



BUS ON TIME

Adrián L



Índice

<u>Índice</u>	2
<u>Especificaciones</u>	3
<u>Elementos innovadores</u>	4
<u>Estudio previo. Prototipo.</u>	12
<u>Backend. Firebase as Service+NodeJS Express API Rest+Google Cloud</u>	16
<u>FrontEnd, Flutter</u>	30
<u>Propuestas de mejora.</u>	44
<u>Bibliografía</u>	45



Especificaciones

Bus On Time es una aplicación para mantener al usuario informado del horario de la línea de autobuses Jaén-Martos en cualquier momento y parada.

Su interfaz es muy amigable para todo tipo de usuarios, teniendo varias opciones de visualización de los horarios, bien a lo tradicional (horario de consorcio), consultando la parada en un mapa interactivo o revisando sus paradas favoritas.

Se basa en la geolocalización del usuario para que el mismo pueda localizar la parada más cercana y revisar sus horarios.

En la parte no visible del proyecto, tenemos un servidor que brinda toda la información a los usuarios desplegado en Vercel. La APP accede a los datos de forma híbrida, lanzando peticiones directas a los servicios de Firebase, para aprovechar así su reactividad o bien lanzando peticiones a nuestra Rest API, para obtener los datos de forma más específica y así ahorra a la APP carga de procesos.

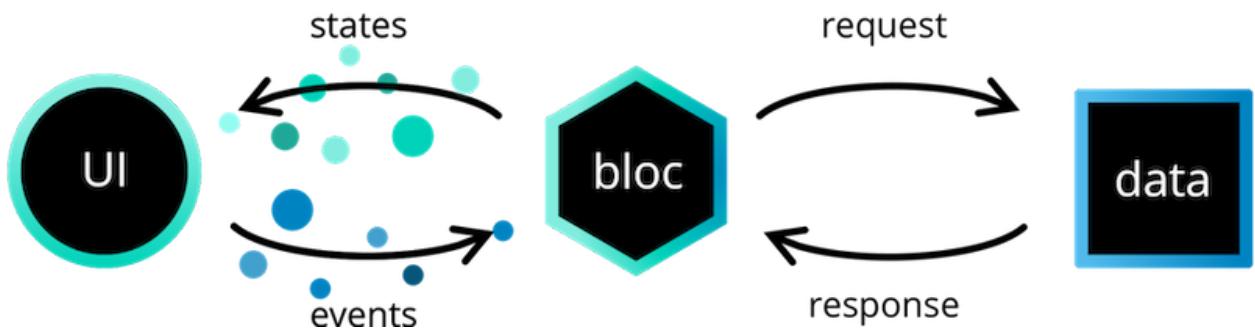
Elementos innovadores

Flutter BloC

Flutter BloC como manejador de estado y estructura.

Sin ninguna duda lo que más tiempo he dedicado, aproximadamente dos meses aprendiendo con mini proyectos de forma casi diaria para conseguir al final saber montar mi propia APP con estructura BloC.

Con BloC se consigue un modelo MVC con la parte UI muy diferenciada de la parte lógica y el manejo de eventos y estados en la APP, consiguiente así control total sobre esta en cualquier contexto.

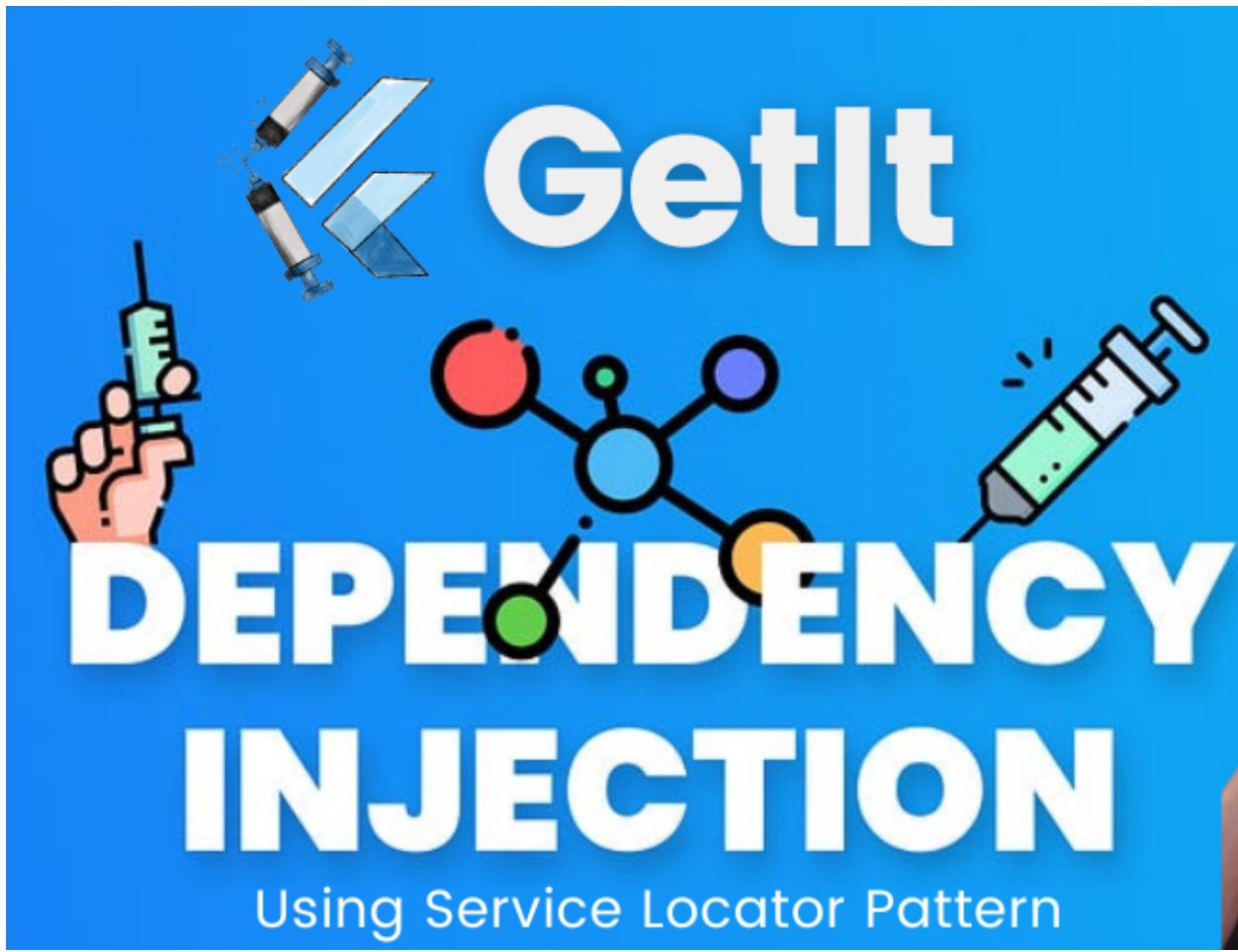


Inyección de dependencias con GetIt

Tras haber conocido un poco su existencia en la empresa, me he lanzado a investigar más a fondo sobre su uso y es lo mejor que pude hacer.

Configurando en el inyector todas las dependencias del proyecto, como singletones, podemos acceder a estas desde cualquier ámbito de uso.

Muy útil cuando se trabaja con Bloc porque por ejemplo se puede acceder al estado de un Bloc desde cualquier punto de la aplicación sin necesidad de instanciar uno nuevo.

