



MÓDULO PROYECTO

CFGS Desarrollo de Aplicaciones
Multiplataforma
Informática y Comunicaciones

<BussAPI>

Tutor individual: <Cristina Silván Pardo>

Tutor colectivo: <Cristina Silván Pardo>

Año: <2024>

Fecha de presentación: (27/05/2024)

Nombre y Apellidos: Álvaro Martín Gómez
Email: alvaro.margom@educa.jcyl.es

**Foto actual
del alumno**

Tabla de contenido

1	Identificación del proyecto	4
2	Organización de la memoria	4
3	Descripción general del proyecto	6
3.1	Objetivos	6
3.2	Cuestiones metodológicas	6
3.3	Entorno de trabajo (tecnologías de desarrollo y herramientas).....	6
4	Descripción general del producto.....	8
4.1	Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.....	8
4.2	Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.	10
4.3	Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha.....	10
5	Planificación y presupuesto	12
6	Documentación Técnica: análisis, diseño, implementación y pruebas.	12
6.1	Especificación de requisitos	12
6.2	Análisis del sistema	13
6.3	Diseño del sistema:	14
6.3.1	Diseño de la Base de Datos	14
6.3.2	Diseño de la Interfaz de usuario.	14
6.3.3	Diseño de la Aplicación.	17
6.4	Implementación:	17
6.4.1	Entorno de desarrollo.	17
6.4.2	Estructura del código.	19
6.4.3	Cuestiones de diseño e implementación reseñables.....	20
6.5	Pruebas.....	20
7	Manuales de usuario	24
7.1	Manual de usuario.....	24

7.2	Manual de instalación	40
8	Conclusiones y posibles ampliaciones	40
9	Bibliografía	40
10	Anexos	41

1 Identificación del proyecto

(Portada)

2 Organización de la memoria

DESCRIPCIÓN GENERAL DEL PROYECTO:

En este apartado se realiza una pequeña descripción sobre el proyecto realizado

- OBJETIVOS: Breve descripción de lo que se ha pretendido alcanzar con el proyecto.
- CUESTIONES METODOLÓGICAS: Forma de realizar el proyecto, metodología de desarrollo que se ha seguido.
- ENTORNO DE TRABAJO: Tecnologías, herramientas de software y lenguajes de programación utilizados.

DESCRIPCIÓN GENERAL DEL PRODUCTO:

- VISIÓN GENERAL DEL SISTEMA: Definición de los límites del sistema, funciones básicas etc.
- DESCRIPCIÓN BREVE DE MÉTODOS Y ARQUITECTURAS: Breve resumen de técnicas utilizadas, incluyendo la justificación de elección.
- DESPLIEGUE DE LA APLICACIÓN: Puesta en marcha de la aplicación, indicando plataforma tecnológica, descripción del proceso de instalación etc.

PLANIFICACIÓN Y PRESUPUESTO:

Tiempo aproximado que ha llevado cada tarea, estructurando esta información por meses y actividades, además del desarrollo del código, coste del software, herramientas utilizadas etc.

DOCUMENTACIÓN TÉCNICA:

- ANÁLISIS DEL SISTEMA: Especificación de requisitos funcionales de la aplicación
- DISEÑO DEL SISTEMA:
 - DISEÑO DE LA BASE DE DATOS: Modelo E / R, creación del esquema de la base de datos etc.
 - DISEÑO DE LA INTERFAZ DE USUARIO: Diagrama de las pantallas y flujo de estas.

- DISEÑO DE LA APLICACIÓN: Enumeración detallada de todos los procesos seguidos de la aplicación.
- IMPLEMENTACIÓN:
 - ENTORNO DE DESARROLLO: Herramientas utilizadas exclusivamente en programación.
 - ESTRUCTURA DEL CÓDIGO: Librerías descargadas, orientación del código etc.
 - CUESTIONES DE DISEÑO E IMPLEMENTACIONES RESEÑABLES: Descripción de elementos reseñables en el diseño o en la implementación del código.
 - PRUEBAS: Descripción de las pruebas realizadas al finalizar la aplicación.

MANUALES DE USUARIO:

- MANUAL DE USUARIO: Manual descriptivo de la interfaz de la aplicación, indicando pantallas, ítems, flujo entre pantallas ...
- MANUAL DE INSTALACIÓN: Requisitos básicos para la instalación.

CONCLUSIONES Y POSIBLES AMPLIACIONES:

Conclusiones que se pueden sacar en función del planteamiento inicial, descripción de las dificultades importantes en el proceso, así como las posibles ampliaciones (si la aplicación es susceptible de ampliar) con su respectiva descripción.

BIBLIOGRAFÍA:

Bibliografía y Webgrafía.

ANEXOS:

Inclusión de los anexos.

3 Descripción general del proyecto

3.1 Objetivos

El objetivo principal del desarrollo de este proyecto es ser capaz de realizar una aplicación móvil con un lenguaje de programación nunca visto hasta ahora, conociendo las bases de dicho lenguaje, su entorno de desarrollo, su despliegue y mantenimiento etc.

3.2 Cuestiones metodológicas

La metodología utilizada para realizar este proyecto ha sido ‘ciclo de vida en cascada’.

Las razones por las que he usado esta metodología son (entre otras):

- Estructura clara y lineal: Este modelo ofrece una estructura fácil de entender, donde cada fase se ejecuta de manera secuencial, es decir, se definen los requisitos de todas las fases hasta la entrega del producto final.
- Requerimientos claros desde el principio: Aunque desde un inicio no se tengan completos y detallados los requisitos del proyecto, este modelo te ofrece la posibilidad de cambiarlos significativamente durante el desarrollo.
- Control de calidad temprano: Dado que cada fase debe completarse antes de pasar a la siguiente, el modelo en cascada permite un control de calidad temprano en el proceso de desarrollo, o lo que es lo mismo, que los problemas y errores pueden detectarse y corregirse en etapas tempranas, lo que puede reducir el riesgo de problemas mayores más adelante

3.3 Entorno de trabajo (tecnologías de desarrollo y herramientas)

- Tecnologías: Para el desarrollo de esta aplicación se ha seguido un modelo cliente-servidor, ya que se necesita conexión a servidores externos para obtener datos actualizados, enviar y recibir notificaciones, autenticación de usuarios etc.
- Herramientas software y lenguajes de programación: Todo el software utilizado para el desarrollo de esta aplicación es de código libre, accesible para cualquier persona.

La principal herramienta usada en este proyecto ha sido Node.js, un entorno de ejecución de JavaScript que permite ejecutar código JavaScript al lado de un servidor.

Node utiliza destinos administradores de paquetes que permiten instalar, administrar y compartir paquetes de código JavaScript reutilizable. En este proyecto se ha utilizado ‘npm’.

La siguiente herramienta usada en este proyecto es el IDE Visual Studio, lugar donde se desarrolla toda la aplicación.

Además, se usan servicios externos, como la conexión a dos APIs y la conexión a una base de datos externa.

Las APIs utilizadas son:

- TFL: Esta API proporciona información sobre toda la red de transporte público de la ciudad de Londres, rutas de autobuses, paradas, puntos de origen y destino de puntos accesibles al transporte público ...
La razón de por qué se ha seleccionado esta API es porque después de muchos días de búsqueda por diferentes sitios web, la API pública relacionada con el transporte público que más se recomendaba para una posible implementación en aplicación móvil o página web era esta, por su acceso a datos en tiempo real, con información detallada sobre el transporte y, sobre todo, por la fácil integración con otros servicios, tanto internos como externos.
- OpenweatherMap: Esta API proporciona datos sobre el tiempo, temperatura, humedad ... de un lugar en concreto. La razón por la que se ha usado esta API es la misma que la razón de por qué he usado la API de TFL, la más recomendada.

Otra herramienta utilizada es Expo Go, que es una aplicación móvil desarrollada por Expo, que actúa como un cliente para ejecutar y previsualizar proyectos de React Native creados con la plataforma Expo. Además, permite a los desarrolladores ver sus aplicaciones en tiempo real en dispositivos móviles IOS y Android durante el proceso de desarrollo. Se recomienda el uso de esta aplicación para el desarrollo de proyectos de React Native si no se tiene dominado el lenguaje, ya que hacer proyectos con React Native CLI (forma nativa de realizar proyectos React Native) puede resultar complicado si no se ha trabajado nunca con el lenguaje.

La siguiente herramienta utilizada es Firebase, en concreto Firebase Realtime Database, una base de datos NoSQL en la nube que permite a los desarrolladores almacenar y sincronizar datos en tiempo real entre usuarios en dispositivos móviles y web.

Las razones de por qué se ha usado Firebase en la realización de este proyecto son varias, como por ejemplo el rápido desarrollo de prototipos, su facilidad de uso, la escalabilidad, pero sobre todo sus actualizaciones en tiempo real.

El lenguaje de programación elegido para desarrollar este proyecto ha sido React Native.

React Native hereda de React, siendo React una librería de JavaScript.

Se podría decir que para poder desarrollar sin dificultades un proyecto en React, se necesita tener conocimientos de JavaScript.

React y React Native pueden parecer similares, pero la diferencia es que React está pensado para realizar interfaces de usuario y React Native para desarrollar aplicaciones móviles.

El por qué se escogió este lenguaje es muy sencillo, en la empresa donde he realizado las prácticas FCT he aprendido React y React Native, por lo que se decidió que, ya que se va a programar en ese lenguaje en la empresa, por qué no aprovechar y hacer el proyecto final con ese mismo lenguaje, siendo una oportunidad para poder practicarlo más y mejor.

React Native tiene una propia sintaxis, pero, aun así, al ser heredado de React y éste a sí mismo de JavaScript, te permite programar en TypeScript y TypeScript JSX, ambos heredados también de JavaScript, pero en este caso se ha utilizado React Native con JavaScript.

