

# **SOUVENIRS CADIZ**

# Índice

<b>1. Introducción.....</b>	<b>2</b>
<b>2. Descripción.....</b>	<b>3</b>
<b>3. Instalación.....</b>	<b>9</b>
<b>4. Prototipado.....</b>	<b>9</b>
<b>5. Diseño funcional.....</b>	<b>11</b>
<b>6. Desarrollo.....</b>	<b>13</b>
<b>7. Pruebas.....</b>	<b>15</b>
<b>8. Distribución.....</b>	<b>15</b>
<b>9. Manual.....</b>	<b>15</b>
<b>10. Conclusiones.....</b>	<b>15</b>
<b>11. Índice tablas e imágenes.....</b>	<b>15</b>
<b>12. Bibliografía.....</b>	<b>16</b>

## 1. Introducción

### ❖ Expectativas

La idea de la aplicación es realizar un catálogo de souvenirs online para una empresa, para que los clientes puedan ver todos los souvenir, guardarlos a favoritos, meterlos en el carrito y pedirlos. Esta información le llegará al administrador junto con su cantidad. El administrador a su vez podrá añadir o eliminar nuevos souvenirs.

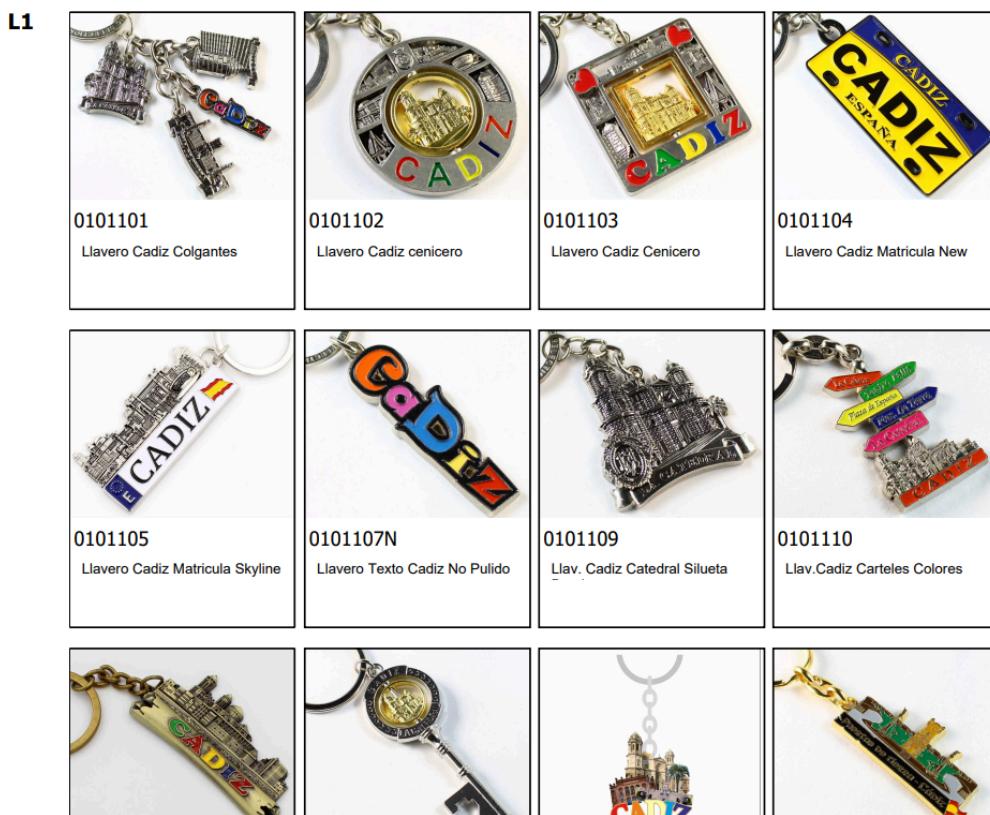
### ❖ Tecnologías

La tecnología usada es **Android Studio** con Jetpack Compose. También he usado **github** para el control de versiones y GitGuardian para la seguridad de Git.

No hay ninguna aplicación que realice esta tarea ya que el sistema de venta de souvenirs no está digitalizado ya que no es de venta online como tal.

### ❖ Idea original

La idea original surge de un PDF que contiene todos los souvenirs con sus respectivas imágenes y nombres

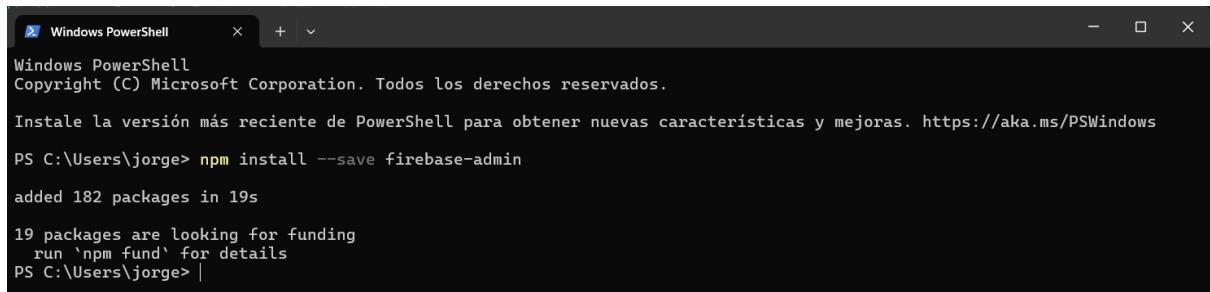


Esta es una captura de un pequeño fragmento del pdf. Me he encargado de sacar todas las imágenes pasándolas a png y todos los nombres en un archivo, el tipo de archivo escogido es csv.

### Actualización:

15/05/2024: He actualizado el proyecto al completo implementando las fotos en la base de datos ya que antes estaban almacenadas en el drawable, también ahora he usado un **JSON** que he cargado en firebase, para ello tuve que:

- 1) Descargar Node js
- 2) pedirle en firebase Configuración > Cuentas de servicio > Generar nueva clave privada (Node js), esto me descarga un archivo que hay que renombrar como key\_service\_account que contiene las claves que te dan los permisos necesarios.
- 3) Crear archivo json al que vamos a acceder en firebase, que tendría los souvenirs.
- 4) upload\_souvenir que se usa para poder dar permisos para abrir los otros archivos.
- 5) Le damos los permisos necesarios

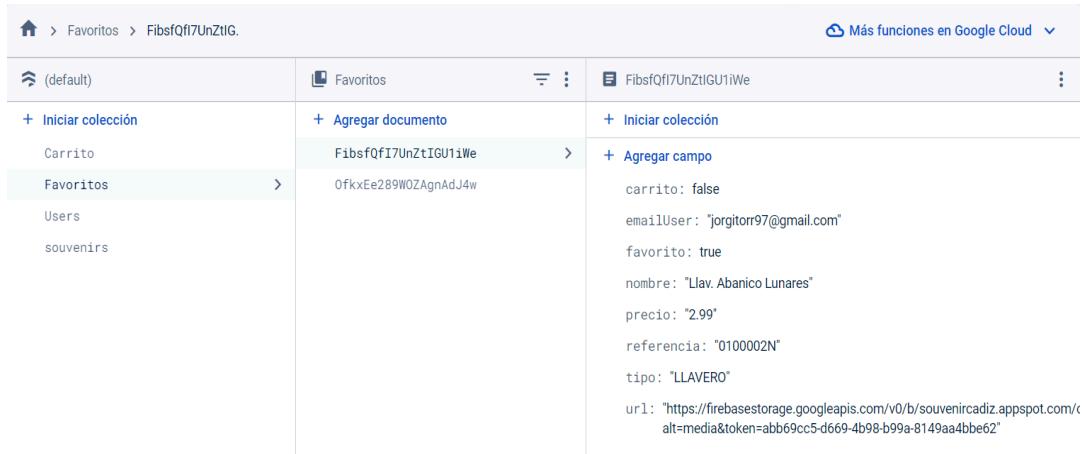


```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows
PS C:\Users\jorge> npm install --save firebase-admin
added 182 packages in 19s
19 packages are looking for funding
  run `npm fund` for details
PS C:\Users\jorge>
```

- 6) Luego ejecutamos con node el archivo javascript que contiene las referencias al resto de archivos.

