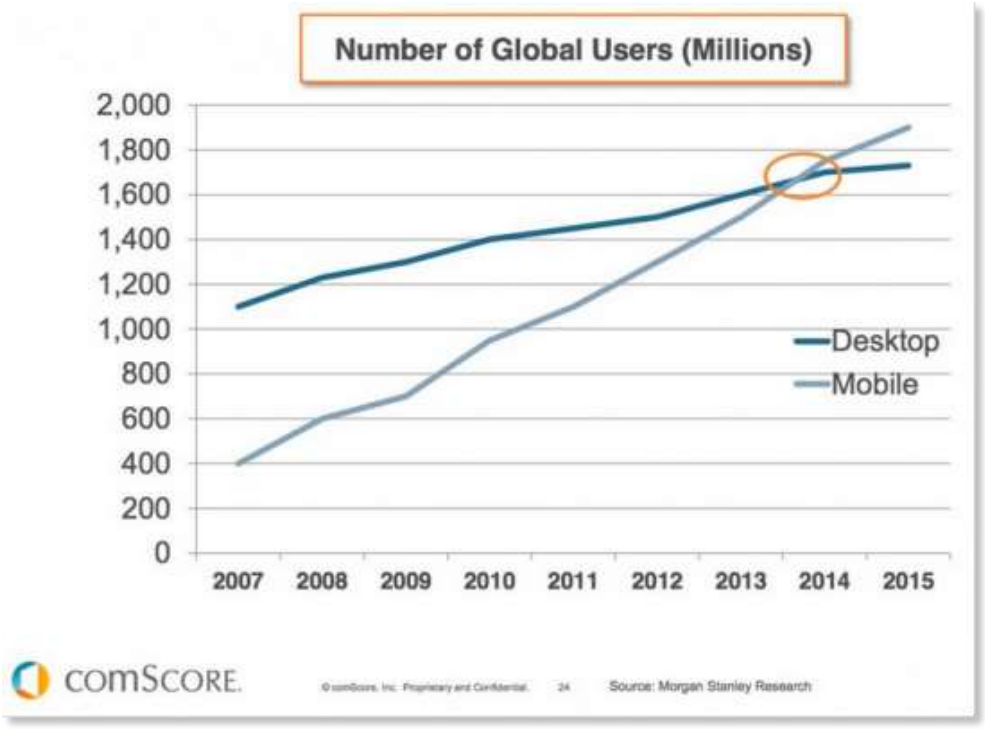

Fig. 1.2: Comparativa entre usuarios móviles y de escritorio (Fuente: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/)

En España las tendencias no son diferentes, empezando con las grandes ciudades como Madrid, Barcelona, Valencia, Málaga, etc. y habiéndose extendido ya hasta los pueblos más pequeños con conexiones básicas a ADSL y cobertura 3G, e incluso 4G, para cuando no estamos en casa.

Como hemos visto antes, la cantidad de cosas que se pueden hacer en esta red mundial es enorme, siendo de lo más básico el proporcionar información útil y relevante a los usuarios. Un claro ejemplo son la cantidad de negocios o comercios, donde es fundamental poner a disposición de los usuarios la información básica de contacto o localización por ejemplo, para asegurar que potenciales clientes puedan encontrarte y hacer uso de sus servicios de manera fácil, donde y cuando quieran. Cualquiera que no aprovecha el Internet móvil está perdiendo grandes oportunidades.

La industria del transporte público (autobuses, taxis, metros, etc.) es consciente del potencial del Internet ofreciendo, en el caso de los autobuses y metros, como mínimo, un sistema donde poder buscar horarios y precios, y en muchos casos, hasta la posibilidad de comprar los billetes con antelación a través de la red.

Los autobuses siguen siendo, desde hace tiempo, uno de los medios de transporte preferido por el público hoy en día, incluso en la Axarquía de la provincia de Málaga. Para moverse entre algunos de sus pueblos y poder llegar hasta la costa o hasta el centro de Málaga, existe una empresa llamada Grupo Loymer que ofrecen varios servicios para turistas tales como alquiler de apartamentos o alquiler de coches, entre ellos la línea de autobuses, Loymerbus, para los habitantes de localidades como Canillas de Albaida, Sayalonga, Algarrobo, Torre del Mar, …

Al vivir en Algarrobo y haber usado con frecuencia estos autobuses, he visto en primera persona a personas equivocarse de hora o parada, y no saber qué autobuses hay por el simple hecho de que esta información no estaba publicada en ningún sitio. Ni los precios ni los horarios están publicados en Internet, teniendo que preguntar en la oficina central o al conductor para sacar esta información, lo que implica un retraso total en cuanto al uso de las últimas tecnologías, siendo totalmente necesario el uso de Internet para poner a disposición de los viajeros los datos mínimos que necesitan.

1.2 Objetivos

A raíz de estos problemas surge la necesidad de una simple aplicación móvil con el objetivo de aportar a los ciudadanos de la Axarquía una forma rápida y sencilla de consultar la hora de llegada y salida de cada autobús, los precios, la localización exacta de cada parada, e incluso el tiempo de llegada estimada de un autobús a una determinada parada para saber el tiempo que tendrá que esperar. Aparte de estas funcionalidades la disponibilidad de la aplicación en varios idiomas, es muy importante debido a la cantidad de extranjeros en Málaga.

Por estas razones se ha desarrollado y documentado para este Trabajo de Fin de Grado una aplicación llamada Loymerbus, pensada para los viajeros donde podrán realizar búsquedas de precios y horarios, y consultar toda la información que se ha comentado antes. Para dotar de este proyecto de aún más funcionalidad, se ha desarrollado otra aplicación móvil secundaria llamada Loymerbus GPS que irá instalada en el móvil del conductor para acceder a la ubicación y llevar un seguimiento

del autobús. De este modo el viajero podrá solicitar también el tiempo que tendrá que esperar hasta la llegada del autobús a la parada en la que se encuentre.

Para cumplir estos objetivos y funcionalidades se ha hecho uso de Ionic, un framework muy popular a día de hoy para crear aplicaciones móviles híbridas haciendo uso de tecnologías web. Se basa en las librerías AngularJS y Apache Cordova para poder exportar nuestra aplicación a las plataformas más usadas en móviles: Android, iOS y Windows Phone.

Para guardar toda esta información necesaria de precios y horarios se ha usado la base de datos y el hosting de Firebase. Encajaba bien ya que permite ahorrar mucho tiempo a desarrolladores aportando un BaaS (Back-end as a Service) para poder centrarnos en el desarrollo de la aplicación en sí y no tener que desarrollar tu propia API con Node.js por ejemplo. Ofrece una gran cantidad de servicios gratuitos como base de datos NoSQL en tiempo real, autenticación, analíticas, hosting, etc.

1.3 Materiales y tecnología usada

- Hardware:
    - Portátil MSI GS70 con sistema operativo Windows 10 Home
    - Teléfono móvil LG G3 con Android 6.0 Marshmallow
    - Teléfono móvil Sony Xperia Z2 con Android 5.1.1 Lollipop
    - Tablet Nexus 9 con Android 7.0 Nougat
- Software:
    - Editor de texto Microsoft Visual Studio Code
    - Navegador web Google Chrome
    - Administrador de paquetes NPM
    - Android Studio
    - Android SDK
    - Microsoft Word 2016
    - Adobe Photoshop CC 2015

1.4 Contenido de la memoria

Este trabajo fin de grado se ha divido en varios capítulos tratando los temas más importantes de cada fase de desarrollo. Se tratan de los siguientes:

- **Capítulo 1. Introducción**: En esta sección se introduce el proyecto a realizar indicando los motivos por los que surge la necesidad de la aplicación, los objetivos del mismo, los materiales utilizados para conseguirlo y un breve resumen del contenido de esta memoria.
- **Capítulo 2. Conocimientos precios**: Aquí tratamos de introducir al lector a las tecnologías usadas para llevar a cabo el desarrollo con definiciones de cada una y algunas nociones básicas de ellas.
- **Capítulo 3. Especificación de requisitos**: En este capítulo detallaremos las funciones que debe cumplir la aplicación con un análisis de requisitos (funcionales y no funcionales).
- **Capítulo 4. Análisis y diseño**: Entramos en la primera fase antes de comenzar el desarrollo comentando los casos de uso que surgen tras el análisis de requisitos seguido de la defínción de la arquitectura de la aplicación y de la base de datos, y establecer una estructura de la misma.
- **Capítulo 5. Desarrollo**: Se trata de la sección más larga de esta memoria ya que explica todo el desarrollo que se ha tenido que llevar a cabo para la aplicación Loymerbus y su aplicación acompañante, Loymerbus GPS. Se comienza hablando de la implementación de la base de datos y la creación y configuración del proyecto, y a partir de aquí explicaremos con detalle cada sección de la aplicación comentando primero los elementos visuales; tras esto explicar con profundidad cómo funciona cada elemento y qué hacen.
- **Capítulo 6. Conclusiones y trabajo futuro**: Por último, comentaremos las conclusiones que se pueden extraer de todo el proceso y discutiremos posibles trabajos futuros que se podrían realizar.
- **Bibliografía**: Material bibliográfico que se ha usado para el desarrollo de la aplicación y la realización de esta memoria.

Capítulo 2. Conocimientos previos

2.1 Ionic 2

Ionic es un SDK open-source para el desarrollo de aplicaciones móviles híbridas. Construido con AngularJS y Apache Cordova, Ionic proporciona herramientas y servicios para desarrollar aplicaciones móviles híbridas usando tecnologías web como CSS, HTML5 y Sass. Las aplicaciones pueden construirse con estas tecnologías y luego ser distribuidas a través de las tiendas nativas para ser instalado en dispositivos aprovechando Cordova.

La primera versión fue creada por Max Lynch, Ben Sperry, y Adam Bradley de Drifty Co. en 2013 y actualmente se encuentra en la versión 2.

La versión 2 está basado en Angular 2 que consistió en un remodelado completo de AngularJS donde todas los conceptos e ideas de Angular siguen, pero hay cambios de estructuración y sintaxis muy grandes que hacen que parezca una librería totalmente diferente a la original. De aquí, y de las renovadas versiones de iOS y Android, y la nueva tendencia del uso de TypeScript, surgió también la necesaria remodelación y renovación de Ionic en forma de Ionic 2.

2.2 Angular 2

AngularJS (o Angular) es un marco de desarrollo de JavaScript de código abierto, mantenido por Google, que se utiliza para crear interfaces de usuario de aplicaciones web cuyo objetivo es simplificar el mantenimiento, el desarrollo, las pruebas y el uso de las aplicaciones. Contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo.

Sigue el patrón MVC (Modelo Vista Controlador), de este modo permite disociar el lado del cliente (frontend) y el lado del servidor (backend) habilitando así la posibilidad de avanzar en el desarrollo de forma paralela.

Veamos dos conceptos básicos de Angular que son los más usados para aplicaciones Ionic y que nos permitirán entender mejor muchas partes del código que veremos a lo largo de la memoria.

- **Controladores**: Son los archivos con formato .ts (TypeScript) y son los encargados de inicializar y modificar la información que contiene nuestra aplicación en todo momento, es decir, la lógica que hay detrás de ella.
- **Enlace de datos bidireccional**: Esto se usa mucho ya que nos permite tener una sincronización de datos entre el modelo y la vista de manera automática. De este modo, cuando el modelo cambia, la vista refleja ese cambio automáticamente y viceversa. Esto nos permite cambiar cualquier dato fácilmente en el controlador o ser notificados fácilmente de algún cambio que se ha producido en la vista.

2.3 TypeScript

TypeScript es un lenguaje de programación gratuito y libre creado por el también creador del lenguaje C#, Anders Heilsberg, y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente se basa en el estándar ECMAScript 6 añadiendo tipado estático y objetos basados en clases.

TypeScript = ES6 + Types + Annotations

Incluye varios aspectos de la programación orientada a objetos como la herencia, interfaces, o incluso funciones lambdas.

A través de un compilador permite traducir el código TypeScript a JavaScript original para que lo entiendan los navegadores web.

Los dos conceptos más usados en los controladores de nuestra aplicación son los siguientes:

- **Tipos**: El tipado en TypeScript nos permite mantener un registro de la intención con la que se crea una variable o función, por ejemplo, si creamos una variable de tipo *string* y se le intenta asignar un *array* el compilador de TypeScript propagará un error. No es necesario asignar el tipo de una variable (como ocurre en JavaScript) pero puede ser útil para asegurarnos de que los tipos de los objetos son los esperados y son correctos.
- **Clases**: Esto nos permite definir clases de un modo similar en la programación orientada a objetos, en JavaScript podemos simular el comportamiento de clases, pero con TypeScript nos lo ponen muy fácil sobre todo para desarrolladores acostumbrados a trabajar con “objetos”.

2.4 Firebase

Fundado en 2011 y comprado por Google en 2014, Firebase es una plataforma de desarrollo móvil en la nube. Es lo que se conoce como “Backend como Servicio”, que básicamente provee de una API para guardar y sincronizar datos en la nube en tiempo real. Entre sus características están:

- Muy fácil implantación.
- Sincronización instantánea. Cuando cambia el valor de una variable, se actualiza en todos los dispositivos automáticamente.
- Provee de un sistema para trabajar offline, lanzando una sincronización cuando se retoma la conexión con el servidor.
- Los datos de guardan en un JSON estándar, por ello es 100% multiplataforma mediante API REST.
- Hosting y dominio personalizado gratis.

Este año han ampliado mucho las funcionalidades de este producto creando así un gran repertorio de herramientas para desarrolladores incluyendo elementos como analíticas, notificaciones push, almacenamiento en la nube, etc.

2.5 Apache Cordova

Originalmente creado por Nitobi, fue comprado por Adobe Systems en 2011 y lo llamó PhoneGap. Más tarde sacaron una versión open-source con el nombre que lleva hoy día.

Es un framework de desarrollo para aplicaciones móviles muy popular por el hecho de que permite a desarrolladores construir aplicaciones usando CSS3, HTML5 y JavaScript en vez de tener que depender de plataformas como Android, iOS o Windows Phone.

Encapsula este código según la plataforma a la que se exporta y extiende el funcionamiento del HTML y del JavaScript para funcionar en el dispositivo. El resultado es una aplicación hibrida que no es realmente nativa ya que el renderizado de las vistas se realiza siempre con *web views* pero tampoco son completamente basadas en web porque podemos acceder a funciones nativas del dispositivo como la cámara, almacenamiento, GPS, etc.

Esto último es uno de los puntos más importantes para nuestra aplicación ya que sin la tecnología de Apache Cordova, que crea la base de Ionic, no podríamos acceder a las funciones del GPS del dispositivo móvil y este trabajo de fin de grado no hubiera sido posible.

2.6 npm (Node Package Manager)

Fue desarrollado por Isaac Z. Schlueter a raíz de la frustración con la administración de paquetes para Node.js. Está escrito a día de hoy enteramente en JavaScript y es instalado de forma automática con la instalación de Node.js. Nos permitirá instalar todos los paquetes de las tecnologías que hemos mencionado antes, mantenerlos actualizados y solucionar dependencias entre ellas, todo a través de la línea de comandos.

Capítulo 3. Especificación de requisitos

La especificación de requisitos de un sistema consiste en una colección de elementos en los que se describen los servicios que ha de ofrecer ese sistema y las restricciones asociadas a su funcionamiento. Esta descripción completa del comportamiento del sistema a desarrollar se suele clasificar en dos grupos:

- **Requisitos funcionales**: Son el tipo más común de requisitos. Identifican los servicios y acciones que el sistema debe proporcionar, normalmente en respuesta a una entrada externa.
- **Requisitos no funcionales**: Este tipo de requisitos nos dicen las restricciones en el diseño o la implementación, es decir, restricciones que afectaran al correcto funcionamiento del sistema.