4 Descripción general del producto

4.1 *Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.*

LÍMITES DEL SISTEMA:

La aplicación desarrollada es una aplicación autónoma, es decir, no necesita interactuar con aplicaciones o sistemas externos para funcionar.

Lo único que necesita la aplicación para que sus funcionalidades básicas se puedan ejecutar correctamente es la conexión a 2 APIs y a una base de datos externa, pero como tal no depende de otra aplicación aparte para interactuar.

FUNCIONALIDADES BÁSICAS DEL SISTEMA:

La finalidad principal de la aplicación es proporcionar a los usuarios información útil y en tiempo real sobre el transporte público de la ciudad de Londres y alrededores.

Entrando más en detalle, dentro de la aplicación los usuarios podrán buscar las paradas de la ciudad de Londres por su nombre (se muestran en forma de tarjetas), proporcionando información como el tipo de transporte, su localización en coordenadas y el identificador numérico de dicha parada.

Si la búsqueda no obtiene resultados, la aplicación notificará a los usuarios de que no se han encontrado resultados para esa parada introducida.

Además, la aplicación proporciona información acerca de los próximos autobuses que llegan a una parada en concreto, mostrando información como la hora de llegada aproximada de dicho autobús a la parada, la ruta que seguirá dentro de la ciudad, el punto de origen desde que sale el autobús (siendo éste la parada buscada) y el destino de dicho autobús.

Esta información también se mostrará a modo de tarjetas, siendo útil para una mejor comprensión de los datos por parte del usuario.

Siguiendo con las tarjetas mencionadas anteriormente, los usuarios podrán añadir la tarjeta a favoritos, para cada vez que quieran ver esa tarjeta en concreto no tener que buscarla de nuevo y tenerla guardada aparte, siendo una especie de acceso directo a dicha tarjeta.

Cada tarjeta dispondrá de una opción para compartir la información de dicha tarjeta a quién sea, teniendo diferentes opciones como por correo, WhatsApp ... como el usuario prefiera.

Al igual que con la ventana anterior, si se realiza una búsqueda de una parada que no existe, la aplicación también avisará a los usuarios de que no se han obtenido resultados, aunque no solo avisará de eso, sino que, si se han realizado muchas peticiones a la API, es decir, se han realizado muchas búsquedas de paradas en muy poco tiempo, la aplicación mostrará un mensaje informando de tal suceso, por lo que se deberá esperar un tiempo hasta que se puedan volver a realizar búsquedas.

Haciendo referencia a la ventana de favoritos mencionada anteriormente, aparte de poder ver las tarjetas que el usuario ha ido añadiendo, también se pueden eliminar si se desea, pudiendo actualizar la lista de favoritos para ver las tarjetas que siguen añadidas a la ventana.

La aplicación también cuenta con una opción de buscar las diferentes rutas que siguen los autobuses entre dos puntos determinados, proporcionando información como la hora estimada de salida y llegada al destino, el tiempo aproximado que tardaría el autobús en llegar de punto a punto y las instrucciones para coger ese autobús a lo largo de su recorrido, como podría ser caminar hasta un punto determinado.

Esta información también se muestra en forma de tarjetas, donde cada una tendrá una opción de ver dichos puntos representados en un mapa 2D, donde habrá un marcador de diferente color exactamente en las coordenadas de los puntos de origen y destino.

Si se introducen dos nombres de lugares que no tienen rutas asociadas uno con el otro, la aplicación también notificará a los usuarios de que no hay rutas entre esos dos puntos introducidos.

USUARIOS:

Por otro lado, la aplicación incluye una autenticación de usuarios para poder gestionar qué usuario interactúa con la aplicación. Esto se hace mediante una ventana de registro y otra de inicio de sesión si el usuario ya tiene cuenta.

La forma de autenticar es muy sencilla, si el usuario ya tiene cuenta, deberá inicial sesión con las credenciales de esa cuenta. Si esos datos están guardados en la base de datos, el usuario accederá correctamente a la aplicación. Si los datos introducidos no son correctos, la aplicación mostrará un mensaje de error indicando dicha situación.

Si no tiene cuenta, deberá registrarse en la aplicación creándose una nueva cuenta en la base de datos.

Para crearse una cuenta, el usuario deberá acceder a la ventana de registro, donde se deben rellenar todos los campos para que el registro sea válido.

Si algún campo se deja en blanco, la aplicación notificará al usuario de la obligación de rellenar todos los campos.

A raíz de esto último, dentro de la aplicación, se proporciona una opción para que el usuario pueda actualizar los datos sin necesidad de volver a entrar en la aplicación. Se pueden modificar todos los datos menos el correo, debido a que es el campo con el que se recogen diferentes datos del resto de funcionalidades de la aplicación, si el correo cambia, las tarjetas asociadas a ese correo ya no serán las mismas, pueden surgir problemas si un usuario se registra con el correo previo a la modificación y le aparezcan en la ventana de favoritos tarjetas que no ha seleccionado.

Cabe recalcar que esta aplicación podría interactuar con cualquier SO (Multiplataforma).

4.2 Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.

Este apartado está referenciado en los puntos 3.2 Y 4.1

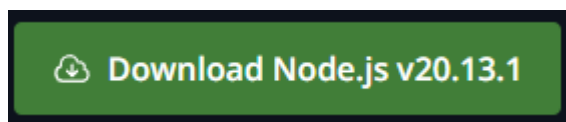
4.3 Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha

PLATAFORMA TECNOLÓGICA:

El sistema operativo en el que se ha realizado la aplicación ha sido Windows 11, aunque la aplicación está pensada que se pueda desplegar en varios sistemas operativos, como MAC o diferentes versiones de Windows (Windows 10, 11 o Windows 11 Pro).

INSTALACIÓN DE LA APLICACIÓN:

Para que esta aplicación se pueda ejecutar se necesita instalar Node.js. Para ello se accederá a la página web oficial (<https://nodejs.org/en/download>) y se pulsará en el botón verde:



Una vez pulsado, se descargará un archivo ejecutable, donde requerirá permisos de administrador para que se instale correctamente.

Dentro del ejecutable habrá varias ventanas en las que el usuario deberá aceptar condiciones de uso, entre otras cosas; se aceptarán todas las políticas de Node.js y se pulsará siempre el botón de siguiente en las ventanas que aparezca dicho botón.

Una vez instalado Node.js en el ordenador, también se han instalado los administradores de paquetes como 'npm' o 'yarn', pero ahora hay que instalar Expo para que facilite el desarrollo del proyecto.

Para instalar Expo simplemente habrá que abrir un terminal (cmd o Windows PowerShell) y 'navegar' hasta la ruta del proyecto:

```
C:\Users\Alvaro\Documents\REACT\primer_proyecto>
```

Una vez se esté en la ruta de deberá ejecutar un comando en concreto para instalar Expo, este comando es el siguiente:

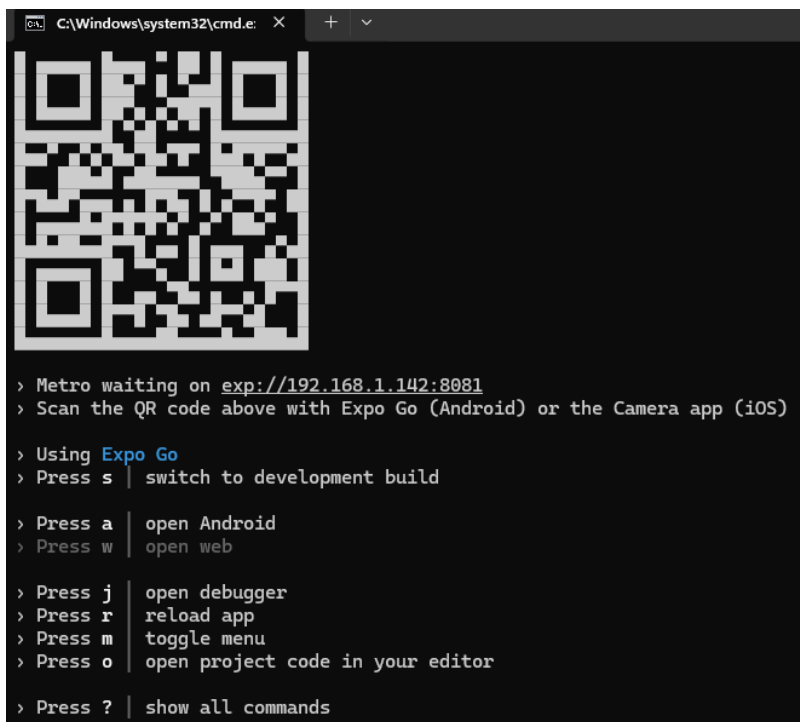
```
npm install -g expo-cli
```

Ya ejecutado este comando, se instalará Expo en nuestro ordenador, por lo que ya se puede ejecutar la aplicación.

Para ejecutar la aplicación se necesita escribir un último comando en el terminal, este comando es el siguiente:

```
C:\Users\Alvaro\Documents\REACT\primer_proyecto>npm start
```

Después de ejecutar el comando anterior, el terminal te preguntará diferentes formas de lanzar la aplicación. Proporcionará un código QR para leer desde la aplicación de Expo (la aplicación de Expo tiene una opción para leer QR) o diferentes opciones para elegir como:



En este caso, se pulsará la tecla ‘a’ para que se lance la aplicación en un ‘device’ o en un móvil físico (se necesita tener las opciones de modo desarrollador activadas).

Una vez pulsada la tecla ‘a’, se lanzará la aplicación.

5 Planificación y presupuesto

PLANIFICACIÓN:

La planificación de este proyecto se ha dividido en diferentes apartados:

- DISEÑO DE LA APLICACIÓN: La duración aproximada de este apartado ha sido de unas 15 horas, aunque al utilizar un modelo de ciclo de vida en cascada, se han podido añadir cambios y mejoras en la aplicación a medida que se desarrollaba dicha aplicación sin que repercuta en el resto del proyecto, por lo que este proceso puede superar las 30 horas.
- POCESO DE EJECUCIÓN: Con diferencia es el apartado que más tiempo ha llevado realizar. Este proceso se ha realizado en 80 horas aproximadamente.

