



ARTNUTRIA

Memoria del proyecto de DAM

Información

Ciclo de Desarrollo de aplicaciones multiplataforma
Tutor: Carlos Tarazona Tarrega

ADRIAN GONZALEZ NUÑEZ

31/05/2023

1 Tabla de contenido

1.	Introducción	4
1.1	Resumen	4
1.1.1	Palabras clave	4
1.2	Resum	4
1.2.1	Paraules clau	4
1.3	Summary	5
1.3.1	Keywords	5
1.4	Descripción	6
1.4.1	Historia de ArtNutria	6
1.4.2	El nombre y el logo	6
2	Estado del arte	7
2.1	Tecnologías	7
2.1.1	Firestore	7
2.1.2	Android Studio	7
2.1.3	Kotlin	7
2.2	Herramienta de apoyo	8
2.2.1	GitHub	8
3	Estudio de viabilidad	9
3.1	Estudio de mercado	9
3.2	Viabilidad técnica/económica del proyecto	11
3.3	Tabla DAFO	11
3.4	Viabilidad temporal	12
4	Análisis de requisitos	14
4.1	Requisitos del usuario	14
4.2	Requisitos del tiempo de desarrollo	14
4.3	Diagrama general de casos de uso	14
5	Diseño	16
5.1	Diagrama entidad-relación y modelo relacional	16
5.2	Diagrama de clases	17

5.3	Descripción de tablas y campos	17
5.4	Mapa conceptual	18
5.5	Diagrama de secuencias	19
5.6	Diagrama de actividades	20
5.7	Diagrama de flujo.....	21
5.8	Mapa de Navegación	21
	22
5.9	Mockups	23
6	Codificación	24
6.1	Diseño grafico.....	24
6.1.1	Selección de plantilla	24
6.1.2	Gama de colores	24
6.2	Proceso de codificación	24
6.2.1	Ejemplos de código interesantes	25
6.3	Documentación del código.....	26
6.3.1	Interna	26
6.3.2	Externa	26
6.4	Seguridad	30
6.4.1	Inyecciones de código	30
6.4.2	Excepciones	31
6.5	Compatibilidad	31
7	Implantación	32
7.1	Clonar el proyecto.....	32
7.2	Despliegue de Firebase	33
7.3	Diagrama de despliegue	41
8	Conclusiones	41
8.1	Ampliaciones y mejoras	41
9	Bibliografía.....	43
9.1	Apuntes y material didáctico	43
9.2	Páginas web y tutoriales	43
9.3	Agradecimientos	45

10	Anexo.....	46
10.1	Codigo	46
10.1.1	MainActivity	46
10.1.2	HomeActivity.....	49
10.1.3	CarouselFragment	51
10.1.4	FollowUserFragment	55
10.1.5	HomeFragment.....	56
10.1.6	LikkeObraFragment	61
10.1.7	MyUserFragment.....	63
10.1.8	ObraFragment	67
10.1.9	UserFragment.....	69
10.1.10	FirebaseViewModel.....	72
10.1.11	ImageViewHolder	73
10.1.12	ObraImageViewHolder	74
10.1.13	ObraViewHolder	75
10.1.14	UserViewHolder	75
10.1.15	FirebaseUserCase	76
10.1.16	FirebaseRepo	77
10.1.17	MyFirebaseMessagingService	82
10.1.18	Usuario	83
10.1.19	Obra.....	83
10.1.20	OnItemClickListenerObra	83
10.1.21	OnItemClickListenerUsuario.....	83
10.1.22	ImageAdapter	84
10.1.23	ObraAdapter	84
10.1.24	ObraImageAdapter.....	85
10.1.25	UserAdapter.....	86

1. Introducción

1.1 Resumen

ArtNutria es una aplicación que pretende ser un escaparate para los artistas permitiendo almacenar imágenes y videos, mostrándolos posteriormente en una página principal donde los usuarios podrán ver y seguir a sus artistas favoritos. El presente trabajo abarca la primera etapa del desarrollo de la aplicación que incluye la gestión de usuarios, archivos y seguimiento. Ha sido realizado bajo el SDK de Android Studio.

Este documento recoge la memoria del desarrollo de la aplicación, desde la planificación inicial, el análisis de las necesidades del cliente, el diseño del sistema que la sustentará, el proceso de codificación y la posterior posible explotación de esta.

1.1.1 Palabras clave

Arte, entretenimiento, redes, social.

1.2 Resum

ArtNutria és una aplicació que pretén ser un aparador per als artistes permetent emmagatzemar imatges i vídeos, mostrant-los posteriorment en una pàgina principal on els usuaris podran veure i seguir als seus artistes favorits. El present treball comprén la primera etapa del desenvolupament de l'aplicació que inclou la gestió d'usuaris, arxius i seguiment. Ha sigut realitzat sota el SDK d'Android Studio.

Aquest document arreplega la memòria del desenvolupament de l'aplicació, des de la planificació inicial, el anàlisi de les necessitats del client, el disseny del sistema que la mantindrà, el procés de codificació i la posterior possible explotació de la mateixa.

1.2.1 Paraules clau

Art, entreteniment, xarxes, social.

1.3 Summary

ArtNutria is an application that aims to be a showcase for artists allowing you to store images and videos, displaying them later a home page where users can see and follow their favourite artists. The present work covers the first stage of the application development that includes the management of users, files, and monitoring. It has been made under the SDK Android Studio.