A continuación vamos a comentar la recopilación de datos que se ha llevado a cabo antes de realizar nada, para ya sacar los requisitos en base a toda esta información.

3.1 Recopilación de datos

El primer paso en la ejecución de este Trabajo de Fin de Grado consistió en entrar en contacto con la empresa Loymerbus para pedirles permiso para usar su nombre en la documentación de este trabajo de fin de grado, ya que figuraba en el listado de marcas registradas en el Gobierno de España. A través de correo electrónico me cedieron esos permisos además de un documento con el horario de todos los trayectos y los precios. Tras varios intercambios de correos conseguí más información necesaria como el descuento que se le aplica a los que tienen la tarjeta de pensionista, los horarios de las rutas a Vélez-Málaga que no figuraban ni en los documentos que me enviaron y algunos logos de la empresa para poder seguir algunas pautas en cuanto a colores para el diseño de la aplicación.

Realicé alguna que otra visita a la oficina central en Torre del Mar para tener fotos de la oficina por fuera y conocer el horario de apertura. Para las localizaciones exactas de las paradas hice uso de Google Maps para encontrar las que ya conocía y el resto, preguntando a amigos de los pueblos que me faltaban.

3.2 Aplicación Loymerbus

3.2.1 Requisitos funcionales

- **RF01 - Mostrar un menú lateral**: La aplicación debe permitir abrir un menú lateral con todas las secciones de la aplicación para poder cambiar de una a otra con facilidad.
- **RF02 - Introducir datos para la consulta de horarios**: El usuario debe poder introducir todos los datos necesarios para buscar información relevante sobre los horarios de los autobuses.
    - **RF02.1 - Introducir parada de salida**: Podrá seleccionar la parada de autobús de salida de una lista de posibilidades.
    - **RF02.2 - Introducir parada de destino**: Podrá seleccionar una parada de destino de los posibles destinos según la salida seleccionada.
    - **RF02.4 - Introducir si es pensionista**: Podrá indicar si dispone o no de la tarjeta de pensionistas de la Junta de Andalucía para disfrutar de un descuento.
- **RF03 - Modificar datos para la consulta de horarios**: El usuario debe poder modificar todos los datos necesarios para buscar información relevante sobre los horarios de los autobuses.
    - **RF03.1 - Modificar parada de salida**: Podrá cambiar la parada de autobús de salida seleccionada de una lista de posibilidades.
    - **RF03.2 - Modificar parada de destino**: Podrá cambiar la parada de destino seleccionada de los posibles destinos según la salida seleccionada.
    - **RF03.3 - Modificar si es pensionista**: Podrá modificar si dispone o no de la tarjeta de pensionistas de la Junta de Andalucía para disfrutar de un descuento.
    - **RF03.4 - Modificar fecha**: Por defecto se muestra la fecha del día actual, pero se podrá modificar la fecha del año actual de un listado de meses y días del mes seleccionado.
- **RF04 - Obtener datos sobre los horarios**: La aplicación debe poder conectarse a la base de datos y obtener los datos de las horas y los precios según las paradas y le fecha que haya introducido el usuario.
- **RF05 - Mostrar información de los horarios**: Una vez introducidos los datos necesarios y pulsado el botón para buscar horarios, se deberán mostrar, en una ventana nueva, los resultados correspondientes cargados de la base de datos.
    - **RF05.1 - Horas de salida**: Se presentarán las horas de salida en una tabla de cada resultado de la búsqueda.
    - **RF05.2 - Horas de llegada**: Se presentarán las horas de llegada en una tabla de cada resultado de la búsqueda.

    - **RF05.3 - Precios**: Se presentarán los precios en una tabla de cada resultado de la búsqueda.
    - **RF05.4 - Error**: Si no hay viaje disponible para la fecha seleccionada se mostrará un texto explicando el error que ha ocurrido.
- **RF06 - Mostrar un mapa con las rutas**: La aplicación debe cargar un mapa de Google Maps donde mostrar información sobre las rutas que siguen los autobuses.
    - **RF05.1 - Localización de paradas**: Se mostrará un marcador en la localización exacta de cada parada de autobús junto con un título de la población a la que corresponde.
    - **RF05.2 - Rutas entre paradas**: Habrá líneas trazadas entre todas las paradas representando el camino que sigue cada autobús.
- **RF07 - Mostrar una imagen del esquema de las rutas**: Se podrá consultar una imagen de un tamaño y una resolución considerable de un esquema ilustrativo de las direcciones que siguen los autobuses y los cambios de autobús necesarios.
- **RF08 - Soporte multilenguaje**: El usuario tendrá la posibilidad de elegir entre varios idiomas (inglés, español, francés y alemán) para cambiar el idioma de la aplicación entera.
- **RF09 - Introducir parada para consultar tiempo de espera**: El usuario insertará la parada en la que se encuentra de un listado de posibles paradas de autobús.
- **RF10 - Calcular tiempo estimado de llegada**: Una vez especificado la parada la aplicación deberá calcular el tiempo que tardará el autobús en llegar a la parada seleccionada haciendo uso de la API de Google Maps, si y solo si se ha determinado que hay un autobús en ruta.
- **RF11 - Mostrar resultados del tiempo estimado de llegada**: Se le comunicará al usuario el resultado de la búsqueda y del cálculo a través de una ventana emergente.
    - **RF11.1 - Tiempo de espera**: Mostrará el tiempo que tardará en llegar el autobús a la parada de interés en minutos.
    - **RF11.2 - Error**: Si ha ocurrido algún error, los autobuses no están en servicio o simplemente se ha determinado que no hay ningún autobús en ruta, se le mostrará al usuario un texto diciendo que no hay ningún autobús en ruta hacia esa parada en ese momento y que consulte por favor el horario.

3.2.2 Requisitos no funcionales

- **RNF01 - Conexión a Internet**: La aplicación necesita conexión a Internet para el acceso a la base de datos. Sería proporcionada por el propio dispositivo (móvil, tablet, u ordenador).
- **RNF02 - Usabilidad**: Dado que la aplicación se puede acceder desde diferentes dispositivos con diferentes resoluciones y tamaños de pantalla, la interfaz debe ser adaptable permitiendo a cualquier usuario usar las funciones y disfrutar de la interfaz de usuario sin importar el dispositivo que use.
- **RNF03 - Tecnologías web**: Las tecnologías y lenguajes de programación deben ser HTML, CSS y JavaScript, es decir, propios de la web. Con esto podremos desarrollar una aplicación móvil hibrida que se pueda exportar a dispositivos con sistemas operativos Android, iOS o Windows Phone, incluso con el propio navegador web.

3.3 Aplicación Loymerbus GPS

3.3.1 Requisitos funcionales

- **RF01 - Indicar línea**: El conductor del autobús deberá poder indicar qué ruta seguirá.
- **RF02 - Activar funciones del GPS**: La aplicación Loymerbus GPS tendrá un botón para activar todas las funciones necesarias para la localización del autobús.
    - **RF02.1 - Mostrar activación del seguimiento por GPS**: Mostrará una ventana emergente indicando que la activación del seguimiento por GPS ha tenido éxito.
    - **RF02.2 - Activar funcionamiento en segundo plano**: Activa el funcionamiento en segundo plano para que se pueda seguir transmitiendo su posición, aunque no esté en primer plano la aplicación.
    - **RF02.3 - Inicializar la base de datos**: Se inicializarán los campos necesarios en la base de datos según el autobús que sea y según la parada desde la que partirá.
    - **RF02.4 - Iniciar la actualización de información**: Se activará una función que se ejecutará cada 10 minutos para actualizar la información en la base de datos de forma periódica.
    - **RF02.5 - Iniciar escucha de eventos**: Si un usuario solicita el tiempo de espera la aplicación deberá ser capaz de escuchar eventos emitidos por la aplicación principal para responder actualizando su ubicación actual.
- **RF03 - Desactivar funciones del GPS**: Habrá otro botón encargado de desactivar todas las funciones necesarias para el seguimiento del autobús.
    - **RF03.1 - Mostrar desactivación del seguimiento por GPS**: Mostrará una ventana emergente indicando que la desactivación del seguimiento por GPS ha tenido éxito.
    - **RF03.2 - Desactivar funcionamiento en segundo plano**: Dejará de permitir que la aplicación siga corriendo en segundo plano.
    - **RF03.3 - Desactivar temporizador para actualización de datos**: Borrará el temporizador de 10 minutos para dejar de actualizar la base de datos.
    - **RF03.4 - Cancelar escucha de eventos**: Desvinculará la aplicación de la base de datos para dejar de recibir emisiones de eventos por parte de la aplicación principal.

3.3.2 Requisitos no funcionales

- **RNF01 - Conexión a Internet**: La aplicación necesita conexión a Internet para el acceso a la base de datos. Sería proporcionada por el propio dispositivo (móvil, tableta, u ordenador).
- **RNF02 - Usabilidad**: Dado que la aplicación se puede acceder desde diferentes dispositivos con diferentes resoluciones y tamaños de pantalla, la interfaz debe ser adaptable permitiendo a cualquier usuario usar las funciones y disfrutar de la interfaz de usuario sin importar el dispositivo que use.
- **RNF03 - Tecnologías web**: Las tecnologías y lenguajes de programación deben ser HTML, CSS y JavaScript, es decir, propios de la web. Con esto podremos desarrollar una aplicación móvil hibrida que se pueda exportar a dispositivos con sistemas operativos Android, iOS o Windows Phone, incluso con el propio navegador web.
- **RFN04 - Conexión a GPS**: El dispositivo deberá permitir el acceso a la ubicación del dispositivo para poder obtener la localización del autobús en cualquier momento.

Capítulo 4. Análisis y diseño

4.1 Casos de uso

En esta sección se muestran los diferentes casos que derivan de los requisitos funcionales especificados en el capítulo anterior.

4.1.1 Aplicación Loymerbus

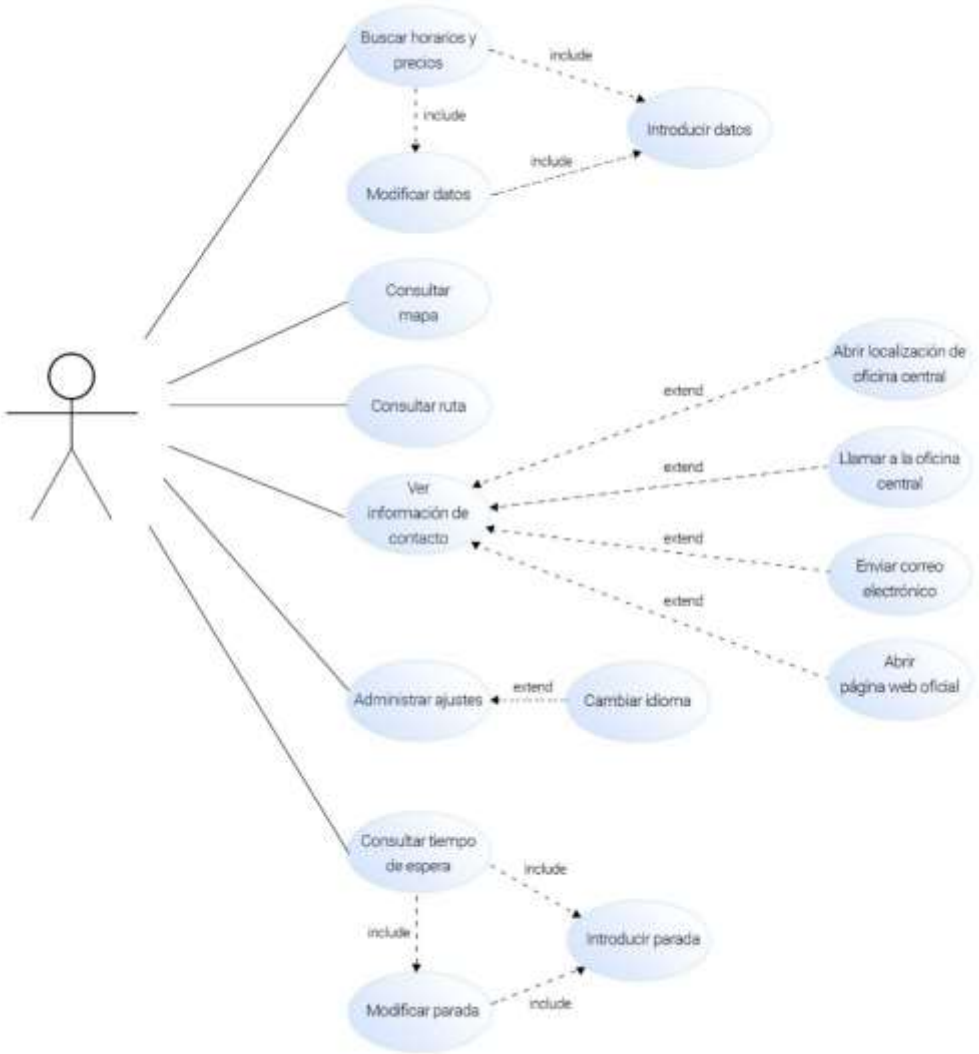

Fig. 4.1: Diagrama de casos de uso de la aplicación Loymerbus

- **CU01 - Buscar horarios y precios**:
    - Descripción: El usuario podrá buscar las horas y precios disponibles para llegar de una parada a otra en un determinado día.
    - Actores: Usuario
    - Precondición: Se deben seleccionar la salida, el destino y la fecha.
    - Postcondición: Se mostrará una nueva ventana con una tabla detallando las horas de salidas y de llegadas, así como los precios.
    - Excepciones:
        1. Si no hay ruta entre la salida y el destino para el día seleccionado se le notificará al usuario con un mensaje.
        2. Si no hay conexión a Internet al usuario no se le cargará ninguna parada de autobús para seleccionar.
    - Secuencia:
        1. Navegar a la sección “Buscar”.
        2. El usuario selecciona desde una lista la parada desde donde sale.
        3. El usuario selecciona de una lista la parada a la que quiere llegar.
        4. Se muestra por defecto la fecha de hoy, el usuario puede cambiarlo, si quiere.
        5. Se muestra por defecto que no es pensionista, el usuario puede indicar que sí lo es, si quiere.
        6. Pulsa el botón de “BUSCAR”.
        7. Los datos devueltos son mostrados en la ventana nueva.
- **CU02 - Modificar datos para la búsqueda de horarios y precios**:
    - Descripción: El usuario podrá modificar los datos introducidos en el CU01 previamente.
    - Actores: Usuario
    - Precondición: La salida, destino, fecha y opción de pensionista tienen que haber sido seleccionados antes.
    - Postcondición: Se mostrará una nueva ventana con una tabla detallando las horas de salidas y de llegadas, así como los precios, teniendo en cuenta los datos nuevos introducidos.
    - Excepciones: Si no hay ruta entre la salida y el destino para el día seleccionado se le notificará al usuario.
    - Secuencia:
        1. Navegamos hacia atrás hasta la sección “Buscar”.
        2. El usuario modifica la parada desde la que sale.
        3. El usuario modifica la parada a la que quiere llegar.
        4. El usuario puede modificar la fecha del viaje.
        5. El usuario puede modificar si es pensionista o no.
        6. Pulsa de nuevo el botón “BUSCAR”.
        7. Los datos devueltos son mostrados en la ventana nueva.