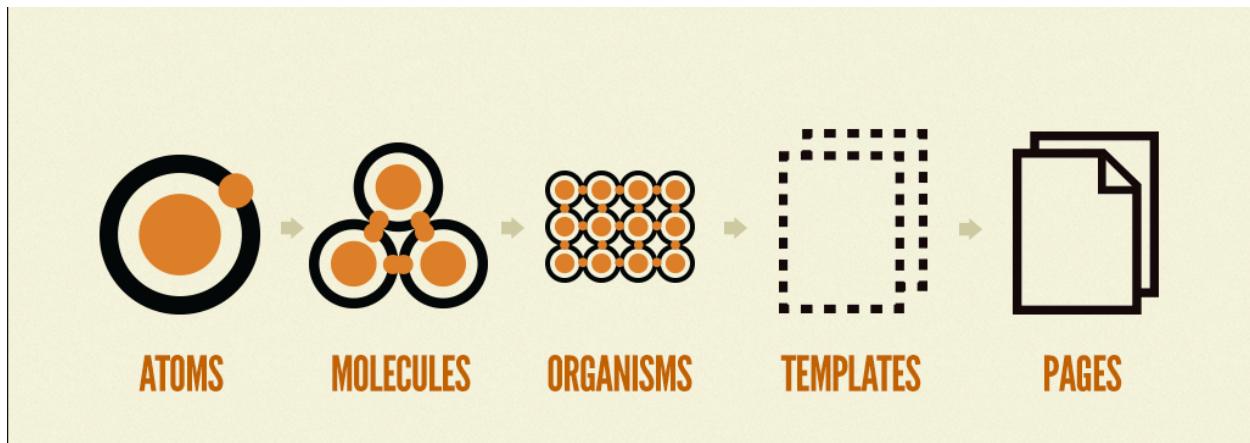


UI Atomic Design

Es una metodología de diseño que consiste en modularizar todas las partes visibles de la APP en medida de lo posible.

Se divide en átomos, moléculas, organismos, plantillas y páginas.

Es decir, pequeños trozos de código moldeable mediante parámetros que se va repitiendo en el desarrollo de la APP o que simplemente es conveniente tener separado para su mantenimiento futuro.



Generative AI

Con el boom de la inteligencia artificial, he decidido subirme al carro y aprender junto a la empresa en prácticas un poco de ingeniería de prompts para sacarle el mejor fruto posible a la generación de contenido con IA.

El 95% de las imágenes e iconos de la APP han sido generadas mediante inteligencia artificial. Obteniendo resultados sorprendentes con un buen manejo y dedicando tiempo.



ImagePicker, acceso a ficheros del dispositivo y cámara.

Leí sobre este paquete y decidí implementarlo para ver que tal funcionaba. A la hora de hacer registro normal sin imagen proporcionada por el perfil de Google, se solicitará añadir una imagen al usuario, bien seleccionándola de su galería o bien directamente abriendo la cámara y tomando una imagen.

Esto también conlleva el haber tenido que indagar en la gestión de permisos del dispositivo.

Geolocalización por GPS del usuario. Google Maps y Google Cloud

El usuario para poder visualizar el mapa, debe aceptar los permisos de localización ya que partimos de su ubicación actual, usando los servicios de Google Maps y Google Cloud. Por desgracia para usar estos servicios hay que pagar cuantías por uso, actualizando el proyecto al plan blaze, aunque tenemos 90 días de prueba con 270 euros de crédito disponible, lo cual es suficiente para este prototipo.



Google Maps

Localización l10n

Algo que he aprendido en estos meses es que es importante saber que a más público pongas tu APP en disposición, mejor será el ratio de descargas / compras. Por lo que, para ampliar conocimiento, la APP está disponible en Inglés también, usando AppLocalizations para ello.



HomePage dinámica respecto a meteorología en Jaén.

En mis proyectos siempre me gusta incluir algo que he aprendido por mi cuenta o me ha gustado como queda y puede verse bien. En este caso, aprendiendo BloC hice una mini APP que hacía llamadas a la API de OpenWeather.

¿Cómo era la forma de incluir esto? Pensando, quise usar todo el potencial de BloC y hacer una HomePage dinámica respecto del tiempo actual en Jaén, es decir, a parte de mostrar la temperatura, el fondo de la pantalla cambiará dependiendo del día que haga.

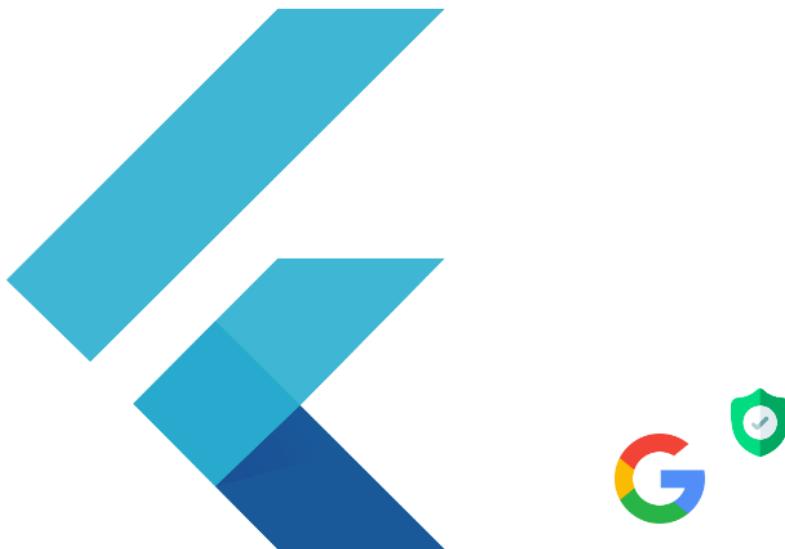
Acceso a API / Servidor a sólo usuarios autenticados.

Para que la API de respuesta, el dispositivo que manda la APP tiene que estar registrado y autorizado en Firebase.

Para ello, cada petición va acompañada de un token OAuth en formato Bearer, el cual tiene el usuario logueado en Firebase.

En la API, un middleware controla todas las peticiones entrantes y comprueba que el token esté autorizado y sea correcto.

Si no es correcto o no está autorizado devuelve Unauthorized como respuesta.



Variables de entorno, DotEnv

Bajo ningún concepto podemos almacenar APIs keys ni contraseñas en el código fuente de la APP, ya que es muy fácil de recoger esta información, para ello, se usan variables de entorno, que hay que configurar y acceder a ellas. Para ello uso archivos .env y DotEnv como paquete para acceder a estas.

Very Good CLI, generador de proyectos Flutter.

Consiste en un generador de proyectos para Flutter, esto genera una estructura base y simplificada, con l10n implementado, bases para flutter y unit testing.

<https://cli.vgv.dev/>

App Features

- **Multi-Platform Support** - Support for iOS, Android, Web, and Windows (macOS and Linux coming soon!)
- **Build Flavors** - Multiple flavor support for development, staging, and production
- **Internationalization Support** - Internationalization support using synthetic code generation to streamline the development process
- **Sound Null Safety** - No more null-dereference exceptions at runtime. Develop with a sound, static type system.
- **Bloc** - Layered architecture with `bloc` for scalable, testable code which offers a clear separation between business logic and presentation
- **Testing** - Unit and widget tests with 100% line coverage (integration tests coming soon!)
- **Logging** - Extensible logging to capture uncaught Dart and Flutter exceptions
- **Very Good Analysis** - Lint rules for Dart and Flutter used internally at Very Good Ventures
- **Continuous Integration** - Lint, format, test, and enforce code coverage using GitHub Actions

Estudio previo. Prototipo.

En cuanto a la base de datos, decido usar Firebase y todos sus servicios ya que es lo más intuitivo debido a que usaremos si o si servicios de Geolocalización de Google.

A parte, decido usar Firebase ya que quiero investigar cómo activar las opciones de analíticas de la APP, para ver cómo funciona Crashlytics por ejemplo, lo cual ayuda mucho cuando se tiene una APP desplegada en producción.

Tuve la idea en un principio de establecer algún sistema / algoritmo que mostrase un autobús a modo Uber APP, pero conforme avanzaba el desarrollo lo veía menos viable por varios motivos, no sé cuántos autobuses se mueven en la línea, el mapa ya tiene mucha carga de asincronía, por lo que añadir muchos autobuses actualizando estado puede ser mucha carga en este caso.