*node upload\_souvenirs.js*



Favoritos > FbsfQfl7UnZtIGU1iWe

(default)	Favoritos	FbsfQfl7UnZtIGU1iWe
+ Iniciar colección	+ Agregar documento	+ Iniciar colección
Carrito	FbsfQfl7UnZtIGU1iWe	+ Agregar campo
Favoritos	OfkxEe289WOZAgndJ4w	carrito: false
Users		emailUser: "jorgitorr97@gmail.com"
souvenirs		favorito: true
		nombre: "Llavero Abanico Lunares"
		precio: "2.99"
		referencia: "0100002N"
		tipo: "LLAVERO"
		url: "https://firebasestorage.googleapis.com/v0/b/souvenircadiz.appspot.com/c..."

## 2. Descripción

### ❖ Funcionalidades

La funcionalidad de la aplicación es permitir la visualización adaptada al estándar actual de un catálogo de souvenirs. Los usuarios tendrán la posibilidad de registrarse a través del correo electrónico o directamente desde google para guardar souvenir en sus favoritos o en el carrito. Los souvenirs guardados en el carrito podrá pedirlos, esta información le llegará directamente al administrador que tendrá una cuenta específica con la que podrá ver los pedidos y la persona que los ha hecho, esta información la recibirá a través de una notificación. También podrá añadir o eliminar souvenirs.

### ❖ Organización

Lo primero que hice en cuanto a organización fue un [diseño de Figma](#) para intentar acercarme a la idea que tenía en mente e implementarla de la forma más sencilla que pudiera siguiendo una base.

Lo primero que escogí fueron los estilos y empecé por la **paleta de colores**:



Silver



Raisan Black



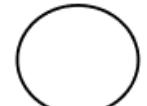
Cerulean



Teal



Redwood



White

Después escogí las **fuentes** que iba a usar:

---

Klee One

Kiwi Maru

*Knewave*

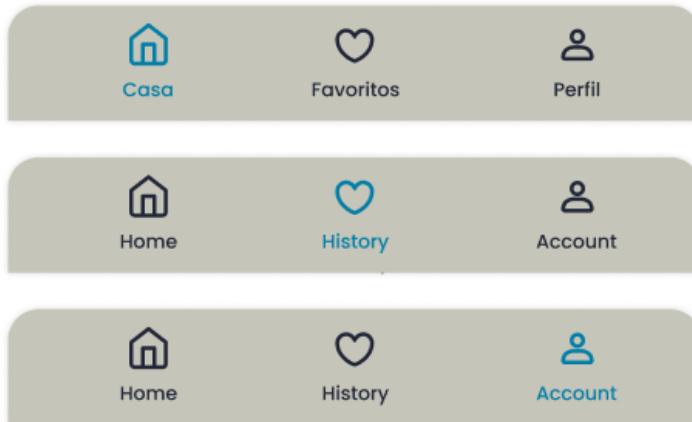
Me quedé con tres fuentes que me resultaban atractivas, las descargué en daFont y las agregue a la aplicación, las agregue en res/raw.

→ Logo



El logo escogido fue este ya que representa la catedral de Cádiz que es uno de los símbolos más importantes de Cádiz, es minimalista y aporta sobriedad al diseño de la aplicación.

## → Footer



### Diseño:

Los colores escogidos para el footer son Silver para el fondo, Raisan Black y Cerulean para los iconos.

Para los iconos e hice las distintas variantes dependiendo de la posición, por ejemplo si el icono de casa es azul y el resto Raisan Black estaríamos en la página principal, si estamos ubicados en favoritos el icono de favoritos se volvería azul mientras el resto en Raisan Black y así también con el icono de perfil, tal y como se muestra en la imagen y en la página de Figma.

### Usabilidad:

El **Footer** quería que fuese sencillo y que cualquiera pudiese entender su uso, ya que es similar al del resto de aplicaciones que estamos acostumbrados a usar, tales como Instagram, Facebook, Whatsapp etc.

**Home** nos lleva a la página principal que tiene todos los souvenirs, junto con los tipos de souvenirs que tenemos. **Favorites** nos lleva a los souvenirs que tenemos guardados como favoritos. **Profile** que nos permite ir a nuestro perfil para saber nuestro nombre de usuario, correo electrónico, añadir una foto o modificar los datos que ya tenemos e incluso borrar nuestra cuenta.

## → Souvenirs



Estas cajas contienen la información del souvenir, en un inicio lo hice pensando en la base de la aplicación ya que solo se podían guardar como favoritos, ya después añadí la cesta para que pudieran guardar en el carrito y pedirlos.

#### → Buscador



Permite buscar el souvenir que te interese y te devuelve el souvenir en detalle, para que puedas añadirlo a favoritos, al carrito o ver más información del souvenir en cuestión.

La versión final implementa el buscador de Jetpack Compose:



Es un SearchBar al que se le introduce una query que muestra uno de los souvenirs si coincide con el introducido.

He preferido usar este buscador ya que no quería implementar el Relay de Figma porque ralentiza

mucho la aplicación y la vuelve difícil de seguir ya que contiene mucho código spaghetti.

## → Input

Nombre:

Introduce el nombre de usuario

Correo electrónico:

Introduce el correo electrónico

Contraseña:

Introduce la contraseña

Input que permite introducir el nombre de usuario, su correo electrónico y contraseña.

## → Botones

Modificar

Suprimir

Cerrar sesion

Botones para modificar, suprimir y cerrar sesión, actualmente tienen ese color puede que en versiones posteriores el color cambie.

→ **Imagen de perfil**



Tendrá la imagen de perfil de los usuarios, que permitirá seleccionar la foto de galería que quiera y la guardará.

→ **Icono de carrito**



Icono del carrito para ver los souvenirs que has añadido a la cesta. Si el ícono se selecciona se verá en Cerule.

→ **Icono de pedidos**



Al seleccionarlo permite ver los pedidos que los clientes han realizado. Si el ícono se selecciona se verá en Cerule.

La aplicación se divide en distintos archivos:

- **Componentes:** Contiene todos los componentes que se repiten de la app, como puede ser el header, footer... Funcionan de forma correcta y son intuitivos de usar.
- **Página principal:** Página principal de la aplicación, que es la que contiene la vista de todos los souvenirs, junto con un pequeño buscador en el que podemos buscar el souvenir que en ese momento nos interese ver.



Tiene el logo a su izquierda con el nombre de la aplicación en el medio, el ícono del carrito

- **Admin:** Página del administrador que permite recibir los pedidos, aceptarlos y anularlos, quizás en un futuro se añadan más posibilidades como el eliminar cuentas etc

Para *entrar como administrador* primero tendremos que iniciar sesión con una cuenta creada para la ocasión, ahora mismo esa cuenta es una constante, pero para aumentar la seguridad de la aplicación recuperaré esos datos de firebase.