En este proceso se ha desarrollado toda la funcionalidad de la aplicación, desde la conexión a las APIs y base de datos, como con las funcionalidades internas de la aplicación (navegación entre ventanas, acciones de los botones al ser pulsados...)

Dentro de las conexiones a los diferentes servicios externos, la conexión a las APIs ha llevado mucho más tiempo que la conexión a la base de datos, ya que había se necesita conocer como ‘devuelve’ la API la información para saber implementarla en la aplicación.

- PERIODO DE PRUEBAS: Este proceso ha sido el más corto, unas 3 horas aproximadamente, comprobando que cada funcionalidad de la aplicación funcione correctamente y no aparecen errores en la interfaz.

PRESUPUESTO:

Haciendo un resumen incluyendo el coste del software, hardware, hosting y desarrollo del código según las horas dedicadas y precio por hora del programador, teniendo en cuenta que es salario medio de un programador de React Native es de unos 17,50 euros / hora y que el tiempo que se ha tardado en realizar esta aplicación ha sido de más o menos unas 120 horas, el coste total sería de unos 2100 euros.

6 Documentación Técnica: análisis, diseño, implementación y pruebas.

6.1 Especificación de requisitos

- Se ajusta a los requerimientos formales establecidos: Portada, índice estructurado, extensión (mínimo 40 páginas), buena redacción, sin faltas de ortografía.

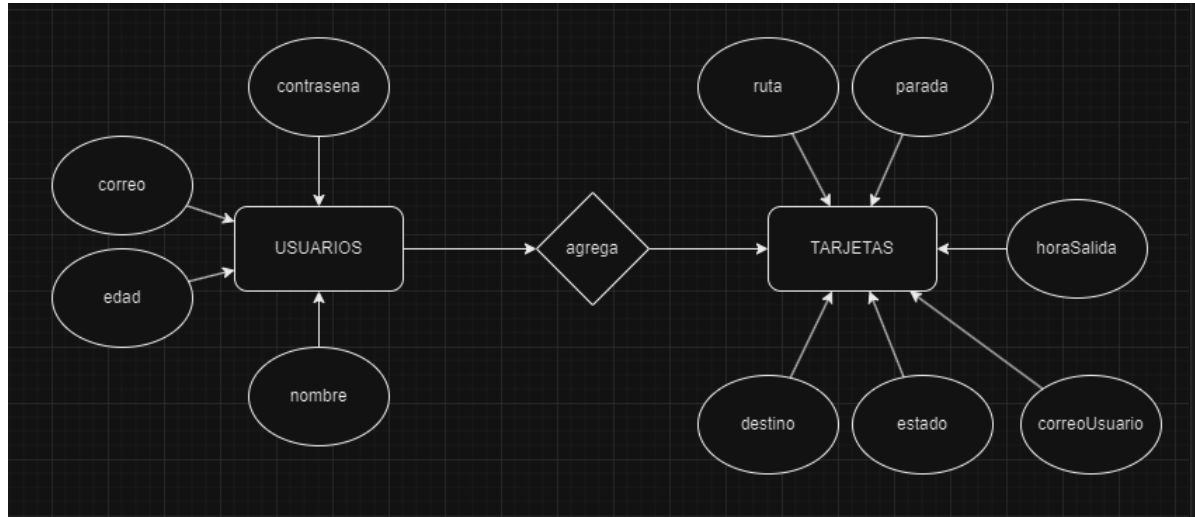
- Se incluyen conclusiones finales sobre el proyecto e investigación realizada, posibles ampliaciones y bibliografía/webgrafía.
- Presenta el proyecto en la forma y plazo establecidos.
- Originalidad del tema elegido (los contenidos no han sido vistos en clase o completan los vistos en clase)
- Grado de dificultad en orden a la investigación de contenidos (es un proyecto que abarca el contenido de diferentes módulos vistos en clase, ...)
- Grado de actualidad del tema elegido (el tema está relacionado con tecnologías actuales y vanguardistas)
- En el caso de una aplicación mobile/web: consume datos de un servicio externo (conexión BBDD o a través de API), implementación CRUD, indica modelo de la base de datos, autenticación de usuarios En caso de otro tipo de aplicación (IA, videojuego...) documenta convenientemente todo el proceso de gestión de datos.
- La aplicación es totalmente funcional, se controlan errores, comentarios del código
- Incluye manual de usuario / Contenido multimedia
- Durante la realización del proyecto, se realizan avances en la ejecución del proyecto (revisión del repositorio del proyecto cada 15 días para comprobar avances). En el README se indican las tareas realizadas (enlaces a tutoriales, esquemas de diseño...).
- El proyecto está subido a GitHub como público. El repositorio contiene el código fuente de la aplicación y un archivo README con el siguiente contenido: Título del proyecto, descripción, enlace a la memoria del proyecto en formato PDF, enlace al manual de usuario en formato PDF
- Se ajusta al tiempo marcado, explica el funcionamiento de la aplicación y no se apoya en vídeos explicativos
- Administra el tiempo en relación a los contenidos del proyecto de forma homogénea.
- Establece un esquema o guía para la exposición, y lo sigue.
- Hace referencia al objetivo del trabajo, los puntos esenciales y las conclusiones finales.
- Responde a las cuestiones planteadas por los miembros del Tribunal con autoridad y de forma razonada. Aspecto físico adecuado a la presentación del proyecto
- Grado de conocimiento y dominio de los contenidos expuestos y lenguaje técnico utilizado.

6.2 Análisis del sistema

Este punto ya se menciona en el apartado 4.

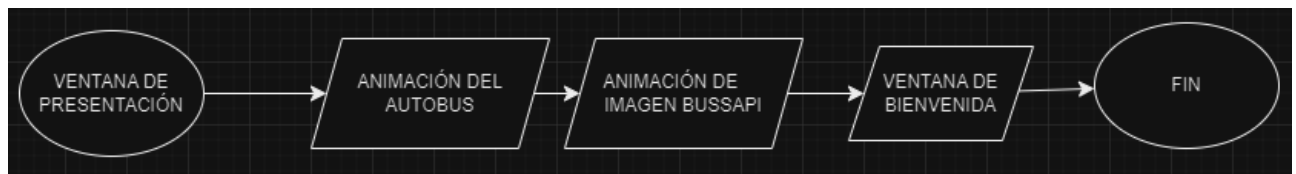
6.3 Diseño del sistema:

6.3.1 Diseño de la Base de Datos

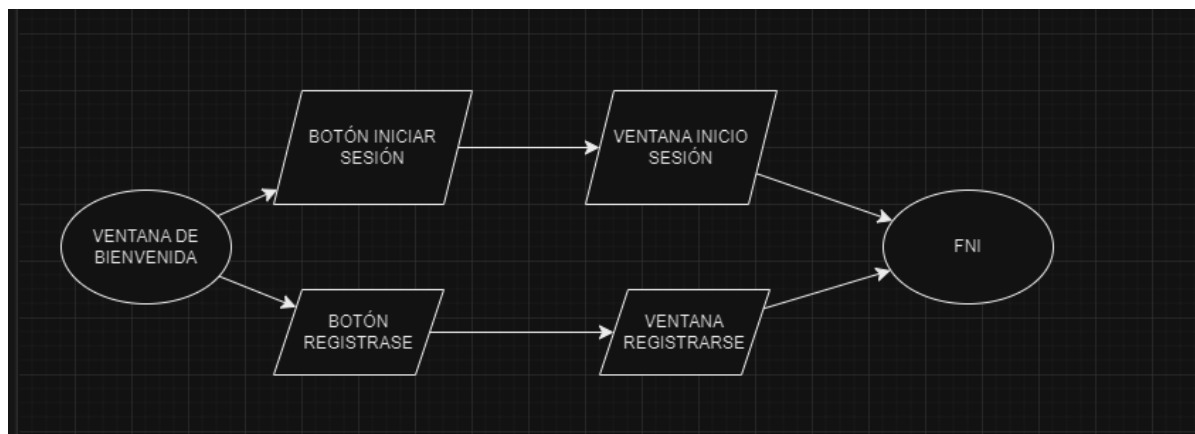


6.3.2 Diseño de la Interfaz de usuario.

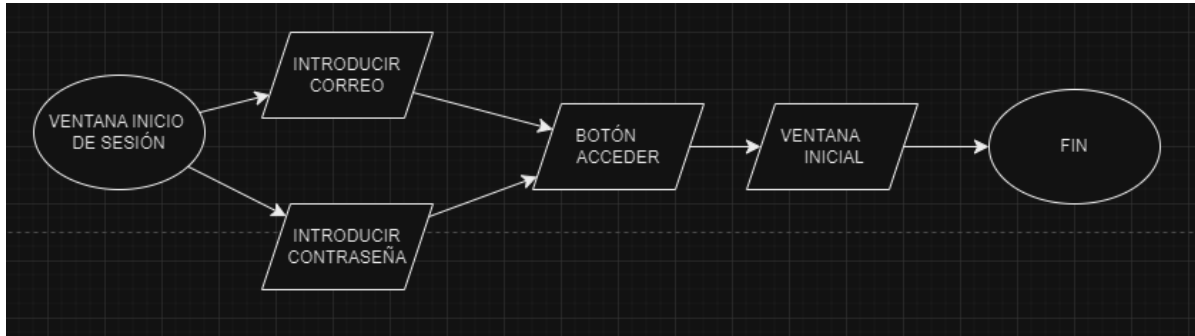
Ventana de presentación:



