**Proyecto de desarrollo de aplicaciones multiplataforma**

**BusTracker**

CICLO FORMATIVO DE GRADO SUPERIOR

**Desarrollo de Aplicaciones Multiplataforma (IFCS02)**

**Curso 2022-23**

Autor/a/es:

**Roberto Blázquez Martin**

Tutor/a:

**Teresa Carrero de las Peñas**

Departamento de Informática y Comunicaciones

**I.E.S. Luis Vives**

# INTRODUCCIÓN

BusTracker es una aplicación que te ayuda a ver los tiempos de los autobuses y metros.  
A parte de eso también permite al usuario crearse una cuenta para poder guardar las paradas que mas le sirvan y poder ver la localización de los autobuses.

## OBJETIVO

El objetivo principal de la aplicación es proveer de una interfaz grafica a los usuarios para poder consultar todo lo relacionado al transporte público.

## ALCANCE

* Recopilación de datos en tiempo real: La aplicación deberá conectarse a los sistemas de seguimiento en tiempo real de las empresas de transporte público para obtener información actualizada sobre los horarios de llegada de los autobuses y el metro.
* Visualización de tiempos de llegada: La aplicación mostrará de manera clara y fácil de entender los tiempos estimados de llegada de los autobuses y el metro en las diferentes paradas. Los usuarios podrán ver la información en tiempo real y realizar un seguimiento de los horarios programados.
* Interfaz intuitiva y amigable: La aplicación deberá contar con una interfaz fácil de usar, con un diseño intuitivo y amigable para los usuarios. Deberá ser accesible tanto para aquellos familiarizados con la tecnología como para aquellos que no lo están.

## JUSTIFICACIÓN

* Mejora de la experiencia del usuario: La aplicación proporcionará a los usuarios información actualizada y precisa sobre los tiempos de llegada de los autobuses y el metro, lo que les permitirá planificar sus viajes de manera más eficiente y reducir la incertidumbre y la espera en las paradas. Esto mejorará la experiencia general de los usuarios del transporte público.
* Ahorro de tiempo y reducción de estrés: Al conocer los tiempos de llegada de los autobuses y el metro en tiempo real, los usuarios podrán optimizar sus desplazamientos y minimizar el tiempo de espera en las paradas. Esto les ayudará a llegar a sus destinos de manera más rápida y eficiente, reduciendo las esperas prolongadas.
* Fomento del uso del transporte público: Al proporcionar información precisa y confiable sobre los horarios de llegada, la aplicación puede fomentar el uso del transporte público al hacerlo más conveniente y atractivo para los usuarios.

# IMPLEMENTACIÓN

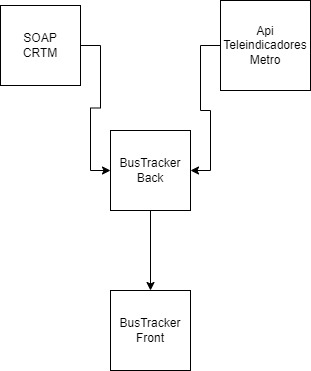
## ANÁLISIS DE LA APLICACIÓN

La aplicación tiene 3 páginas principales.

* Búsqueda de parada de autobús
* Búsqueda de estación de metro
* Inicio de sesión

## DISEÑO

El principal diseño de la aplicación seria:



La idea de hacer un back era unificar y normalizar los datos que proveen los diferentes proveedores, y de esta manera simplificar el uso.

## IMPLEMENTACIÓN

A la hora de implementar la aplicación el principal problema era obtener los datos de los diferentes proveedores ya que no hay ningún tipo de documentación.

Para obtener los datos de los autobuses la única opción era consumir la api de la web de crtm.

El problema es que la api se cae constantemente, entonces ¿cómo solucionamos eso?.

Resulta que la api es solo un wrapper hecho en php sobre un servicio soap que es el que realmente provee los datos.

Y la prueba de ello es ver la aplicación descompilada que la propia CRTM publicó hace tiempo.

Aunque ya no esté en Google Play, el apk sigue estando en páginas como apkpure.  
<https://m.apkpure.com/es/transporte-de-madrid-crtm/com.crtm.mitransporte>

Y si lo descompilamos y buscamos entre los archivos, obtenemos una url y una clave privada.   
Texto

Descripción generada automáticamente

También parece haber varios entornos y un servidor para consultar los abonos transporte.

A partir de aquí podemos generar todas las clases del servicio SOAP y consultarlo

Ahora podemos obtener los datos de los autobuses, pero ¿cómo obtenemos los datos del metro?

Tampoco parece haber ningún tipo de api, pero usando la herramienta de búsqueda de github encontré este repositorio.  
<https://github.com/Caul58/ScriptableElements>

El cual obtiene información de los teleindicadores.   
  
A partir de aquí solo había queda normalizar estos datos para el consumidor de la api.

Primero se implementó las paradas y las líneas, luego el endpoint del metro, y por ultimo los endpoints de los usuario y las paradas favoritas.

Paradas

Para las paradas de autobuses no hubo mucho problema en obtener los datos, el problema principal era los tiempo de carga, las excepciones y los tiempos de carga.

Aunque sea un servicio SOAP totalmente tipado, si introduces parámetros erróneos te devuelve excepciones.

El formato para consumir los ids de las paradas es “${codMode}\_${stopCode}"

Codmode refiere a el tipo de transporte, bus, metro…

Stopcode refiere a la parada, siempre tiene el mismo formato y sabiendo que los buses del CRTM tienen codmode 8 ya podemos consultar las paradas.