La forma en que emitiría coordenadas GPS para escucharlas mediante un Bloc o directamente un Stream a un campo de Firebase la tenía bastante clara y dediqué varios días a investigar formas y ver opciones, descubriendo aquí las cloud functions, las cuales tengo reservadas para implementar en otras APPs / APIs. Con esto se podría haber logrado el resultado, pero no tenía claro cuántas coordenadas emitir ni en qué momento emitirlas, por lo que decidí centrarme en el resto de la APP.

Para la API, me he basado en lo que hemos realizado en clase y en proyectos anteriores, ya que considero que es una base y estructura muy sólida y sencilla de aplicar.

Generando los métodos y endpoints que necesito, consigo una API robusta y sencilla de mantener.

Esta API, está desplegada en Vercel, los cuales proporcionan un servicio excelente sin interrupciones.

Para el resto de peticiones a la base de datos o los servicios de Google Auth, Geolocator, Maps... La misma APP hace el trabajo usando métodos directos de Auth ya que es más reactivo y rápido para el usuario, un onboarding ligero y rápido es lo que buscaba.

Para el diseño de la APP:

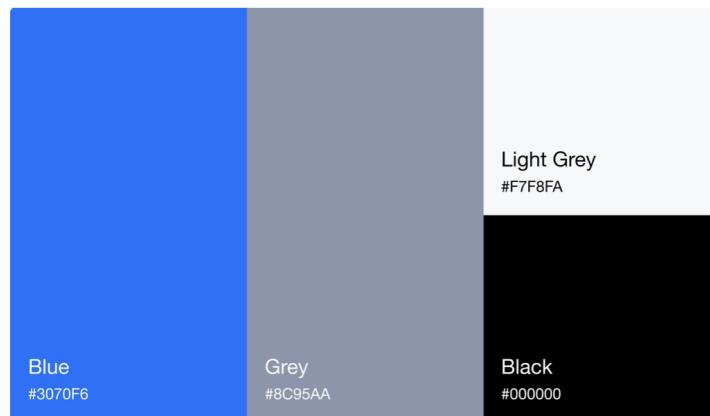
Principalmente me he basado en [Dribbble](#), donde se suben diseños y se venden, pero yo he ido mirando para aplicar estilos y estructuras variadas en mi APP.

Para los colores y el desarrollo de la interfaz del Login, registro... etc me he basado en este diseño: [Colores y onboarding](#)

Sustituyendo el color azul por el verde que hay en la APP, logro el resultado de la APP.

El resto de los átomos, pantallas y flujos de la APP no son de ningún sitio externo.

Colors & Typography

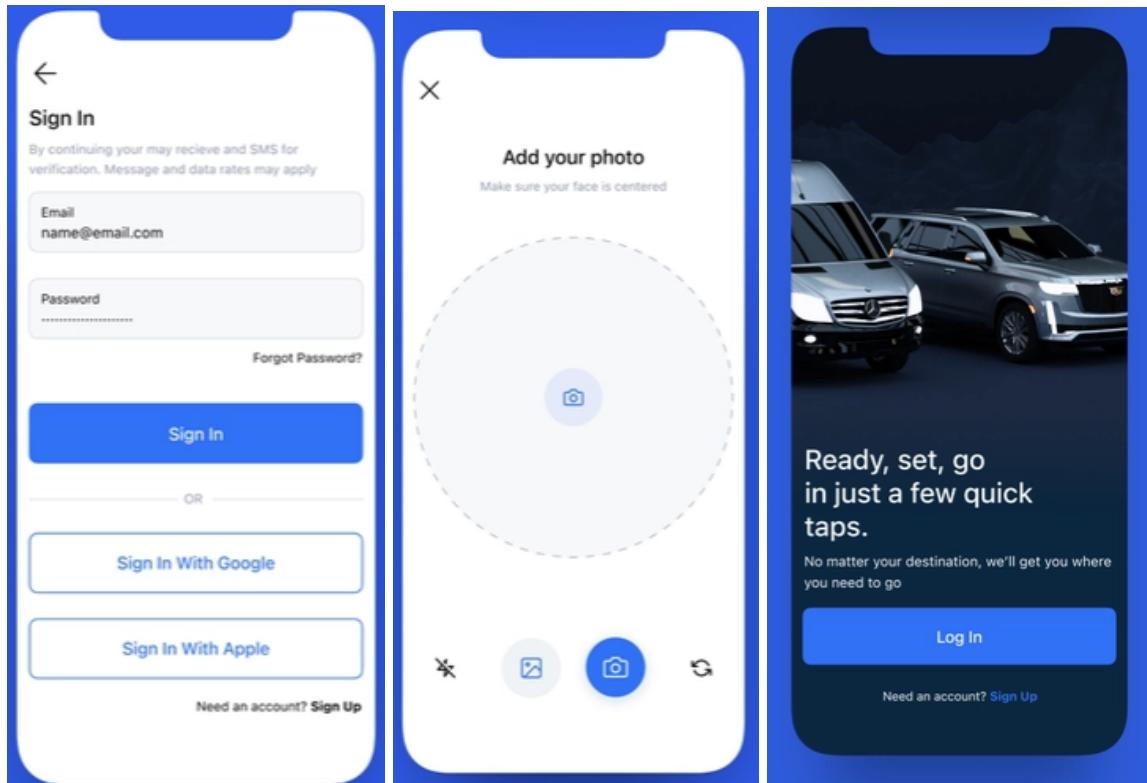


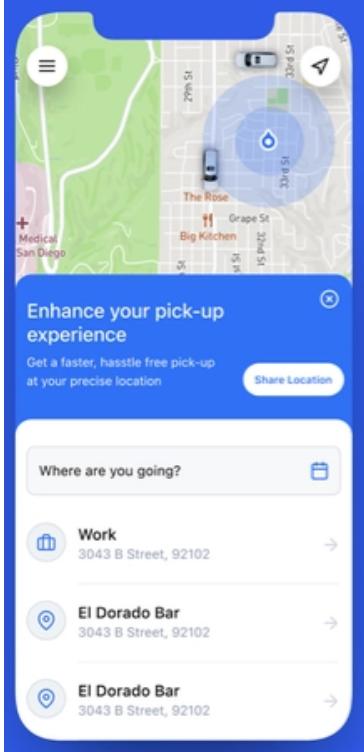
Tipografía de la APP

Poppins
Designed by Indian Type Foundry, Jonny Pinhorn

Whereas disregard and contempt for human rights
have resulted

Diseño base de Dribble





El resto del diseño de la APP es fruto de mi imaginación, ya usando todos los elementos que tengo disponibles al haber generado todos estos componentes.

Como funcionalidades extra y que la visualización de los horarios sea más intuitiva, decidí añadir timers, calculando el tiempo restante a cada salida de autobús en cada parada, parte de esto lo hace la API.

La última funcionalidad fue la de la Home Page dinámica, que, analizando la respuesta de OpenWeather, logro generar un mapa donde según el tiempo que haga, muestro una imagen u otra de fondo, lo cual da un efecto muy fresco cada vez que se abre la APP en diferentes días.

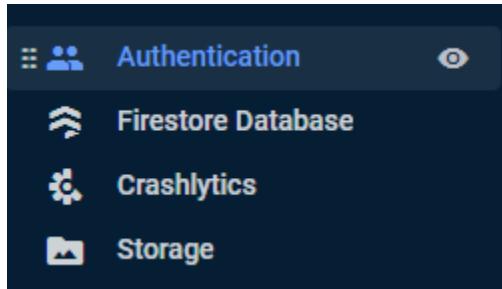
Backend. Firebase as Service + NodeJS Express API Rest + Google Cloud

Esta combinación, me proporciona todos los elementos y respuestas necesarias para que la APP sea muy escalable y modificable.

Firebase as Service.

Con esto se refiere a que se accede a Firebase como servicio desde todos los ámbitos del proyecto, tanto desde la API como desde la APP directamente.

De Firebase, utilizo los siguientes servicios para autenticación de usuarios, base de datos NoSQL, almacén de imágenes de usuarios e informes de errores y crasheos en la APP.



Servicio de Autenticación de usuarios.

Se encarga la APP directamente, hay un BloC que se encarga de chequear el estado del usuario y emitir estados dependiendo de este, aumentando así la velocidad de respuesta y reactividad a logins, registros, envíos de correos y verificación de usuarios.