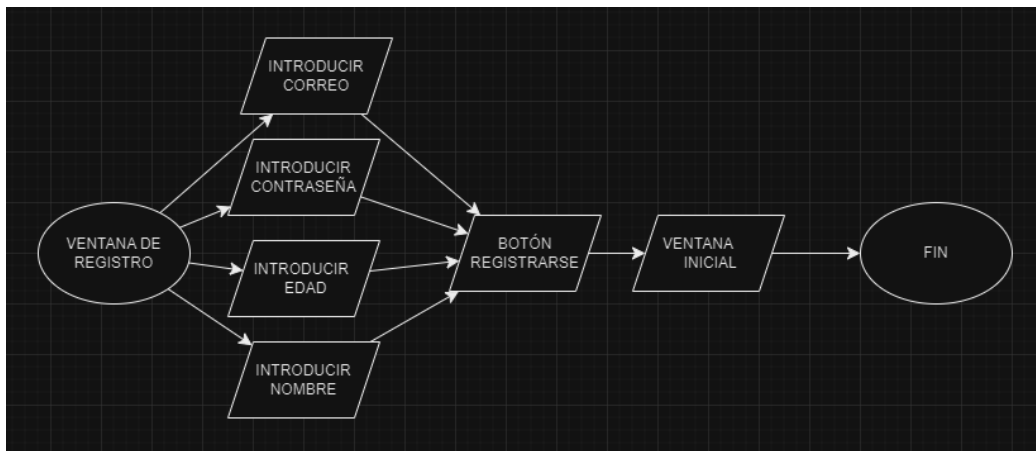
Ventana de bienvenida:



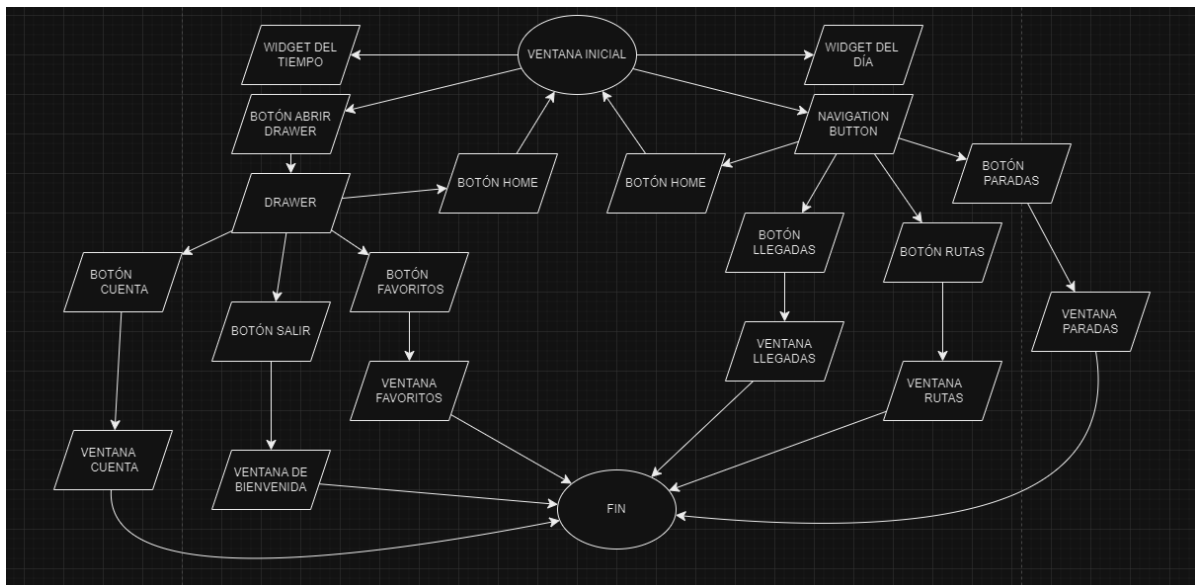
Ventana de inicio de sesión:



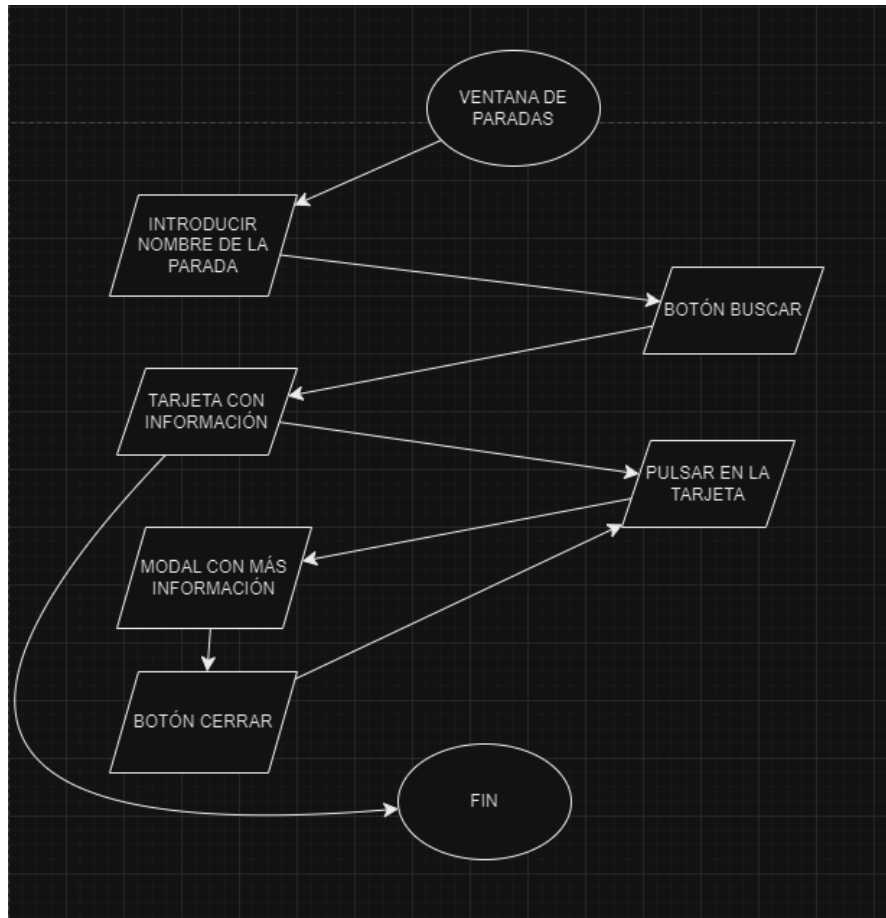
Ventana de registro:



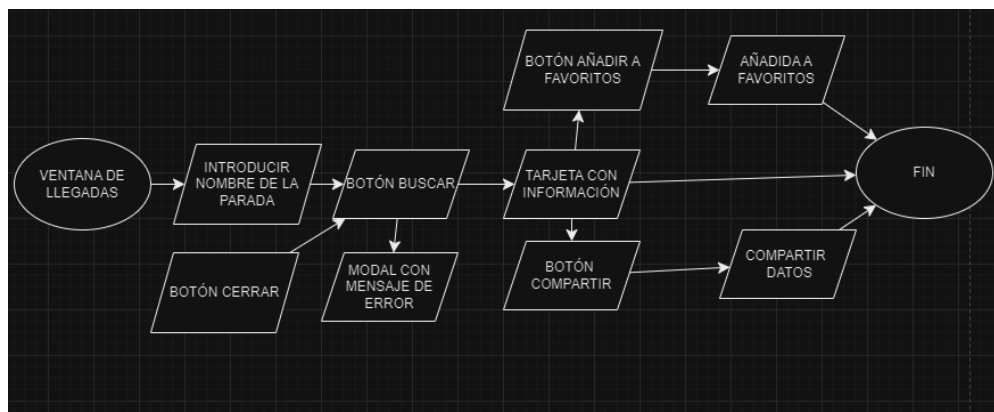
Ventana inicial:



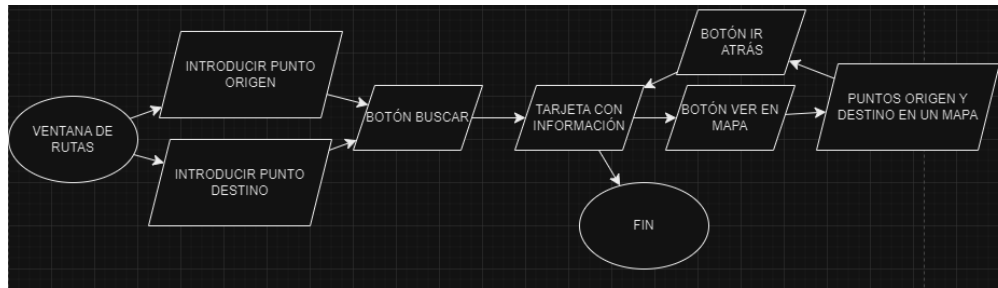
Ventana de paradas:



Ventana de llegadas:



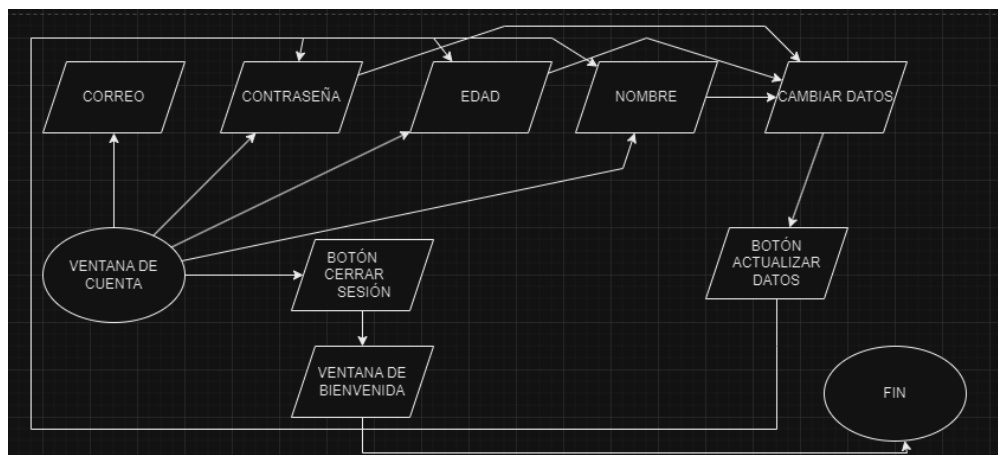
Ventana de rutas:



Ventana de favoritos:



Ventana de cuenta:



6.3.3 Diseño de la Aplicación.

Este apartado se referencia en el punto 4.