- **Actualizaciones:**

13/05/2024:

- ➔ Se ha añadido la posibilidad de ver las cuentas actuales, más adelante se añadirá la posibilidad de eliminar cuentas.
- ➔ Recuperar los datos con un usuario en firebase que tiene el correo del administrador que es una constante en el programa, entonces no necesito tener la contraseña como constante, ya que solo necesito el correo y compararlo con el email del auth.
- ➔ Eliminar las constantes innecesarias y que pueden provocar fallos en la seguridad.

Podremos iniciar sesión con la cuenta del administrador y nos saldrá una pantalla como esta:



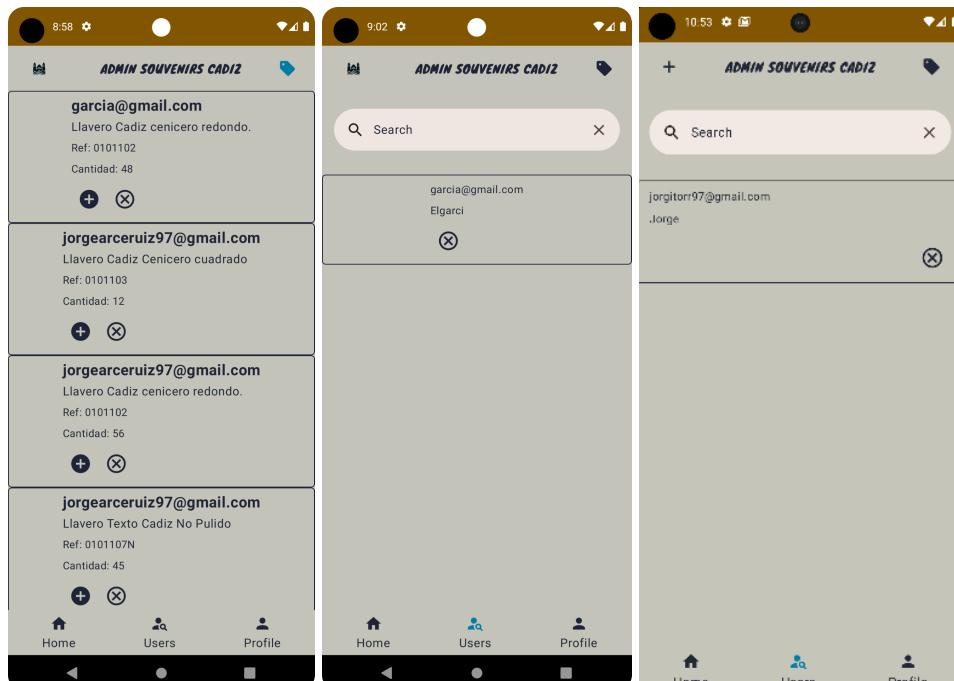
#### ❖ Header:

- **Pedido:** Página que te permite ver los pedidos realizados por los clientes. Esta página solo está disponible para el administrador de la aplicación. Ve el usuario que ha hecho el pedido y el pedido correspondiente, el administrador puede aceptar o cancelar el pedido

#### ❖ Footer:

- **Users:** Permite ver todos los usuarios.
- **Home:** Es la página principal, permite ver todos los souvenirs.
- **Profile:** Es la página de la cuenta del administrador.

Como podemos ver la pantalla del administrador es prácticamente igual a la pantalla principal, la diferencia más notable es que el administrador en vez de tener el icono del carrito tiene el de pedidos, que le permite ver los pedidos que sus clientes le han hecho.



Estas son pantallas que únicamente ve el administrador, que es un ejemplo de los pedidos realizados por los usuarios. Y otra en la que podemos ver un usuario registrado.

### **Actualización:**

21/05/2024:

- He arreglado la página de usuarios para que permita eliminar el usuario que quiera.
- Está bien organizado y el botón de eliminar aparece donde tiene que aparecer gracias a la correcta colocación con un Box que lo contiene.

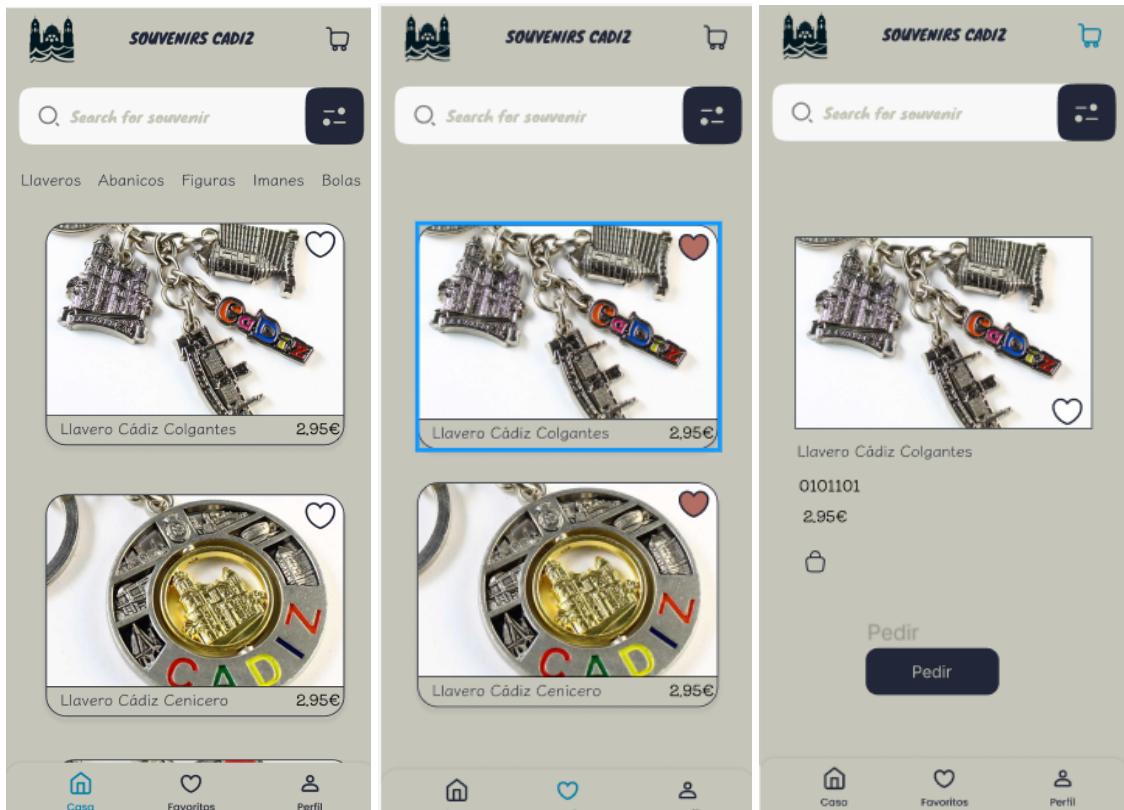
### **3. Instalación**

El enlace al github de la aplicación es el siguiente: [enlace app](#)

### **4. Prototipado**

El prototipo wireframe está desarrollado en figma: [enlace figma](#)

Primero tenemos la pantalla principal, a la que podemos acceder sin iniciar sesión, para ver los souvenirs pero si queremos guardarlos en favoritos o en el carrito tenemos que registrarnos o en el caso de tener ya una cuenta iniciar sesión.