This document gathers the record of the development of the application, from the initial planning, the analysis of the client's needs, the design of the system that will support it, the coding process, and the possible future exploitation of it.

1.3.1 Keywords

Art, entertainment, networking, social.

1.4 Descripción

ArtNutria surge como una opción para las necesidades de un usuario específico. Se trata de una aplicación que permite a los usuarios poder compartir y mostrar su arte de una manera sencilla, dándose a conocer para posibles trabajos, por lo tanto, la función principal será de escaparate, mostrando las obras de los usuarios a otros usuarios, esperando que esto ayude al usuario final para darse a conocer y llegar a la máxima cantidad de usuarios posibles.

1.4.1 Historia de ArtNutria

Un usuario se ve en la necesidad de buscar nuevas formas de mostrar su arte para darse a conocer al mundo, después de probar varias opciones, ha visto que varias de las aplicaciones no muestran bien el arte de los usuarios, dando prioridad a los gustos directos del usuario, impidiendo los nuevos descubrimientos.

Por este motivo, se ha decidido crear ArtNutria, buscando ser una opción viable para compartir el arte de los usuarios, permitiendo una opción de favoritos, pero siguiendo, dando importancia a todos los artistas de la aplicación.

1.4.2 El nombre y el logo

ArtNutria hace referencia a la palabra arte en inglés “Art” y la palabra “Nutria”, el nombre viene a simbolizar a como las nutrias guardan varias piedras y seleccionan su piedra favorita, simbolizando la piedra como el trabajo de un artista.

El logotipo de ArtNutria está diseñado para mostrar una nutria sujetando un pincel.



Imagen 1: Logo de la aplicación

2 Estado del arte

2.1 Tecnologías

2.1.1 Firebase

Plataforma en la nube para desarrollo de aplicaciones web y móvil. Contiene diferentes funciones que podremos utilizar para el desarrollo de nuestra aplicación.

Se utilizará principalmente para dos propósitos:

- Base de datos: Guardara toda la información necesaria de los usuarios y las obras subidas.
- Almacén: Guardara las imágenes y videos subidos, permitiéndonos tener un fácil acceso a estos.



Firestore

Imagen 2: Logo de Firebase

Se ha escogido ya que se ha trabajado con anterioridad con la plataforma y se ha decidido, tras la experiencia pasada, ampliar conocimientos en esta y utilizarla.

2.1.2 Android Studio

Entorno de desarrollo de aplicaciones móvil que ayudará al desarrollo de la aplicación.

Se utilizará para crear la aplicación junto al lenguaje de programación Kotlin.

Android Studio



Imagen 3: Logo de Android Studio

Se ha escogido por que se ha trabajado anteriormente con él.

2.1.3 Kotlin

Lenguaje de programación que se ha decidido implementar para la creación del código de nuestra aplicación, junto al entorno de Android Studio.

Se utilizará para crear la aplicación junto el entorno de Android Studio.



Imagen 4: Logo de Kotlin

Se ha escogido por que se ha trabajado anteriormente con él.

2.2 Herramienta de apoyo

2.2.1 GitHub

Repositorio online que permite gestionar proyectos y controlar versiones de código.

Sera utilizado para subir, llevar una gestión y tener un control de versiones del proyecto.



Imagen 5: Logo de GitHub

Se ha escogido por la familiaridad que tienen los desarrolladores con éste.

3 Estudio de viabilidad

3.1 Estudio de mercado

Se realizó un estudio de mercado para sondear la existencia de otras aplicaciones similares a ArtNutria.



Imagen 6: Logo de ArtStation

Entre las existentes en el mercado, posiblemente la más destacable sea <<ArtStation>> (<https://www.artstation.com>) una aplicación multiplataforma muy completa que permite a los usuarios compartir su arte de manera fácil y sencilla, pero si cabe destacar algo son las herramientas que tiene para darlo a conocer y como gestiona las relaciones con los usuarios, permitiendo enviar mensajes entre los usuarios y pudiendo hacer compras directamente en la aplicación. Cualquier usuario puede vender su arte en la plataforma, pero esta cobra una tasa del 18% anual de las ventas generadas. Presenta cuatro tipos de servicio:

1. “Gratis” servicio estándar con ciertas restricciones dentro de la aplicación.
2. “Plus” con un gasto de 6.99\$ al mes desbloquea portafolios en la aplicación, imágenes y videos a 4k y postear blogs.
3. “Pro” tiene un coste de 9.95\$ al mes y sus beneficios más destacables son, Más visibilidad, ganar más dinero con las comisiones y poder generar analíticas profesionales.
4. “Estudio” desde 14.99\$ añade las opciones, agregar miembros a equipo, aparecer en el directorio de estudios y recomendación a empresas que confían en la plataforma.

Una de sus mayores desventajas es que solo está en inglés, dificultando el aprendizaje del manejo de la aplicación a los usuarios que no son de habla inglesa.

Otra de las más destacables y conocidas es <<Pinterest>> (<https://www.pinterest.es/>), esta aplicación multiplataforma permite la difusión de imágenes y videos de manera fácil y sencilla, lo más destacable que tiene la aplicación es su adaptabilidad al usuario, mostrándole imágenes relacionadas con sus intereses al abrir y buscar otras imágenes o videos. Aparte de esto cabe destacar que cuenta con un chat para los usuarios y la aplicación está disponible en español.