6.4 Implementación:

6.4.1 Entorno de desarrollo.

Para el desarrollo de la aplicación se han utilizado diferentes elementos, estos son los siguientes:

- **BOTONES:** Se han utilizado a lo largo de toda la aplicación, algunos con la misma funcionalidad, como la de buscar paradas o llegadas, y otros como cerrar sesión o actualizar los datos.
- **FLAT LIST / SCROLLVIEW:** Ambos elementos funcionan de manera similar, crean una lista que se puede deslizar hacia arriba y hacia abajo. Se han utilizado para mostrar las tarjetas.
- **CAMPOS DE TEXTO:** Se han utilizado a lo largo de toda la aplicación para introducir datos que se usan para desarrollar diferentes acciones.
- **MODAL:** Es parecido a un 'Alert', una ventana emergente que aparece cuando se acciona un botón o se realiza una acción, contienen mensajes de texto para informar al usuario de algo en concreto.
- **DRAWER:** Es el menú desplegable que aparece si se pulsa el botón situado arriba a la izquierda en la ventana de 'home'. Contiene diferentes botones que realizan diferentes acciones.
- **BUTTON NAVIGATOR:** Son los diferentes botones que aparecen abajo del todo de la pantalla. Se llaman así porque su función es navegar a diferentes ventanas cuando los botones son pulsados.
- **TOAST:** Son mensajes que aparecen en la aplicación cuando se realiza una acción, por ejemplo, al iniciar sesión con un usuario.
- **VIEW:** Son 'vistas' que envuelven el resto de los elementos. Dentro de estos elementos se pueden incluir todo tipo de cosas, textos, botones, modales...
- **TOUCHABLEOPACITY:** La traducción al castellano sería 'zona tocable'. Es un elemento que permite crear una zona en la que el usuario puede pulsar y realizar una acción. Es parecido a los botones, solo que se pueden hacer tocables más cosas, como tarjetas, imágenes etc.
- **IMAGE BACKGROUND:** Este elemento permite establecer imágenes de fondo.
- **STACK:** Es simplemente un conjunto de pantallas apiladas que permiten la navegación fluida dentro de una aplicación móvil.
- **MAPVIEW:** Como un 'view' pero en el que solo se puede 'meter' mapas.
- **MARKER:** Marcador de puntos de origen y destino en los mapas.
- **SHARE:** Elemento para compartir la información de las tarjetas de diferentes formas elegidas por el usuario.
- **HOOKS:** Los 'hooks' son funciones especiales que te permiten usar el estado y otras características de React Y React Native en componentes de función. Antes de la introducción de 'hooks', las características como el estado y el ciclo de vida solo

estaban disponibles en componentes de clase. Los 'hooks' cambiaron eso, permitiendo a los desarrolladores usar el estado, efectos y otros aspectos de React Native en componentes funcionales sin tener que convertirlos en componentes de clase.

6.4.2 Estructura del código.

LIBRERÍAS DESCARGADAS:

- @expo/vector-icons (^14.0.0): Iconos vectoriales personalizables para Expo.
- @react-native-async-storage/async-storage (^1.23.1): Un sistema de almacenamiento asincrónico persistente para React Native.
- @react-navigation/bottom-tabs (^6.5.20): Navegación mediante pestañas inferiores para React Navigation.
- @react-navigation/drawer (^6.6.15): Componente de navegación de cajón para React Navigation.
- @react-navigation/native (^6.1.17): Navegación nativa para React Native.
- @react-navigation/native-stack (^6.9.26): Navegación de pilas nativas para React Navigation.
- @react-navigation/stack (^6.3.29): Navegación en pila para React Navigation.
- expo (~50.0.14): Plataforma para aplicaciones universales React.
- expo-linear-gradient (~12.7.2): Componente de gradiente lineal para Expo.
- expo-sqlite (^13.4.0): Funciones de SQLite para Expo.
- expo-status-bar (~1.11.1): Barra de estado personalizada para Expo.
- firebase (^10.10.0): Plataforma de desarrollo de aplicaciones móviles y web de Google.
- react (18.2.0): Biblioteca principal de React.
- react-native (0.73.6): Marco de desarrollo de aplicaciones móviles.
- react-native-gesture-handler (~2.14.0): Manipuladores de gestos declarativos para React Native.
- react-native-linear-gradient (^2.8.3): Componente de gradiente lineal para React Native.
- react-native-maps (^1.11.3): Componente de mapas para React Native.

- react-native-reanimated (~3.6.2): Animaciones declarativas para React Native.
- react-native-sqlite-storage (^6.0.1): Funciones de almacenamiento SQLite para React Native.
- react-native-svg-animations (^1.0.0): Animaciones SVG para React Native.
- react-native-weather (^0.2.11): Componente de pronóstico del tiempo para React Native.

LIBRERÍA DE FUNCIONES DESARROLLADA POR EL USUARIO:

No se ha desarrollado ninguna librería de funciones.

ORIENTACIÓN DEL CÓDIGO:

El código está orientado a eventos, ya que se utilizan bibliotecas como 'React Navigation', que gestionan la navegación y las transiciones en función de eventos de usuario, y 'Firebase', que proporciona una base sólida para manejar eventos en tiempo real.

6.4.3 Cuestiones de diseño e implementación reseñables.

Se debe resaltar la realización de las ventanas de llegadas y de rutas, ya que la información que se muestra en la 'interfaz' no la proporciona la API directamente. Se han realizado dos 'conversiones' (una en cada ventana) para adaptar la información recibida por la API a lo que requería la aplicación. Lo que se quiere dar a entender es que la API de TFL trabaja mucho con identificadores, coordenadas, nombres en clave para diferentes datos etc., por lo que se han tenido que realizar cambios importantes para que la aplicación sea lo más sencilla de usar para el usuario.

6.5 Pruebas.

PRIMERA PRUEBA:

Comprobar que aparecen tarjetas con el nombre de paradas en la ventana de paradas:

Para realizar esta prueba, se introduce un nombre de una parada, por ejemplo 'London Eye'.

Pero también hay que comprobar que si se introduce el nombre de una parada que no existe en la API, la aplicación debe detectarlo y debe avisar al usuario, por lo que se introduce el nombre de la parada 'Madrid' para forzar el 'error' al realizar una búsqueda de una parada con ese nombre.

Las imágenes de la prueba:

Paradas

Buscar Parada por Nombre

BUSCAR

London Eye Waterloo Pier



Home



Paradas



Llegadas



Rutas

Paradas

Buscar Parada por Nombre

BUSCAR

No se encontraron paradas



Home



Paradas



Llegadas



Rutas

SEGUNDA PRUEBA:

Actualizar datos del usuario en la ventana de cuenta.

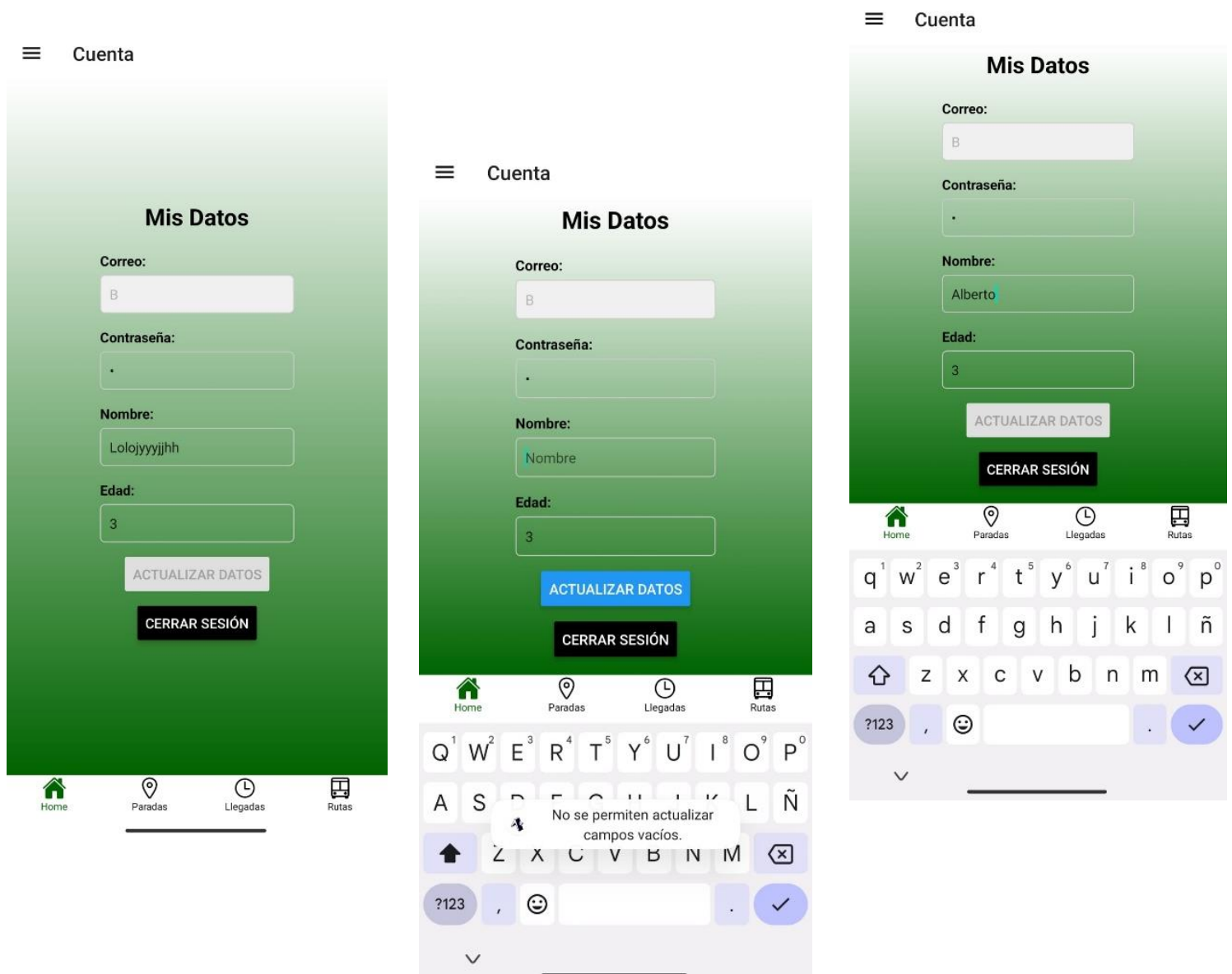
Para ello, se cambiará el nombre del usuario a 'Alberto'.