- **CU03 - Consultar mapa**:
    - Descripción: El usuario podrá consultar un mapa con la localización exacta de todas las paradas de autobús y ver las rutas entre cada una de ellas.
    - Actores: Usuario
    - Precondición: Ninguna
    - Postcondición: Un mapa es mostrado al usuario con marcadores en cada parada de autobús y líneas trazadas entre cada una de ellas para indicar las rutas.
    - Excepciones: Si no hay conexión a Internet no se mostrará el mapa.
    - Secuencia:
        1. Navegar a la sección “Mapa”.
        2. Explorar mapa y contenido de la misma.
- **CU04 - Consultar ruta**:
    - Descripción: El usuario podrá consultar un esquema ilustrativo de las paradas y rutas.
    - Actores: Usuarios
    - Precondición: Ninguna
    - Postcondición: Una imagen es mostrada con el esquema de las paradas y rutas donde poder hacer scroll en eje X e Y.
    - Excepciones: Ninguna
    - Secuencia:
        1. Navegar a la sección “Ruta”.
        2. Se muestra la imagen
- **CU05 - Ver información de contacto**:
    - Descripción: Se puede consultar toda la información necesaria de contacto de Loymerbus.
    - Actores: Usuario
    - Precondición: Ninguna
    - Postcondición: Se mostrará una tarjeta informativa con todos los datos de contacto de Loymerbus.
    - Excepciones: Ninguna
    - Secuencia:
        1. Navegar a la sección “Contacto”.
        2. Se muestra la tarjeta informativa con todos los datos necesarios.
- **CU06 - Abrir localización de la oficina central**:
    - Descripción: Pulsando en un botón se puede abrir la localización de la oficina central de Loymerbus en alguna aplicación de mapas.
    - Actores: Usuario, Aplicación de mapas/Navegador web
    - Precondición: Estar en la sección “Contacto”.
    - Postcondición: Se le abrirá la aplicación de mapas que use de forma predeterminada (normalmente Google Maps) con la localización de la

oficina central de Loymerbus marcada, o si no tiene ninguna instalada, se abrirá en la página web de Google Maps a través del navegador.
    - Excepciones: Si no tiene conexión a Internet no cargará la ubicación.
    - Secuencia:
        1. Navegar a la sección “Contacto”.
        2. Pulsar el botón rosa con el icono del marcador de un mapa.
        3. Se abre la ubicación en la aplicación de mapas que corresponda.
- **CU07 - Llamar a la oficina central**:
    - Descripción: El usuario puede llamar a la oficina central pulsando en el número de teléfono.
    - Actores: Usuario, Aplicación telefónica
    - Precondición: Estar en la sección “Contacto”.
    - Postcondición: Al usuario se le abrirá la aplicación predeterminada de llamadas con el número marcado para poder realizar la llamada.
    - Excepciones: Ninguna
    - Secuencia:
        1. Navegar a la sección “Contacto”.
        2. Pulsar en el número de teléfono.
        3. Abre la aplicación de llamadas del dispositivo.
- **CU08 - Enviar correo electrónico**:
    - Descripción: Podremos pulsar encima del correo electrónico que aparece para enviarles directamente un correo.
    - Actores: Usuario, Aplicación de correo electrónico
    - Precondición: Estar en la sección “Contacto”.
    - Postcondición: Si pulsamos en el correo se nos abrirá la aplicación que tengamos configurada para el correo electrónico con el campo de destinatario ya rellenado.
    - Excepciones: Si no tiene ninguna aplicación de correo electrónico configurada no abrirá ninguna aplicación.
    - Secuencia:
        1. Navegar a la sección “Contacto”.
        2. Pulsar sobre el correo electrónico.
        3. Se abre la aplicación de correo electrónico.
- **CU09 - Abrir página web oficial**:
    - Descripción: Pulsando en el enlace a la página web de Loymerbus se nos abrirá el enlace en un navegador web.
    - Actores: Usuario, Navegador web
    - Precondición: Estar en la sección “Contacto”.
    - Postcondición: Si pinchamos en el enlace se nos abrirá la URL.
    - Excepciones: Si no tenemos conexión a Internet no cargará la página web.
    - Secuencia:
        1. Navegar a la sección “Contacto”.

2. Pinchar en el enlace web.
3. Apertura de la página web en el navegador del dispositivo.

- **CU10 - Administrar ajustes**:
  - Descripción: El usuario puede revisar y modificar cualquier ajuste desde su correspondiente sección.
  - Actores: Usuario
  - Precondición: Ninguna
  - Postcondición: Aparecen los ajustes de la aplicación y como están configuradas las mismas.
  - Excepciones: Ninguna
  - Secuencia:
    1. Navegar a la sección “Ajustes”.
    2. Se muestra los ajustes configurables de la aplicación.
- **CU11 - Cambiar idioma**:
  - Descripción: Será posible cambiar el idioma de toda la aplicación desde la sección de “Ajustes”.
  - Actores: Usuario
  - Precondición: Ninguna
  - Postcondición: El idioma seleccionado cambiará automáticamente todos los textos de la aplicación al idioma correspondiente de forma instantánea.
  - Excepciones:
  - Secuencia:
    1. Navegar a la sección “Ajustes”
    2. Seleccionar el idioma deseado.
    3. El idioma de todos los textos de la aplicación cambiará al idioma seleccionado.
- **CU12 - Consultar tiempo de espera**:
  - Descripción: El usuario podrá seleccionar una parada de autobús para ver si hay algún autobús en ruta y si es así, cuánto le faltaría para llegar.
  - Actores: Usuario
  - Precondición: Se deben seleccionar la parada de interés.
  - Postcondición: Se mostrará una ventana emergente indicando el tiempo de llegada estimada del autobús en llegar a esa parada.
  - Excepciones:
    1. Si no hay ningún autobús en ruta se le notificará al usuario con un mensaje en la ventana emergente.
    2. Si no hay conexión a Internet al usuario no se le cargará ninguna parada de autobús para seleccionar.
  - Secuencia:
    1. Navegar a la sección “ETA”.
    2. El usuario selecciona una parada desde una lista.
    3. Pulsa el botón de “BUSCAR”.

4. Los datos devueltos son mostrados en la ventana emergente.

- **CU13 - Modificar datos para la consulta del tiempo de espera**:
  - Descripción: El usuario podrá modificar los datos introducidos en el CU12 previamente.
  - Actores: Usuario
  - Precondición: La parada de interés debe de haber sido ya seleccionada.
  - Postcondición: Se mostrará una nueva ventana emergente con los datos devueltos de la nueva parada introducida.
  - Excepciones: Si no hay ningún autobús en ruta se le notificará al usuario con un mensaje en la ventana emergente.
  - Secuencia:
    1. Volver a la parte principal de la vista “ETA” pulsando el botón “OK” de la ventana emergente.
    2. Seleccionar la nueva parada de la lista.
    3. Pulsa el botón “BUSCAR”.
    4. Los nuevos datos devueltos son mostrados en la ventana emergente.

4.1.2 Aplicación Loymerbus GPS

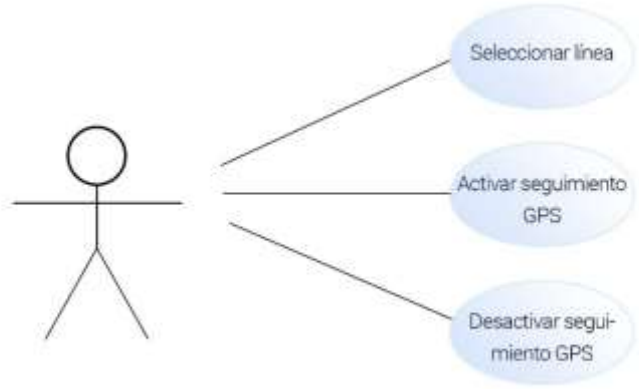

Fig. 4.2: Diagrama de casos de uso de la aplicación Loymerbus GPS

- **CU01 - Seleccionar línea del autobús**:
  - Descripción: El conductor del autobús podrá seleccionar una de dos opciones para indicar la ruta que seguirá.
  - Actores: Conductor
  - Precondición: Ninguna
  - Postcondición: Se guardará la opción seleccionada para el funcionamiento interno de la aplicación.

- Excepciones: Ninguna.
- Secuencia:
    1. Seleccionamos una de las dos opciones que se presentan en la única vista de la aplicación.
    2. Se sincroniza la opción guardada con el controlador.

- **CU02 - Activar seguimiento GPS**:
    - Descripción: El conductor del autobús debe pulsar el botón “ACTIVAR GPS” para activar todas las funciones necesarias internamente para el seguimiento de la ruta del autobús.
    - Actores: Conductor
    - Precondición: Tener el GPS del dispositivo activado.
    - Postcondición: Se activará el funcionamiento en segundo plano y se ejecutarán varias funciones internas.
    - Excepciones: Si no está activado el GPS se le notificará al usuario con una ventana emergente, ya que sin activar el GPS no se puede acceder a la ubicación del dispositivo.
    - Secuencia:
        1. Activamos el GPS del dispositivo.
        2. Pulsamos el botón “ACTIVAR GPS”.
        3. Activación del funcionamiento en segundo plano.
- **CU03 - Desactivar seguimiento GPS**:
    - Descripción: El conductor del autobús debe pulsar el botón “DESACTIVAR GPS” cuando haya terminado su ruta para detener las funciones internas que están en ejecución y desactivar su funcionamiento en segundo plano.
    - Actores: Conductor
    - Precondición: Tener el seguimiento por GPS ya activado.
    - Postcondición: Se desactivará el funcionamiento en segundo plano y se cancelarán las funciones internas que están en ejecución.
    - Excepciones: Ninguna
    - Secuencia:
        1. Pulsamos el botón “DESACTIVAR GPS”.
        2. Desactivación del funcionamiento en segundo plano.

4.2 Arquitectura

Podríamos distinguir dos tipos de arquitecturas en este proyecto: por un lado la arquitectura cliente-servidor en la que los clientes son las aplicaciones de los usuarios y la de los conductores, y el servidor hemos decidido que será Firebase. Por otro lado en cuanto a organización de clases y funciones, se ha seguido el patrón que establece Angular, MVC (Modelo Vista Controlador).

4.2.1 Arquitectura cliente-servidor

Es una estructura de software en el que aislamos al cliente del servidor y los conectamos a través del Internet. Esta conexión solo se realiza cuando el cliente envía una petición al servidor, en cuyo caso, ya sea la aplicación Loymerbus o Loymerbus GPS, el servidor responde con lo que se haya solicitado, pero nunca podrá iniciar la conexión el servidor.

Por lo tanto, nuestro servidor Firebase nos suministra un servicio: darnos la información de la base de datos, para que cuando todos los clientes que estén en ejecución así lo soliciten, siempre tengamos a nuestro servidor respondiendo a todas las peticiones.

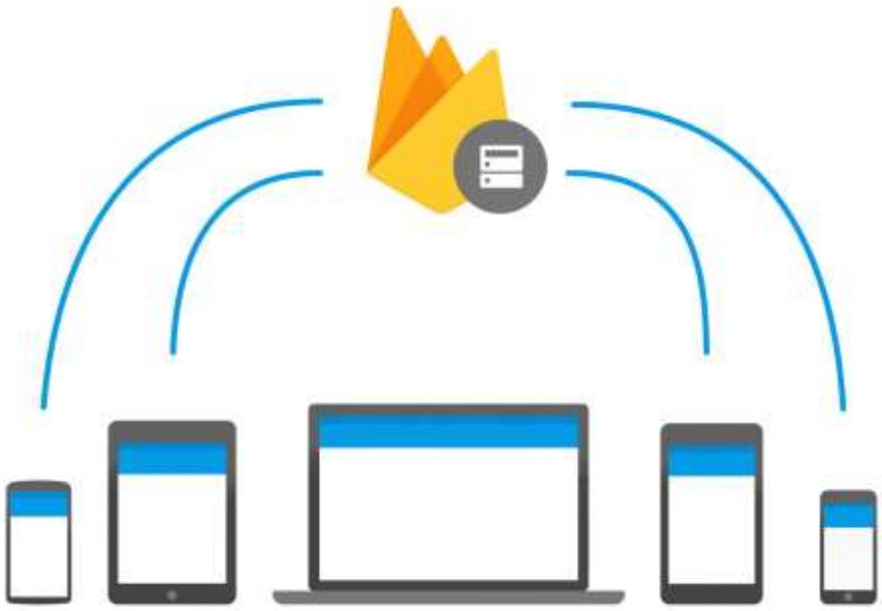

Fig. 4.3: Illustración de la base de datos en tiempo real de Firebase

4.2.2 Arquitectura Modelo Vista Controlador

Al estar usando Angular, seguimos el conocido patrón Modelo Vista Controlador. Un patrón muy conocido en cuanto a la arquitectura de software donde define tres partes interconectadas para separar el funcionamiento interno del software y aislarla de la información a la que se accede en la base de datos y de lo que es presentado al usuario.

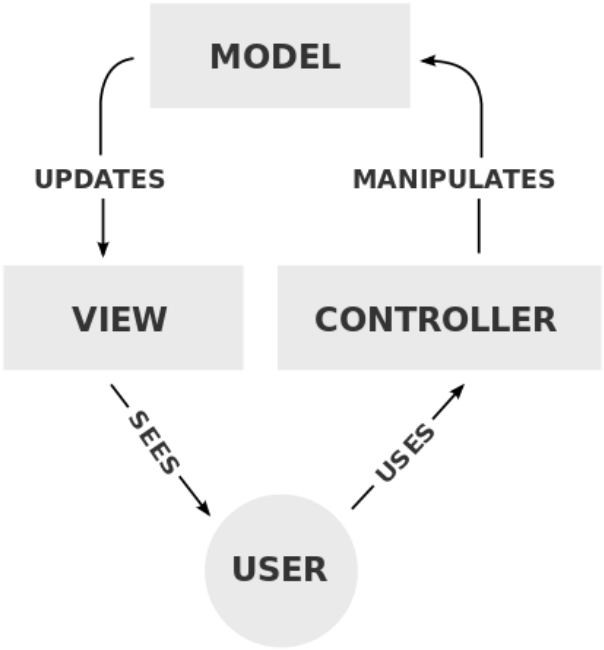

Fig. 4.4: Arquitectura Modelo Vista Controlador

4.2.3 Modelo

Es la capa donde se trabaja con los datos, es decir, contiene los métodos para acceder a la información y también para actualizarla. Estos datos son los que tendremos almacenados en Firebase y en nuestro caso los correspondientes *selects*, *updates*, *inserts*, etc. lo hace la API de Firebase. Por lo tanto, aunque a lo largo de esta memoria veremos métodos como `update()`, son simplemente métodos de la librería de AngularFire (la librería que conecta Firebase con Angular) y pertenecen al controlador.

4.2.4 Vista

La vista, como su nombre indica, contiene el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, es decir el código HTML y CSS. Desde la vista se trabaja con los datos del modelo, pero no se realiza un acceso directo a ellos.

4.2.5 Controlador

Contiene todo el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una búsqueda, etc.

Es la capa que sirve de enlace entre el modelo y la vista, aunque normalmente su responsabilidad no es manipular directamente los datos, en nuestro caso, al no tener servidor dedicado a la manipulación de datos, sino que tenemos un servidor para proveer los datos, todos los cálculos y llamadas a la API para obtener datos de Firebase se han realizado desde los controladores.

4.3 Estructura y diseño

Gracias a Ionic, la interfaz de usuario, así como los estilos usados para Android, iOS y Windows Phone, se adaptan según la plataforma en la que estén. Ionic ayuda mucho con la estructura de la aplicación ya que crea una plantilla limpia desde cero siguiendo las pautas de diseño nativas de iOS y Windows, y en el caso de Android, las de Material Design. Esto nos permite tener una base con estilos ya aplicados, pero a la vez 100% personalizables al gusto del desarrollador. En este caso se han mantenido la mayoría de los estilos que trae Ionic por defecto con el objetivo de ofrecer una experiencia de usuario familiar y conocida, y mantenerlo lo más simple posible para que todo tipo de persona pueda usarlo con facilidad.

El hecho de usar estilos de la interfaz nativa de cada plataforma es muy importante, ya que cualquier persona podrá coger la aplicación y aprender fácilmente cómo funciona y dónde está el menú, por ejemplo, porque sigue la misma estructura que las demás aplicaciones modernas.