Imagen 7: Logo de Pinterest



Imagen 8: Logo de Instagram

Otra de las posibles aplicaciones que podemos encontrar es <<Instagram>> (<https://www.instagram.com>), cabe destacar que esta esta más encarada como una aplicación móvil, aunque cuenta con una versión web. Permite la subida y difusión de videos e imágenes de manera fácil con un estilo más dedicado a la difusión en móvil.

Los motivos principales del rechazo de los usuarios a estas aplicaciones tienen que ver con el idioma (ArtStation solo está en inglés), la curva de aprendizaje en el manejo de la aplicación (dificultándose más por el idioma) y el desbloqueo por medio de pagos mensuales de ciertas herramientas de la aplicación.

Aunque el cliente está más interesado en una aplicación que pudiera manejarse desde móvil, es posible su implantación más adelante en ordenado.

3.2 Viabilidad técnica/económica del proyecto

ArtNutria está pensado para ser una aplicación que no cobre por su servicio, y tampoco tiene coste para el programador, pues se realiza con material del que ya se dispone previamente, tanto en software como en hardware.

Recursos de hardware

Recurso	Coste
Ordenador de sobremesa	0€ (ya se disponía de él)
teléfono móvil Galaxy A52	0€ (ya se disponía de él)

Recursos de software

Recurso	Coste
Android Studio	0€ (software gratuito)
Firebase	0€ (versión gratuita)

3.3 Tabla DAFO

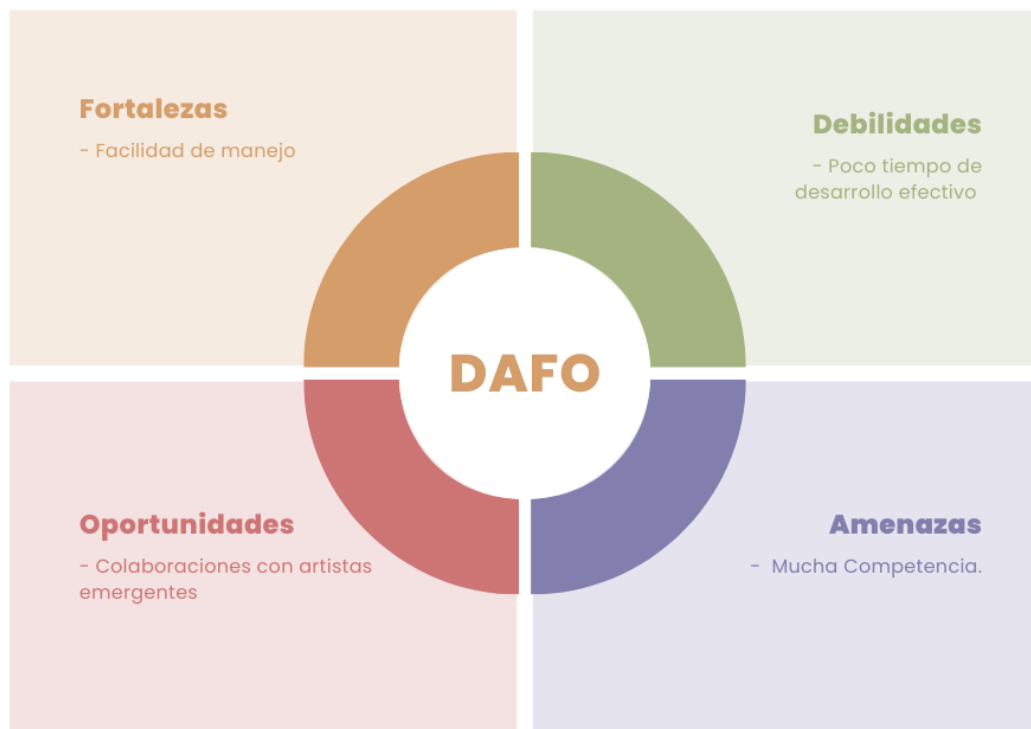


Imagen 9: Tabla DAFO

3.4 Viabilidad temporal

El tiempo efectivo para la realización de la aplicación fue de tres meses, entre marzo y mayo de 2023. Con este límite temporal se planteó el desarrollo de una parte de la aplicación, que puede tornarse mucho más compleja con mayor tiempo de codificación. Se planteó el desarrollo completo de la parte de gestión de obras y usuarios. En las últimas semanas se testeó el programa con varias pruebas, corrigiendo los errores de la aplicación.

El último mes se dedicó gran parte a la ejecución de la documentación, creando esquemas, diagramas y buscando información.

Para gestionar la viabilidad temporal se utilizó la herramienta web <<Team Gantt>> (<https://www.teamgantt.com/>), que permite de forma gratuita elaborar diagramas de Gantt para un proyecto.