El principal problema son los tiempos de carga, como el cliente del servicio SOAP esta generado en java hay un principal problema, y es que no se puede cancelar una petición.

Así que había que ejecutarlo en un Contexto de IO para poder cancelarlo a los 30 segundos.

Si aun así no funcionaba el valor que devuelve es el de la cache, provisionando cuando fue la última actualización de los tiempos.

Lineas

Para las líneas se usa lo mismo pero el formato de las lineas es "${codMode}\_\_${lineCode}\_\_\_"

Así que sabiendo que codMode siempre es 8, podemos obtener la línea solo con el número, por ejemplo si tenemos las línea 468 y queremos saber su localización podemos hacer una petición

GET V1/bus/lines/468/mode/8/locations

Metro

Para el metro solo hay que consultar la api de serviciosapp.metromadrid.es que por suerte ya devuelve los datos en formato json.

La única manera de buscar es

* Obtener todas las estaciones
* Buscar por id

Usuarios

Con los usuarios hace falta implementar la autenticación, un sistema de verificación para los usuarios

Y una base de datos.

La base de datos usa mongo, la autenticación jwt y el sistema de verificación envía un correo para validar a la persona.

Favoritos

Para guardar las paradas favoritas había que limitar los accesos de los usuarios, es decir, que una persona no pueda acceder a las paradas de otra. Para ello podemos extraer los datos usuario a partir de jwt y validar que las paradas favoritas que solicita el usuario pertenecen a el mismo y no a otro.

Tests

Los tests usan una libraría llamada tests container para hacer las pruebas de integración con Docker.

Dan un coverage de mas del 80%

Interfaz de usuario gráfica

Descripción generada automáticamente  
  
Front

La parte de front esta hecha con react y typescript y la razón principal es que puede usarse desde cualquier sistema operativo/dispositivo ya que esta hecho para la web.

## IMPLANTACIÓN

Back

Para la implantación de la api he usado Docker y Github CI/CD para poder desplegar las nuevas versiones.

Cada vez que se hace un commit se ejecutan todos los tests.

Si el commit se hace con un tag, se ejecuta una pipeline que sube una imagen a Docker hub de la api, y acto seguido se baja en la maquina en la que se va a desplegar ejecutando el Docker compose del repositorio.

Al desplegarse se usa nginx como proxy para securizar la api con certificados.

Front

En la parte de front ocurre exactamente lo mismo, al subirse un tag se sube una nueva versión de la imagen a Docker hub y acto seguido se despliega en la maquina usando nginx y securizando la conexión.

## DOCUMENTACIÓN

Los endpoints que quedan son:

GET V1/bus/lines/{lineCode}/mode/{codMode}/locations

GET V1/bu /lines/{lineCode}/mode/{codMode}/stops

GET V1/bus/lines/{lineCode}/mode/{codMode}/itineraries

GET V1/bus/stops/{stopCode}/times

GET V1/bus/stops/{stopCode}/times/cached

GET V1/metro/times

GET V1/metro //times/{id}

POST V1/users/register

GET V1/users/verify

POST V1/users/login

Authenticated

GET V1/favorites

POST V1/favorites

GET V1/favorites/{id}

DELETE V1/favorites/{id}

¿Como ver los tiempos?

Al entrar en la aplicación se puede ver una barra de búsqueda por código de parada.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Si metemos el código de una parada

Nos saldrán los tiempos y la ultima vez que se actualizo   
Imagen que contiene Texto

Descripción generada automáticamente

Si clicamos en un bus, podremos ver las localizaciones

Mapa

Descripción generada automáticamente

Si vamos a la parte de metro podemos buscar por nombre de la estación.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Y obtendremos los tiempos para las líneas de metro

Texto

Descripción generada automáticamente

Si queremos acceder a la funcionalidad de guardar las paradas en favoritos, tenemos que hacer login.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Como no tenemos una cuenta todavía, le damos a crear.

Introducimos los datos   
Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Y nos dira que miremos el correo  
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En el correo le daremos a verificar  
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Y nos llevara a la pagina para iniciar sesión con las credenciales que pusimos.

Si consultamos una parada nos saldrá la opción de guardar en favoritos.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Al hacerlo, ya nos saldrán las paradas guardadas en la pagina principal   
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# TRABAJO FUTURO (Opcional)

Para el futuro me gustaría implementar otros proveedores como EMT o RENFE.

También hay que mejorar la interfaz y a la vez añadir otros funcionalidades como ver el saldo de la tarjeta o recopilar información de cuando llegan buses, que tipo de bus son...

# CONCLUSIONES

Con este proyecto he aprendido a realizar una aplicación que cumpla las necesidades de la personas usan el sistema de transporte publico, he aprendido a conectar backend con frontend usando bases de datos, haciendo un despliegue continuo con Docker y una integración continua con los tests.

# BIBLIOGRAFÍA

Para ver como consumir los datos de CRTM: <https://github.com/jvicentem/citram-python-api>

Para ver como consumir los datos de Madrid: <https://github.com/Caul58/ScriptableElements>

Para ver como testear la aplicación desde un punto de vista “real”: <https://tyrrrz.me/blog/unit-testing-is-overrated>

Para ver como consumir el SOAP de CRTM: <https://gist.github.com/elraro/020b220d21761051b05f1044d4fea690>

Para ver como generar el código a partir de la definición SOAP

<https://medium.com/swlh/wsdl-client-generation-with-kotlin-and-gradle-398483e32191>