Está habilitado sólo el login con correo / contraseña y con Google.

En la APP está disponible el botón para Apple, pero no lo he activado ya que no dispongo de dispositivo Apple para pruebas.

Authentication					
Users	Sign-in method	Templates	Usage	Settings	Extensões <small>NUEVA</small>
<input type="text"/> Buscar por dirección de correo electrónico, número de teléfono o UID de usuario Agregar usuario C ⋮					
Identificador	Proveedores	Fecha de creación	↓	Fecha de acceso	UID de usuario
adrianlopez95@hotmail.es		11 jun 2023		11 jun 2023	OjddqopsodYGrfrNExVaHimpbjm32
adrianlpr95@gmail.com		11 jun 2023		11 jun 2023	f12kvrrZXrdzAVqzaaxAlcHv8wC3
adrianlopezruiz.dev@gmail...		10 jun 2023		10 jun 2023	igz67kpRVyalHnIFGUL080YFmgf1
Filas por página: 50 1 – 3 of 3 < >					

Base de datos

Firestore es la elección, ya que proporciona una base de datos NoSQL muy robusta y escalable.

La base de datos consta de 3 modelos o colecciones(tablas) base.

- **Users:** esta colección es la suplementaria al usuario generado por Authenticacion, es decir, aquí genero y almaceno el perfil final de usuario. Guardando sus datos básicos, su última posición conocida en Latitud y Longitud, su lista de favoritos...

users	OjddqopsodYGfrNEXvaHlmpbjm32
+ Agregar documento	+ Iniciar colección
0jddqopsodYGfrNEXvaHlmpbjm32	+ Agregar campo
igz67kpRVyalHn1FGUL	createdAt: 1686494303248
l12kvrrZXrdzAVqzaax	email: "adrianlopez95@hotmail.es"
	favoriteStops
	0 "gvgkckN3wwnSikfkbHhm"
	id: "OjddqopsodYGfrNEXvaHlmpbjm32"
	lastLocation
	0 37.7173367
	1 -3.9747583
	name: "adri"
	profileImage: "https://firebasestorage.googleapis.com/o/profile/d1201.appspot.com/o/profile?alt=media&token=e4c3ba2f-55"
	updatedAt: 1686494442340

- **Línes:** En esta colección, se almacena la línea o futuras líneas que disponga la APP. De ellas, almaceno el nombre de la línea, su fecha de creación y el horario completo, como un árbol de Mapas que acaban en una lista de horas en formato String, la cual es la que se usa para desplegar los horarios dinámicos en la APP.

```

{
  "lines": [
    {
      "_id": "GOVQeo7St9oYvT00tKH",
      "createdAt": 1686150020536,
      "name": "M02-1 - Jaén",
      "schedule": {
        "toJaen": {
          "saturdays": ["07:50", "08:50", ...]
        },
        "sundays": {
          "jaen": ["07:50", "11:50", "14:50", "19:20"]
        },
        "martos": ["07:00", "11:00", "14:00", "18:30"]
      }
    }
  ]
}
  
```

Como se puede observar, el campo **schedule**, almacena **2 mapas de mapas**, los cuales son la **dirección de la línea**, hacia **martos** o hacia **Jaén**.

Cada dirección tiene unos **“días”**, ya que **el horario varía según el tipo de día**.

Y estos “días” **almacenan los horarios** en las diferentes paradas, como **listas de horas**.

- **Stops:** Esta colección almacena los datos básicos de las paradas de autobús, como su nombre, línea a la que pertenece, dirección y punto geográfico, para después pintarla en el mapa.

 stops	 8zAFkiB0905ohCJRyikh	
+ Agregar documento	+ Iniciar colección	
8zAFkiB0905ohCJRyik	+ Agregar campo	
SPZ9nohuyVUKPDtykCU	databaseName: "jaen"	
gvgkckN3wwnSikfkbHh	line: "GOVQeo7St9oYvT0OtKHI"	
pKJuvjuksC7oy0Ae82D	location: [37.77176635325846° N, 3.7866687868666133° W]	
	name: "Jaén"	
	street: "Plaza Coca de la Piñera"	

Firebase Storage

Firebase también nos da opciones para almacenar ficheros o documentos.

Para ello tenemos storage, que en mi caso, solo lo uso para almacenar las imágenes que los usuarios suben tras el registro.

Esto genera un link, que durante el registro, almaceno en el documento correspondiente al usuario como link de acceso al “bucket” y a la foto correspondiente.

gs://fb-auth-bloc-d1201.appspot.com > profileImages				
<input type="checkbox"/>	Nombre ↑	Tamaño	Tipo	Modificación más reciente
<input type="checkbox"/>	 Adri1686088451388	4.69 MB	image/jpeg	6 jun 2023
<input type="checkbox"/>	 Adri1686176254833	1.01 MB	image/jpeg	8 jun 2023
<input type="checkbox"/>	 Adrian1686427656559	4.66 MB	image/jpeg	10 jun 2023
<input type="checkbox"/>	 adri1686494355304	407.63 KB	image/png	11 jun 2023

CrashLitics

Firebase nos ofrece un servicio de recopilación de errores y crasheos de nuestras APP, brindando unos resúmenes muy detallados de los errores que ocurren en los dispositivos de los usuarios, desde el tipo de dispositivo, el usuario específico, versión de la APP, versión de Android y muchísimos más datos.

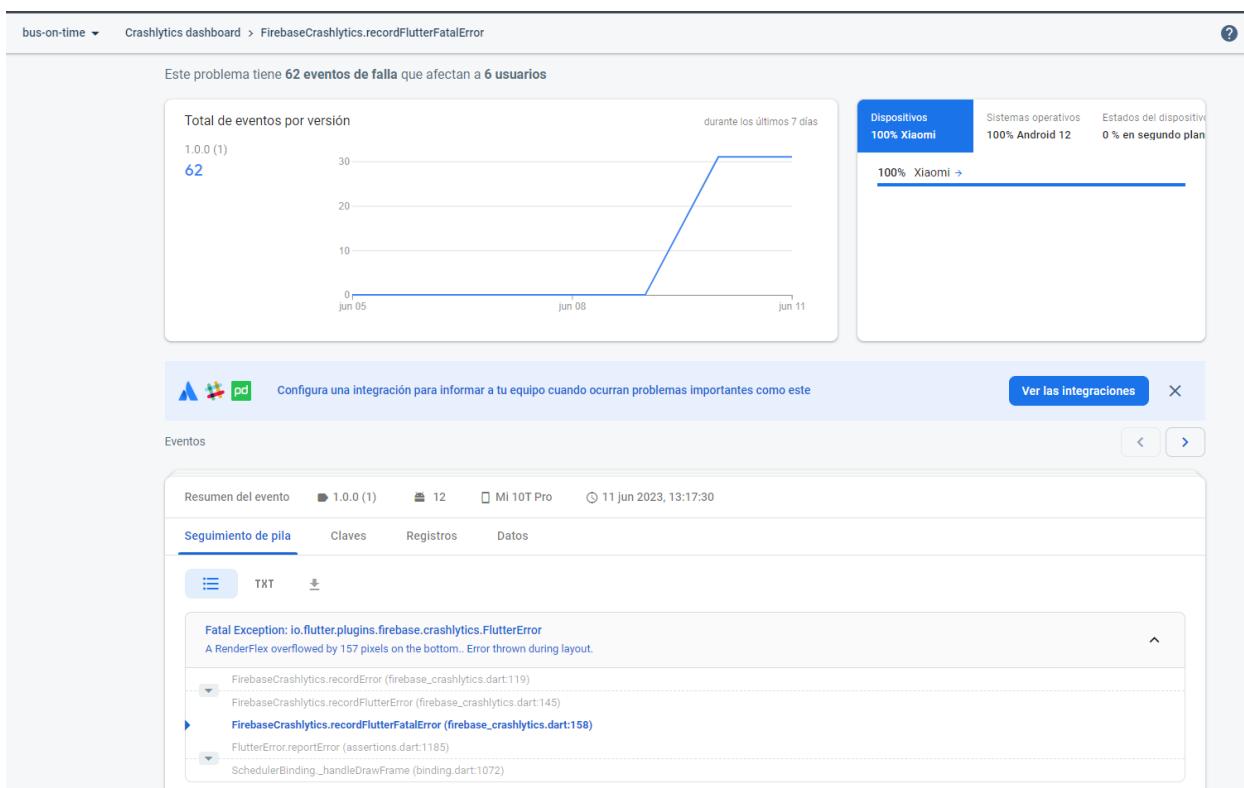
Estos datos nos ayudan a ser muy eficaces resolviendo incidencias y bugs en la APP o en los usuarios.