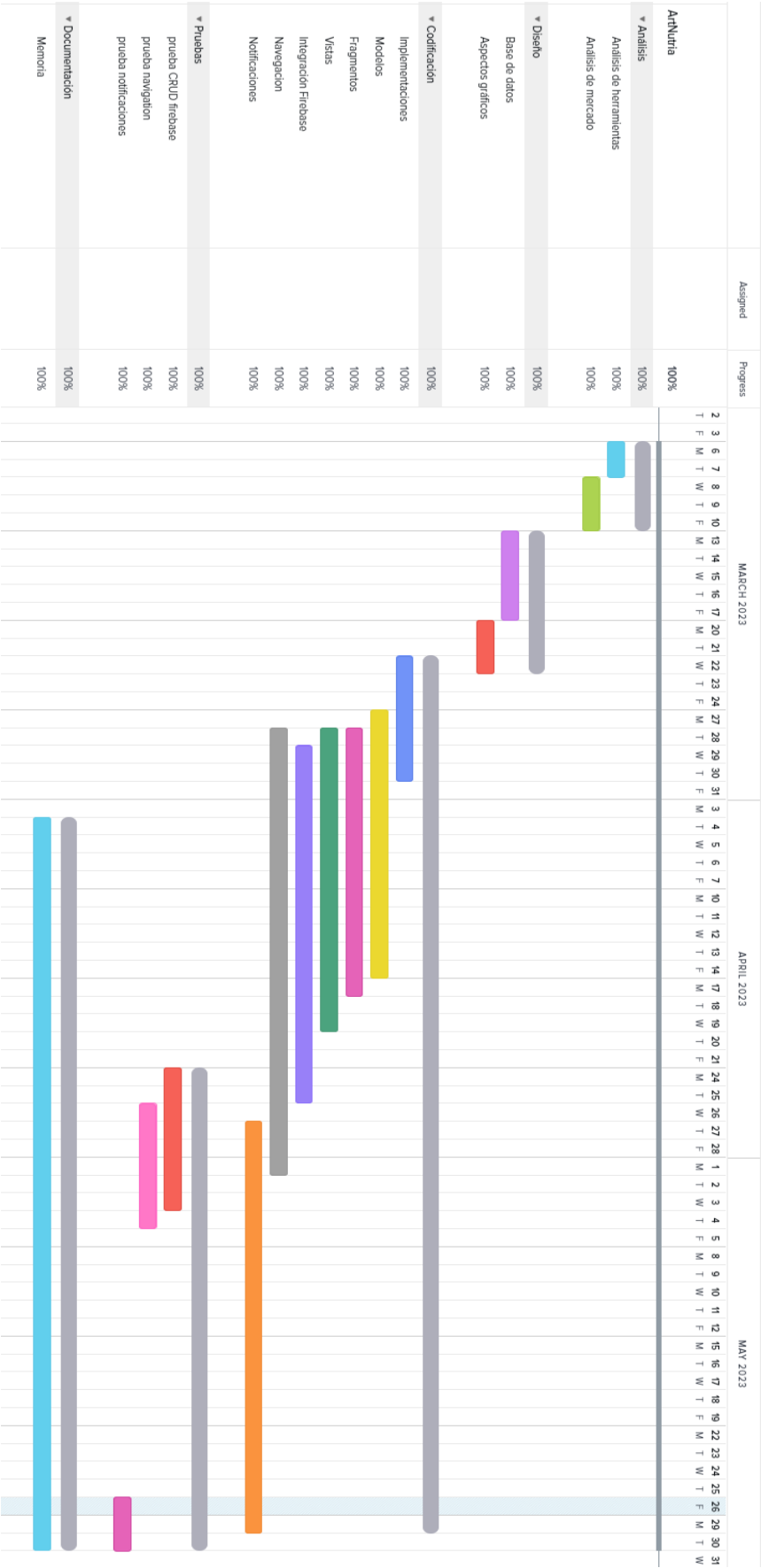


Imagen 10: Diagrama de Gantt

4 Análisis de requisitos

En esta etapa encontramos dos vertientes analíticas: las que atañen a los requisitos del usuario, y el tiempo de desarrollo del que se dispone para realizar la aplicación.

4.1 Requisitos del usuario

En la etapa inicial no se tenía una idea clara de las necesidades del usuario. Lo que teníamos claro era el requisito que debía tener el usuario que quiera utilizar la aplicación, el arte, es algo abstracto, pero al final lo que busca el usuario es ver y compartir arte (ya sea imágenes o videos).

En términos generales, el usuario tiene que poder gestionar su información personal y obras, dejando siempre la opción de poder compartir sus obras y ver las obras de los otros usuarios.

A si mismo el usuario debía poder crear una obra y añadirle los elementos que esta necesita (nombre, etiquetas e imágenes) y mostrarlas a los demás usuarios, también resultaba interesante algún sistema que permitiera la valoración de estas.

Otro de los requisitos es el seguimiento de los usuarios y alertar de cuando alguno de los usuarios seguido suba una obra.

4.2 Requisitos del tiempo de desarrollo

Debido al tiempo de desarrollo del que se disponía para la realización del proyecto se definieron sólo algunas de las entidades que formarían parte del diseño final de la aplicación, las cuales deberían funcionar al final de la fase de codificación del programa.

