universidad de león

MEMORIA FINAL DEL ALUMNO

ALUMNO: Alejandro García Fernández			
NIF: 02770973W	**TELEFONO: 693751397**	**E-MAIL: alex1998@gmail.com**	
TITULACIÓN: INGENIERÍA INFORMÁTICA			**CURSO:** 2020/21
FECHA DE INICIO/FIN REALIZACIÓN PRÁCTICAS: 1 ER CUATRIMESTRE 20/21			**HORAS REALIZADAS: 150**
ENTIDAD COLABORADORA: PROCONSI			**LOCALIDAD: León**

IDENTIFICACIÓN DE LAS APORTACIONES QUE, en materia de aprendizaje, han supuesto las prácticas:

He aprendido una nueva forma de crear aplicaciones para dispositivos móviles, de manera que, programando una única vez, esta aplicación funcione de forma nativa en Android como en iOS. También, he visto los plazos y la forma de funcionar de una empresa tecnológica, en el ámbito del mantenimiento y actualización de aplicaciones.

VALORACIÓN PERSONAL: (Indicar vuestra valoración personal, puntos a favor y en contra. Este informe es confidencial, por lo que no dudéis en poner todo lo que queráis...)

En lo personal, me he sentido muy cómodo y arropado por la empresa desde el minuto 0, ya que siempre me han dado todo el apoyo necesario y me han brindado facilidades cuando he encontrado un problema, o me he atascado. También, he visto que, en muchos casos, la autonomía a la hora de recrear una aplicación desde cero es necesaria y a la vez no, ya que, en mi caso, he sido demasiado ambicioso planteando objetivos y funcionalidades.

Por otro lado, siento que no he rendido todo lo que podría haber rendido, ya que ha habido días en los que no avanzaba todo lo necesario, generándome frustración y dejando de trabajar, retrasando la evolución de las aplicaciones.

No obstante, también me han servido para ver cómo es el ser programador, y que puede que ese no sea el puesto de trabajo que deseo.

La Descripción detallada del trabajo (objetivos, tareas desarrolladas, así como la relación entre los problemas planteados y el procedimiento seguido para su resolución) se incluyen en la memoria del trabajo que se adjunta a continuación.

León, a 22 de enero de 2021

Fdo.: Alejandro García Fernández

universidad de león

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

PRÁCTICAS EXTERNAS

APLICACIONES HÍBRIDAS

HYBRID APPLICATIONS

Alejandro García Fernández

(Enero, 2021)

universidad de león

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

RESUMEN:
En este proyecto se ha llevado a cabo la migración parcial de la APP Buskbus, desarrollada por Proconsi, a React Native, permitiendo así la hibridación de esta, para así poder ser compatible tanto con Android como con iOS. Además, facilitará la mantenibilidad de ésta, puesto que React Native ofrece soporte para ambas plataformas simultáneamente, pudiendo también decidir que funciones afectan a un sistema operativo o a otro. Otro propósito de este proyecto es brindar a la aplicación una nueva interfaz, ya que la interfaz actual, pese a ser 100% funcional, se consideró desactualizada estéticamente.

Palabras clave: React Native, Android, iOS, BuskBus, migración

Contenido

1 Introducción.

1.1 Propósito

El propósito de este proyecto es la migración de la APP BuskBus, desarrollada por Proconsi, a React Native, permitiendo así la hibridación de la aplicación, para así poder ser compatible tanto con Android como con IOS. Además, facilitará la mantenibilidad de ésta, puesto que React Native ofrece soporte para ambas plataformas.

1.2 Alcance

El alcance del proyecto es el mismo que tuvo la aplicación BuskBus cuando fue creada, proveer un servicio de información, para los usuarios, del transporte urbano.

1.3 Personal involucrado

Este proyecto ha sido planteado para una sola persona

Nombre	Alejandro García Fernández
Rol	Programador
Categoría	Estudiante en prácticas de Ingeniería Informática
Responsabilidades	Diseño y desarrollo
Información de contacto	alex1998gf@gmail.com

2 Objetivos

El objetivo principal de este proyecto es la migración del código de la aplicación, de los lenguajes en los que se implementó tanto la versión de Android (Java) como la de iOS (Swift), a un lenguaje que permita la hibridación de este, es decir, que el código al compilarse pueda ejecutarse en ambos sistemas operativos, en este caso, React Native.

3 Planificación

En este proyecto se podrían definir dos etapas:

3.1 Aprendizaje con React Native

Durante esta etapa, que comprendió desde el inicio de las prácticas, el día 14 de octubre, hasta el 16 de diciembre, cuando decidí avanzar a la siguiente etapa.

En esta etapa, se me asignó la creación de una aplicación sencilla, en React Native, que solicitará mediante un formulario un listado de canciones a la API de iTunes, y mostrará los datos solicitados. Para el aprendizaje no diseñé una planificación como tal, ya que su desarrollo fue muy ilimitado y al ser solo para aprendizaje no pensé que fuera necesario. Esto fue un error, y por culpa de ello, esta etapa se demoró más de lo pensado.