También se va a forzar el 'error' al intentar actualizar el nombre a un campo vacío, para que la aplicación indique al usuario que no se pueden actualizar datos cuyos campos estén vacíos.

En la primera imagen aparecen los datos del usuario antes de ser cambiados.

En la segunda imagen aparece el campo de nombre vacío con el mensaje de 'error' avisando al usuario de que no se pueden actualizar datos cuyos campos de texto estén vacíos.

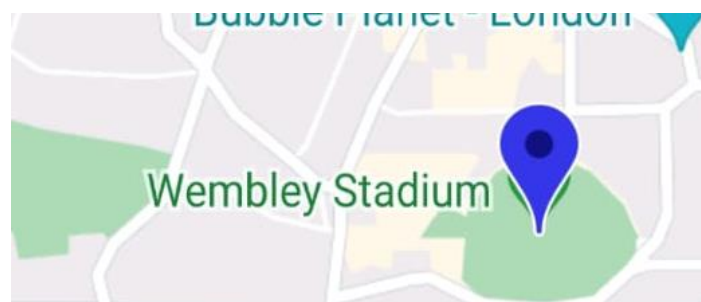
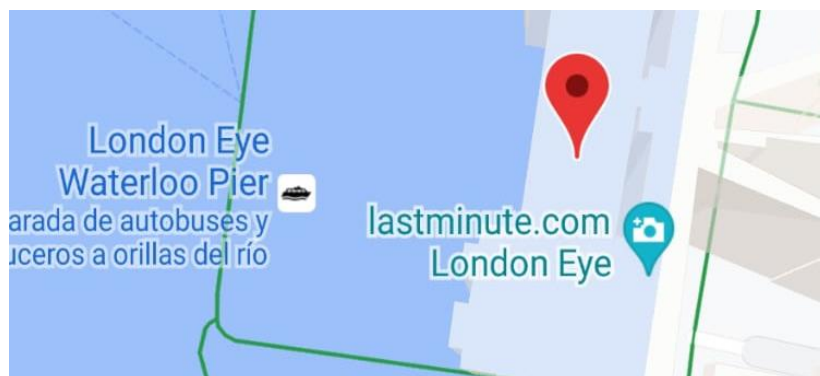
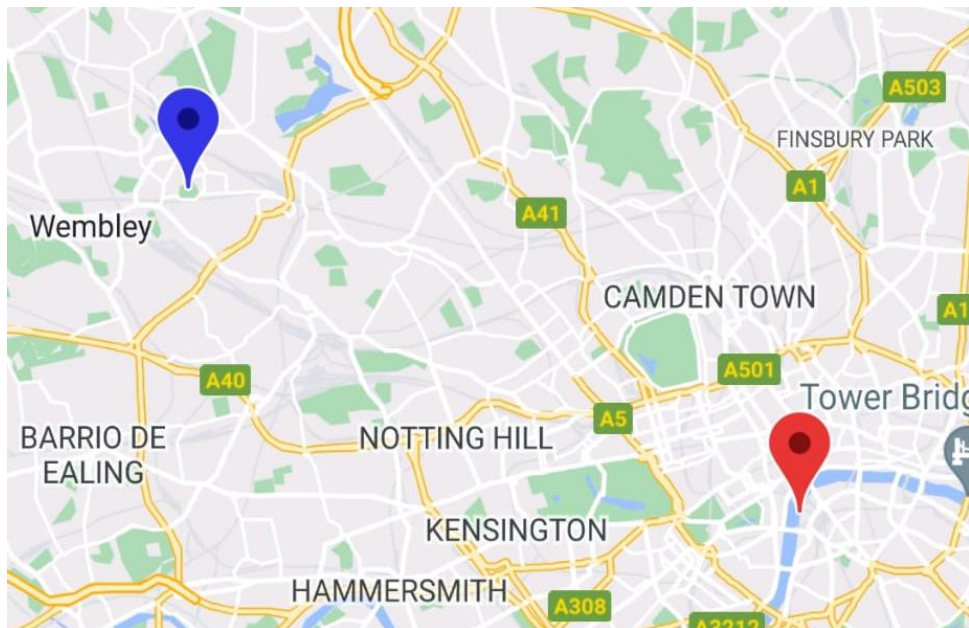
Por último en la tercera imagen, aparece el campo de texto nombre con el nombre 'Alberto' actualizado correctamente.



TERCERA PRUEBA:

Comprobar que aparecen bien los marcadores de los mapas en los puntos de origen y destino introducidos en la ventana de las rutas.

Para ello se establecerá como punto de origen 'London Eye' y como punto de destino 'Wembley Stadium'



7 Manuales de usuario

7.1 *Manual de usuario*

A continuación, se explicará el funcionamiento de la aplicación ventana por ventana, aportando imágenes de las mismas con todos los detalles que el usuario necesita para una mejor comprensión de la funcionalidad del proyecto.

VENTANA DE PRESENTACIÓN:

Esta ventana no tiene nada para que el usuario interactúe con la aplicación, simplemente es una animación, primero aparece un autobús de izquierda a derecha, y más tarde aparece el logo de la aplicación, que acaba desapareciendo, mostrándose la ventana de bienvenida.



VENTANA DE BIENVENIDA:

En esta ventana aparecen dos opciones para que el usuario elija cómo entrar a la aplicación, si mediante un inicio de sesión (se necesita tener una cuenta) o mediante un registro (para usuarios que no tienen cuenta).

Dependiendo de la opción que seleccionen, deberán pulsar un botón u otro, si se pulsa el botón de inicio de sesión, la aplicación llevará al usuario a la ventana de inicio de sesión. Por el contrario, si el usuario selecciona la opción de registro, la aplicación le llevará a la ventana de registro,



VENTANA DE INICIO DE SESIÓN:

En esta ventana aparecen dos campos de texto que el usuario debe rellenar para poder acceder a la aplicación siempre y cuando el usuario tenga una cuenta creada, de no ser así, deberá pulsar la flecha situada arriba a la izquierda para volver a la ventana de bienvenida y pulsar el botón de registrarse.

Si el usuario ya tiene una cuenta creada, deberá indicar el correo electrónico y la contraseña de dicha cuenta para acceder a la aplicación.

Una vez rellenados ambos campos, se deberá pulsar el botón de acceder para poder entrar a la aplicación.

Si se introducen valores en los campos de texto que no coinciden con los datos de la cuenta, la aplicación informará al usuario de que no existe un usuario con esos datos en la base de datos, por lo que deberá revisar los datos que ha introducido.

Si los datos introducidos son correctos, se mostrará la ventana inicial.



The screenshot shows a mobile application interface for a login screen. At the top left, there is a back arrow icon and the text "Login". The main area has a green gradient background. In the center, the text "Inicio de Sesión" is displayed. Below this, there are two text input fields: the first is labeled "Correo electrónico" and the second is labeled "Contraseña". At the bottom center, there is a black button with the white text "ACCEDER".

VENTANA DE REGISTRO:

El usuario accederá a esta ventana si no tiene una cuenta creada, de ser así pulsará la flecha situada arriba a la izquierda, mostrando de nuevo la ventana de bienvenida, en la que pulsará el botón de iniciar sesión.

En esta ventana aparecen cuatro campos de texto para rellenar, siendo obligatorio rellenar todos para que el registro sea completo. La aplicación también notifica al usuario de que es obligatorio rellenar todos los campos en el caso de que se pulse el botón de acceder con algún campo vacío.

Solamente puede haber un correo asociado a una cuenta, por lo que, si el usuario utiliza un correo ya existente en la base de datos, no podrá acceder a la aplicación. La aplicación se encarga de avisar al usuario de que ese correo ya pertenece a otra cuenta, que debe cambiarlo.

Si todos los datos que introduce son válidos, el usuario entrará en la aplicación, a la ventana de inicio.

← Registro

Registrarse

Nombre

Edad

Correo electrónico

Contraseña

ACCEDER

VENTANA DE INICIO

En esta ventana, lo primero que aparece son dos ‘widgets’ con diferente información, el primero indica el día de semana, día del año y la hora actual de Londres, ya que la aplicación está realizada a partir de la API de TFL de Londres.

El segundo ‘widget’ indica el tiempo actual de Londres, su temperatura, humedad...

El usuario no puede interactuar con estos ‘widgets’, son solo informativos.

Por otro lado, en esta ventana aparecen cuatro botones en la parte inferior de la pantalla. Cada botón hace referencia a una ventana distinta, por lo que, si se pulsa un botón, se mostrará la ventana asociada a ese botón.

Además, en la parte superior izquierda aparecen tres barras horizontales, es un botón, que, si se pulsa, aparece un ‘drawer’ (menú lateral) con más botones que referencian a otras ventanas.