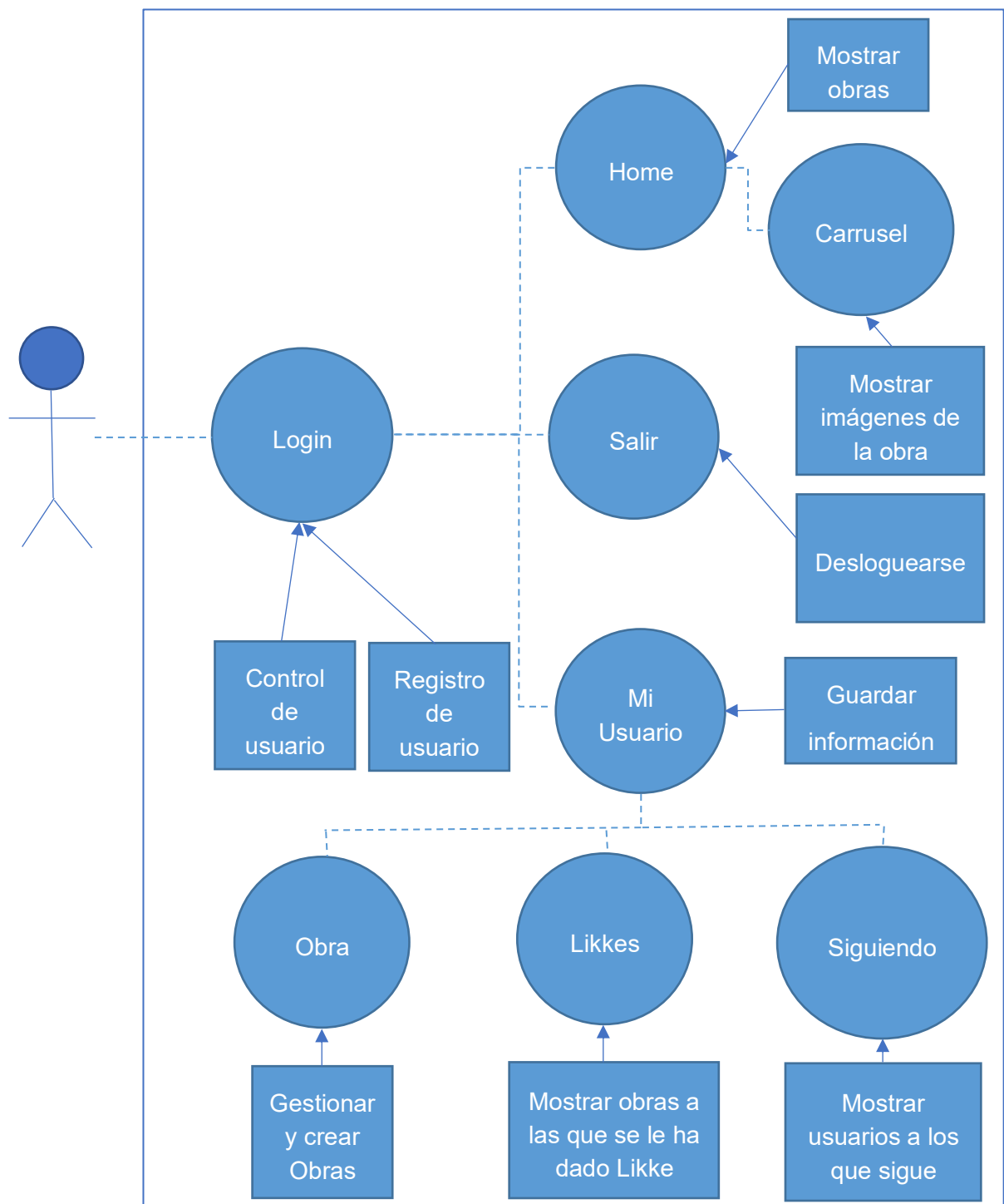
Éstas incluirán a los usuarios y sus obras (gestión de la información de los usuarios, creación y gestión de las obras, seguimiento de las obras, seguimiento de los usuarios y notificación de nueva obra de un usuario seguido).

4.3 Diagrama general de casos de uso

Se muestra a continuación el diagrama de casos de uso genérico para la aplicación. El usuario tendrá dos formas de acceder a la aplicación, mediante credenciales (correo electrónico y

contraseña) o mediante un usuario limitado (este usuario solo podrá ver los datos, pero no podrá subir o modificar ninguno).

Una vez verificado (con correo y contraseña), el usuario podrá acceder mediante un menú desplegable a todas las funciones de la aplicación, inicio (donde podrá ver las obras, darles *Likke* y seguir a los usuarios), mi perfil (donde podrá gestionar su información personal y subir obras) y salir (desloguea de la aplicación).



5 Diseño

El diseño de la base de datos de ArtNutria fue una de las fases más delicadas, se estudió con cuidado para la implementación final. Incluso en la codificación hubo que realizar cambios menores en el diagrama entidad relación debido a situaciones que no se habían tenido en cuenta.

5.1 Diagrama entidad-relación y modelo relacional

El diagrama entidad-relación de ArtNutria abarca dos entidades, la primera corresponde a los usuarios y la segunda a las obras, vinculándose con una relación uno a muchos, ya que un usuario podrá tener muchas obras, pero no al revés.