Pantallas principales antiguas

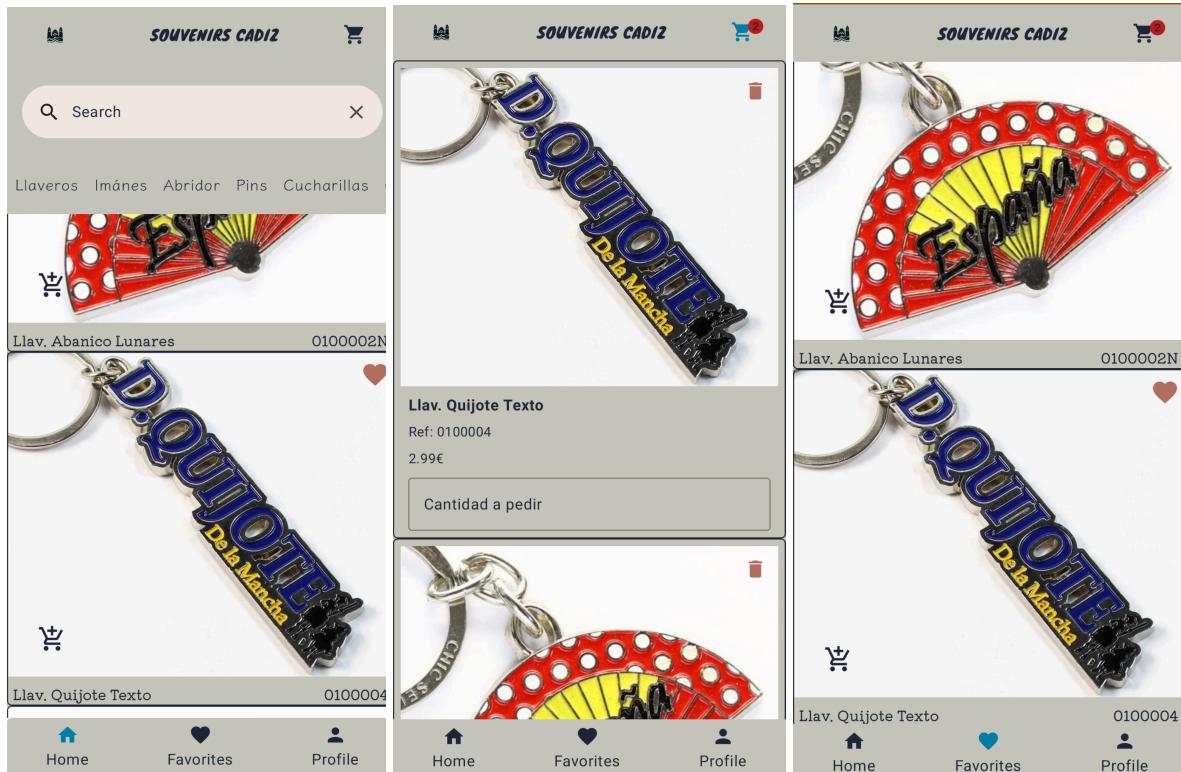
La **pantalla principal** tiene el conjunto de souvenirs que los saca de un csv, un buscador, un componente de categorías para acceder a todos los tipos de souvenirs y que al seleccionar uno te devolverá todos los souvenirs de ese tipo.

Podemos seleccionar el souvenir que queramos y añadirlo en favoritos, actualmente en esta versión solo se puede guardar en el carrito al entrar en un souvenir en concreto por temas de diseño, pero en la versión final esto también será añadido para que puedas añadir al carrito directamente el souvenir que quieras.

### **Actualización:**

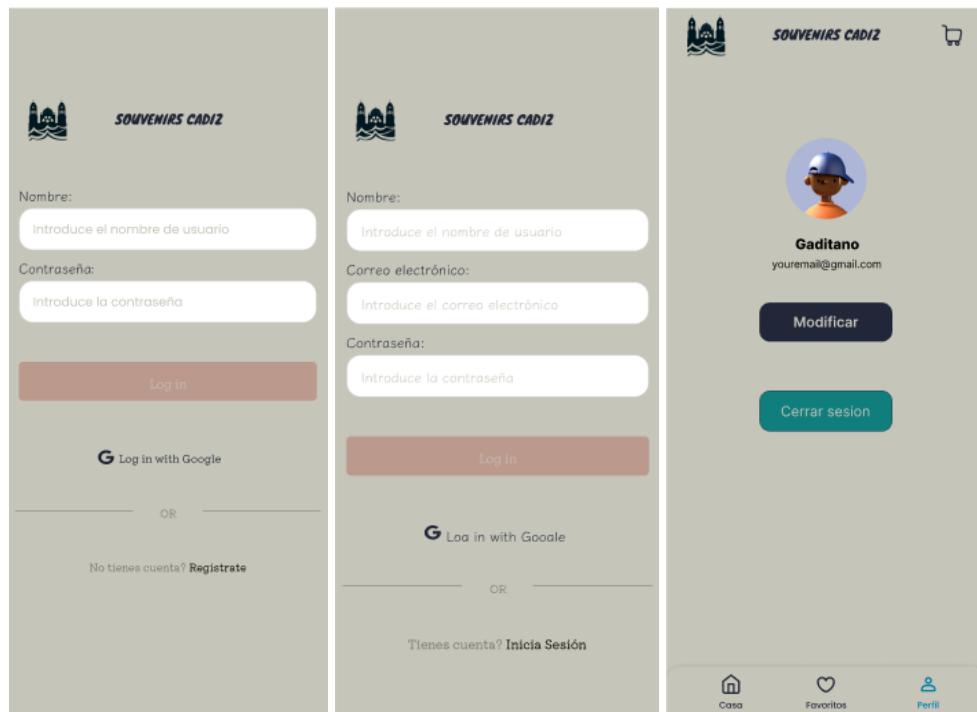
- En la actual se pueden guardar en el **carrito** los souvenirs.

En la zona de arriba o **header** tenemos el logo de la empresa que te muestra una página de detalle con la localización de la empresa y más información, el nombre de la aplicación y el ícono de carrito que te lleva a otra pantalla.



Pantallas actuales

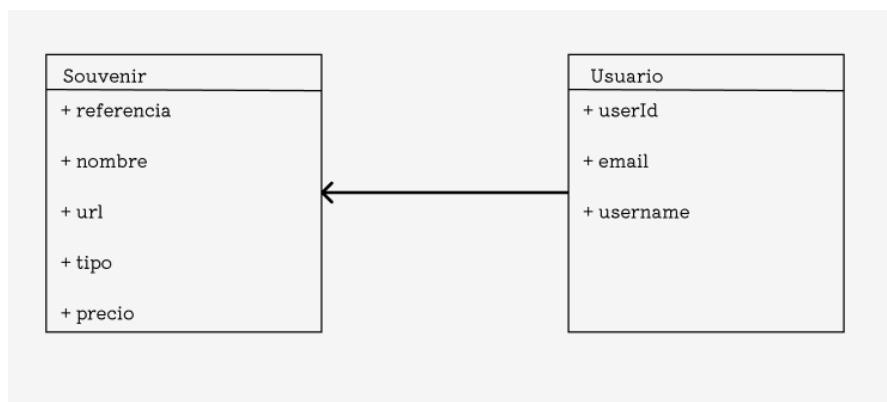
En la zona de abajo tenemos el **footer** que ya hemos explicado con anterioridad, que posee el icono que te lleva a la pantalla de inicio con todos los souvenirs, el icono de favoritos que te lleva a una pantalla que te muestra los souvenirs favoritos y el de perfil que te lleva a una pantalla donde puedes registrarte o iniciar sesión.