Llegó un momento, en el que resultaba imposible seguir avanzando, ya que uno de los componentes se implementó erróneamente y para que se pudiera seguir el desarrollo era necesario empezar desde el principio. Por otro lado, estipulé que esta aplicación de aprendizaje había llegado a un punto en el cual, lo que estaba intentando no era necesario para el desarrollo de la aplicación final.

Por tanto, esta aplicación de aprendizaje no fue terminada y fue “abandonada”.

3.2 Desarrollo de la aplicación

A la hora de comenzar con esta etapa, preparé una planificación desde el principio, para así no cometer los errores que aparecieron en la etapa anterior.

3.2.1 Diseño de prototipos

Para no comenzar sin una base, se diseñaron los siguientes prototipos, que fueron realizados a mano por mí.

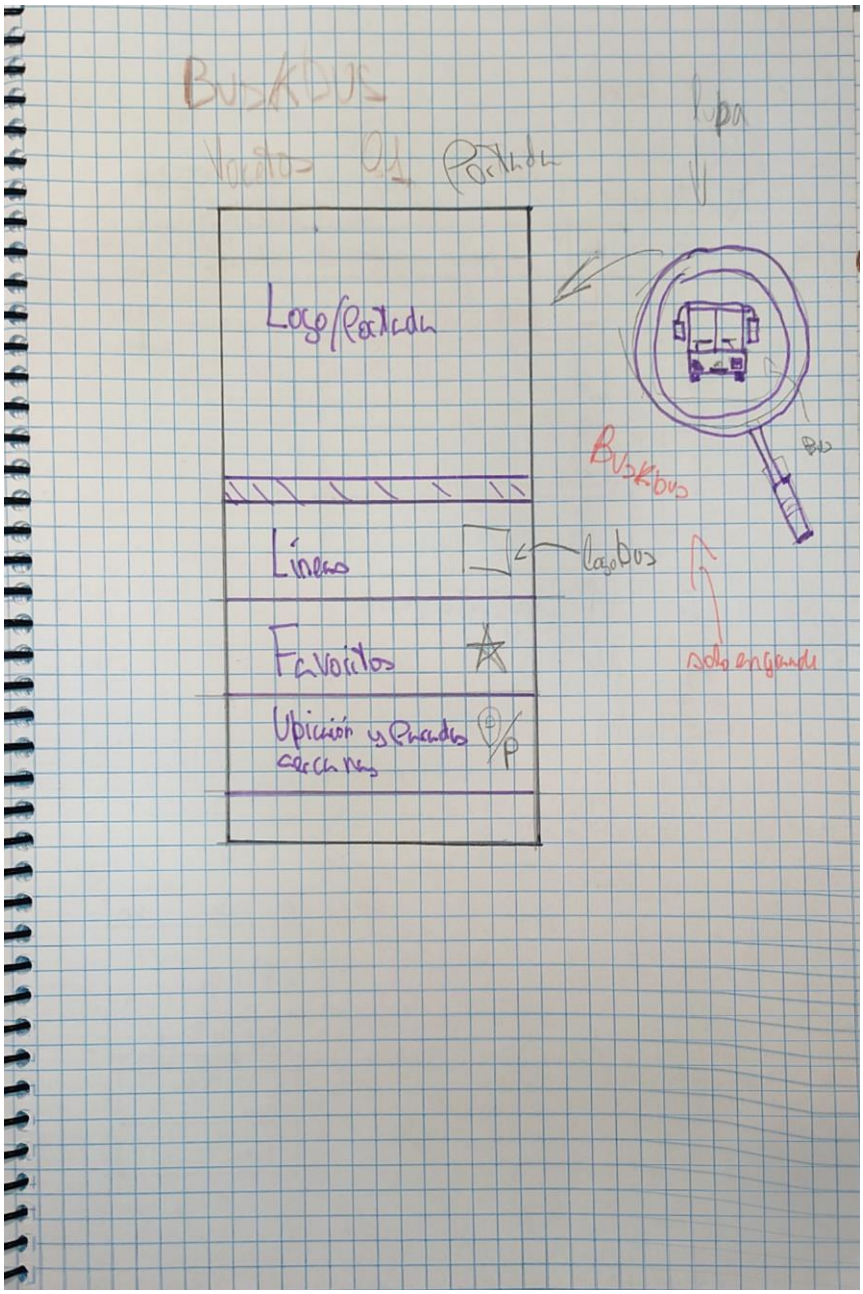

Ilustración 1 Prototipo portada BuskBus

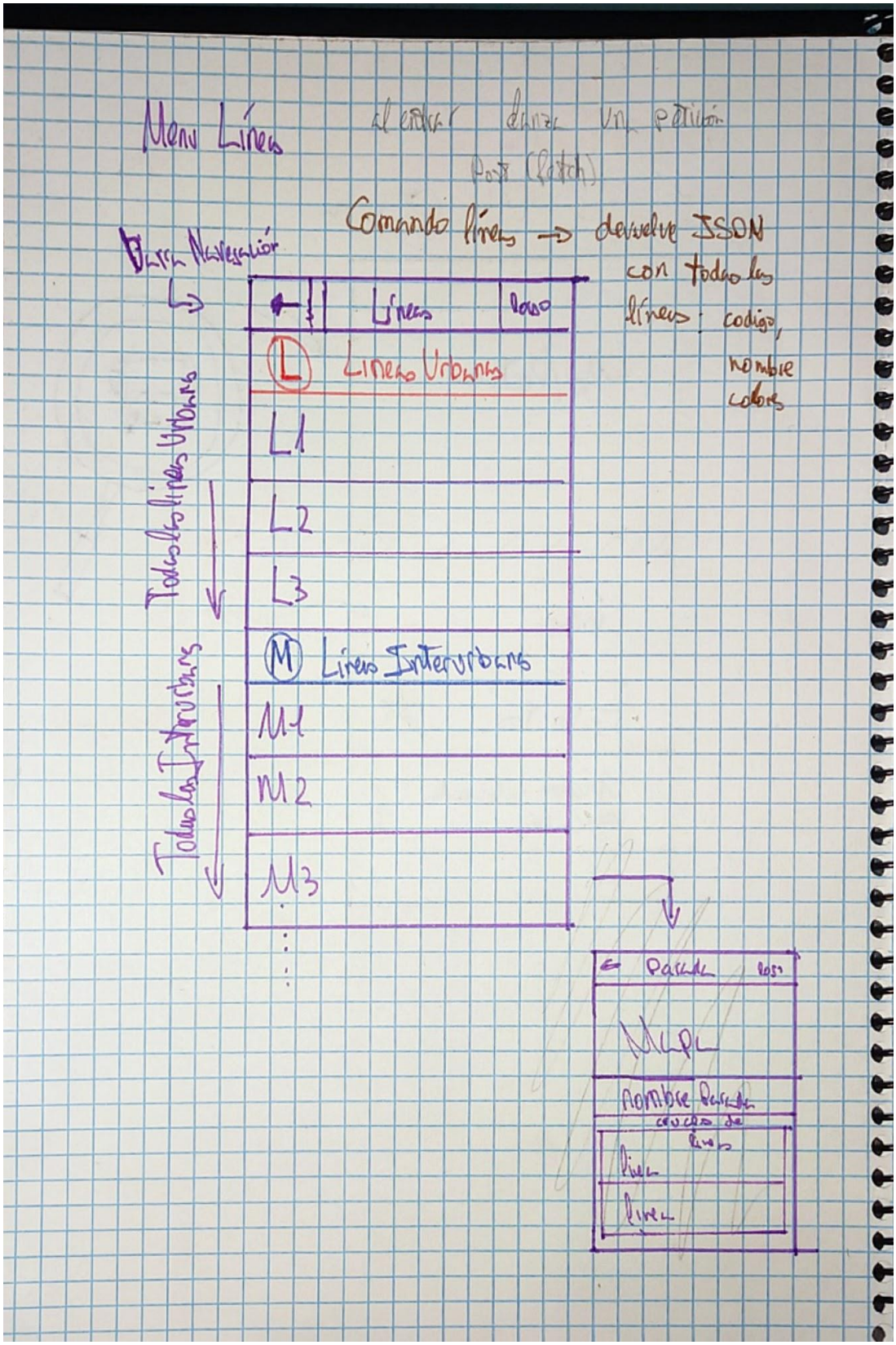

Ilustración 2 Prototipo vista lineas

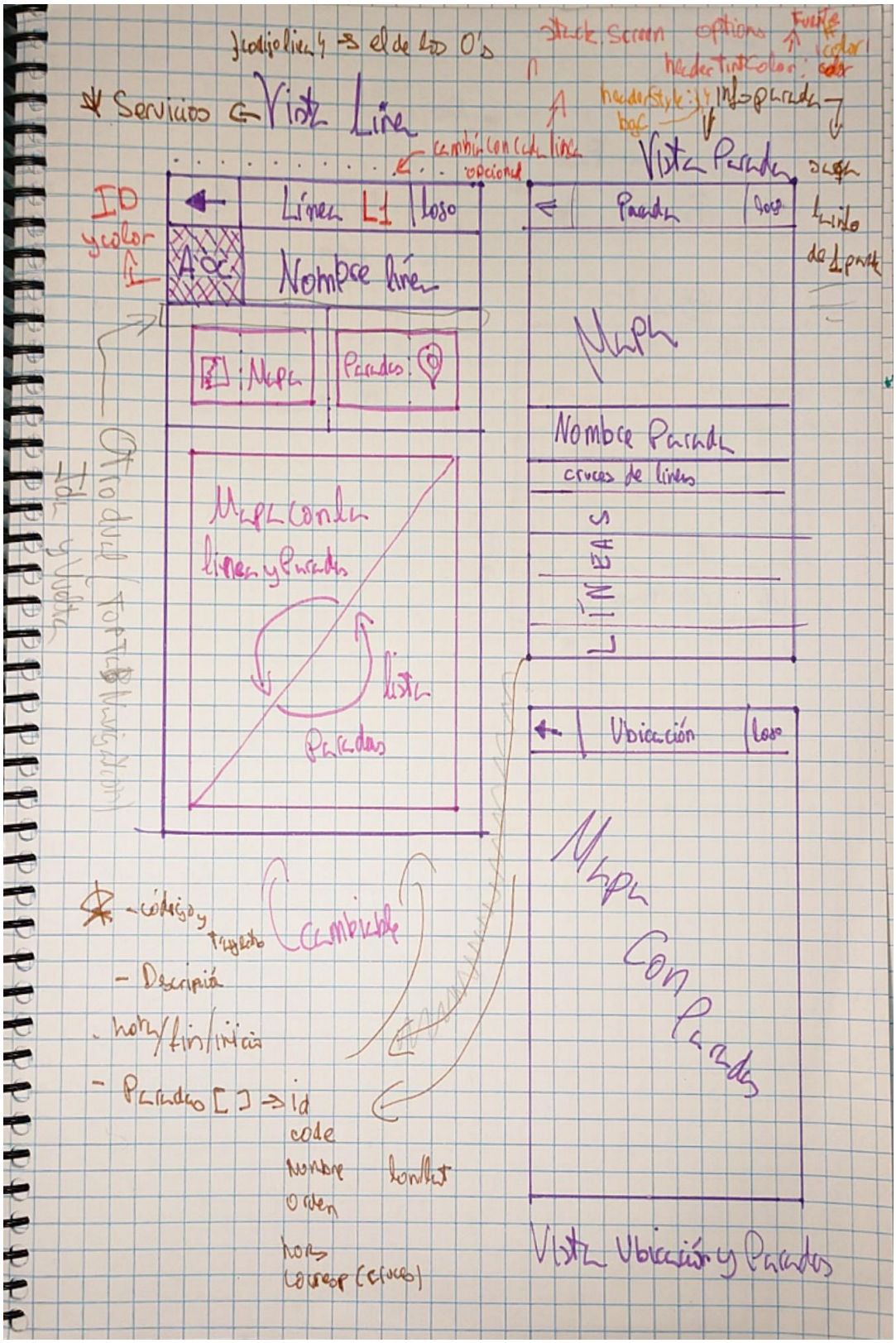

Ilustración 3 Prototipo vista linea, parada y mapa

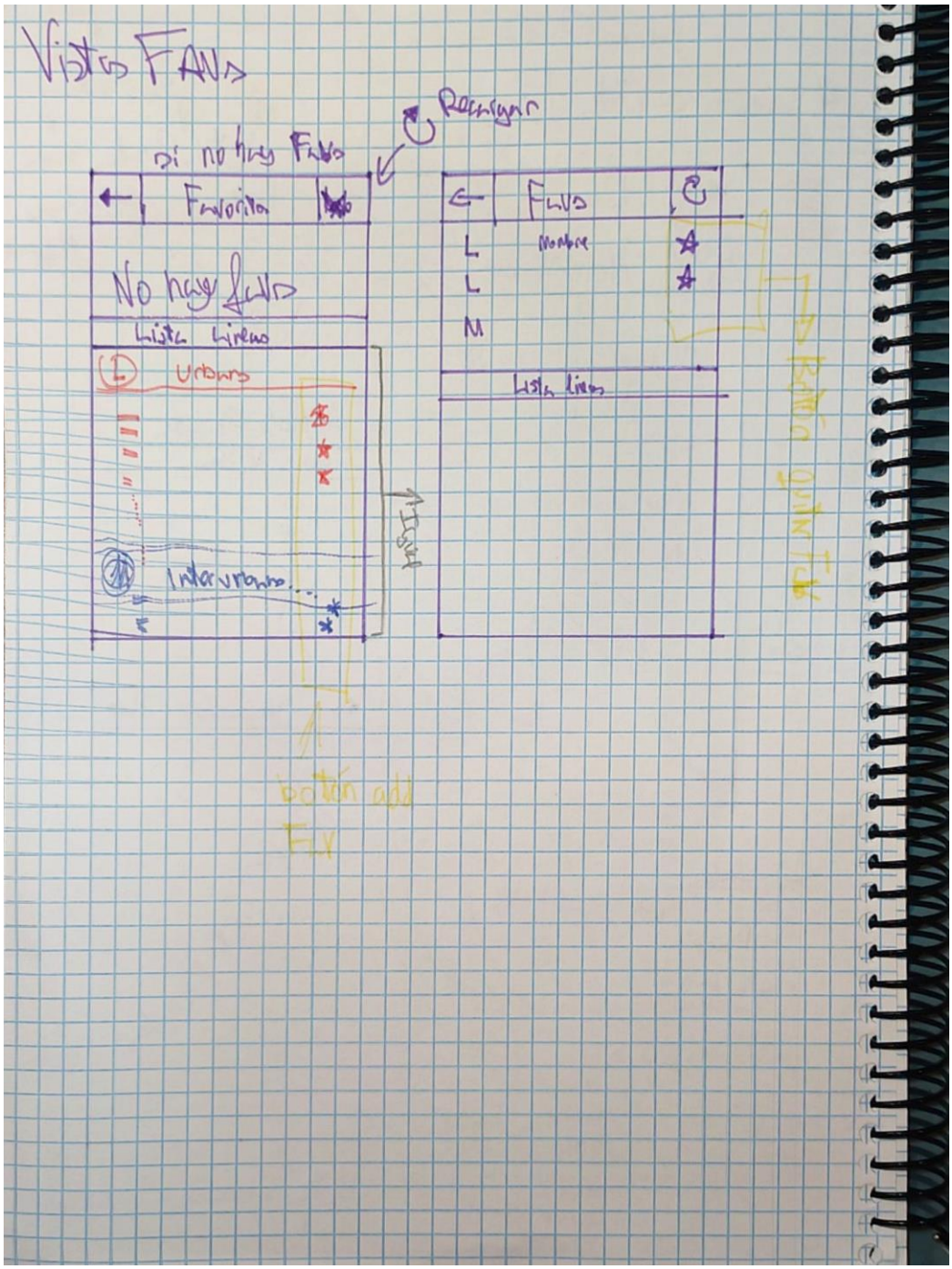

Ilustración 4 Prototipo Vista favoritos

Estos prototipos han sido tenidos en cuenta, pero no se han seguido al 100%, ya que ha habido partes que no se han podido implementar.

3.2.2 Creación de la estructura del proyecto

En esta fase, se estructuró todo el proyecto, es decir, se crearon las carpetas y las clases dejando la siguiente estructura

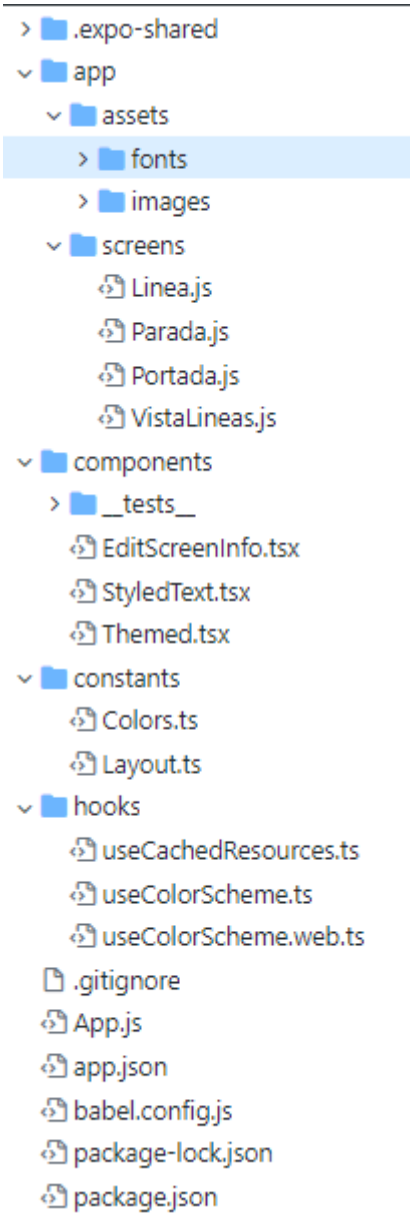

Ilustración 5 Estructura del proyecto

Originalmente se crearon también las clases necesarias para las vistas de Favoritos y de Ubicación actual y paradas cercanas.