Para implementarla, se añade en el void main de la APP una línea, la cual abre un Stream en la APP que escucha todos los throws y manda analítica a Firebase.

The screenshot shows the Firebase Crashlytics dashboard for the app 'bot_main_app (android)'. At the top, there's a summary section with 'Estadísticas sin fallas' (Statistics without errors) and 'Tendencias' (Trends). The trends chart shows 157 errors and 10 users from June 5 to June 11. Below this, there's a section for querying data in BigQuery and a 'Problemas' (Problems) section. The 'Problemas' section lists open issues, including a new error related to Flutter's RenderBox size and another about FirebaseCrashlytics recording a fatal error. A search bar at the top right allows users to search by title, subtitle, or key words.

Estado del problema	Detalles	Versión	Eventos	Usuarios
Abiertos	Falla com.example.ver... package:firebase_crashlytic... Problema nuevo fallas que ocurren de forma reiterada	1.0.0 - 1.0.0	62	6
	FirebaseCrashlytics.recordFlutterFatalError io.flutter.plugins.firebaseio.crashlytics.FlutterError - A RenderFlex overflowed by 46 pixels on the right.. Error thrown durin...			
	Falla com.example.verysgoodcore.bot_main_app package:flutter/src/rendering/box.dart:1966 Problema nuevo	1.0.0 - 1.0.0	34	3
	RenderBox.size io.flutter.plugins.firebaseio.crashlytics.FlutterError - RenderBox was not laid out: RenderViewport#f7553 NEEDS-LAYOUT ... → Más de 10 variantes			

En estas imágenes se pueden observar todos los datos y estadísticas que nos ofrece.



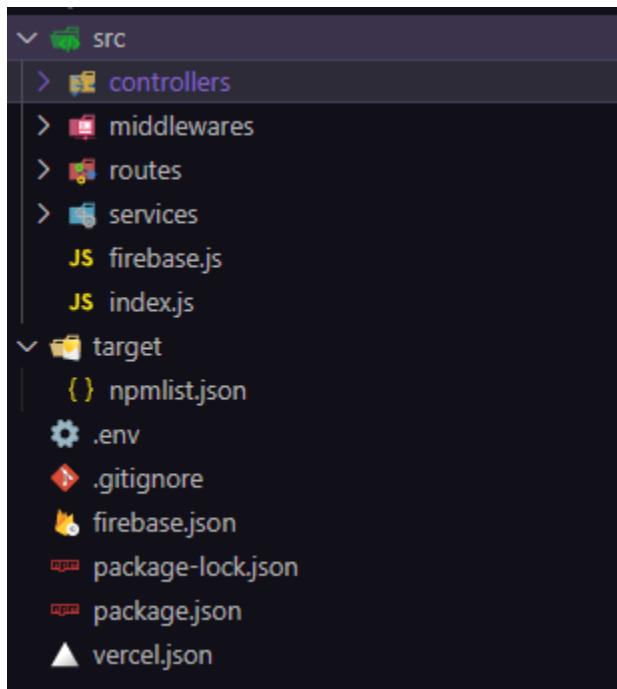
API Rest, NodeJS + Express

La API, está construida a partir de la base que hemos estado fabricando durante el curso, obteniendo así una API bastante robusta y fácil de mantener.

Estructura

La API, también se basa en arquitectura MVC (Model - View - Controller).

Con esto conseguimos modularizar y separar la lógica de tratamiento de los datos que obtenemos de la base de datos con la lógica de enrutamiento y respuestas hacia la APP.



La api cuenta con control de excepciones completo, véase la imagen.

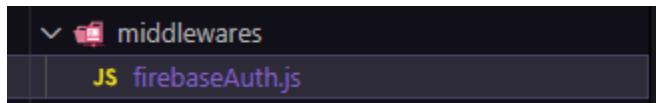
```
//Create one
const addStop = async (req,res) => {
    const { body } = req;
    if (
        !body.name ||
        !body.location ||
        !body.street ||
        !body.line
    ) {
        res.status(400).send({
            status: 'FAILED',
            data: { error: 'One of the keys is missing.' }
        });
        return;
    }
    const newStop = {
        name: body.name,
        line: body.line,
        location: body.location,
        street: body.street,
        createdAt: new Date().toLocaleString('es-ES'),
    };
    try {
        const createdStop = await addStopToDB(newStop);
        res.status(201).send({
            status: 201,
            data: createdStop
        });
    } catch (error) {
        res.status(error?.status || 500).send({
            status: 'FAILED',
            data: {
                error: error?.message || error
            }
        });
    }
};
```

Se valoran todas las posibles fallas que traiga una petición compleja, en este caso un POST de una nueva parada.

EndPoints

Siempre partimos de la base /api/v1 , para localizar que estamos en la primera versión de la API y establecer un endpoint genérico.

Cada endpoint, usa el middleware firebaseAuth, que se encarga de comprobar si el usuario que lanza la petición está autorizado, mediante el Token Bearer.



Cada colección tiene su propio endpoint, los cuales vamos a detallar:

*No podré añadir capturas de PostMap o ThunderClient, por algún motivo el token que le paso por cabecera me lo rechaza siempre.

- **/api/v1/users :** métodos crud básicos para control de los datos de usuario, tirando peticiones a la colección de firebase.

GET /api/v1/users/:userId

Recibe la id de usuario por parámetro y devuelve toda la información de ese usuario.

PUT /api/v1/users/:userId

Actualiza los datos de un usuario, los datos vienen en el body de la petición.

DELETE /api/v1/users/:userId

Borra un usuario de firestore.

GET /api/v1/users

Devuelve todos los usuarios, no usado ni recomendado usar.

- **/api/v1/lines**

GET /api/v1/lines/todaySchedule

Este método, tiene que recibir los parámetros mediante query parameters.

Params:

lineId : String

stopName : String

direction : String (martos o jaen)

Ejemplo de petición

https://APIURL/api/v1/lines/todaySchedule?lineId=LINEID&stopName=martos&direction=martos

GET /api/v1/lines/:lineId

Devuelve la linea completa por ID.

GET /api/v1/lines/

Devuelve todas las lineas

POST /api/v1/lines/

Inserta una nueva linea, los datos vienen por el body

PUT /api/v1/lines/:lineId

Actualiza los datos de una linea.

DELETE /api/v1/lines/:lineId

Elimina los datos de una linea

- **/api/v1/stops**

GET /api/v1/stops/:lineId

Devuelve los datos de la parada pasada por parametro

GET /api/v1/stops/

Devuelve todas las paradas

POST /api/v1/stops/

Inserta una nueva parada.

PUT /api/v1/stops/:stopId

Actualiza los datos de una parada

DELETE /api/v1/stops/:stopId

Elimina los datos de una parada.

Google Cloud Services

Aunque esté directamente enlazado con Firebase, cuenta con una consola a parte, de momento sólo uso la API de geolocalización y Maps. Aunque se registra todo el tráfico del proyecto.

The screenshot shows the Google Cloud Platform (GCP) dashboard. At the top, there's a message about the free trial credit: "Estado de la prueba gratuita: Te quedan €270.69 de crédito y 61 días. Con una cuenta completa, obtendrás acceso ilimitado a todas las funciones de Google Cloud Platform." Below the header, there's a search bar and a "DESCATAR" button.