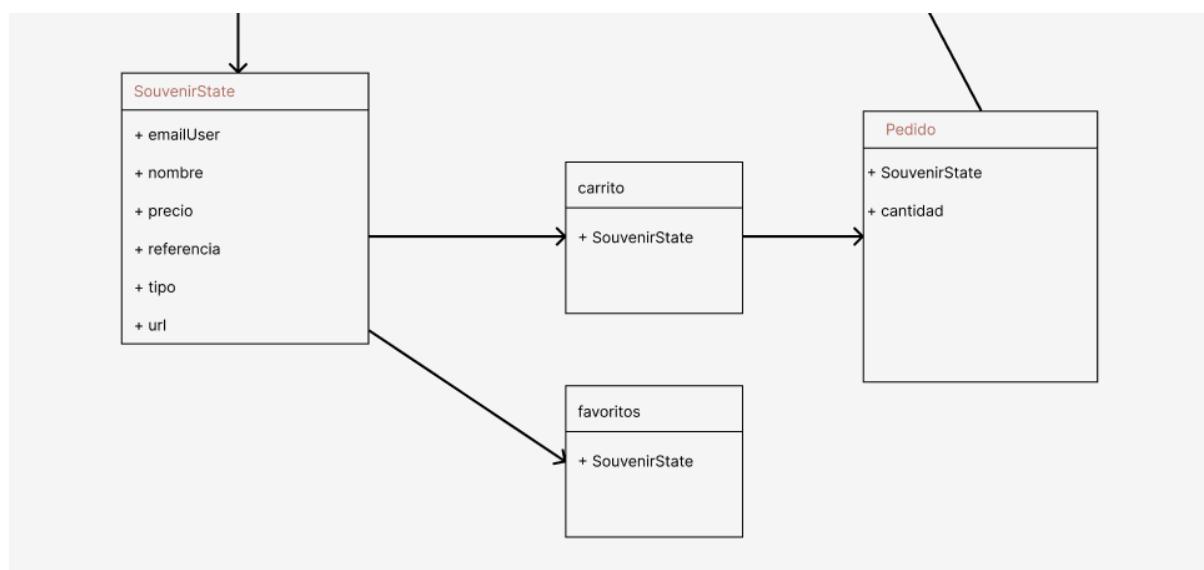
Pantallas de Figma

Aquí podemos ver la pantalla de inicio de sesión, de registro y de perfil en la que se muestra nuestro perfil con su email, su nombre y su foto.

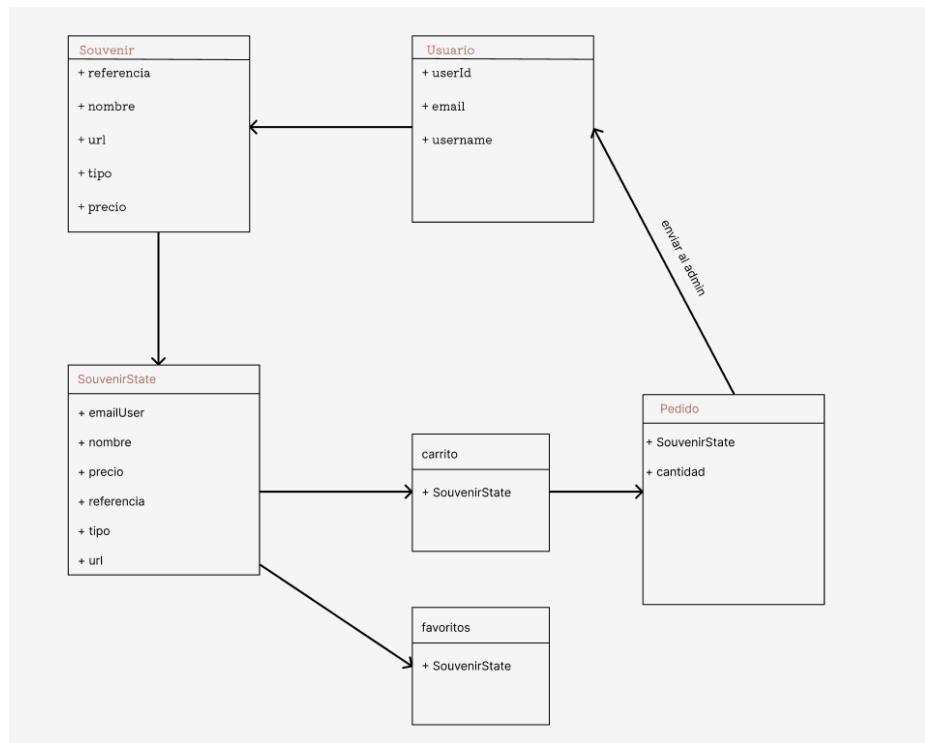
## 5. Diseño funcional



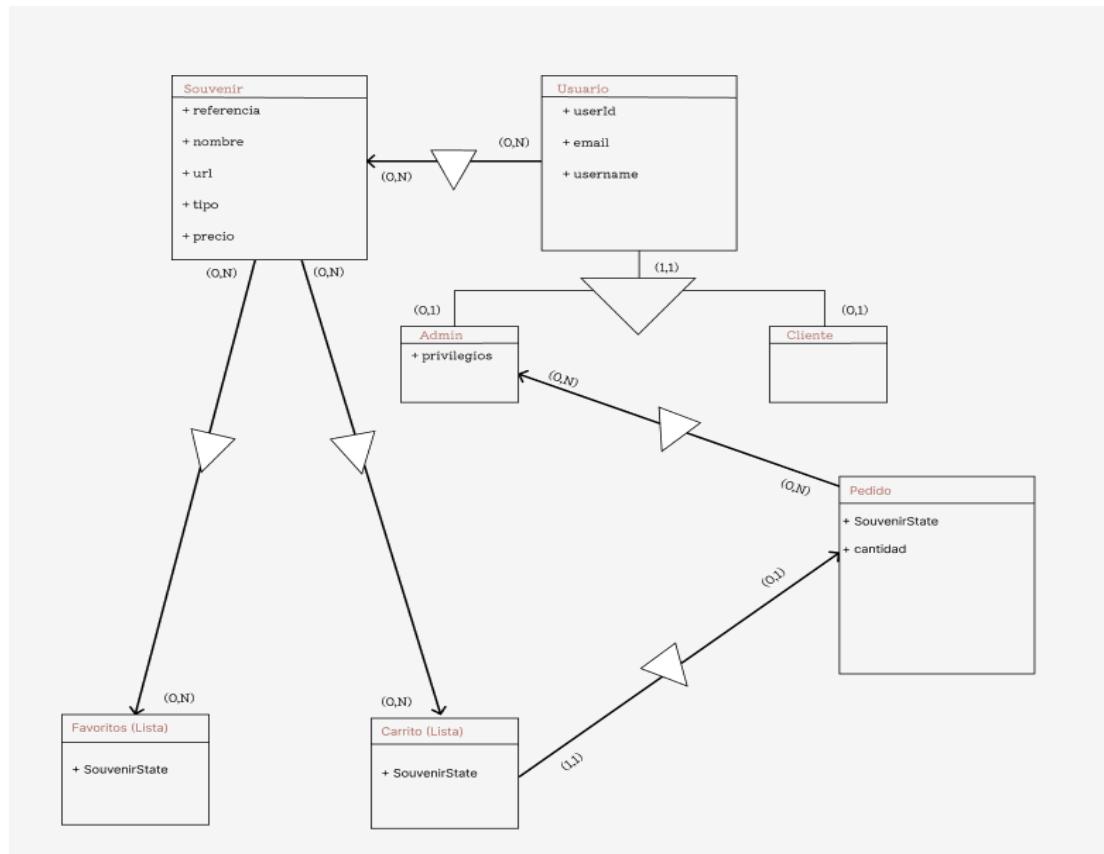
En el esquema UML básico podemos ver que tiene dos entidades, **Usuario** y **Souvenir**. El resto de entidades pertenecen a la base de datos.