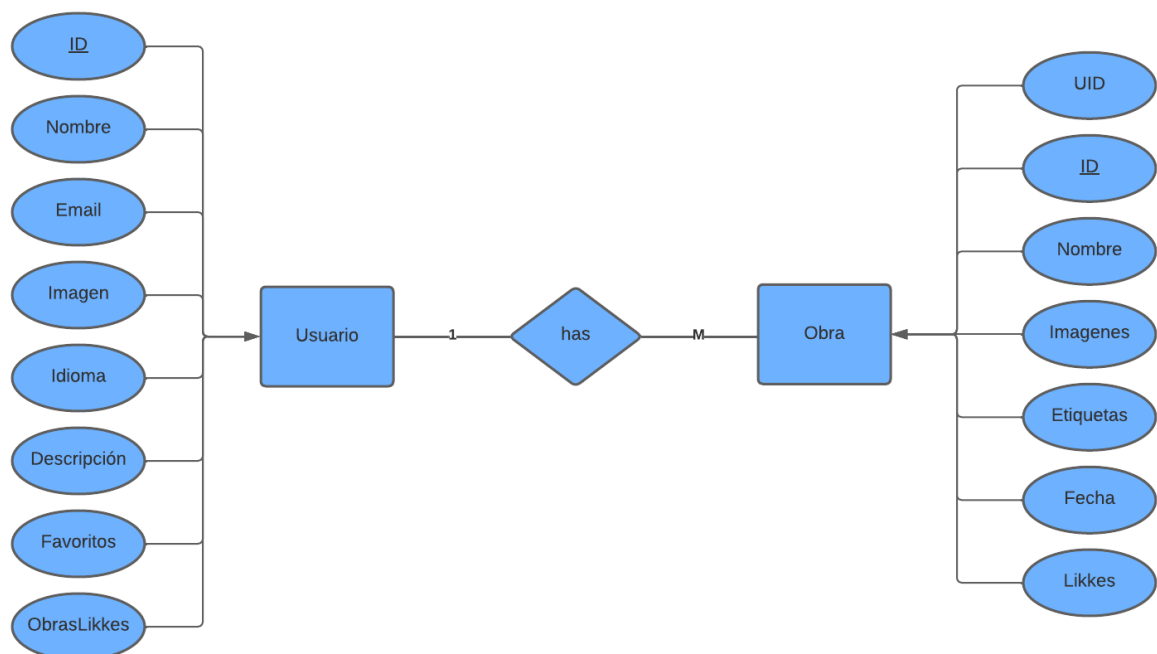


Imagen 11: Diagrama entidad-relación

5.2 Diagrama de clases

Firebase no necesita crear clases para la tabla pivoté, pues permite la creación, modificación y eliminación de los datos y tablas.

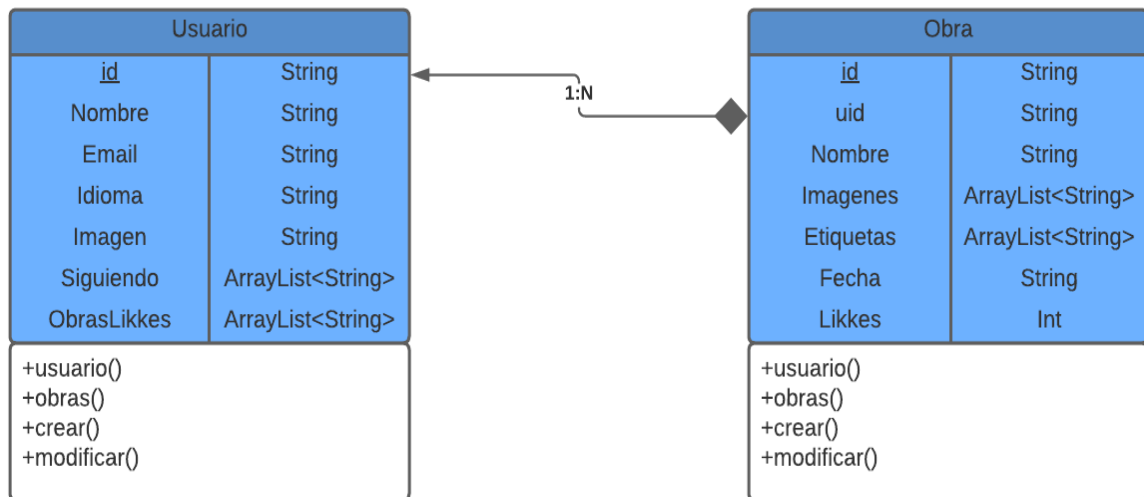


Imagen 12: Diagrama de clases

5.3 Descripción de tablas y campos

En ArtNutria encontramos dos tablas:

La primera es la tabla de usuarios, que contendrá los datos de los usuarios, estos datos serán, **id** (identificador único del usuario), **nombre** (nombre dentro de la aplicación del usuario), **email** (correo electrónico del usuario), **idioma** (idioma principal del usuario), **imagen** (foto de perfil del usuario), **siguiendo** (id de los usuarios que sigue) y **obrasLikkes** (id de las obras a las que el usuario ha dado *Likke*)

La última es la tabla de Obra, que contendrá todos los datos relacionados con las obras de los usuarios, estos datos son, **id** (identificador único de la obra), **nombre** (nombre de la obra), **imágenes** (conjunto de urls que apuntan a las imágenes guardadas en Firestore), **etiquetas** (lista de etiquetas de la obra), **fecha** (fecha en *string* de la creación de la obra), **likkes** (número de *likkes* que ha recibido la obra) y **uid** (identificador único del usuario que ha creado la obra).

5.4 Mapa conceptual

El mapa conceptual de ArtNutria, es muy similar al diagrama de casos de uso. En la imagen inferior cuando se menciona la palabra <<gestiona>> se refiere a las operaciones básicas de lectura, escritura y modificación.

Como podemos ver el usuario es capaz de gestionar su propia cuenta y las obras que el crea, mientras que los datos de otros usuarios solo los podrá leer.