The main area is divided into several sections:

- PANEL:** Shows "Información del proyecto" (Project info), including the project name "bus-on-time", project number "1017991836538", and project ID "fb-auth-bloc-d1201". It also has a "AGREGA PERSONAS A ESTE PROYECTO" (Add people to this project) button and a link to "Ir a la configuración del proyecto" (Go to project settings).
- ACTIVIDAD:** Displays a chart titled "API APIs" showing "Solicitudes (solicitud/s)" (Requests per second) over time. The chart shows a peak around 19:30 and another around 20:00. A note below says "Solicitudes: 0.009/s".
- RECOMENDACIONES:** Includes sections for "Estado de Google Cloud Platform" (All services are functioning normally), "Facturación" (Billing information), "Monitoring" (Create my panel, Set up alert policies, Create availability checks), and "API Error Reporting" (No errors found). There's also a "PERSONALIZAR" (Customize) button.
- RESOURCES:** Lists various Google services: BigQuery, SQL, Compute Engine, Storage, Cloud Functions, App Engine, and Trace. Under Compute Engine, it shows VM, GPU, TPU, and disks. Under Storage, it shows multi-regional object storage. Under Cloud Functions, it shows event-based serverless functions. Under App Engine, it shows the managed platform.
- Trace:** A section stating "No hay datos de seguimiento de los últimos 7 días" (There are no tracing data from the last 7 days).

At the bottom, there's a detailed view of API usage statistics:

API APIs y servicios		APIs y servicios		+ HABILITAR APIs Y SERVICIOS																																																		
<ul style="list-style-type: none"> APIs y servicios habilitados Biblioteca Credenciales Pantalla de consentimiento ... Acuerdos de uso de páginas 		<p>Tráfico</p> <p>Errores</p> <p>Mediana de latencia</p>		<p>1 hora 6 horas 12 horas 1 día 2 días 4 días 7 días 14 días 30 días</p>																																																		
		<p>Filtro Filtro</p> <table border="1"> <thead> <tr> <th>Nombre</th> <th>Solicitudes</th> <th>Errores (%)</th> <th>Latencia mediana (ms)</th> <th>95% de latencia (ms)</th> </tr> </thead> <tbody> <tr> <td>Cloud Firestore API</td> <td>348</td> <td>0</td> <td>46</td> <td>65</td> </tr> <tr> <td>Identity Toolkit API</td> <td>151</td> <td>11</td> <td>33</td> <td>481</td> </tr> <tr> <td>Directions API</td> <td>87</td> <td>0</td> <td>84</td> <td>130</td> </tr> <tr> <td>Maps SDK for Android</td> <td>41</td> <td>0</td> <td></td> <td></td> </tr> <tr> <td>Firebase Installations API</td> <td>13</td> <td>0</td> <td>311</td> <td>502</td> </tr> <tr> <td>Token Service API</td> <td>5</td> <td>0</td> <td>24</td> <td>31</td> </tr> <tr> <td>Cloud Storage for Firebase API</td> <td>4</td> <td>0</td> <td>393</td> <td>511</td> </tr> <tr> <td>App Engine</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>App Engine Admin API</td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		Nombre	Solicitudes	Errores (%)	Latencia mediana (ms)	95% de latencia (ms)	Cloud Firestore API	348	0	46	65	Identity Toolkit API	151	11	33	481	Directions API	87	0	84	130	Maps SDK for Android	41	0			Firebase Installations API	13	0	311	502	Token Service API	5	0	24	31	Cloud Storage for Firebase API	4	0	393	511	App Engine					App Engine Admin API					
Nombre	Solicitudes	Errores (%)	Latencia mediana (ms)	95% de latencia (ms)																																																		
Cloud Firestore API	348	0	46	65																																																		
Identity Toolkit API	151	11	33	481																																																		
Directions API	87	0	84	130																																																		
Maps SDK for Android	41	0																																																				
Firebase Installations API	13	0	311	502																																																		
Token Service API	5	0	24	31																																																		
Cloud Storage for Firebase API	4	0	393	511																																																		
App Engine																																																						
App Engine Admin API																																																						

FrontEnd, Flutter

La APP está diseñada en Flutter en su totalidad. El lenguaje utilizado por este es Dart.

El prototipo está desarrollado en la última versión de Flutter y Dart.

Para instalar flutter, es recomendable seguir los pasos detallados en su documentación oficial: [Instalar flutter](#)

Dependencias o Paquetes instalados

Estos son los paquetes que se usan en el desarrollo de la APP.

Cada dependencia tiene un enlace a su documentación oficial para más info.

[awesome_dialog](#): Se encarga de estilizar los mensajes de error, info, pop ups de formularios

[flutter_bloc](#): Manejador de estados de la APP

[cloud_firestore](#): Todas las funciones para acceder a las funciones de firestore desde la APP.

[convex_bottom_bar](#): Ayuda para generar la barra de navegación inferior, la cual el estado se controla mediante BloC en mi caso.

[equatable](#): Permite comparar objetos entre sí, evitando el problema de comparar referencias en memoria, Object == Object => true

[firebase_auth](#): Da acceso a todos los métodos y funcionalidades de Authentication, controlar el estado del usuario, guardar en cache su sesión... etc

[firebase_core](#): Es la base de Firebase, sin esta no podemos usar ninguna app de firebase.

firebase_crashlytics: Permite añadir crashlytics a la APP, esta empieza a mandar datos a Firebase en cada exception lanzada.

firebase_storage: Proporciona todas las funciones para acceder al storage de Firebase.

flutter_dotenv: Permite acceder a variables de entorno y utilizarlas en cualquier parte del código.

flutter_localizations: Acceso a la parte de localizaciones de Flutter.

sdk: flutter

flutter_native_splash: Permite modificar de manera sencilla el inicio de la APP, es decir, si se muestra una imagen, un video o animación durante la carga de la APP.

flutter_polyline_points: Mediante una API, nos devuelve una lista de Latitudes y longitudes, esto nos ayuda a generar una ruta entre dos o varios puntos.

flutter_svg: Para cargar archivos SVG o animaciones json.

geolocator: Proporciona servicios necesarios para obtener la localización del dispositivo, enviar datos en tiempo real de ubicación... etc

get_it: Inyector de dependencias. Al iniciar la APP, todas las declaradas se cargan de la forma que hayamos indicado en el setup.

go_router: Navegación declarativa, proporciona más flexibilidad por si la APP escala a web, podemos obtener rutas tipo web.

google_maps_flutter: Proporciona el Widget de GoogleMaps y todas sus funcionalidades

google_sign_in: Proporciona los servicios de google para hacer login con cuentas de Google.

http: Librería para hacer peticiones HTTP / HTTPS

image picker: Proporciona funcionalidades para acceder a la galería de imágenes o a la cámara del usuario.

permission handler: Ayuda a tener todos los permisos necesarios de la APP necesarios, así como solicitar estos y ver su estado antes de realizar ciertas operaciones. Como abrir el mapa.

toggle switch: Librería que proporciona diferentes estilos de botones de tipo Switch.

validators: Proporciona métodos para validar Strings, por ejemplo, isEmail

video player: Instancia un reproductor de video con posibilidad de añadir controles. Lo uso para el Splash Screen.