Al igual que los botones de abajo, si se pulsa algún botón del ‘drawer’, se mostrará la ventana asociada a ese botón.



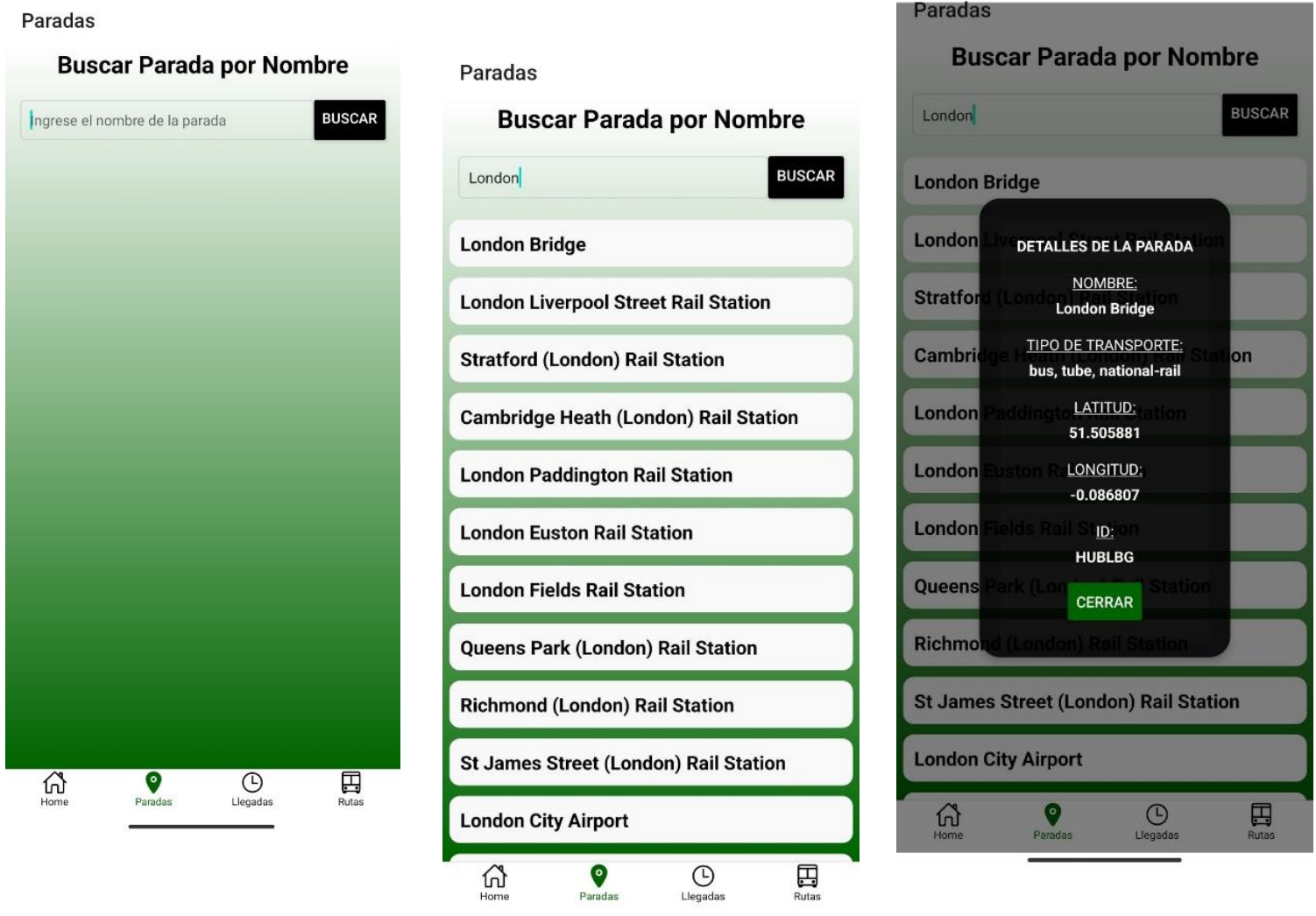
VENTANA DE PARADAS:

La finalidad de esta ventana es que el usuario pueda ver las diferentes paradas que hay en la ciudad de Londres y sus alrededores, por lo que deberá rellenar el campo de texto con la parada que desea consultar.

Si el usuario escribe el nombre de una parada que no existe, por ejemplo ‘Madrid’, la aplicación mostrará un mensaje diciendo que la parada no existe.

Si, por el contrario, el usuario escribe el nombre de una parada que, sí que existe, aparecerán tarjetas asociadas al nombre introducido, por ejemplo, si se introduce el nombre de London, aparecerán muchas tarjetas que empiezan por el nombre London, o si el usuario introduce tal cual el nombre de una parada solamente aparecerá la tarjeta de esa parada.

Además, si se pulsa en una tarjeta, aparecerá una ventana emergente con más datos sobre esa parada, como el tipo de transporte que llega a esa parada, como sus coordenadas, pudiendo ser útil para que el usuario se ubique mejor en la ciudad.



VENTANA DE LLEGADAS:

La finalidad de esta ventana es que el usuario pueda ver los autobuses que llegan a la parada/s que quiera.

Para ello el usuario deberá introducir el nombre de la parada que quiera consultar. Si escribe un nombre de una parada que no existe, la aplicación mostrará un mensaje indicando que no se encontraron resultados para la búsqueda.

Por otro lado, si el usuario escribe el nombre de una parada que, sí que existe, aparecerán tarjetas con la información de los autobuses que llegan a esa parada, mostrando información como la ruta que seguirá ese autobús, la hora aproximada a la que se espera que ese autobús llegue o el destino al que se dirige.

Cada tarjeta cuenta con dos botones, uno para añadir a favoritos y otro para compartir la tarjeta.

Si se pulsa la estrella, la estrella se pondrá en amarillo y se añadirá a la ventana de favoritos (a los 2,5 segundos la estrella vuelve a estar amarilla ella sola).

Si el usuario pulsa en el botón de compartir, aparecerán varias opciones para que se compartan los datos de esa tarjeta, por correo electrónico, WhatsApp etc.

Hay una cosa importante a mencionar en esta ventana, y es que la búsqueda de paradas tiene un límite, es decir, cuando se realizan búsquedas de paradas, se realizan llamadas a la API para obtener esos datos, pero la API impone un límite de tiempo entre llamada y llamada (o búsqueda y búsqueda, que es lo mismo) que hay que respetar. El tiempo que hay que esperar no se sabe, varía, hay veces que el usuario puede hacer en 20 segundos 3 llamadas a la API y otras veces que tiene que esperar 2 minutos para poder realizar búsquedas, por lo que la aplicación informa al usuario cuando se supera el límite de búsquedas mediante una ventana emergente con un mensaje indicando dicho problema.

Esto ocurre porque las peticiones a la API (en este caso) son solicitudes síncronas, no se puede ejecutar otras acciones hasta que la API devuelva los datos que quiere el usuario.

Las solicitudes síncronas pueden influir en el límite de peticiones por diferentes motivos:

- Mayor impacto en el límite de peticiones: Cuando las solicitudes son síncronas, cada solicitud bloquea el hilo de ejecución hasta que se complete la respuesta. Esto significa que el cliente puede estar esperando más tiempo para recibir una respuesta, lo que podría resultar en un mayor número de solicitudes simultáneas y, por lo tanto, un mayor uso de los recursos de la API.
- Mayor riesgo de superar los límites: Si las solicitudes síncronas se realizan de manera agresiva o sin control, existe un mayor riesgo de superar los límites de las peticiones impuestos por la API. Esto se debe a que el cliente puede enviar

solicitudes más rápido de lo que la API puede manejar, lo que lleva a un exceso en el número de solicitudes permitidas en un período de tiempo determinado.

- Necesidad de administración de solicitudes: Con solicitudes síncronas, es importante implementar una gestión adecuada de las solicitudes para evitar exceder los límites establecidos por la API. Esto puede implicar la implementación de estrategias como el control de velocidad, la cuota de solicitudes por usuario, o la pausa entre solicitudes para cumplir con los límites de la API.

Otra cosa a tener en cuenta en esta ventana es que, si se realiza la búsqueda de una parada, se busca otra distinta y se quiere buscar de nuevo la primera parada, es posible que aparezca el mensaje de que no se encontraron resultados. Esto ocurre porque los datos de la API se están actualizando constantemente, y cuando se actualizan, la API no devuelve datos, hay que esperar un tiempo a que la API vuelva a pasar información sobre esa parada.

Llegadas

Buscar Próximos Autobuses

No se encontraron resultados

Home

Paradas

Llegadas

Rutas

Llegadas



Buscar Próximos Autobuses

Ruta:
RB1

Hora de llegada:
2024-05-14T17:15:04Z

Parada:
London Eye Waterloo Pier

Destino:
Barking Riverside Pier





Ruta:
RB1

Hora de llegada:
2024-05-14T17:57:00Z

Parada:
London Eye Waterloo Pier

Destino:
Barking Riverside Pier



Ruta:
RB1

Hora de llegada:

Home

Paradas

Llegadas

Rutas

VENTANA DE RUTAS:

La funcionalidad de esta ventana es que el usuario pueda consultar rutas de los autobuses que tienen como punto de origen y destino los puntos introducidos en los campos de texto.

Una vez introducidos los nombres de los puntos de origen y destino, se pulsa al botón buscar ruta para realizar la búsqueda.

Si entre esos puntos introducidos no hay rutas disponibles, la aplicación notificará al usuario de que no se han encontrado rutas.

Si por el contrario si que se han encontrado rutas, se mostrarán una serie de tarjetas con la información de dicha ruta, como la hora a la que sale el bus desde el punto de origen, la hora a la que llega al punto de destino, la duración del trayecto de un punto a otro y las instrucciones que se deben seguir para coger ese autobús. Estas instrucciones hacen referencia al lugar donde se podrían ubicar las paradas por las que pasa el autobús, pero hay veces que el autobús para en un lugar donde no existe parada, por lo que es mejor seguir las instrucciones que mostrar directamente las paradas.