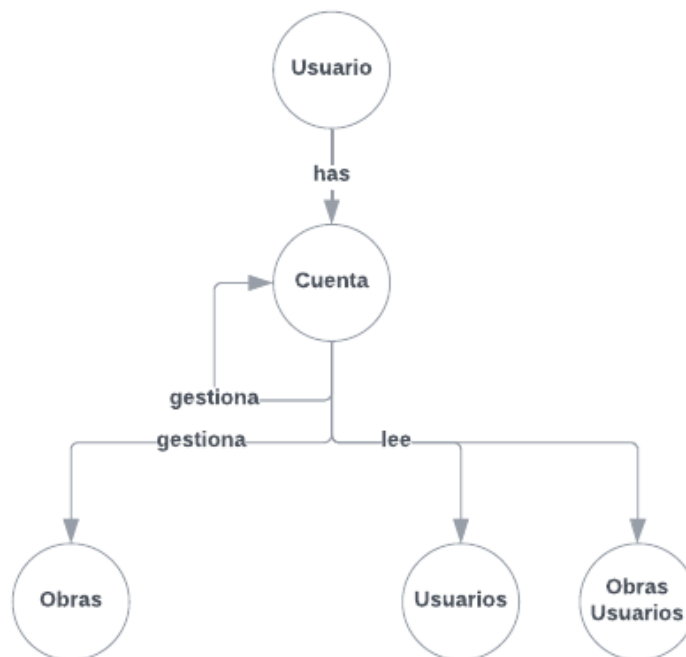


Imagen 13: Mapa conceptual

5.5 Diagrama de secuencias

Las operaciones más importantes de la aplicación son las que conciernen leer, escribir y modificar, así pues, se ha optado por realizar un diagrama de secuencias genérico, que sirve para cualquiera de las operaciones. Como podemos ver no hemos añadido la función eliminar, esto es porque no nos interesa que el usuario pueda ejecutar esta acción.

La interacción comenzará con el usuario enviando una petición de tipo *GET* o *POST* a través de una vista, esta contactará con el método adecuado del controlador de la vista, y a su vez este efectuará una validación y posteriormente procederá a crear el objeto oportuno a través del modelo. Una vez creado el objeto, el controlador interactuará con la base de datos para realizar la petición adecuada. Si la operación tiene éxito, se retorna al usuario un mensaje de éxito que se mostrara en la vista.

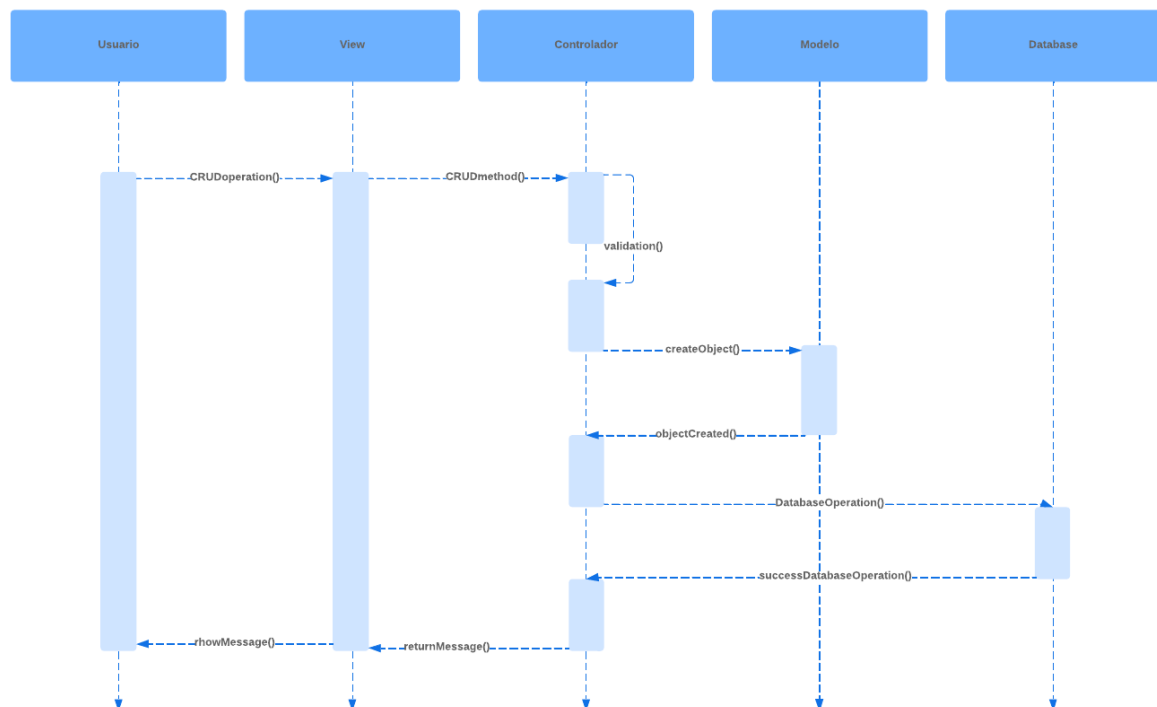


Imagen 14: Diagrama de secuencias

5.6 Diagrama de actividades

A continuación, se muestra tres diagramas de actividades para ArtNutria. El primero, es un diagrama genérico y simple que podría ser trasladado a cualquier otro proyecto con cambios leves, el segundo, muestra la creación de un usuario dentro de la aplicación, y por último se podrá ver como una obra es creada o modificada dependiendo si existe previamente.