En la figura 4.1 se puede ver un ejemplo de las diferencias según la plataforma y de la estructura inicial que nos aporta nuestro proyecto:

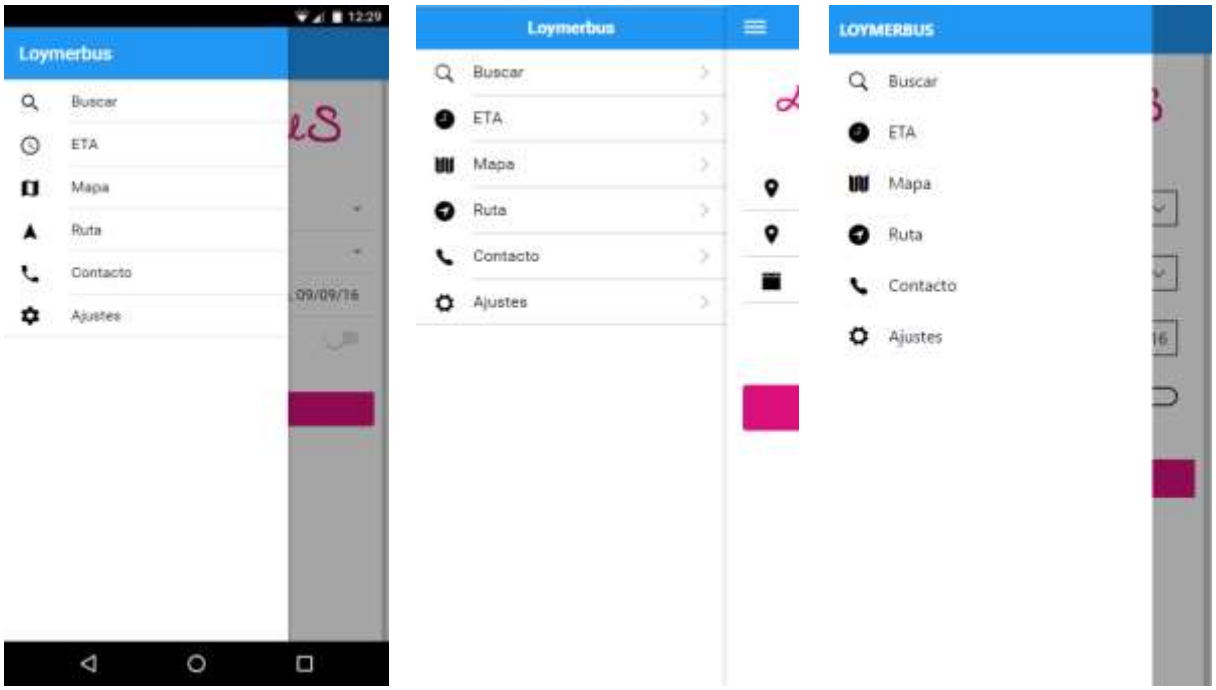

Fig. 4.5: Menú lateral en Android, iOS y Windows Phone

En cuanto a los colores, para mantener el *branding* de la empresa Loymerbus se han extraído los colores de este logo, que son los mismos usados en todos los autobuses:

Fig. 4.6: Logo de Loymerbus

Siendo los colores primarios de la aplicación el rosa y el azul, estos se pueden usar en la aplicación a través del fichero `themes/app.variables.scss` donde Ionic nos lo pone fácil para modificar los colores primarios, secundarios, etc. que se usan en toda la aplicación, así como componentes más específicos. Se aplica el rosa a los componentes primarios (botones, textos importantes, …) y el azul para la barra de navegación.

```
$toolbar-background: #2196F3;
$toolbar-text-color: white;
$toolbar-md-text-color: white;
$toolbar-md-button-color: white;
$toolbar-ios-button-color: white;

$colors: (
  primary:    #db107c,
  secondary:  #32db64,
  danger:     #f53d3d,
  light:      #f4f4f4,
  dark:       #222,
  favorite:   #69BB7B
);
```

Capítulo 5. Desarrollo

5.1 Base de datos

Para las funcionalidades de esta aplicación no es necesario una base de datos muy complicada con la que poder hacer operaciones SQL complejas o crear tablas con relaciones entre sí, ya que las consultas necesarias son bastantes sencillas de realizar debido a la simplicidad de la información, siendo la mayor parte cadenas de texto representando los nombres de los pueblos de salida y llegada, las horas de llegada y salida, números con las coordenadas exactas, y los precios. Es decir, las operaciones que se realizarán serán simplemente para consultar textos y números en concreto según los que se introducen en los campos de texto.

Por esta razón basta con un modelado NoSQL con el que poder escalar horizontalmente con facilidad en caso de añadir más líneas de autobuses o más paradas.

Como hemos mencionado en la introducción, Firebase encaja perfectamente con este perfil ya que nos permite tener una base de datos en forma de JSON sobre la que podemos realizar consultas e incluso actualizarla haciendo uso de su API y los métodos correspondientes que nos proporcionan, ahorrando así el trabajo de crear nuestra propia API para traer información de nuestra base de datos. En la figura 5.1 podemos ver la cantidad de funcionalidades que trae Firebase:

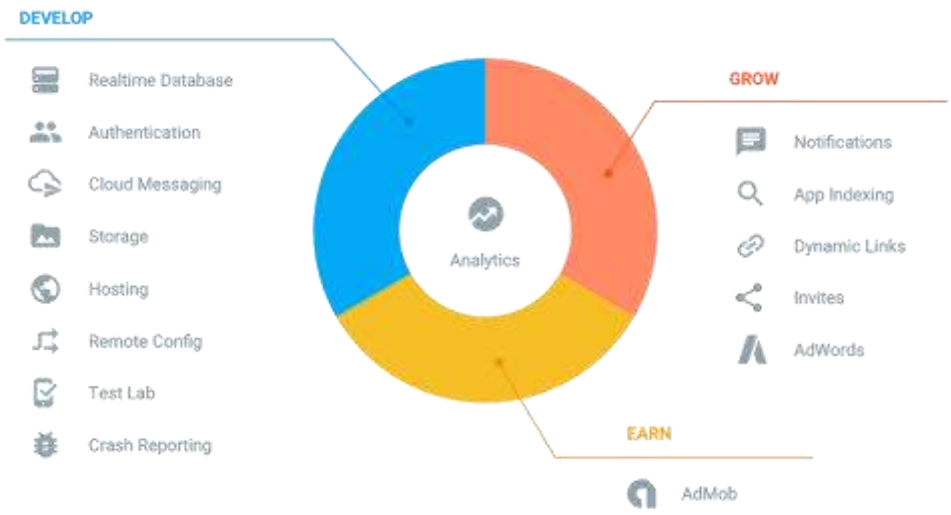

Fig. 5.1: Características de Firebase

La información recopilada se ha transformado en un fichero JSON con una estructura específica para facilitar la lectura y acelerar las búsquedas (más adelante en la sección 5.3.1 de búsqueda se explicará porque se ha estructurado de esta manera) quedando así:


```
{
  "trayectos": [
    {
      "salida": "Canillas de Albaida",
      "lat": 36.845524,
      "lng": -3.985473,
      "destinos": [
        {
          "destino": "Cómpeta",
          "precio": 1.16,
          "dias": [
            {
              "lmxjv": [
                {"horaSalida": "07:00", "horaLlegada": "07:15"},
                {"horaSalida": "09:30", "horaLlegada": "09:45"},
                {"horaSalida": "15:30", "horaLlegada": "15:45"}
              ]
            },
            {
              "sabado": [
                {"horaSalida": "09:00", "horaLlegada": "09:10"},
                {"horaSalida": "15:30", "horaLlegada": "15:45"}
              ]
            },
            {
              "domingo": [
                {"horaSalida": "09:00", "horaLlegada": "09:10"},
                {"horaSalida": "18:00", "horaLlegada": "18:15"}
              ]
            }
          ]
        },
        {
          "destino": "Sayalonga",
          "precio": 1.27,
          "dias": [
            {
              "lmxjv": [
                {"horaSalida": "07:00", "horaLlegada": "07:40"},
                {"horaSalida": "09:30", "horaLlegada": "10:05"},
                {"horaSalida": "15:30", "horaLlegada": "16:05"}
              ]
            },
            {
              "sabado": [
                {"horaSalida": "09:00", "horaLlegada": "09:25"},
                {"horaSalida": "15:30", "horaLlegada": "16:05"}
              ]
            },
            {
              "domingo": [
                {"horaSalida": "09:00", "horaLlegada": "09:25"},
```

```
                {"horaSalida": "18:00", "horaLlegada": "18:30"}
              ]
            }
          ]
        }
      ...
```

Como se puede, ver hay varios trayectos, todos con su correspondiente pueblo de salida que representan a las diferentes paradas de autobús, identificadas por el nombre de ese pueblo y las coordenadas exactas de la parada en sí. Desde cada parada de autobús se puede llegar a otras por un determinado precio, y estas son accesibles a través de diferentes viajes a diferentes horas del día, según el día de la semana, con sus correspondientes horas de salida y horas de llegada.

5.2 Creación y configuración del proyecto

La creación y configuración del proyecto se podría separar en tres fases, todas ellas con ciertas instalaciones y configuraciones que realizar:

- Ionic: creación del proyecto base.
- Firebase: conexión con la base de datos y hosting.
- GitHub: creación y subida a repositorio privado para control de versiones.

Antes de iniciar la creación del proyecto se necesita NPM, un administrador de paquetes JavaScript, que nos permitirá instalar y mantener actualizados los paquetes necesarios de Ionic y Apache Cordova, y más adelante los de Angular 2, Firebase y AngularFire 2. Es el instalador de paquetes predeterminado de Node.js por lo que basta con instalar Node.js para tener también instalado NPM.

Una vez instalado NPM se procede a instalar los paquetes de Ionic 2 y Apache Cordova (desde una línea de comandos):

```
npm install -g ionic@beta cordova
```

Para iniciar un proyecto nuevo ejecutamos:

```
ionic start loymberbus sidemenu --v2
```

Con las opciones “`sidemenu`” indicamos que nos cree una plantilla de un proyecto que tenga menú lateral, “`v2`” para indicar que vamos a usar la versión nueva de Ionic que hace uso del remodelado Angular 2, y por último indicar que ahora se usa por defecto el lenguaje de programación TypeScript en vez de JavaScript frente a

versiones anteriores, debido a que el desarrollo de Angular 2 se ha realizado con este lenguaje de Microsoft.

Ya está el proyecto creado y listo para su uso y para la creación de las diferentes páginas (secciones) de la aplicación, pero antes de nada hay que indicar para qué plataformas se estará desarrollando:

```
ionic platform add android

ionic platform add ios

ionic platform add windows

ionic platform add browser
```

Dentro de la carpeta que se ha creado hay una típica estructura de proyecto Cordova donde podemos instalar plugins nativos para extender la funcionalidad de la aplicación, y crear archivos específicos para nuestro proyecto.

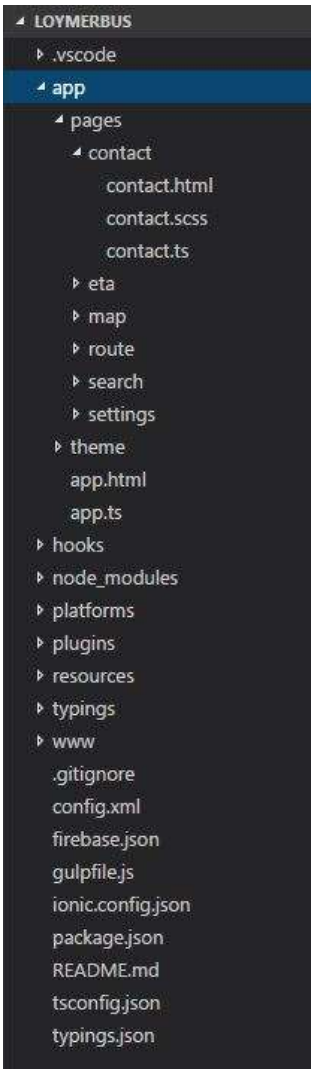

Fig. 5.2: Jerarquía de ficheros del proyecto

Para el uso de Firebase se tiene que usar AngularFire, la librería de conexión entre AngularJS y Firebase. Para ello instalamos sus correspondientes paquetes con NPM igual que antes:

```
npm install angularfire2 firebase --save
```

Inicializamos el proyecto Firebase:

```
firebase init
```

Siguiendo las instrucciones que se muestran e indicando que la carpeta "www" será la que se suba a Firebase, ya que es la que contiene todos los ficheros necesarios para ser ejecutado en la web. Por último, con "`firebase deploy`" se envía el contenido de la carpeta "www" a nuestro proyecto Firebase en la nube. Este comando se ha ejecutado de forma continua durante la realización de este proyecto para aprovechar el hosting gratuito y tener siempre la última versión de la aplicación web ejecutándose en el servidor de Firebase (https://loymerbus.firebase.com).

Ahora podemos incluir en nuestro proyecto Ionic las referencias a nuestro Firebase para poder hacer uso del JSON en nuestra base de datos.

Con indicar en el fichero `app.ts` el proveedor que usaremos, ya se puede inyectar al constructor de cualquier página la referencia a la base de datos de Firebase para su uso en cualquier sección de la aplicación.

```
ionicBootstrap(
    MyApp,
    [
      FIREBASE_PROVIDERS,
      defaultFirebase('https://loymerbus.firebaseio.com'),
      {
        provide: TranslateLoader,
        useFactory: (http: Http) => new TranslateStaticLoader(http,
'assets/i18n', '.json'),
        deps: [Http]
      },
      TranslateService
    ],
    {}
);
```

```
constructor(@Inject(FirebaseRef) public ref:Firebase, public af:
AngularFire…
```

Ya está la base de datos y el proyecto Ionic creado, ambos funcionando y ejecutándose en Firebase. Como último paso, antes de iniciar el desarrollo de la aplicación, para asegurar un correcto entorno de desarrollo y flujo de trabajo, se ha creado un repositorio privado en GitHub donde tener alojado el código y poder hacer

uso del SCM (control de versiones) *git*. A lo largo de la realización del proyecto se ha usado para tener siempre en cuenta todos los cambios realizados y si es necesario, volver a una versión anterior del proyecto.