Aquí podemos ver que hay una entidad llamada **SouvenirState** que se encarga de recoger la información de la base de datos que en este caso es la información del souvenir y el email del **usuario** que ha guardado el souvenir. El souvenir se puede guardar en dos entidades diferentes, una es en la de **favoritos** y otra es la de **carritos**, los souvenirs guardados en la entidad del carrito permiten ser pedidos. Los souvenirs pedidos los recibirá el administrador una vez sean comprados.



Al añadir al administrador en la **jerarquía** del usuario, podemos tener una relación como este:



El administrador y el cliente son parte de los usuarios, un **usuario** puede guardar cero o muchos *souvenirs* en una lista de **favoritos** y otra de **carrito**, los souvenirs en el carrito se pueden pedir, una *lista de carrito* hace un **pedido** que se le manda al **administrador** que a su vez es un *usuario con privilegios*.

## 6. Desarrollo

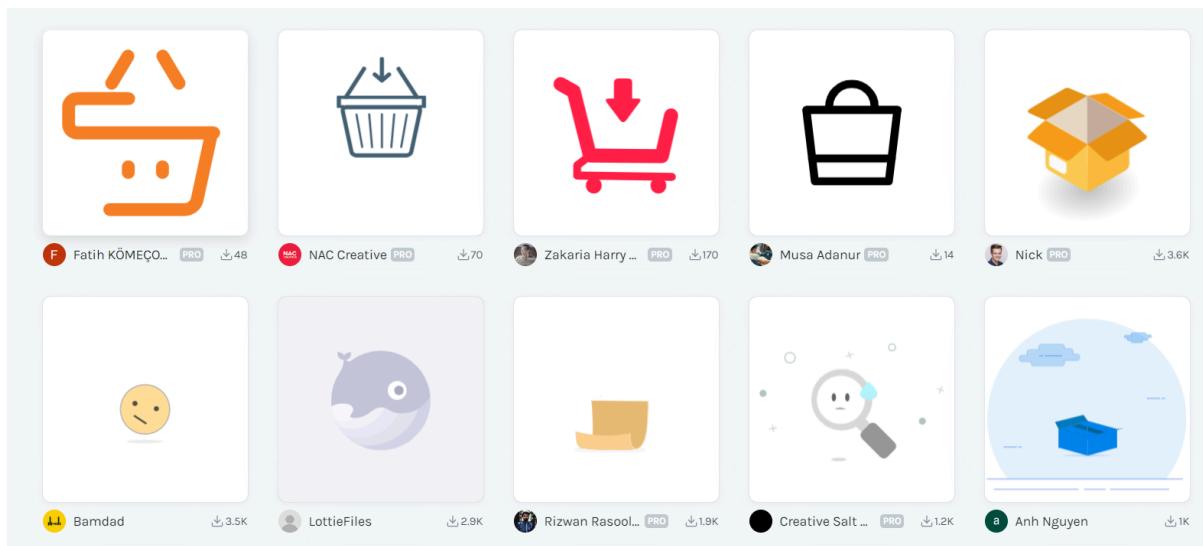
El primer paso fue crear el diseño que está en el enlace de figma, después pasé al código creando las bases de la aplicación, su estructura es la siguiente:

- **model:** Contiene las data class de la aplicación.
- **util:** Contiene las clases de utilidades como son las constantes.
- **navigation:** Contiene el sistema de navegación de la aplicación.
- **ui:** contiene los.viewmodel, las pantallas y temas.
  - **model:** Contiene los ViewModel.
  - **theme:** Contiene los ficheros que vamos a utilizar después para los colores de la aplicación, las fuentes etc.
  - **view:** Contiene todos los ficheros y Screens de la aplicación que es lo que vemos como usuarios.

El siguiente paso fue crear las data class, que en este caso son Souvenir y User, que son las dos clases fundamentales de la aplicación.

Creé las constantes que iba a usar y empecé con los viewModel para pasar a crear los archivos, añadir la base de datos de firebase etc.

- La versión actual de la aplicación no tenía **dagger Hilt** que es un conjunto de técnicas para disminuir el acoplamiento, la idea de la inyección de dependencias es que no haya instancias ya que todas se las pase dagger, de esta forma hace varias llamadas a una sola instancia.
- Animaciones añadidas con la dependencia de **Lottie**, desarrollada por airbnb, Lottie es una biblioteca para Android, iOS, Web y Windows que analiza animaciones de Adobe After Effects exportadas como JSON con Bodymovin y las representa de forma nativa en dispositivos móviles y en la web. Podemos encontrar más información en su github: [github lottie](#)
- La página que he usado para las **imágenes animadas** es Lottie Files: [lottie files](#)



(Ejemplo al buscar empty basket)

- Uso de **corrutinas**: Las corrutinas son una forma avanzada de trabajar con operaciones asíncronas, lo que significa que nos permiten realizar tareas en el fondo sin interrumpir el flujo principal de nuestro programa.
- ❖ Algunas de las **dificultades** encontradas en la programación de la aplicación fueron:
  - Digitalizar la información, tuve que sacar toda la información con respecto de los souvenirs de un pdf:



Primero saqué las imágenes y guardé la información de los souvenirs en un csv ya que es la manera más sencilla y fácil de ampliar que encontré, además de que ya tenía un csv con la información y para no cometer fallos lo preferí, en ediciones posteriores la información quizás se recoja directamente de la base de datos.

En el pdf viene la imagen del souvenir, su nombre y la referencia.

- Google: Problemas al identificarse con google, ya que me da un error de que no hay suficiente espacio de memoria. La solución la encontré en una página de stackoverflow: [solución](#). Uno de los usuarios dice que entre en la carpeta .gradle y borres la carpeta daemon que son los demonios, esto me ha funcionado. Gracias a esto puedo recoger el SHA que es para la autenticación con google.

- Seguridad: Uno de los fallos de este tipo es la API KEY de google, use un plugin para github llamado GitGuardian para la seguridad de la aplicación:

The screenshot shows the GitGuardian interface with a sidebar containing 'Incidents', 'Perimeter', 'Analytics', 'Honeytoken (beta)', and 'taC'. The main area is titled 'Secrets' with 1/1 result. It displays a table with columns: DATE, SECRET, SEVERITY, INFO, TAGS, and STATUS. A single row is shown for a Google API Key found on March 23rd, 2024, at 11:55. The secret is labeled 'Critical' and is associated with user 'jorge' and file 'app/src/main/AndroidManifest.xml'. Tags include 'From historical scan', 'Publicly exposed', and 'Default branch'. The status is 'Triggered'.

Las constantes estaban guardadas al principio en un archivo, pero esto puede ser peligroso ya que son accesibles para cualquiera.

```
class Constant {
    /**
     * @param MAX_SOUVENIRS max de souvenirs que tengo actualmente
     * @param MIN_SOUVENIRS minimos souvenirs que tengo actualmente
     * @param EMAIL_ADMIN email del administrador
     * @param CONTRASENIA_ADMIN contraseña del administrador
     * @param NUMERO_TLF numero de telefono de la empresa
     */
    @ jorge *
    companion object{
        const val MAX_SOUVENIRS: Int = 115
        const val MIN_SOUVENIRS: Int = 6
        const val EMAIL_ADMIN:String = "arsenogue@gmail.com"
        const val CONTRASENIA_ADMIN: String = "arsenoque" |
        const val NUMERO_TLF: String = "617759036"
        const val TOKEN:String = "802100001314-0168ghbg5joghv56bgjpf7ib4qbdotv0.apps.googleusercontent.com"
    }
}
```