Finalmente, no fueron implementadas.

3.2.3 Implementación de prototipos

En esta fase se introdujo el código base necesario para poder visualizar correctamente las peticiones de datos al servidor TimeBus. Este código es sencillo, y simplemente servirá como "chasis" de la vista final.

No he podido añadir capturas de la vista de este "chasis" puesto que el desarrollo avanzó.

3.2.4 Implementación del código funcional de las clases

En esta fase, se implementaron las peticiones de datos al servidor y el muestreo de estas. Las direcciones a las cuales hacer las peticiones fueron proporcionadas por el documento anexo a este. A la hora de realizar y mostrar las mismas, encontré el inconveniente de que, al pedir las paradas, idealmente se deberían ordenar por la ID de la línea, pero este comando devuelve todas las líneas que pasan por esa parada hasta el fin de las mismas, es decir, puede darse el caso de que devuelva 3 marcas de la misma línea, pero de diferentes horas, por tanto la única solución que encontré fue ordenar estas líneas mostradas por la hora a la que pasan. Al ordenar los datos de esa forma, se ven primero todas las pasadas restantes de una línea, luego de otra, y así con todas las líneas que coinciden en esa parada.

3.2.5 Implementación de prototipos II

En esta última fase, las vistas son refinadas para proporcionar una interfaz visual y sencilla para el uso.

4 Núcleo del trabajo.

4.1 Descripción general

4.1.1 Perspectiva de producto

El sistema, en principio, no está diseñado para interactuar con otro sistema informático.