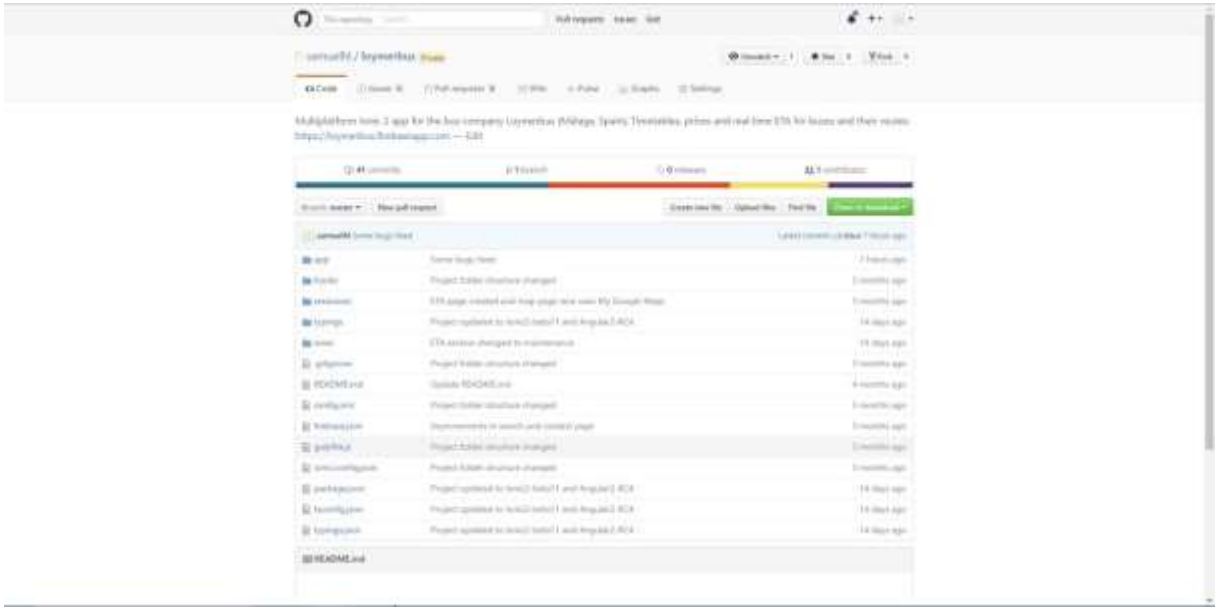

Fig. 5.3: Proyecto en GitHub

5.3 Aplicación Loymerbus

5.3.1 Buscar

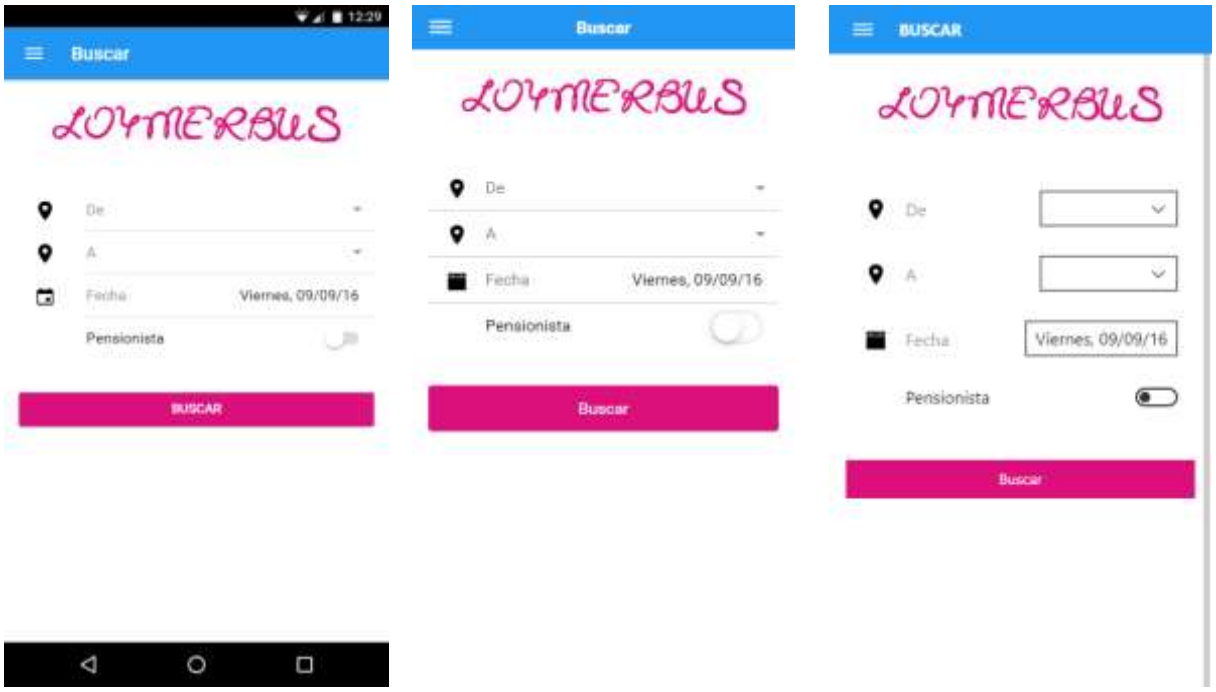

Fig. 5.4: Sección "Buscar" en Android, iOS y Windows Phone

Nada más abrir la aplicación se nos muestra la sección principal de la aplicación, la de búsqueda, que permite a los usuarios consultar los horarios que hay disponibles para un determinado trayecto y día. Quizás sea la sección más útil, pero a la vez una de las más largas de desarrollar.

En primer lugar, introducimos la parada de la que salimos, seleccionando de una lista que se carga al iniciar la aplicación extrayendo el listado de salidas que existen en la base de datos que hemos visto antes. Al seleccionar un elemento del primer campo ya tendremos disponibles un listado de destinos disponibles según la que se ha seleccionado. Esto evita que el usuario elija una salida, un destino y que se encuentre con que no es posible realizar ese viaje, sino que se va encontrar solamente con los destinos a los que sí es posible.

Por defecto, se selecciona la fecha del día actual, ya que en la mayoría de los casos el usuario tiene interés en ver los horarios de ese mismo día. No obstante, se puede cambiar sin problemas introduciendo la fecha deseada. El único dato de real interés aquí es el día de la semana, ya que los autobuses tienen el mismo horario de lunes a viernes, pero diferentes sábado y domingo.

Como se ha comentado antes, la razón por la que la base de datos estaba estructurada de esa forma es para que la introducción de datos siga el mismo orden, reduciendo drásticamente el tamaño de datos a manejar cada vez que se introduce uno nuevo. Primero la salida, ya podemos descartar todos los demás pueblos y solo tener en cuenta aquellos a los que se puede viajar desde allí. Con ese dato tendríamos disponible ya el precio del viaje (no cambia según el día), y tres opciones de fechas: entre semana (en la base de datos llamado “lmxjv”), sábado o domingo, cada uno con horarios diferentes. Una vez seleccionado la fecha, se descartan las demás horas y tendríamos disponible las horas de llegada, de salida y precios disponibles.

La última opción que se puede rellenar, de forma opcional, es indicar si eres pensionista y tienes la correspondiente tarjeta de la Junta de Andalucía para disfrutar de un descuento. Esto es meramente por mostrar los precios directamente con un 50% de descuento si el usuario lo desea.

El código HTML de esta sección se encuentra en `loymerbus/pages/search/search.html` y su correspondiente controlador, encargado de manejar los datos y cargar una cosa u otra en `loymerbus/pages/search/search.ts`.

Al hacer clic en BUSCAR, obtenemos los datos introducidos a través de un típico formulario HTML que es pasado al controlador de la página de detalle de búsqueda (se encuentra también en el fichero `pages/search/search.ts`).

A continuación, se ve en código el proceso de obtención de los datos que se ha explicado antes:

```
class SearchDetailPage {
  salida: string;
  destino: string;
  precio: string;
  viajes: boolean;
  horas: FirebaseListObservable<any[]>;

  constructor(@Inject(FirebaseRef) public ref: Firebase, public af:
AngularFire, public params: NavParams, public viewCtrl: ViewController)
{
    let idDestino: string = params.get('idDestino');
    let idSalida: string = params.get('idSalida');
    let fecha = moment(params.get('fecha'));
    let desc: boolean = params.get('desc');
    let               firebaseRef:               Firebase               =
this.ref.child('trayectos').child(idSalida);

    // Obtenemos nombre de salida
    firebaseRef.on('value', snapshot => {
      this.salida = snapshot.val().salida;
    });

    // Obtenemos nombre del destino y precio
    firebaseRef.child('destinos').child(idDestino).on('value',  snapshot
=> {
      this.destino = snapshot.val().destino;
      this.precio = snapshot.val().precio;
    });

    // Aplicamos descuento
    if (desc) {
      this.precio = (parseFloat(this.precio)/2).toFixed(2);
    } else {
      this.precio = (parseFloat(this.precio)).toFixed(2);
    }

    // Obtenemos fecha
    if (fecha.day() == 6) {
      this.horas  =  this.af.database.list('/trayectos/'  +  idSalida  +
'/destinos/' + idDestino + '/dias/1/sabado');
    } else if (fecha.day() == 0) {
      this.horas  =  this.af.database.list('/trayectos/'  +  idSalida  +
'/destinos/' + idDestino + '/dias/2/domingo');
    } else {
      this.horas  =  this.af.database.list('/trayectos/'  +  idSalida  +
'/destinos/' + idDestino + '/dias/0/lmxjv');
    }
```

```
    this.horas.do(snapshots => {
      if (snapshots.length == 0) {
        this.viajes = false;
      } else {
        this.viajes = true;
      }
    }).subscribe();
  }
```

La página de detalle trata un componente *modal*, una especie de pantalla temporal que muestra información relevante de una sección en particular. Por ello tiene su propio fichero HTML (`loymerbus/pages/search/searchDetail.html`) que es llamada por el controlador de la página de búsqueda y muestra de forma ordenada y clara el viaje que se ha seleccionado, así como los horarios de salida y llegada, y los precios, en una tabla. También dispone de un botón para retroceder con facilidad a la pantalla anterior ya que, como se ha dicho antes, es solo una pantalla temporal desde la cual consultar los resultados de nuestra búsqueda.

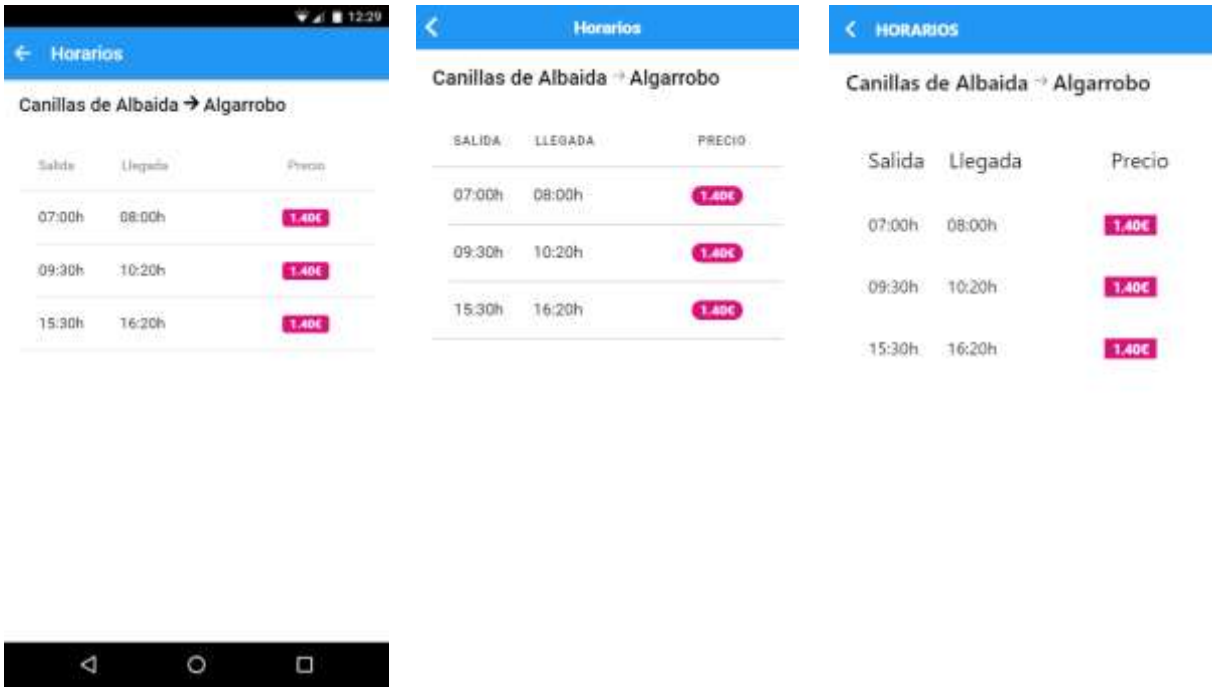

Fig. 5.5: Detalle de una búsqueda en Android, iOS y Windows Phone

5.3.2 Mapa

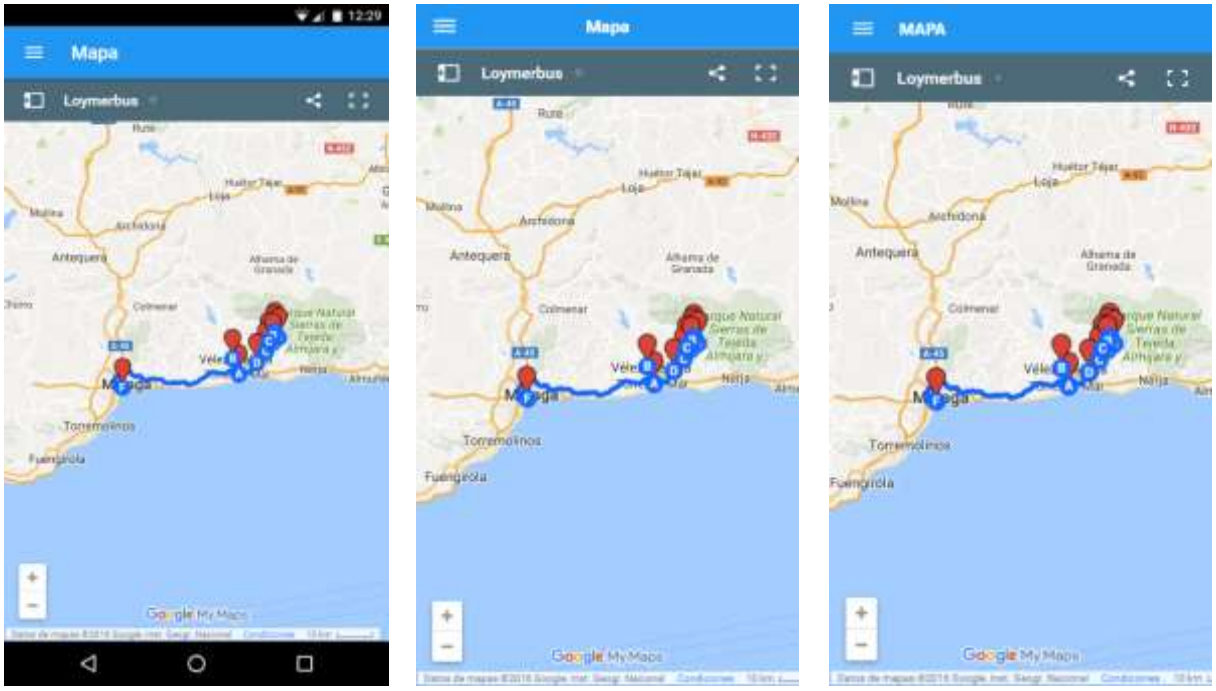

Fig. 5.6: Sección "Mapa" en Android, iOS y Windows Phone

Otro problema a resolver ha sido la localización de las paradas de autobús, no están señalizadas en ningún lado y no está claro si los autobuses paran o no, ya que en la gran mayoría de los casos se tratan de cabinas que parecen estar abandonadas. Incluso en el caso de Canillas de Albaida, no hay parada, se trata de una rotonda donde para el autobús. Si no eres de la zona no hay posibilidad de saber esto. Por ello esta información se ha obtenido también preguntando a personas que viven por la zona para obtener la posición exacta de las paradas de autobús.

Primero se han mostrado incluyendo Google Maps en la aplicación, pero a la hora de mostrar también las rutas exactas entre paradas o añadir información adicional sobre cada parada se empieza complicar la cosa. Google Maps tiene un simple objetivo único, mostrar rutas entre dos posiciones. Entonces para mostrar el cambio de autobús para Vélez-Málaga o para Árchez por ejemplo, es imposible representar las rutas tal y como la queremos.

Debido a esta complicación se ha optado por usar Google My Maps, una plataforma de Google en la que puedes crear, de forma interactiva con una interfaz de usuario en Internet, mapas con marcadores en las posiciones que queramos y todas las rutas que queramos en las posiciones que nos hagan falta. De esta forma se ha conseguido un mapa con un nivel de detalle mucho más amplio y con más información en menos tiempo.