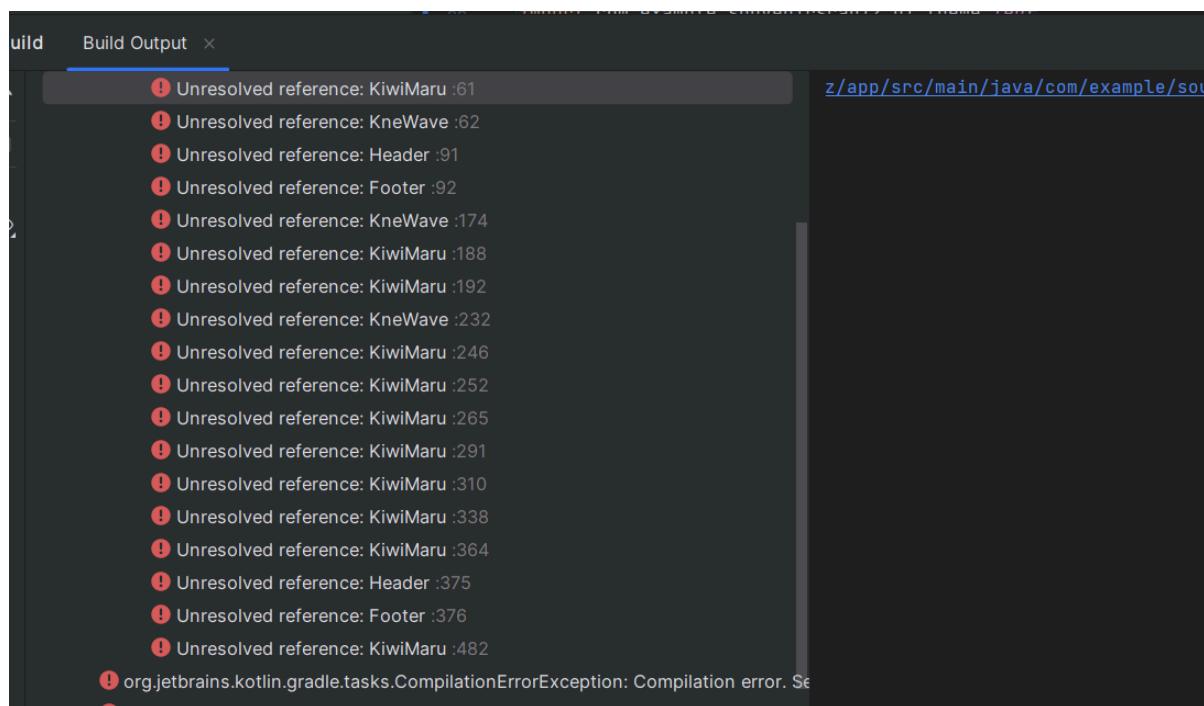
El archivo de constantes actual es así:

```
/**  
 * constantes  
 * @param EMAIL_ADMIN email del administrador  
 * @param TOKEN token de google  
 */  
@jorge  
class Constant {  
    @jorge  
    companion object{  
        const val EMAIL_ADMIN:String = "arsenogue@outlook.com"  
        const val TOKEN:String = "802100001314-0168qhb5joghv56bgjpf7ib4qbdoty0.apps.googleusercontent.com"  
    }  
}
```

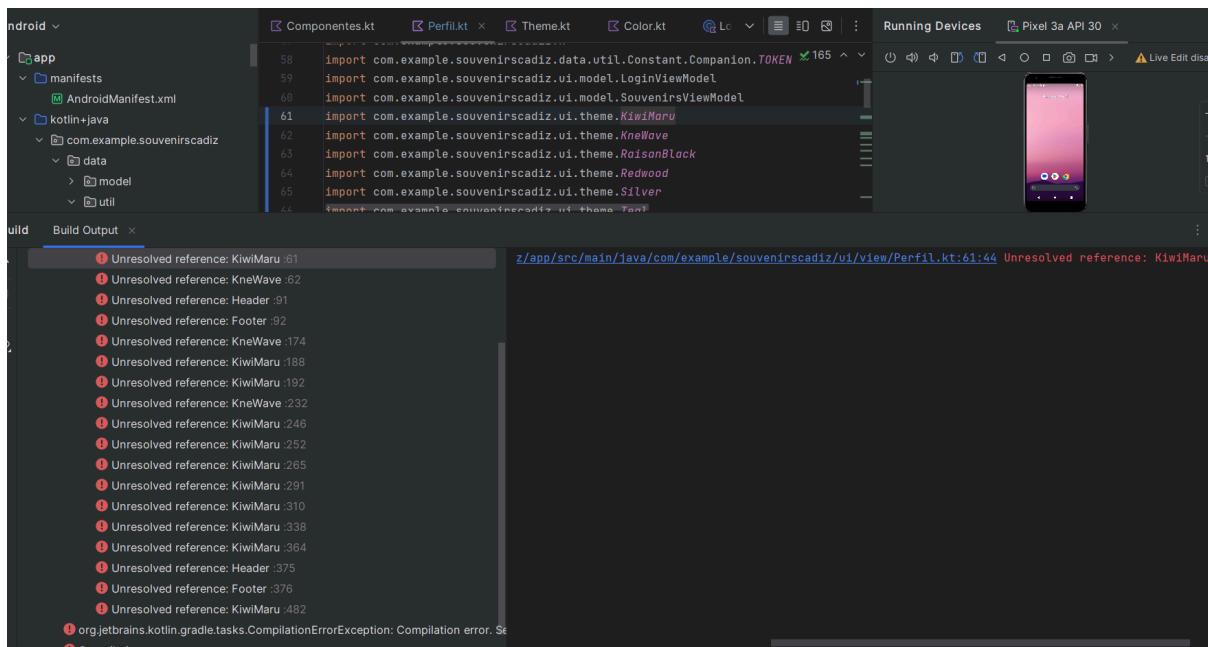
Hemos eliminado las constantes que pueden provocar fallos de seguridad, como la del número de teléfono y la contraseña del administrador, ya que solo necesitamos el correo.

Fallo de referencias de fuentes y métodos:

- Unresolved references: Este fallo surgió de repente, con cosas como la fuente, el header y el footer. La solución fue sencilla, acceder a los métodos o variables que creaban esto y modificarlos.



Está es la pantalla con las importaciones y aún así teniendo el error.



➤ Que me modifique el color de los iconos y su tipo cuando pulsase y si pulso en otra pantalla actualizase, tuve muchos problemas con esto. La manera de resolverlo en la misma pantalla fue creando dos variables de tipo Boolean que cambian cada vez que el componente se actualiza. Estas variables están guardadas en el viewmodel y son llamadas en el composable con una condición que establece que cada vez que cambian me llama al ícono, entonces el ícono en el momento que se guarda o se elimina un elemento de la base de datos se actualiza el ícono ya que vuelve a llamar al composable.

❖ Algunas de las **mejoras** que ha recibido la aplicación son:

- Que me aparezcan los corazones en rojo cuando estuviesen guardados en la base de datos, para ello tuve que crear un nueva variable en las data class llamada guardado y que si esa variable estaba en true me mostraba el corazón en rojo. Tenía que hacer un fetch de los souvenirs guardados en favoritos antes de imprimir todos los souvenirs por pantalla y comprobar uno a uno si tenía la misma referencia que el mostrado por pantalla, en el caso de tenerla me cambiaba el ícono del corazón junto con su color.
- He tenido que realizar la misma función para los guardados en el carrito agregando el carrito al cuadrado con su página principal, de esta forma sabré que souvenirs están guardados y que la aplicación lo sepa para poner el corazón o el carrito de otro color.