4.1.2 Funcionalidad del producto

En origen, se plantearon las siguientes funcionalidades:

- Consulta de las líneas disponibles
- Consulta de los horarios de una línea
- Consulta de las paradas de una línea
- Consulta de la ubicación de una parada
- Consulta de las paradas cercanas
- Consulta de las líneas cercanas
- Consulta de las líneas que coinciden en una parada
- Búsqueda de paradas por el nombre de la calle en la que se encuentra
- Guardado y consulta de líneas favoritas

De estas funcionalidades, han sido implementadas las siguientes:

- Consulta de las líneas disponibles
- Consulta de los horarios de una línea
- Consulta de las paradas de una línea
- Consulta de la ubicación de una parada
- Consulta de las líneas que coinciden en una parada

4.1.3 Suposiciones y dependencias

Se asume que los requisitos descritos en este documento son estables.
El sistema dependerá de la conexión a internet del dispositivo móvil.

4.2 Requisitos específicos

Número de requisito	RF 01
Nombre de requisito	Consulta del listado de líneas
Tipo	Requisito
Fuente del requisito	Usuario
Prioridad del requisito	Alta/Esencial

Número de requisito	RF 02
Nombre de requisito	Consulta de la información de una línea
Tipo	Requisito
Fuente del requisito	Usuario
Prioridad del requisito	Alta/Esencial

Número de requisito	RF 03
Nombre de requisito	Consulta de la información de una parada
Tipo	Requisito
Fuente del requisito	Usuario
Prioridad del requisito	Media/Deseado
Número de requisito	RF 04
Nombre de requisito	Consulta de las correspondencia de una línea
Tipo	Requisito

Fuente del requisito	Usuario
Prioridad del requisito	Alta/Esencial

Número de requisito	RF 05
Nombre de requisito	Búsqueda de paradas por nombre de calle
Tipo	Requisito
Fuente del requisito	Usuario
Prioridad del requisito	Media/Deseado

Número de requisito	RF 06
Nombre de requisito	Correcto funcionamiento de las peticiones al servidor
Tipo	Requisito
Fuente del requisito	Usuario
Prioridad del requisito	Alta/Esencial

Número de requisito	RF 07
Nombre de requisito	Guardado y consulta de favoritos
Tipo	Requisito
Fuente del requisito	Usuario
Prioridad del requisito	Media/Deseado

Número de requisito	RF 08
Nombre de requisito	Consulta de paradas cercanas
Tipo	Requisito
Fuente del requisito	Usuario
Prioridad del requisito	Alta/Esencial

4.3 Requisitos no funcionales

4.3.1 Requisitos de rendimiento

La app necesitará de una conexión a internet para su correcto funcionamiento.
El tiempo de respuesta en las operaciones debe ser inferior o igual a 30 segundos.

4.3.2 Requisitos de seguridad

No será necesario tomar medidas de autenticación, ya que los datos de las líneas son comunes a todos los usuarios. Para guardar los datos de favoritos, se necesitará dar permiso a la app desde el dispositivo, para que esta pueda almacenar datos en la memoria de este.

4.3.3 Requisitos de fiabilidad

Al detectar un problema, antes de repararlo, se evaluará la gravedad de este. Esto puede deberse a fallos del servidor, cuyo tiempo de reparación es ajeno al proyecto, fallos en petición, cuyo tiempo de reparación será un par de horas, o fallos en interfaz, que tendrán un tiempo de reparación de entre 1 a 3 días, en función de la gravedad de este.

4.3.4 Requisitos de disponibilidad

Esta app estará disponible las 24h del día, ya que apenas requiere recursos. No obstante, y dado que no se ofrecen servicios de transporte urbano en horario nocturno se recomienda cargar actualizaciones o reparaciones del sistema en horario nocturno.

4.3.5 Requisitos de mantenibilidad

Las labores de mantenimiento serán realizadas por los responsables de la aplicación, o por personal cualificado de la empresa propietaria.

4.3.6 Requisitos de portabilidad

Puesto que React Native proporciona soporte para los dos sistemas operativos principales para telefonía móvil, no se plantea realizar otro tipo de portabilidad. Pendiente quedaría una compatibilidad con Harmony OS, de Huawei, aunque este se espera que corra aplicaciones Android sobre una máquina virtual, garantizando la portabilidad.

4.4 Arquitectura de la aplicación.

En este caso, la arquitectura utilizada sigue el patrón Cliente-Servidor.
El **servidor** al que se realizan las peticiones de datos es TimeBus, el cual es propiedad de Proconsi. Ofrece, entre otras opciones, información en tiempo real sobre las líneas de bus de la ciudad/municipio indicada en la petición, siempre que este se encuentre cargado en él.
El **cliente** está construido sobre EXPO en React Native. Expo es el complemento ideal de React Native, ya que permite ejecutar las apps en cualquier dispositivo (si tiene la app de expo instalada) sin necesidad de instalar Android Studio o Xcode.
A Expo y React Native, sumamos los siguientes componentes:

- React Navigation
- React Native Maps
- React Native Paper

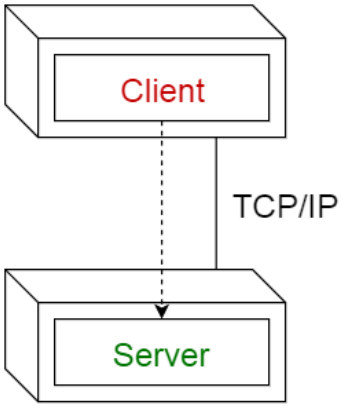

Ilustración 6 Arquitectura Cliente Servidor

El entorno de desarrollo ha sido Visual Studio Code, y el entorno de pruebas ha sido una Android Virtual Device, de Android Studio, configurada para emular un Google Pixel 2 con Android 10 Q

5 Resultados

La portada de la aplicación sería la siguiente.