Flujos de la APP

La APP la podemos dividir en 2 flujos base:

- **Usuario nuevo / no logueado**, debe proceder o bien a hacer login o a registrarse.
 - **Usuario con estado logueado**, es decir, previamente usó la APP y el estado de esta queda en memoria, evitando nuevos logins.
-

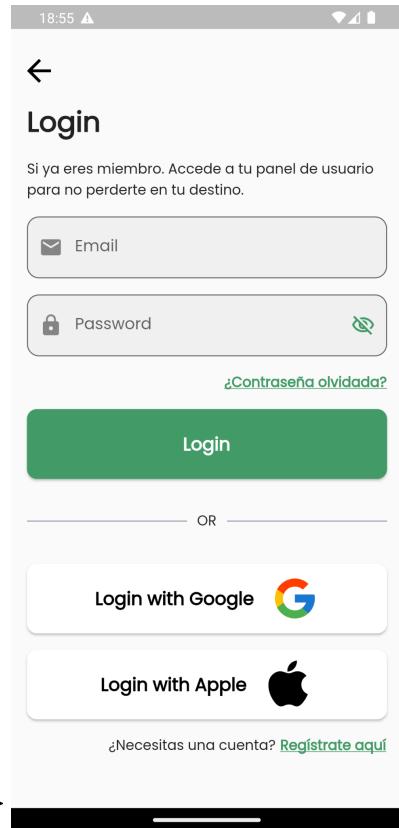
Ahora en detalle:

- **Usuario nuevo / no logueado**

Landing Page



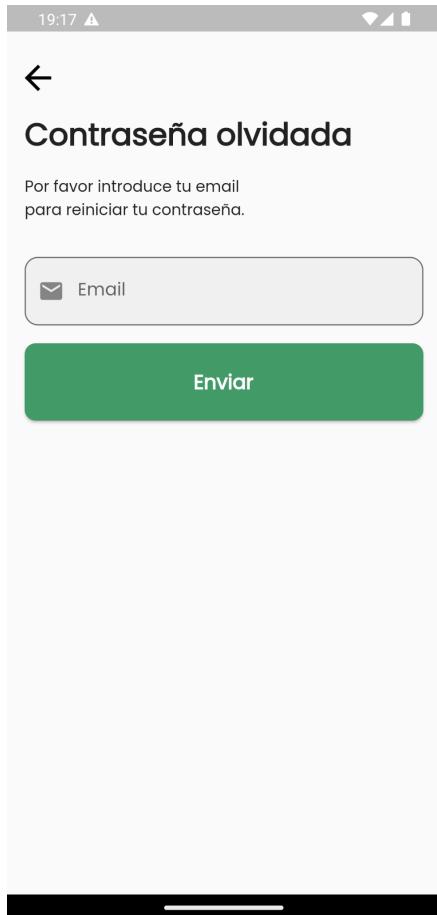
Register Page



El usuario ve la **Landing Page** al entrar a la APP después del **Splash**.

Una vez en la Landing, elige si hacer **login o registrarse**, aunque en la LoginPage **también existe la posibilidad de registrarse**.

Aquí se encuentra también el flujo hacia **contraseña olvidada**.



Este flujo acaba aquí, una vez recibido el email **depende del usuario volver al login**.

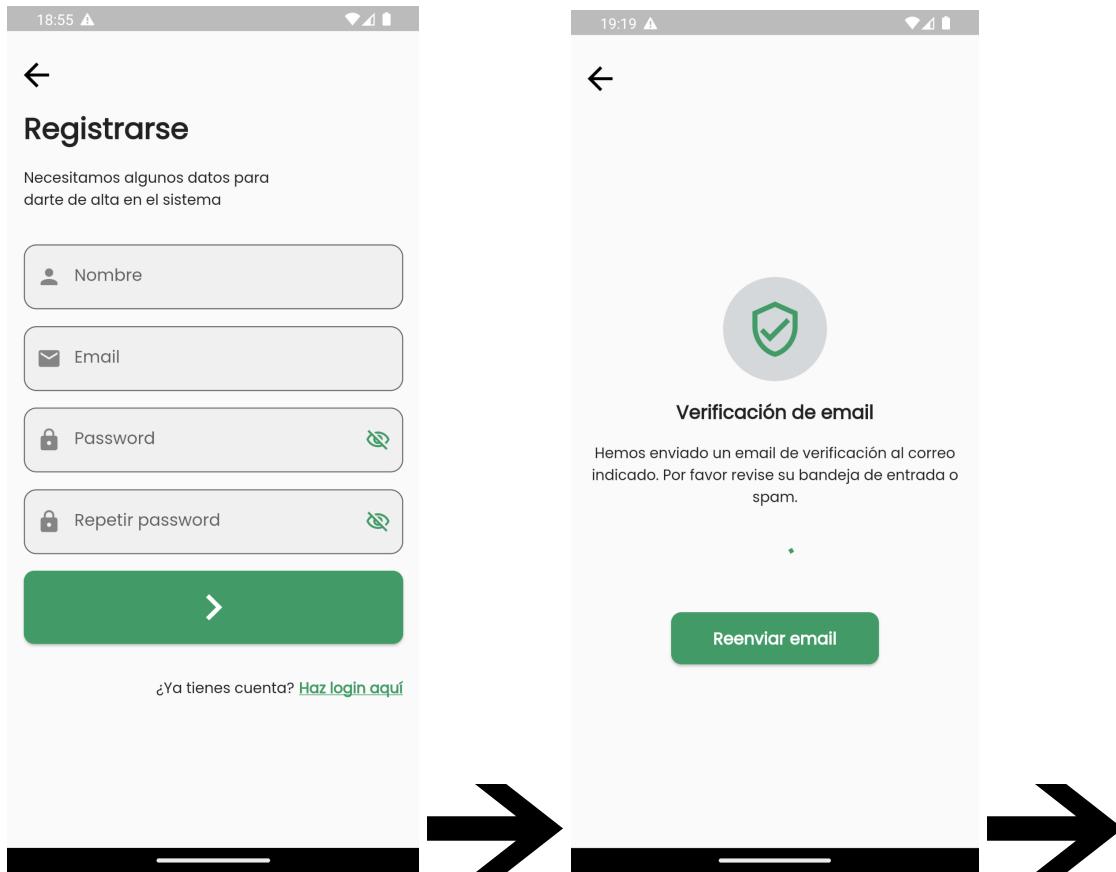
No hay redirección.

Ahora hay 2 opciones.

Google Login / Registro

Si hace login con **Google**, todo el proceso de registro **se salta** y va directo a la **Home Page**.

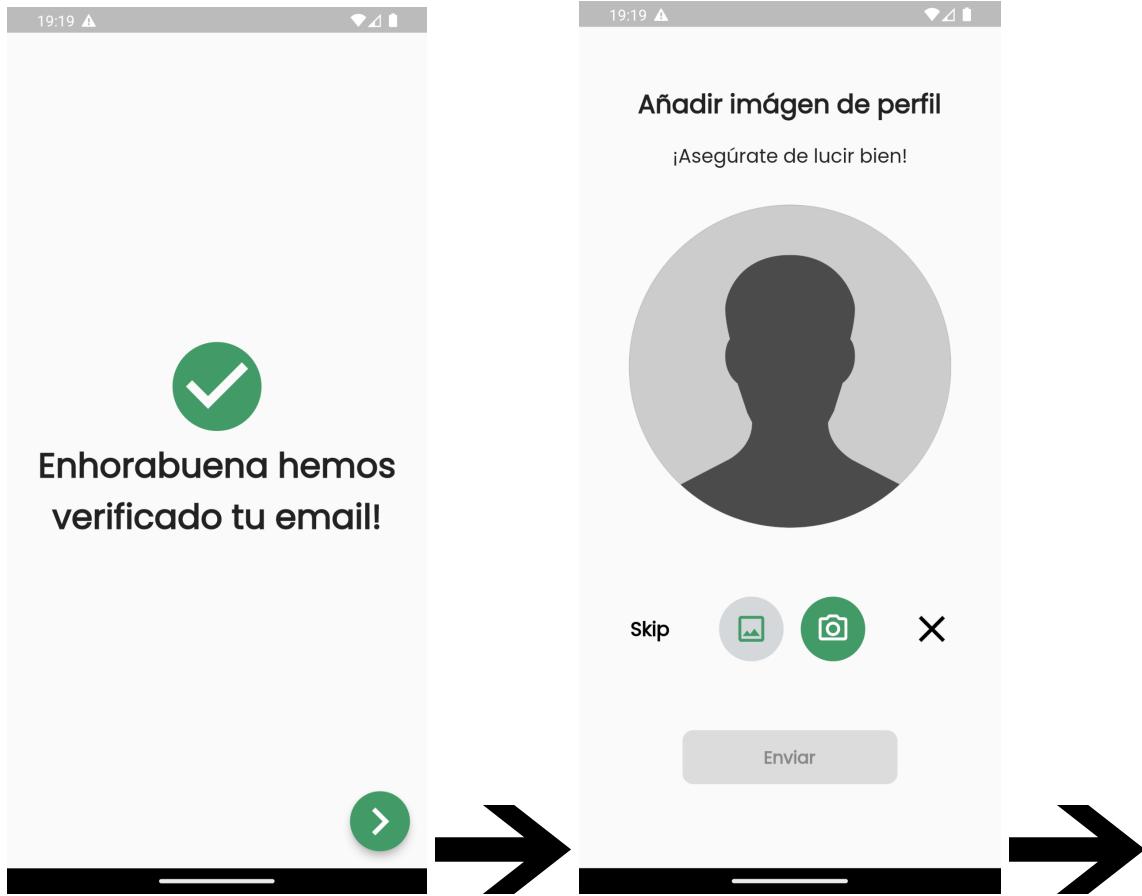
Si elige registrarse...