Se puede exportar este mapa con facilidad para incluirlo en nuestra aplicación en forma de *iframe* que encontraremos en el HTML de esta sección (`loymerbus/pages/map/map.html`).

```
<ion-content class="map">
  <iframe                                                    id="map"
src="https://www.google.com/maps/d/embed?mid=1sk6tmdPMR5-
zYAqS3xoQtIplIUI"></iframe>
</ion-content>
```

El controlador de esta sección (`loymerbus/pages/map/map.ts`) no realiza ninguna acción respecto al mapa ya que se carga automáticamente en el HTML, pero sí que tiene un controlador para que aparezca una pequeña ventana de “Cargando”, ya que esta es la sección que tarda más en cargarse por tratarse de una petición HTTP al mapa de Google que hemos creado y luego el renderizado de la misma. Con 2 o 3 segundos de espera es suficiente, pero con este efecto que hemos creado hace que la transición a esta vista sea mucho más amigable y placentera para el usuario.

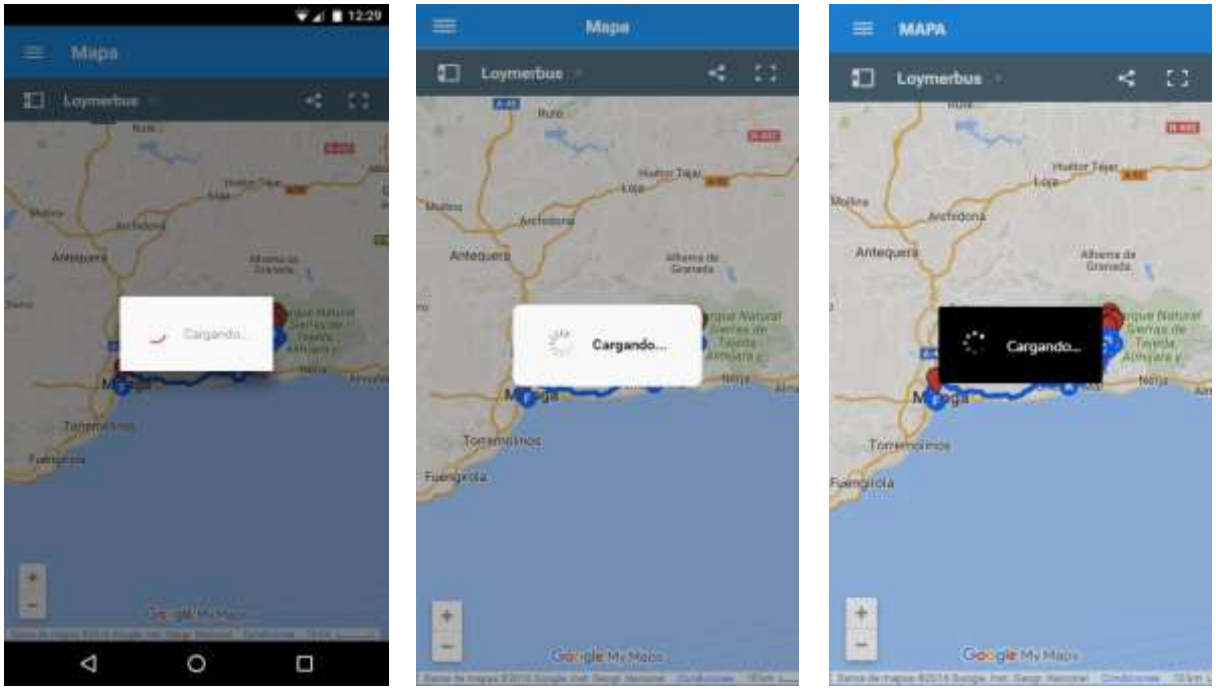

Fig. 5.7: Cargando sección "Mapa" en Android, iOS y Windows Phone

5.3.3 Ruta

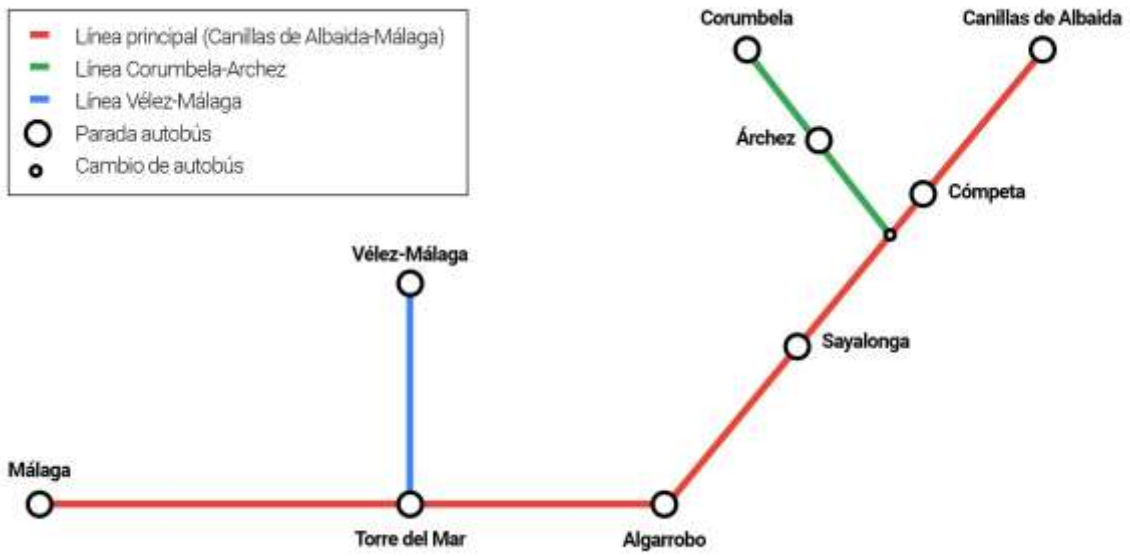

Fig. 5.8: Esquema illustrativo de rutas y paradas

Para entender mejor la ruta que hacen los autobuses y ver lo desvíos y cambios de autobuses que son necesarios hacer para llegar a determinadas paradas, se ha realizado el típico esquema que suelen tener las líneas del metro o autobuses colgadas en las paradas. Se ha realizado desde cero con Adobe Photoshop CC 2015 en forma de imagen JPEG que va incluido en la sección de “Ruta” con un tamaño considerable para tener suficiente calidad. Se puede hacer scroll tanto en el eje Y como en el eje X para ver al completo la ruta.

El código HTML del mismo se encuentra en el fichero `loymerbus/pages/route/route.html`, sin necesidad de controlador, debido a que se trata de mostrar simplemente una imagen informativa con ciertos estilos aplicados para darle ese tamaño.

```
<ion-content class="route">
    <ion-scroll scrollX="true" scrollY="true" zoom="true" overflow-
scroll="false" style="width: 100%; height: 100%">
      <div class="route-img"></div>
    </ion-scroll>
</ion-content>

<style>
  .route-img {
    width: 1920px;
    height: 1080px;
    background: url('img/ruta.jpg') no-repeat;
  }
```

```
</style>
```

5.3.4 Contacto

La aplicación incluye también una sección de contacto con una ficha informativa sobre la oficina central del Grupo Loymer que se encuentra en Torre del Mar, para poder hacer cualquier tipo de consulta sobre sus servicios.

Para ello se incluye una foto de la oficina, la dirección exacta de la misma, así como un botón que nos abre Google Maps para poder navegar hasta su ubicación exacta (si está Google Maps instalado en el dispositivo se abre en ella, sino se abre en el navegador predeterminado que tengamos). La página web oficial (en la fecha de la realización de este proyecto no estaba en funcionamiento), el número de teléfono sobre el cual podemos pinchar para abrir la aplicación de teléfono que se tenga en el dispositivo y realizar directamente la llamada; un correo electrónico que al igual que en el caso del número de teléfono, podemos pinchar sobre él para abrir la aplicación de correo predeterminada que tengamos configurada. Por último, el horario de apertura de la oficina, que se ha obtenido preguntando a uno de los empleados.

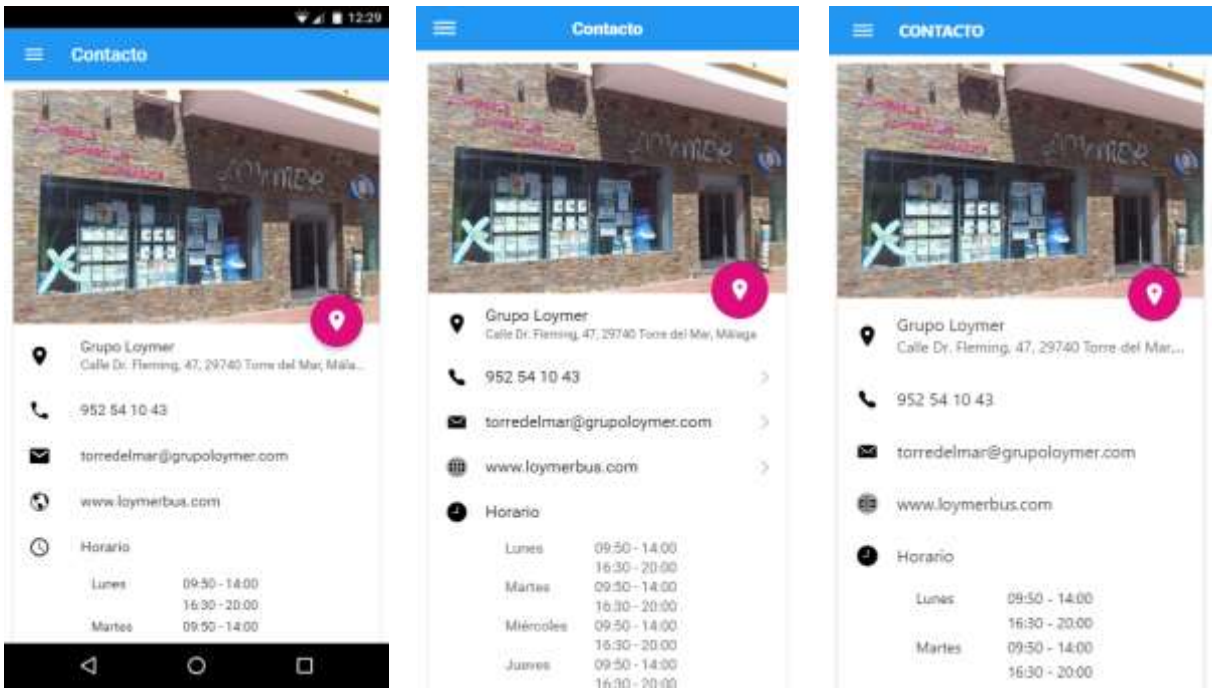

Fig. 5.9: Sección "Contacto" en Android, iOS y Windows Phone

5.3.5 Ajustes

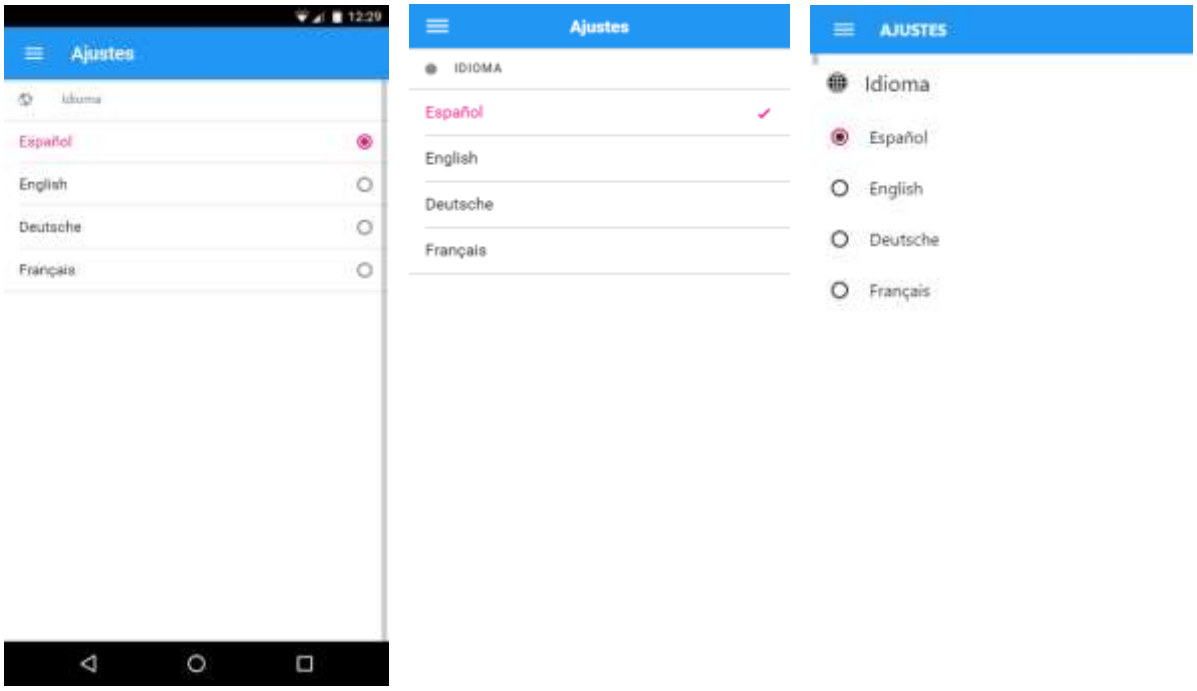

Fig. 5.10: Sección "Ajustes" en Android, iOS y Windows Phone

El único cambio de configuración de interés en esta aplicación es el cambio de idioma, pudiendo elegir entre: español, inglés, alemán y francés.

Para la traducción se ha empleado una librería de NPM llamado ng2-translate:

```
npm install ng2-translate --save
```

Este paquete permite usar el estándar i18n de internacionalización en aplicaciones que usan Angular 2.

Para ello se ha tenido que crear una carpeta assets dentro de www, y dentro de esa, otra carpeta llamada i18n, que contendrá un fichero JSON por cada uno los idiomas que queremos incorporar a nuestra aplicación. Con el siguiente formato "CÓDIGO_IDIOMA".json. En este caso cuatro:

- es.json (español)
- en.json (inglés)
- de.json (alemán)
- fr.json (francés)

Cada una de ellas sigue la misma estructura: una clave y su correspondiente valor en el idioma que corresponda. Por ejemplo, cada fichero contiene todos los títulos de las secciones traducidas, es decir, todos tienen la clave "search", y de valor "Search" en inglés, "Buscar" en español, "Suche" en alemán, y "Chercher" en francés.

Así con todos los nombres de las secciones, así como cualquier otra palabra estática que se encuentra en la aplicación.

Para poder hacer uso de estas traducciones lo primero es realizar las correspondientes importaciones y configuración de los proveedores en el fichero `app.ts`:

```
translateConfig() {
    let userLang = navigator.language.split('-')[0];
    userLang = /(de|en|es|fr)/gi.test(userLang) ? userLang : 'es';
    this.translate.setDefaultLang('en');
    this.translate.use(userLang);
  }
}
```

Además de la creación de un método nuevo para cargar y usar el idioma del navegador (si está disponible), y en caso de que no, o en caso de ser un idioma que no soporte nuestra aplicación se usará el inglés por defecto. Llamamos este método en el constructor de nuestra aplicación.

Es necesario también añadir el símbolo “pipe” a cada página de nuestra aplicación que requiera traducciones para poder usar expresiones del tipo `{{"key" | translate}}` en el HTML correspondiente de la página. Con esta simple expresión estamos indicando a nuestra aplicación que coja la traducción para la clave “key” (por ejemplo, ”search”, “cancel”, “monday”, etc.), usando el idioma que esté configurado en ese momento, ya sea porque se haya cargado al iniciar la aplicación o porque el usuario ha seleccionado otra diferente en esta sección.

Para cambiar de idioma basta con seleccionar la que se desea activar y el idioma de la aplicación entera es cambiada de forma instantánea. Esto ocurre ya que al cambiar una de las opciones de la lista de idioma se llama al método `onChange` del controlador:

```
  onChange(lang) {
    this.translate.use(lang);
  }
```

Como podemos ver, hace uso de uno de los métodos de la librería ng2-translate para cambiar el idioma al valor que se le haya pasado como parámetro.