Para poder tener una aproximación de como quedaría se diseño el logo que se ve en la parte superior. La idea de ese logo surge del simple razonamiento del nombre "BuskBus" == Buscar (lupa) Bus (autobús)

En ella, se aprecian 3 acciones. Líneas, que muestra las líneas de la ciudad asignada. Favoritos, que muestra las paradas favoritas del usuario, pudiendo añadirlas desde la línea (esta función no se ha implementado). Paradas cercanas, que muestra las paradas cercanas a la ubicación del usuario (esta función no se ha implementado).

Para las funciones no implementas, se muestra la siguiente alerta:

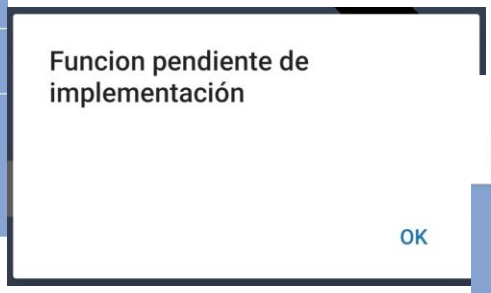

Ilustración 7 Portada aplicación

Ilustración 8 Alerta

Para la visualización de las líneas se plantea la siguiente vista.

En ella se muestra, en una lista dinámica, las líneas disponibles para la ciudad o municipio que se estipule.

La forma de mostrar estos datos es mediante un componente "Flatlist" de React Native, y un componente "List.Item" de la librería React Native Paper, el cual recibe tanto el ID de la línea, como su Código de panel, su color asignado, su nombre…

El componente se muestra centrado en la pantalla.

Ilustración 9 Vista lineas

Para mostrar la información de la línea seleccionada se implementó la siguiente vista.

En ella se muestra, en el caso de que la línea seleccionada contenga solo una ruta, es decir, sea de un único bus, en la pantalla se mostrarán los datos de la siguiente manera.

Inicialmente, en lo alto, se muestra el nombre de la línea, con su color correspondiente. Susto debajo, se muestra mediante dos elementos Button, de la API React-Native-Paper, los cuales al ser pulsados uno u otro, cambian el estado de una variable booleana, cambiando la vista, de la lista de paradas al mapa de la línea.

Para el resto de la explicación usaremos una imagen de una línea con 2 rutas.

Ilustración 10 Vista listado paradas

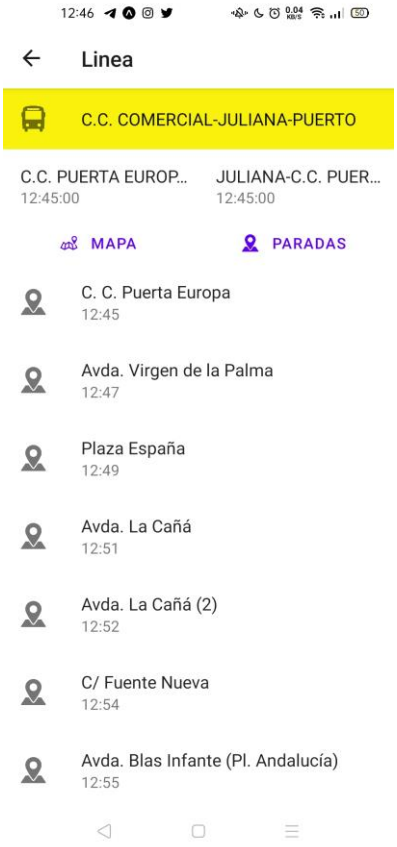

Ilustración 11 Vista listado paradas

Para mostrar las líneas de dos rutas se añade entre el elemento que muestra el nombre de la línea, y los botones que cambian la vista entre la lista de paradas y el mapa, dos elementos de tipo List.Item, también de React-Native-Paper, que alterna la vista completa entre una ruta y otra.

Para el caso del mapa, se utiliza la API React-Native-Maps para mostrar los datos en un mapa, en este caso tomado de Google, que muestra la ubicación concreta mediante longitud y latitud. Estos datos se muestran mediante elementos Marker de la API mencionada anteriormente. Cabe destacar que, al pulsar en cualquiera de estos marcadores, se mostrará el nombre de la parada y la hora de paso estimada del autobús.