Una vez el usuario revise el email y verifique, pasamos de screen.

Si el usuario **no verifica su email**, al intentar hacer login volverá a la página de verificación.

Confirmación de verificado y selección de imagen.



Hay **varias opciones** ahora, si presiona en “**Skip**” o “**Saltar**”, no se **aplica una nueva imagen de perfil** y se va a la HomePage ya **logueado**.

Si se selecciona un tipo de imagen, se cargará y al enviar, se actualizará y se redireccionará al HomePage.

Aquí se usa el paquete **Image Picker** que he mencionado antes.

HomePage y Navbar



Ahora el usuario **ya puede navegar libremente por la APP**, no hay flujo establecido a partir de aquí.

El resto de pantallas las vemos en la siguiente sección, cuando un usuario ya está logueado desde caché.

El fondo de pantalla de esta página varía dependiendo de la meteorología del momento en Jaén.

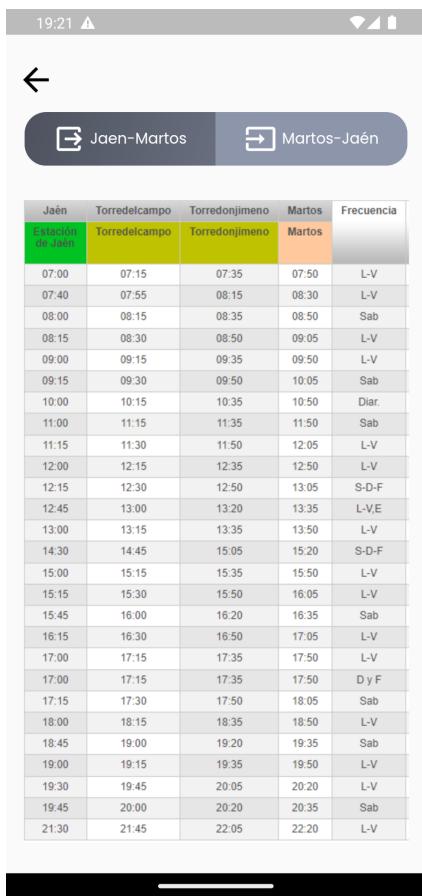
Los datos que se muestran en la HomePage son la temperatura actual, un mensaje de bienvenida al usuario y los botones de redirección al resto de la APP.

- **Usuario logueado en caché, vuelve a abrir la APP**

Si el login no ha expirado en Firebase, va **directo a la HomePage**.

Detallamos ahora las pantallas existentes y las funcionalidades.

Horario



Para no perder la esencia de mirar el lioso horario que el consorcio proporciona, se ha añadido a modo información adicional.

Es una imagen, pero tiene una funcionalidad, se puede zoomear.

Para ello se usa el Widget **InteractiveViewer**

Con un switch button permitimos navegar entre ida y vuelta.

Favoritos

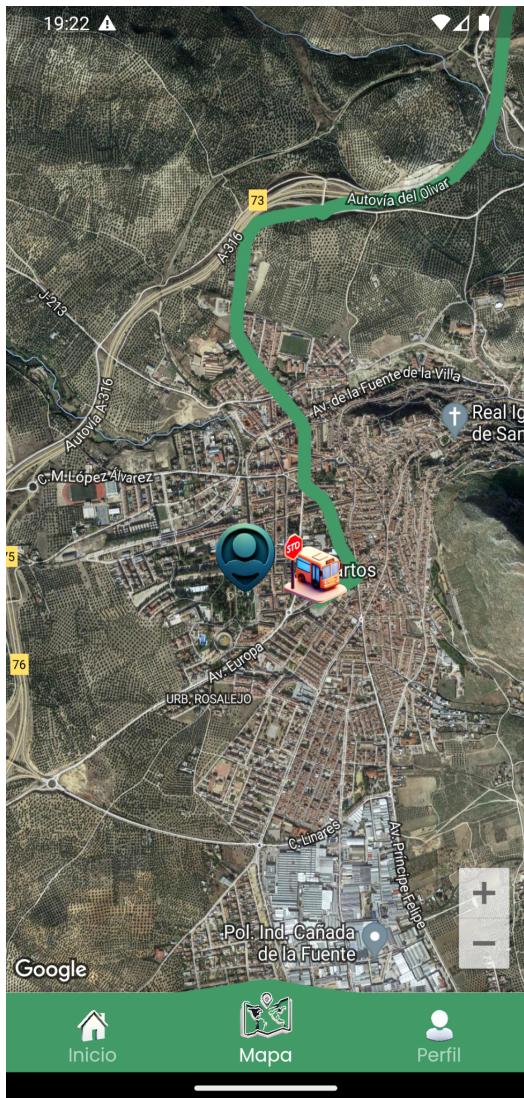


En esta screen, como se observa, se **detallan las paradas** de autobús que el usuario ha **añadido previamente a favoritos**.

En cada una, a parte de todos los datos de la parada, se muestra el **tiempo restante** hasta la salida del próximo autobús desde esa parada hacia la dirección que indica.

Estas tarjetas son un widget **Dismissible**, si las deslizamos hacia la derecha, podemos **borrar de favoritos la tarjeta**.

Mapa



El mapa **sólo se mostrará** si el usuario ha aceptado los **permisos de ubicación**, que se le preguntan al navegar al mapa por **primera vez**.

Una vez cargado, ocurren variastareas asíncronas que cargan:

- Ubicación del usuario.
- Ubicación de las paradas.
- Datos base de las paradas.
- Generación de rutas entre paradas.

Ahora el usuario, al **tocar en cualquier parada**, se desplegará la **información** y el **horario dinámico** de esta, que detallamos a continuación.



Al hacer **tap en una parada** se despliega esta **ventana**, en la cual vemos el **horario dinámico** según la **dirección** en la que queramos ir.

Es decir el próximo autobús hacia Martos desde Torredonjimeno sale a las 11:35 y el próximo hacia Jaén a las 11:15.

Al lado de esta información aparece un **temporizador** que indica cuánto queda para la salida.

En esta ventana también el usuario es donde selecciona si es su favorita o no, dando tap en el corazón.

Perfil



En esta página se ven datos básicos del perfil del usuario y se le brindan ciertas opciones de modificación o fin de sesión.

Si elige cambiar contraseña, un email se le enviará para que lo haga vía web.

Propuestas de mejora.

La APP se encuentra en fase de prototipo.

Como propuestas de mejora o **futuras features** tengo un pequeño listado:

- Mostrar en el mapa los autobuses en movimiento.
 - Posibilidad de modificar la imagen de perfil, ha quedado pendiente.
 - Admin section, para poder actualizar los horarios de las líneas desde una interfaz de administración.
 - I0s login y soporte.
 - Firebase Analytics, analizamos con esto los movimientos del usuario dentro de la APP.
-

Bibliografía

- [Flutter Bloc Doc](#)
 - [Pub Dev](#)
 - [How to get user current location](#)
 - [devciencia](#)
 - [Dribble UI example](#)
 - [Bing chat & Bing Create for generative AI](#)
-