Cada tarjeta cuenta con un botón con la imagen de un mapa. Si este botón se pulsa, se mostrará la ventana de mapas.

Rutas

Buscar Rutas de Autobús

BUSCAR RUTA

Rutas

Buscar Rutas de Autobús

BUSCAR RUTA

Salida:

2024-05-14T16:42:00

Llegada:

2024-05-14T17:07:00

Duración:

25 minutos

Instrucciones / Guía :

- Walk to Waterloo Station
- Jubilee line to London Bridge
- Walk to 1 London Bridge, City

**Salida:**

2024-05-14T16:44:00

Llegada:

2024-05-14T17:09:00

Duración:

25 minutos

Instrucciones / Guía :

- Walk to Waterloo Station
- Jubilee line to London Bridge



Home



Paradas



Llegadas



Rutas



Home



Paradas



Llegadas



Rutas

VENTANA DE MAPAS:

La finalidad de esta ventana es que el usuario pueda ver los puntos de origen y destino introducidos en la ventana de rutas en un plano/mapa.

Aparece un marcador de color rojo en el punto de origen de la ruta.

Ocurre de la misma manera en el punto de destino, pero el color es azul.

En este mapa se puede hacer zoom, por lo que, si el usuario desea ampliar más el plano para ver más de cerca los puntos, lo podrá hacer, además de girar el plano en el orden que sea.

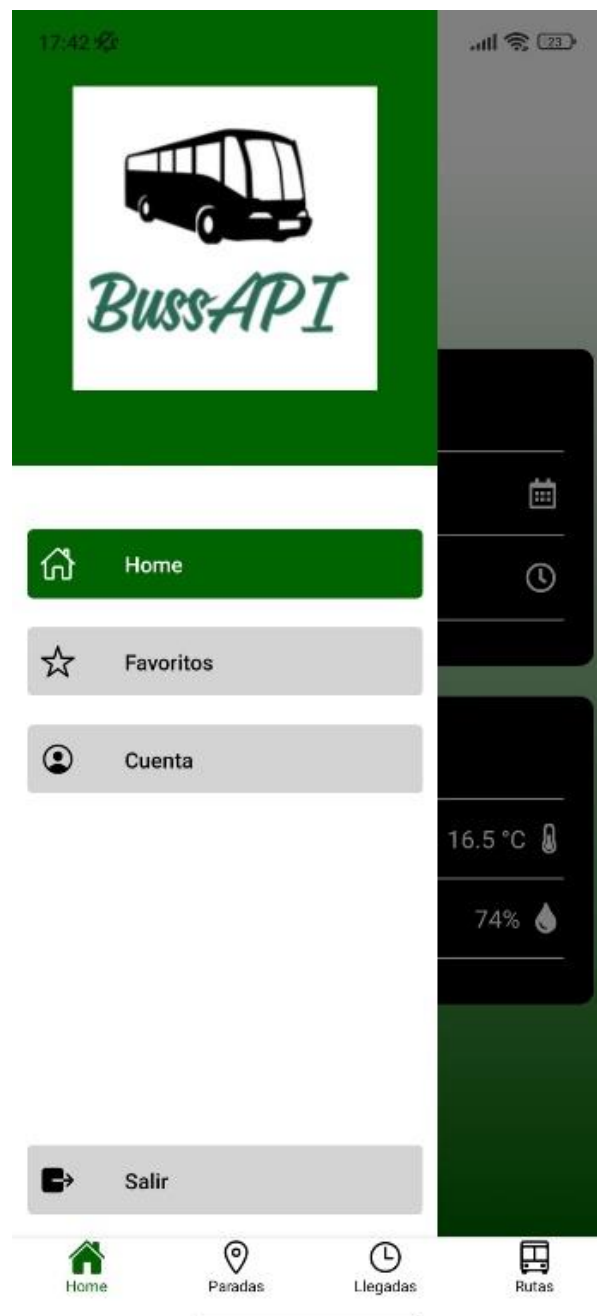
La ventana también cuenta con un botón de retroceso situado arriba a la izquierda, que si se pulsa se mostrará de nuevo la ventana de rutas.



VENTANA DEL DRAWER:

En este menú lateral aparecen 4 botones diferentes, cada botón haciendo referencia a una ventana diferente.

La finalidad de esta ventana es simplemente que el usuario pueda navegar a las diferentes ventanas de manera más sencilla y cómoda posible.



VENTANA DE FAVORITOS:

La finalidad de esta ventana es que el usuario pueda tener un acceso directo a las tarjetas con los datos de los autobuses que lleguen a una parada en vez de buscarlas cada vez que se entre a la aplicación.

Para que aparezcan las tarjetas en la ventana de favoritos, se deberá buscar una tarjeta en la ventana de llegadas.

Una vez encontrada la tarjeta que se desea, se pulsará la estrella de esa tarjeta para que aparezca en esta ventana.

Además, en esta ventana aparecen tarjetas con una estrella, pero la estrella está siempre amarilla, eso quiere decir que está añadida a favoritos.

Si se pulsa esa estrella, la estrella pasará a estar blanca, eso quiere decir que se habrá eliminado de favoritos, pero no desaparecerá de la ventana, aunque ya no esté en favoritos, esto se debe a que se tiene que actualizar la ventana.

Para actualizar la ventana, se debe pulsar el botón que aparece en la parte superior de la misma (la flecha).

Al pulsar esa flecha, las tarjetas en las que se hayan quitado la estrella amarilla desaparecerán y solo se mostrarán aquellas cuya estrella siga estando amarilla.

Si por lo que sea se ha borrado una tarjeta sin querer o simplemente porque el usuario se ha confundido, se deberá ir de nuevo a la ventana de llegadas, buscar esa tarjeta y volver a pulsar la estrella de esa tarjeta.

Otra cosa que comentar es que se puede añadir a favoritos la misma tarjeta, por lo que debe ser el usuario el que decida si eliminar las tarjetas repetidas.

≡ Favoritos



Ruta:

RB1

Hora de salida:

2024-05-03T11:37:00Z

Parada:

London Eye Waterloo Pier

Destino:

North Greenwich Pier



Ruta:

RB1

Hora de salida:

2024-04-25T18:22:00Z

Parada:

London Eye Waterloo Pier

Destino:

Barking Riverside Pier



VENTANA DE CUENTA:

La finalidad de esta ventana es que el usuario pueda consultar los datos de su cuenta.

En esta ventana aparecen varios campos de texto con los datos de ese usuario.

Estos campos son los mismos que aparecen en la ventana de registro.

Tres de los cuatro campos que aparecen se pueden editar. Estos campos son la contraseña, el nombre y la edad del usuario.

Para actualizar esos campos simplemente se debe pulsar en los campos y cambiarles de valor, por ejemplo, cambiar el nombre de Alberto a José.

Una vez cambiado el valor de aquellos campos a actualizar, se pulsará en el botón de actualizar datos que aparecerá de color azul (previamente estaba de color gris, esto es porque estaba deshabilitado, no se podía pulsar hasta que le hiciese algún cambio)

Si se decide cambiar algún campo y establecer el valor a vacío, la aplicación notificará al usuario de que no se pueden actualizar campos vacíos.

Si los valores que se introducen son válidos, al pulsar en el botón de actualizar datos, se modificarán los datos antiguos por los nuevos.

Además, en esta ventana aparece un botón de cerrar sesión, que, si se pulsa, la aplicación cerrará la sesión al usuario y mostrará la ventana de bienvenida.



Cuenta

Mis Datos

Correo:

B

Contraseña:

.

Nombre:

Alberto

Edad:

3

ACTUALIZAR DATOS

CERRAR SESIÓN



Home



Paradas



Llegadas



Rutas

BOTÓN DE SALIR:

Este botón aparece en la ventana del ‘drawer’.

La funcionalidad de este botón es salir de la aplicación cuando es pulsado, mostrando de nuevo la ventana de bienvenida.

La idea de este botón es que el usuario pueda salir de la aplicación sin entrar a la ventana de cuenta.

7.2 Manual de instalación

Este apartado ya se explica en el punto 4.3

8 Conclusiones y posibles ampliaciones

En cuanto a las conclusiones, he aprendido bastante el lenguaje de ‘React Native’, siempre está bien aprender un lenguaje de programación nuevo. He cumplido mis objetivos, realizar una aplicación funcional que consulte datos de una API y que implemente una base de datos externa, aunque ha sido complicado adaptar la información recibida de la API a la aplicación para que sea más fácil de usar etc., así que, por lo general, estoy bastante satisfecho.

Por otro lado, hay una implementación que de cara a cambios futuros podría mejorar la aplicación. Esta implementación tiene que ver con los mapas. Cuando se pulsa en el botón del mapa de la tarjeta en la ventana de rutas, aparece un mapa con dos marcadores, uno haciendo referencia al punto de origen y otro al punto de destino, la implementación sería unir esos 2 puntos con una línea, pero no una línea recta, si no por las paradas que proporciona la API, que aparecen en modo de instrucciones.

9 Bibliografía

<https://youtu.be/psSO3T7gslU>

<https://youtu.be/VE7J0SA1PRQ>

<https://youtu.be/8GAualqYB10>

https://youtu.be/Ewv2_b3N3Q8

<https://youtu.be/RdwDoyLsY6E>

<https://stackoverflow.com/>

<https://chatgpt.com/>

10 Anexos

Página para crear el logo de la aplicación: <https://www.tailorbrands.com/>

Repositorio del código: https://github.com/MartinGomezAlvaro/primer_proyecto

Base de datos utilizada: <https://firebase.google.com/?hl=es>

Mapa: <https://leafletjs.com/>