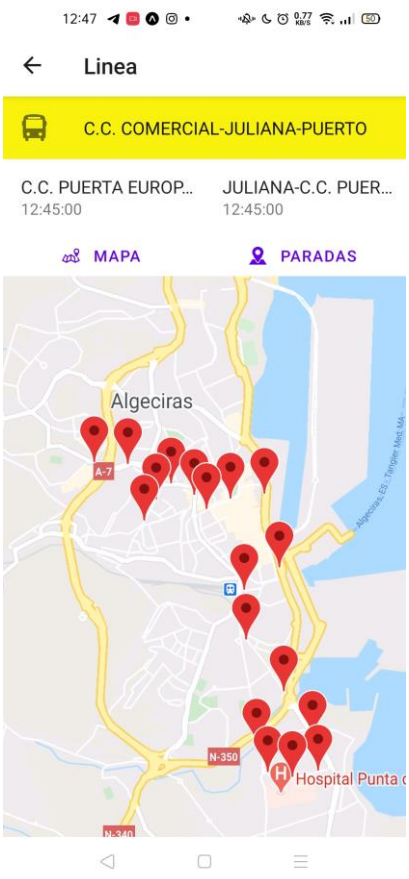

Ilustración 13 Vista Mapa

Ilustración 12 Vista Mapa

En el caso de que no se encuentre disponible esa línea se mostrara la siguiente imagen.

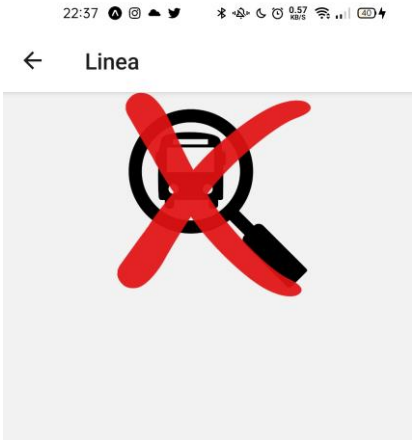

Ilustración 14 Vista Linea no disponible

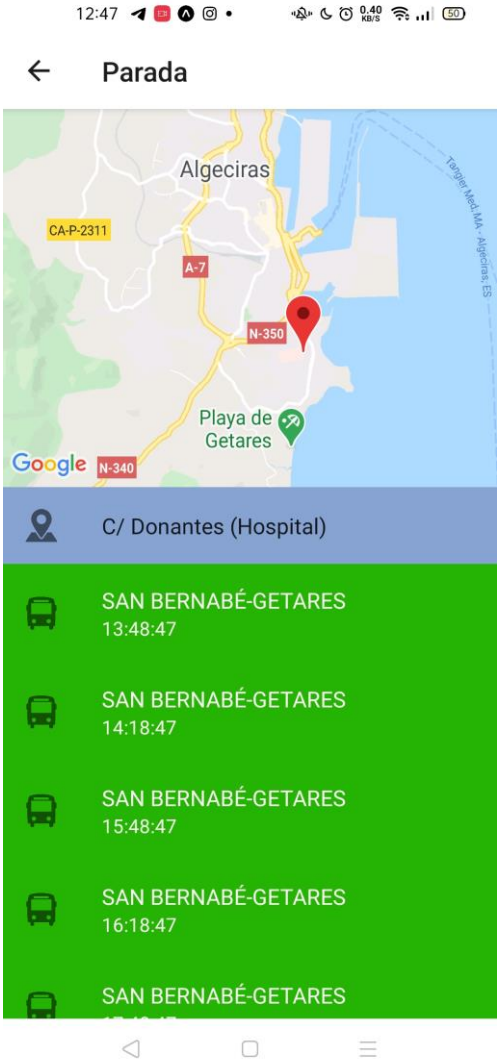

Ilustración 15 Vista parada

En el caso de acceder a una parada, se muestra primero el mapa ubicación de la parada. Bajo este, el nombre de la parada y, debajo de este, mediante una lista los datos de las correspondencias entre líneas por orden de líneas y cronológico.

6 Conclusiones

A lo largo de este documento, hemos visto el desarrollo, en este caso parcial, de la migración del código de la aplicación BuskBus, de los lenguajes Java y Swift a React Native. Para facilitar este desarrollo, se han utilizado los componentes propios de React Native, añadiendo a estos tanto React Navigation, que facilita la navegación entre vistas React Native Paper, para la mejora visual de las vistas y React Native Maps, para la visualización de los mapas.

Esta aplicación, pese a ser sencilla en diseño, ofrece muchas posibilidades ya que, cuando esté completa, no solo se podrá obtener la información de las líneas y paradas, también se podrá tener un listado de paradas favoritas para facilitar un seguimiento de estas, un buscador de paradas por nombre para facilitar precisamente la búsqueda de paradas; y un buscador de paradas por ubicación, es decir, las paradas más cercanas disponibles.

7 Bibliografía y referencias

www.youtube.es

https://reactnative.dev/

https://reactnavigation.org/

https://callstack.github.io/react-native-paper/

https://github.com/react-native-maps/react-native-maps

https://www.npmjs.com/

https://stackoverflow.com/

Anexos:

API BuskBus v2.pdf
Bus.docx