5.3.6 Tiempo estimado de llegada

A los usuarios se les permite también consultar el tiempo estimado de llegada (o ETA de sus siglas en inglés, *Estimated Time of Arrival*) del autobús a la parada en la que se encuentran. Si hay algún autobús en camino, se mostrará al usuario un tiempo aproximado que se ha calculado en tiempo real empleando Google Maps, que tiene en cuenta la distancia y el tráfico que hay en ese momento.

Se ha intentado emular la mayoría de líneas de autobuses existentes en los que basta con introducir la parada de interés y pulsar el botón “BUSCAR”.

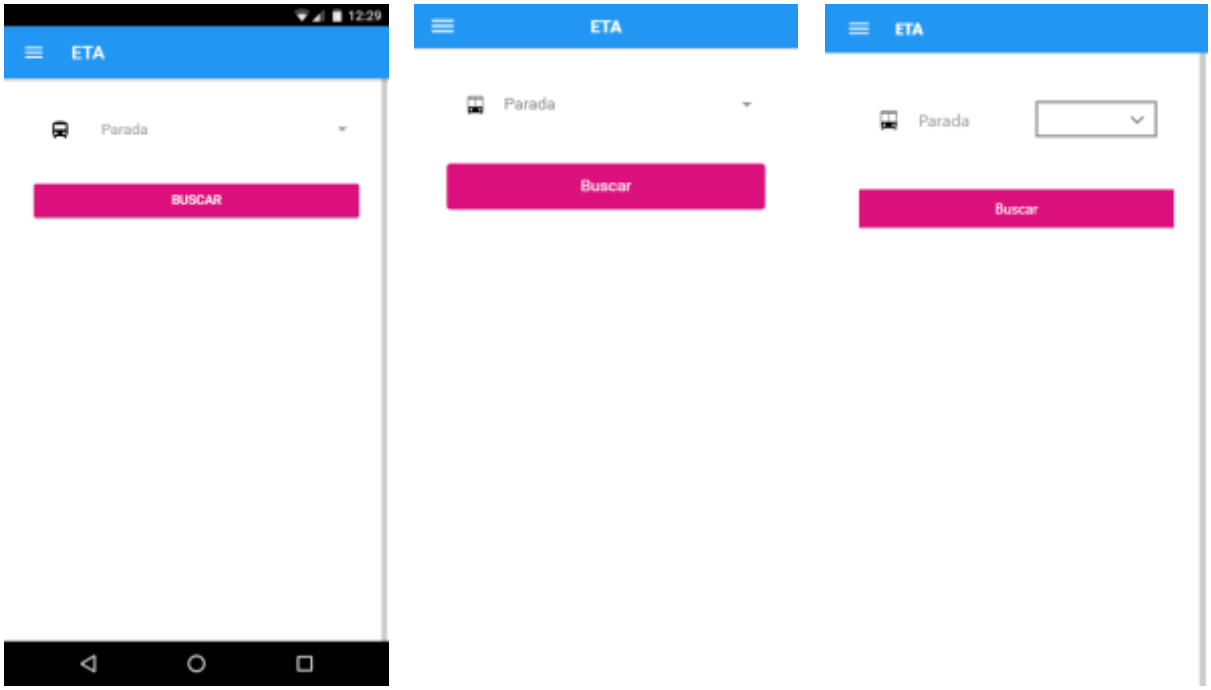

Fig. 5.11: Sección "ETA" en Android, iOS y Windows Phone

Para conseguir esto, es necesaria la localización del autobús. En el caso de la empresa Loymerbus con la que estamos tratando, no tienen GPS instalado en los autobuses. Por esta razón es imposible extraer su ubicación a través de este medio. Sin embargo, se puede aprovechar un dispositivo que sí que tendría todo conductor con él y que llevaría en el autobús: su móvil.

Los teléfonos móviles hoy día permiten acceder fácilmente a su geolocalización a través de su GPS interno. Por tanto, con una aplicación secundaria instalada en el móvil del conductor (o en un móvil secundario que se dejara en el autobús para hacer la función de GPS), podríamos actualizar en nuestra base de datos la ubicación del autobús cuando haga falta y desde la aplicación del usuario realizar los cálculos necesarios.

Se han estudiado dos soluciones para realizar esta tarea, de las cuales se ha elegido una y se ha descartado la otra. A continuación, se explican cada una de ellas y por qué se ha usado una y no otra:

- Tener en ejecución, en segundo plano, nuestra aplicación y que vaya actualizando la ubicación del autobús cada cierta cantidad de metros recorridos.
    - Vigilancia constante de trayecto de autobús.
    - Ubicación de autobús actualizada constantemente.
    - Mayor consumo de batería.
    - Mayor consumo de datos.
- Tener en ejecución, en segundo plano, nuestra aplicación y que actualice la ubicación del autobús solo cuando se solicita el ETA desde la aplicación de usuario.
    - Solo actualiza ubicación cuando es requerida.
    - Menos consumo de batería.
    - Menor consumo de datos.

Es obvio que tener una aplicación en segundo plano consume más batería, en la primera opción algo más que en la segunda debido a que actualizaría la base de datos muchas más veces, esto también conllevaría más consumo de datos. La primera opción es interesante para tener también un seguimiento de los trayectos de los autobuses en caso de que la empresa lo requiera, pero ahora mismo no buscamos eso.

La segunda opción tardaría algo más de tiempo, ya que tenemos que lanzar el evento a la aplicación secundaria para indicarle que alguien está solicitando el tiempo de espera que le queda y se necesita la ubicación actualizada para realizarlo. En el primer caso tendríamos siempre la ubicación actualizada. A pesar de esto se ha sacrificado un poco de velocidad de respuesta y se ha optado por la segunda opción.

En la siguiente sección explicaremos con detalle cómo funciona esta aplicación secundaria que hemos comentado y cómo se actualiza todos los datos necesarios para darle al usuario esta opción.

5.4 Aplicación Loymerbus GPS

Como se ha dicho anteriormente, es necesario disponer de una segunda aplicación móvil, que llamaremos “Loymerbus GPS”, y que iría instalada en el móvil del conductor del autobús. La base de datos y la creación y configuración del proyecto sería exactamente de la misma forma que se ha visto en las secciones 5.1 y 5.2 de esta memoria.

Esta aplicación secundaria estaría ejecutándose en todo momento en segundo plano y tiene que acceder a la geolocalización del dispositivo, por lo que tendremos que hacer uso de algún plugin de Cordova para acceder a funciones nativas del dispositivo. En este caso background-mode y geolocation:

```
cordova plugin install cordova-plugin-background-mode

cordova plugin install cordova-plugin-geolocation
```

Con los plugins ya instalados podemos hacer las correspondientes importaciones de sus librerías para acceder a sus métodos.

Antes de nada, se han tenido que realizar algunos cambios en la base de datos. Hemos añadido un apartado “gps” donde hay dos campos “bus1 y “bus2”; cada uno representa un autobús para una ruta diferente. Dentro de cada autobús tienen los mismos campos que detallaremos a continuación:

- **lat**: la latitud del autobús. Se actualiza, si y solo si, un usuario ha solicitado un tiempo de espera.
- **lng**: la longitud del autobús. Se actualiza, si y solo si, un usuario ha solicitado un tiempo de espera.
- **latA**: latitud del autobús. Cada 10 minutos se le asigna latB.
- **lngA**: longitud del autobús. Cada 10 minutos se le asigna lngB.
- **latB**: latitud del autobús. Cada 10 minutos es actualizado.
- **lngB**: longitud del autobús. Cada 10 minutos es actualizado.
- **direction**: dirección en la que se dirige el autobús. Se indica con el índice de la parada en nuestra base de datos y varía según el autobús. Para el “bus1” puede valer 0 o 5, Canillas de Albaida o Málaga, respectivamente. Para el “bus2” puede valer 7 o 5, Corumbela o Málaga, respectivamente.
- **nextStop**: nos indica la próxima parada hacia la que se acerca el autobús.

Con solo tener un teléfono móvil para sacar la ubicación de los autobuses y no un GPS con una vigilancia constante de cada uno, los recursos son bastantes limitados y se ha realizado un largo estudio sobre cómo sacar toda la información

necesaria para obtener unos resultados óptimos y satisfactorios para el usuario a la hora de simplemente introducir la parada en la que se encuentra.

Vamos a justificar los datos anteriores que hemos tenido guardar en la base de datos y demostrar por qué son necesarios. Para ello es muy importante tener en cuenta las rutas que siguen los autobuses y el nombre de las paradas, por lo que se recomienda consultar de nuevo la imagen de la ruta en el apartado 5.3.3 Ruta:

- De primeras, es necesario saber en qué dirección va el autobús en todo momento y cuál es la siguiente parada hacia la que se dirige. De aquí surgen las variables “direction” y “nextStop”. Esto es debido a limitaciones de Google Maps donde introducimos dos coordenadas y nos devuelve el tiempo y distancia entre ellos, pero no nos dice nada más. El fallo está claramente en que, si un usuario solicita, por ejemplo, ver cuánto le falta al autobús en llegar a Algarrobo el autobús podría estar según Google Maps a 2 minutos, pero ¿dónde se encuentra respecto a la parada de interés exactamente?, ¿se acerca o se aleja? Si está en ruta hacia Torre del Mar y el autobús se encuentra al oeste de Algarrobo está claro que el usuario ha perdido el autobús. Esto lo sabemos porque conocemos hacia qué parada se dirige el autobús y en qué dirección va. Veamos otro ejemplo para dejarlo lo más claro posible. Seguiremos usando Algarrobo como ejemplo. Si la siguiente parada es Torre del Mar igual que antes, pero va en dirección Canillas de Albaida, sabemos que el autobús se acerca hacia Algarrobo y todavía no se lo ha pasado. En este caso podemos mostrarle con éxito al usuario cuánto le falta al autobús en llegar a la parada de Algarrobo.

  Cabe destacar que en líneas más avanzadas de autobuses no ocurren estos problemas, ya que siempre hay dos paradas de autobús, uno en cada lado de la carretera con dirección constante. Aquí no ocurre esto, se trata de pueblos pequeños de la Axarquía y en la gran mayoría de los casos solo tienen una parada para ambos sentidos.

  También suele haber GPS integrado en todos los autobuses que realizan una especie de *check-in* automático cuando paran en determinadas paradas. Esto suprime la necesidad de saber en cada momento hacia qué parada va, ya que sabríamos en todo momento si ya ha visitado una parada o no.

- Estos dos últimos datos como hemos visto son fundamentales y se consiguen con las coordenadas del autobús cada 10 minutos. De aquí las variables “latA”, “lngA”, “latB” y “lngB”. Es necesario actualizar cada 10 minutos estas coordenadas para saber en qué dirección va el autobús en cada momento. Se ha elegido 10 minutos porque es un tiempo bastante considerable para asegurarnos de que el autobús no esté parado más de ese tiempo y un tiempo que no sobrecargará la base de datos ni el móvil.

Veamos como hacemos esto con un par de ejemplos:

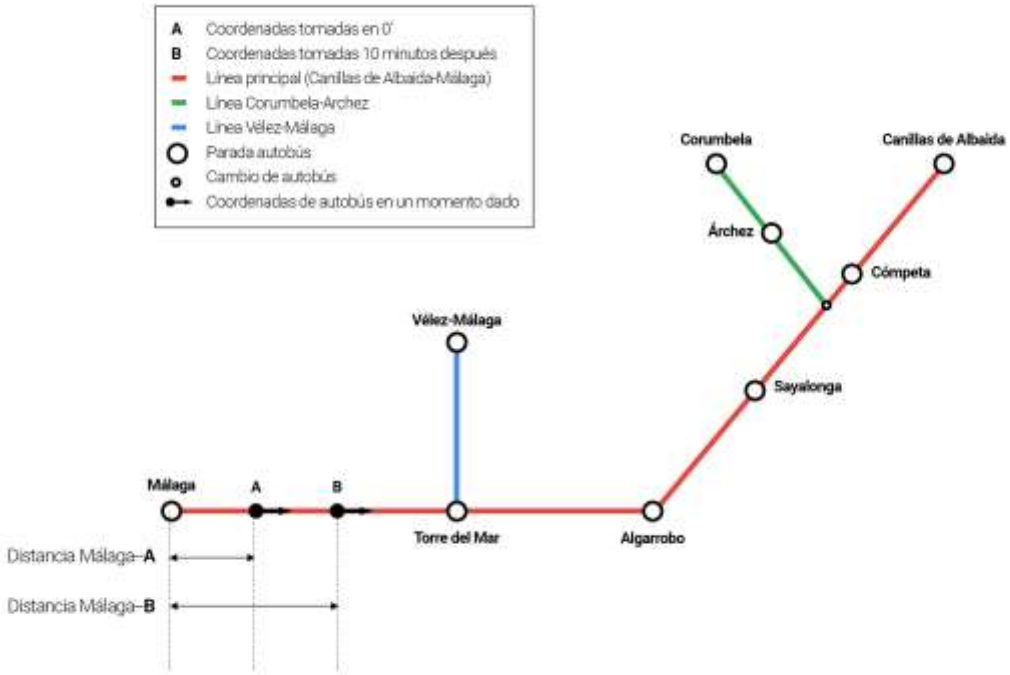

Fig. 5.12: Ejemplo 1 para calcular dirección del autobús

Tomamos como punto de referencia Málaga (para la línea Corumbela - Árchez se toma como punto de referencia el punto de “cambio de autobús”, pero funciona de la misma manera) y calculamos la distancia del autobús en el punto A a Málaga y lo mismo del punto B. El punto B es el más actualizado, entonces si es mayor esa distancia podemos afirmar que el autobús se aleja de Málaga, o lo que es lo mismo, se acerca a Canillas de Albaida. Es decir:

SI Distancia Málaga-B > Distancia Málaga-A **ENTONCES** direction = Canillas de Albaida

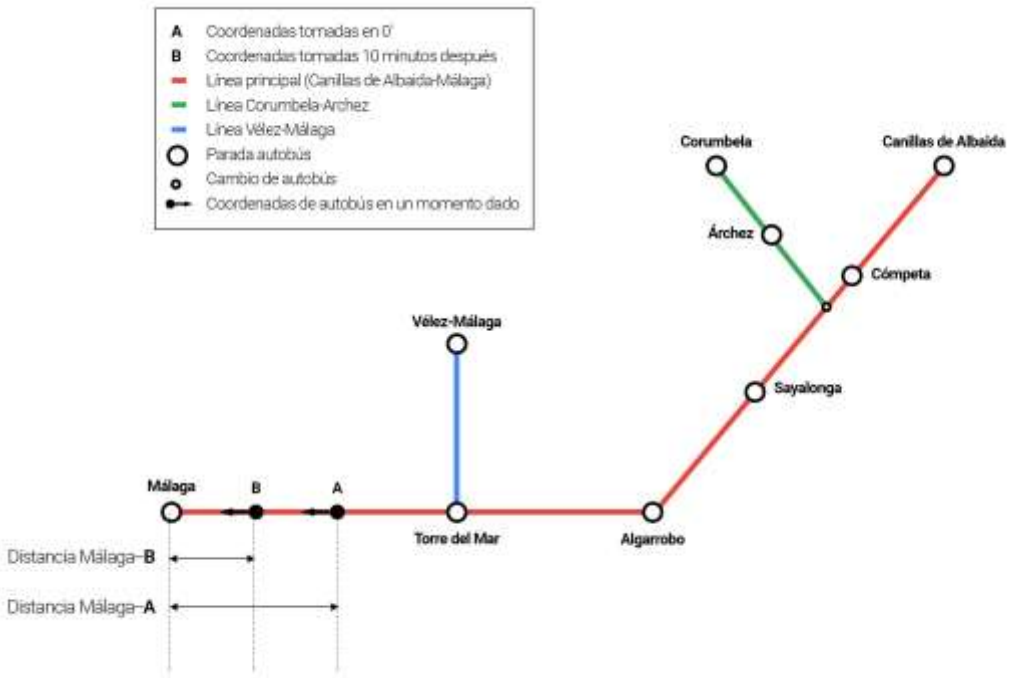

Fig. 5.13: Ejemplo 2 para calcular dirección del autobús

Si el autobús va en el sentido contrario, funciona de la misma manera:

SI Distancia Málaga-B < Distancia Málaga-A **ENTONCES** direction = Málaga

Con esto ya tenemos la variable “direction” actualizada, vamos a explicar cómo sacamos la variable “nextStop” actualizada.