- He añadido un elemento **Icon Toggle Button** que es un tipo de elemento que permite colocar el icono encima de una foto o elemento, se usa normalmente para mostrar las opciones de la aplicación pero en este caso lo he usado para poner los iconos por encima de la foto y que el diseño tenga un aspecto más moderno.
- He añadido un elemento en **google maps** para la posición de la cámara y a su vez he activado las propiedades del satélite en Google Maps.
- Todos los iconos se actualizan en tiempo real.
- Todos los datos se recogen de la **base de datos de firebase (Firestore Database)**, como las imágenes con **Storage** y todos los souvenirs, por eso se han podido eliminar algunas data class innecesarias que hacían referencia al objeto original y al de la base de datos, tales como Souvenir, como ahora los souvenir solo salen de la base de datos no necesitan otra clase, también se ha eliminado la de usuario ya que ya tenía UserState y todos los usuarios se guardan directamente en la base de datos, no hay nada local.

❖ Interacción con usuario:

- Test para saber la opinión de los usuarios, la página usada fue [test](#) (Tengo que hacerlo)

❖ Avances y cosas que quedan por hacer:

- ~~El admin puede añadir souvenirs nuevos,~~
- El admin puede modificar los souvenirs que hay y decir si tiene o no stock de algunos de los souvenirs
- Añadir reglas de seguridad a firebase
- ~~Pedido contiene varios souvenirs~~
- Al pedir los souvenirs del carrito pregunta si estás seguro.
- Casos de uso -> caja negra, caja blanca etc.
- ~~Pueda guardar una imagen de perfil para el usuario~~
- ~~Una vez pedidos los productos que quieras ya desaparecen del carrito~~
- Antes de hacer un pedido te pregunta si estas seguro que quieres realizarlo en un Alert Dialog

- Mejorar login con Google para que al iniciar sesión no entre directamente en la anterior antes seleccionada, sino que me deje seleccionar de nuevo.
- Información delicada en la base de datos y que acceda a esos datos cada vez que los necesite.
- Detekt: Analizador de código
- Gargar las imágenes y el csv desde la base de datos → al final he elegido un JSON
  
- Los productos en pedido los puedo ver el admin para que sepa que productos quieren los clientes
- Hacer que tanto el botón de fav y el del carrito se vean en pantalla nada más iniciarlos(????)
- Poner sonidos, al pulsar en el corazón, el carrito, al mostrar la estrella, la caja de que no hay, al pedir etc
- Añadir notificación del usuario tiene pendientes souvenirs en el carrito
- Añadir notificación del admin (en el caso de que estés registrado como administrador) ha recibido nuevos souvenirs y sale los que son la cantidad
- La pantalla de pedidos te pide el número de souvenirs que quieres y esa info también le llega al administrador.
- Que funcione bien el botón de eliminar en el Carrito en todas las pantalla
- Que funcione bien el botón del corazón en favoritos en todas las pantalla
- Hay un error que es al dejar de ver el textField con los souvenirs se pierde la cantidad del que no se ve, arreglado con un label que pone la cantidad que has introducido anteriormente
- En el caso de que elimine un elemento de favoritos o del carrito me tiene que actualizar el ícono en la pantalla principal
- Si pulsas el corazón rojo de nuevo se quita de los guardados
- Arreglar que no se vea el nombre y correo del usuario una vez has salido de la app
- Configurar botón de eliminar souvenir del carrito
- Probar si funciona el botón de pedir bien y pide y borra los souvenirs
- Añadir animación cuando no tengas souvenirs guardados
- Añadir animación cuando no tengas elementos en el carrito
- Añadir Alertdialog antes de pedir, para que el cliente tenga que confirmar que quiere pedir los productos
- Poner que me salga el corazón en rojo cuando los souvenirs están guardados
- Cambiar el shopping Basket por add Shopping Cart

- Añadir lottie e iconos con lottie File
- Crear un administrador que pueda añadir un souvenir nuevo
- Añadir carrito y pedido a la base de datos
- Que me salga el corazón en rojo cuando haya elementos guardados
- Cambiar Shopping Basket por addShoppingCart y cuando esté guardado el elemento remove Cart por Shopping Cart
- Los títulos del login dentro de cada TextField
- Arreglar Google Maps para que me salga la ubicación señalada correctamente
- Meter Dagger Hilt
- Meter un número que indique el número de elementos que tienes en el carrito
- Meter Splash
- Poner el ícono de fav dentro de la foto arriba a la derecha
- Poner el ícono de carrito dentro de la foto abajo a la izquierda
- Arreglar tamaño de imágenes
- Cambiar color de la letra del search para que se ponga blanca
- Tengo que darle función al botón de ir hacia atrás
- Meter un número que indique el número de elementos que tienes en el carrito en ese momento y que se vea

## 7. Pruebas

Tengo que hacerlo todavía.

## 8. Distribución

descarga de apk.

## 9. Manual

La aplicación tiene cinco pantallas a las que se accede a través de iconos, tres de esos iconos forman parte de un componente **Footer** y otros dos forman parte del **Header**. El footer contiene el ícono de **Home**, **Favorites** y **Profile**. El **Header** contiene el **carrito** que te muestra todos los elementos que tiene guardado el usuario en el carrito, el **logo** que te lleva a la página de detalles de la empresa y el **título** de la aplicación.

**Home** navega a la pantalla principal que contiene cajas con las imágenes de cada souvenir junto con su referencia, si seleccionas el souvenir va a la página de **detalle del souvenir** que contiene la imagen del mismo souvenir, su referencia y precio.

El icono de **Favorites** navega a la pantalla de favoritos que contiene todos los souvenirs que ha seleccionado el usuario y que ha decidido guardar en favoritos.

**Profile** navega a la pantalla del perfil del usuario, que contiene una imagen de perfil, su nombre de usuario y el correo introducido.

Gitbook: [gitbook](#)

## 10. Conclusiones

El proyecto está realizado pensando en una empresa de souvenirs, al principio todo se desarrolla con cierta facilidad y parece que en apenas unos días tienes lista la aplicación, el problema comienza con los detalles que son las cosas con las que más se tarda, al final es como una carrera de fondo en la que avanzamos lentamente, hay días que ni siquiera avanzas, pero pensar que el proyecto servirá a alguien y tiene una función más allá de la propia satisfacción ayuda a seguir el proyecto.

## 11. Índice tablas e imágenes

1. [pdf todos los souvenirs](#)
2. [comando para pasar json a la BDD](#)
3. [ejemplo BDD con listas y souvenirs guardados](#)
4. [colores](#)
5. [fuentes](#)
6. [logo](#)
7. [Footer](#)
8. [Cajas](#)
9. [Buscador](#)
10. [Inputs](#)
11. [Botones](#)
12. [Imagen de perfil](#)
13. [Carrito](#)
14. [Pedidos](#)

15. [Página principal](#)
16. [Pantalla usuarios](#)
17. [Pantallas actuales](#)
18. [Pantallas de Figma](#)
19. [UML Usuario Souvenir](#)
20. [UML Listas](#)
21. [UML Sin jerarquia](#)
22. [UML final con jerarquías](#)
23. [Lottie](#)
24. [GitGuardian](#)
25. [Archivo de constantes antiguas](#)
26. [Archivo de constantes actuales](#)
27. [Unresolved References](#)
28. [Unresolved References](#)

## 12. Bibliografía

- 1) [solución fallo de gradle](#)
- 2) [autenticación de Google](#)
- 3) [inspeccionar rendimiento](#)
- 4) [lottie tutorial](#)
- 5) [número de elementos en un icono](#)
- 6) [¿Cuál es mejor csv o json?](#)
- 7) [Corrutinas](#)
- 8) [drop down menu](#)

- 9) [photo picker](#)
- 10)[Subir json a firebase](#)