- En el mismo proceso anterior sacamos también la próxima parada del autobús, aprovechando las coordenadas A y B. El algoritmo consiste en un bucle en el que comparamos la distancia del autobús a nuestro punto de referencia Málaga, con la distancia del mismo punto a la siguiente parada. En el mejor de los casos la primera distancia será menor, por lo tanto, podemos confirmar que está entre Torre del Mar y Málaga, es decir su próxima parada es Torre del Mar (si tiene dirección Canillas de Albaida claro). Si la distancia del bus a Málaga es mayor, entraríamos en el bucle y calculamos la distancia a la siguiente parada, aumentando el rango; volvemos a comparar distancias y seguimos hasta que la distancia del autobús sea menor. Cuando consigamos eso sabremos entre qué dos paradas está, y con la dirección, si se acerca hacia una parada u otra.
- Las variables “lat” y “lng” serán las coordenadas del autobús cuando un usuario solicite el tiempo de espera. Podríamos usar las coordenadas B que hemos mencionado antes, pero podrían estar, en el peor de los casos, 9 minutos equivocados.

- Para esto último sirve la variable “update”. Se mantiene a “false” la mayoría del tiempo, pero si es cambiado a “true”, desde la aplicación del usuario se propaga un evento en la aplicación del conductor para actualizar su posición actual en las variables “lat” y “lng” de la base de datos. Tambien es usado a lo largo de todas las operaciones para asignarle el valor en forma de *string* “error”. Cualquier tipo de fallo de Google Maps de que la posición no está disponible en ese momento, etc. convierte el valor de “update” en “error”.

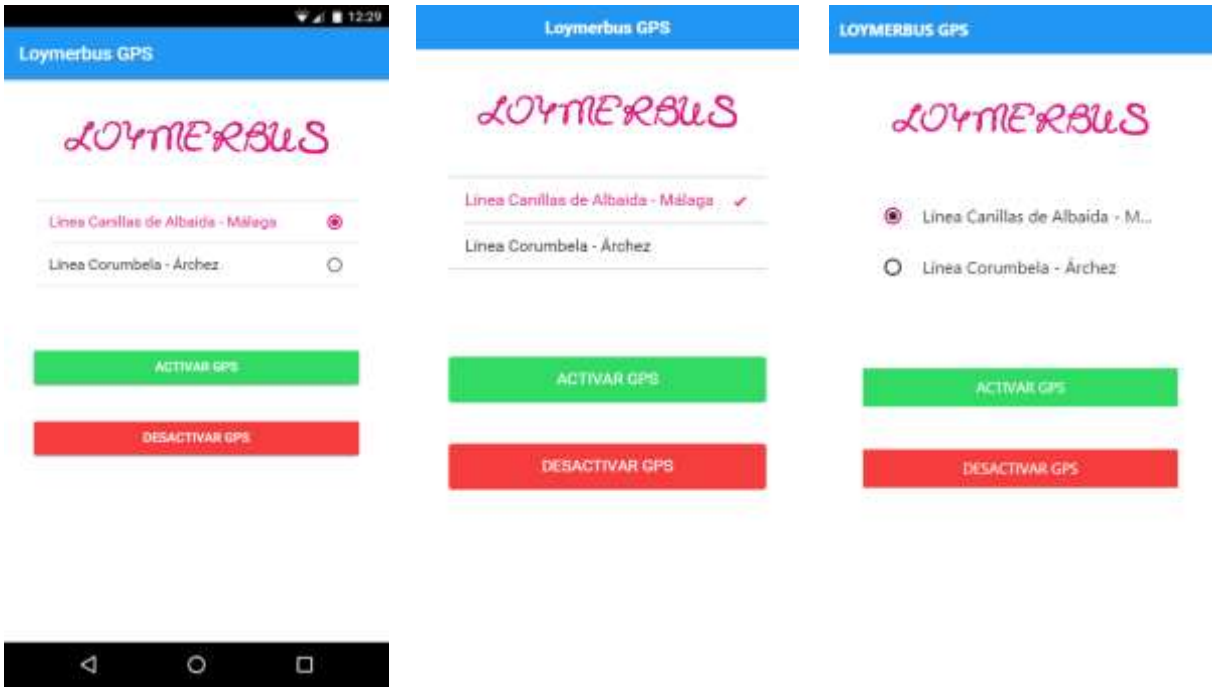

Fig. 5.14: Aplicación Loymerbus GPS

Antes de nada, es importante saber qué autobús es, ya que podría ser la línea principal o la secundaria para conectar Corumbela y Árchez (cabe destacar que aunque haya otro autobús aparte según se ha podido ver en la figura 5.8 para llegar hasta Vélez-Málaga, no se puede coger ningún autobús desde allí por lo tanto no tiene sentido tener también un seguimiento de este autobús). Como se puede ver en las capturas, esto se indica fácilmente indicando una opción u otra, antes de darle al botón “ACTIVAR GPS”. Este botón está pensado para ser pulsado en cuanto el conductor sale de uno de los extremos de su ruta. Por lo tanto, lo primero que debemos hacer es guardar el autobús (1 para la ruta principal de Canillas de Albaida - Málaga y 2 para Corumbela - Árchez).

Vamos a ver qué ocurre internamente al pulsar este botón en orden de ejecución:

- Se ejecuta el método `start` que llama al método `enable()` del plugin que hemos instalado para iniciar su ejecución en segundo plano.
- Debemos inicializar los datos de la base de datos que se realiza en el método `initialize(bus)`, donde según el autobús que sea inicializa unos datos u otros.
- Tras esto usamos el método `setInterval` para ejecutar el comando `update(bus)` cada 10 minutos, lo que actualizará nuestros datos de la base de datos como se ha explicado antes.
- Se está escuchando al objeto “gps” en forma de Observable, es decir, cuando cambie algún valor en la base de datos (en este caso el valor del campo “update”) salta un evento y se ejecuta el código que haya dentro, accediendo a la ubicación del autobús con el plugin Geolocation y actualizando las variables “lat” y “lng” en la base de datos con el método `update` de Firebase.
- Por último, el método `stop()` desactiva el temporizador, desconecta el vinculo entre la aplicación y la escucha de eventos de la base de datos, y cancela la ejecución en segundo plano llamando al método `disable()` del BackgroundMode. También actualiza el parámetro “update” del autobús correspondiente al valor “error”, esto lo usamos para simplemente indicar que ha habido un error al intentar acceder a los datos de esta aplicación, en este caso debido a que el autobús no esta en funcionamiento en ese momento. Todo esto se consigue pulsando el botón “DESACTIVAR GPS”.

IMPORTANTE: Esta aplicación “Loymerbus GPS” no actualiza la ubicación en segundo plano cuando está siendo tratada por la aplicación de Google WebView. Esta aplicación es la encargada de renderizar y mejorar la experiencia de uso de las aplicaciones hechas con tecnologías web. Sin embargo, no permite acceder a la geolocalización del dispositivo en segundo plano debido a que trata a las aplicaciones como si fuera de un navegador web. Para que funcione correctamente en Android se debe instalar en un móvil con versión Android 4.4 KitKat o inferior con la que no es compatible la aplicación WebView, o simplemente desinstalar la aplicación WebView que viene instalado por defecto si se está usando una versión de Android superior.

Capítulo 6. Conclusiones y trabajo futuro

6.1 Conclusiones

En este trabajo de fin de grado se han descrito todas las fases para poder desarrollar una aplicación funcional desde cero con tecnologías web para dispositivos móviles.

Comenzando por comentar lo importante que es Internet hoy en día y lo fundamental que es para cualquier tipo de negocio poder tener su información en la palma de la mano de cualquier usuario esté donde esté, sobre todo cuando se trata de una línea de autobuses transitada todos los días por cientos de personas.

Al estar ya situado en el contexto, se ha podido explicar con claridad cómo se iba a solucionar este problema que tiene la empresa Loymerbus, detallando todas las fases de análisis de información y de la estructura de la aplicación, siempre teniendo en cuenta el rango de edades que podrían usarlo y manteniéndola lo más simple y fácil de usar posible. Una vez definido esto último hemos podido comenzar todo el desarrollo necesario para llevarlo a cabo.

Haciendo uso de herramientas como Ionic y Firebase, se han simplificado muchas tareas que suelen ser tediosas para un desarrollador. Desarrollar desde la nada una API en un servidor para acceder a la información de nuestra base de datos hubiera requerido mucho tiempo y el aprendizaje de otra tecnología más. Firebase me ha permitido simplificar mucho esto, ofreciéndome un alojamiento gratuito de mis datos que necesitaba para la aplicación y una tremenda cantidad de procedimientos y funciones ya hechas para acceder a ellas, actualizarlas, borrarlas, etc. Por otro lado, Ionic une lo mejor de AngularJS y Apache Cordova, dos tecnologías muy usadas y con grandes comunidades detrás de ellas. Desde el momento que ejecuté el primer comando para iniciar mi proyecto me di cuenta de la potencia de Ionic, creándome un proyecto entero con todas las carpetas y archivos necesarios, un menú lateral ya hecho y todos los estilos nativos de Android, iOS y Windows Phone ya confirgurados y programados para ser usados según el dispositivo que estés usando, ya sea porque te has bajado la aplicación como si has visitado el enlace de Firebase para la versión web.

En este trabajo de fin de grado se ha desarrollado no solo una aplicación móvil, sino tres, y una versión web. Esto ha sido posible gracias a los grandes avances en el desarrollo web y la posibilidad de crear aplicaciones hibridas y visualizarlas con los *web views*. Esto es muy interesante para pequeñas empresas que necesiten

aplicaciones multiplataforma, para asegurarse un alcance de público bastante mayor, y en un tiempo y con un coste bastante inferior al que necesitaríamos para desarrollar aplicaciones nativas de cada plataforma.

Está claro que para conseguir rendimientos superiores es necesario desarrollar implementaciones nativas. Sin embargo, para el caso de este trabajo de fin de grado bastaba con una aplicación simple para mostrar información de interés y que sea fácilmente accesible.

La realización de este proyecto ha supuesto un aumento de mis conocimientos en muchos ámbitos del desarrollo de software, ya que el trabajo realizado en cada fase ha tocado alguna rama del desarrollo de una aplicación completa: el diseño, la base de datos, la configuración inicial y mantenimiento de los paquetes, el desarrollo con MVC (Modelo Vista Controlador), etc.

En resumen, un proyecto bastante completo donde se ha desarrollado una aplicación móvil completamente funcional usando las tecnologías que más me gustan y que mejor manejo, las tecnologías web.

He aprendido a ver un problema real y solucionarlo usando la tecnología y me ha permitido trabajar en un proyecto que podría ser de real interés para el público que use esta línea de autobuses.

He disfrutado mucho aprendiendo TypeScript y AngularJS para este proyecto y el haber tenido la oportunidad de usar Ionic para desarrollar aplicaciones móviles me ha abierto a un nuevo mundo donde las posibilidades con lenguajes tan básicas como HTML, CSS y JavaScript son infinitas

6.2 Trabajo futuro

- **Modo offline**: Está claro que un sector del público, como con cualquier línea de autobuses, son extranjeros, ya sean residentes en España o turistas. De aquí surgió la necesidad de implementar la aplicación en varios idiomas. Pero en este último caso de los turistas, una funcionalidad que podría ser muy interesante para ellos es la posibilidad de tener un modo *offline* debido a que normalmente no usan la tarifa de datos en el extranjero por los precios tan elevados del *roaming*. Sin acceso a Internet no se podría consultar el tiempo de espera del autobús debido a la necesidad de acceder a la API de Google Maps pero se podría guardar una copia en local de los horarios de los autobuses cuando se active este modo para poder consultar por lo menos los precios y horarios entre los destinos deseados.

- **Servidor y API con Node.js**: Como hemos mencionado varias veces a lo largo de este documento, Firebase ha simplificado mucho la parte del servidor y de la API con tal de mantener el proceso lo más simple posible. Aún así, hay muchos cálculos y llamadas a la base de datos y a la API de Google Maps que

se realizan desde el móvil del conductor, en caso de ser la aplicación Loymerbus GPS o desde la del usuario en el caso de la aplicación principal. Esto conlleva más uso de datos y menos rendimiento. Lo ideal de cara al rendimiento y la experiencia de usuario hubiera sido desarrollar nuestra propia API con Node.js y alojarla en un servidor dedicado a realizar todas las llamadas a APIs externas necesarias y todos los cálculos que hagan falta para que las aplicaciones solo tengan que recibir información y mostrarla.

- **Versión de escritorio**: Uno de los inconvenientes del framework Ionic es que solo sirve para móvil, es decir, si abrimos el enlace https://loymerbus.firebase.com desde un navegador de escritorio, funciona igual, pero se ve todo muy estirado y no queda bien. Sin embargo, si lo abres desde un móvil todos los estilos se muestran correctamente tal y como lo veríamos desde la aplicación. Para las personas que quieran consultar la información que ofrece esta aplicación desde un navegador de escritorio sería útil una versión web de escritorio dividida en las mismas secciones. Se podría desarrollar con simple código HTML, CSS y JavaScript, o incluso con AngularJS.

Bibliografía

[1] Beati, H. (2015). HTML5 y CSS3 para diseñadores. (1ª edición). Argentina: Alfaomega Grupo Editor Argentino

[2] Sawyer McFarland, D. (2014). Programación JavaScript y jQuery. (3ª edición). España: Ediciones Anaya Multimedia

[3] Tomás Gironés, J. (2015). El gran libro de Android. (4ª edición). España: Marcombo

[4] Documentación Ionic 2. https://ionicframework.com/docs/v2/. Agosto 2016

[5] GitHub. https://github.com/. Septiembre 2016

[6] Integrating Firebase with AngularFire2 into AnhgularJS & Ionic 2. http://www.clearlyinnovative.com/integrating-firebase-with-angularfire2-into-angularjs-ionic2. Mayo 2016

[7] Ionic 2 | Integrating Google Maps. http://www.gajotres.net/ionic-2-integrating-google-maps/2/. Enero 2016

[8] GitHub | Cordova Plugin Splashscreen. https://github.com/apache/cordova-plugin-splashscreen. Septiembre 2016

[9] Angular. https://angular.io/. Septiembre 2016

[10] Documentación Firebase. https://www.firebase.com/docs/. Julio 2016

[11] Canal de YouTube de Firebase. https://www.youtube.com/user/Firebase. Septiembre 2016

[12] Documentación de NPM. https://docs.npmjs.com/. Septiembre 2016

[13] Google Maps Distance Matrix API. https://developers.google.com/maps/documentation/distance-matrix. Septiembre 2016

[14] Google Maps Directions Service. https://developers.google.com/maps/documentation/javascript/directions. Agosto 2016

[15] Google Maps Distance Matrix Service. https://developers.google.com/maps/documentation/javascript/distancematrix. Septiembre 2016

[16] Ionic 2 & Angular2 Translate - Internationalize and Localize Your App. https://blog.thecodecampus.de/ionic2-angular2-translate-internationalize-localize-app/. Junio 2016

[17] GitHub | ng2-translate. https://github.com/ocombe/ng2-translate. Septiembre 2016