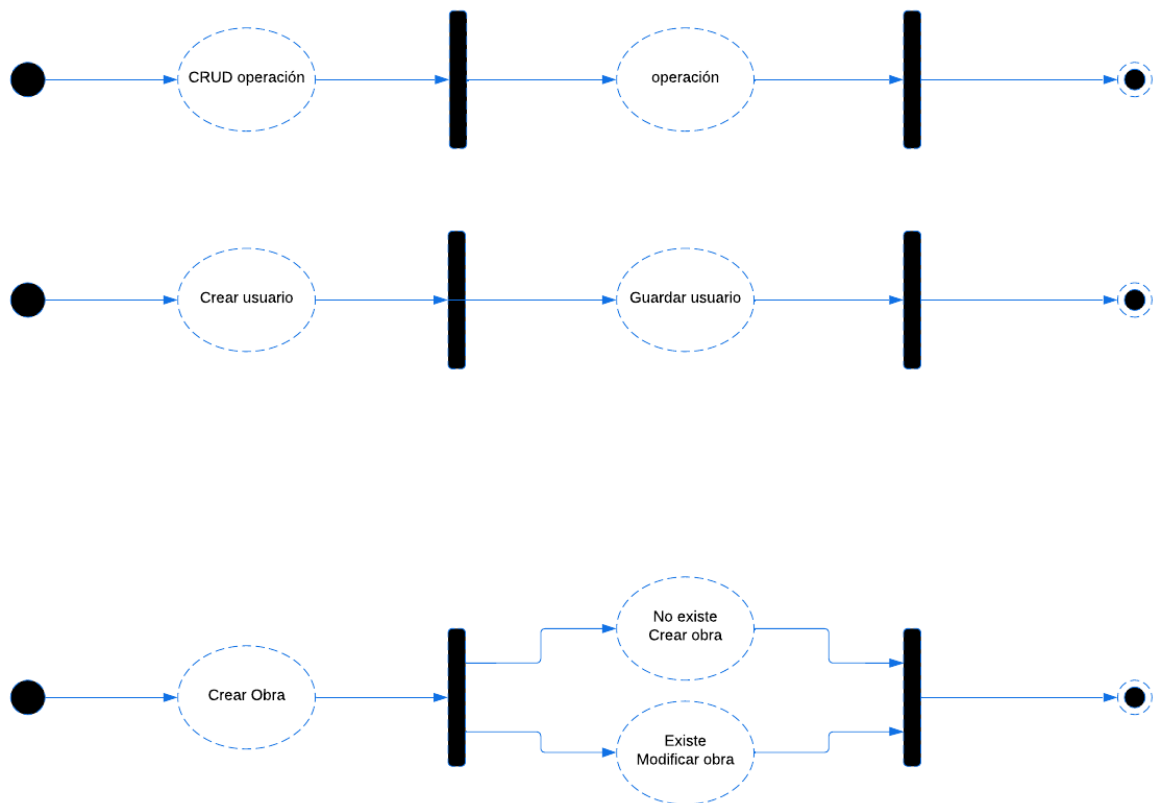


Imagen 15: Diagrama de actividades

5.7 Diagrama de flujo

El diagrama de flujo de la aplicación es bastante sencillo, ya que todo girara en torno a el CRUD.

Así que un posible diagrama de flujo genérico en la aplicación sería el que se muestra a la izquierda.

El usuario accedería a la aplicación e introduciría su correo y contraseña. Si estos son verificados, podría acceder a las operaciones CRUD disponibles para las entidades de la aplicación, los datos son validados, y si la validación resulta exitosa la operación se completa con la base de datos. El cierre y salida del sistema lo efectúa el propio usuario, deslogueandose de la aplicación con el botón.

5.8 Mapa de Navegación

El mapa de navegación de la aplicación muestra los enlaces entre las diferentes vistas de esta, de forma que es una representación gráfica de la red que se teje en el sistema a través de los vínculos que lo componen.

Así que en esta sección se muestra tanto el menú desplegable, como los elementos que nos dirigirán a otras vistas.

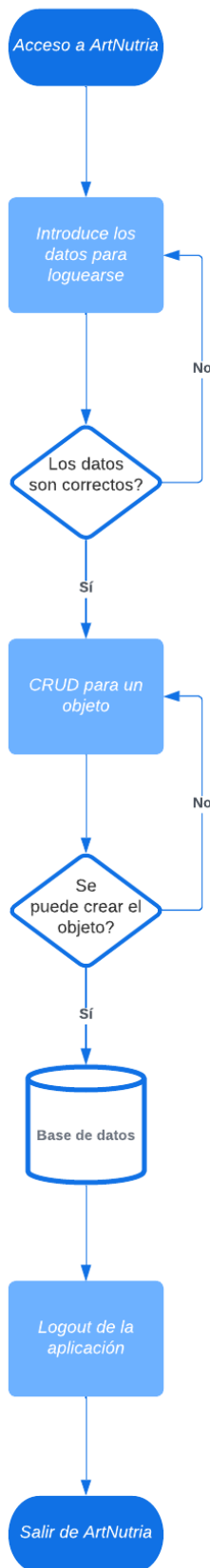


Imagen 16: Diagrama de Flujo

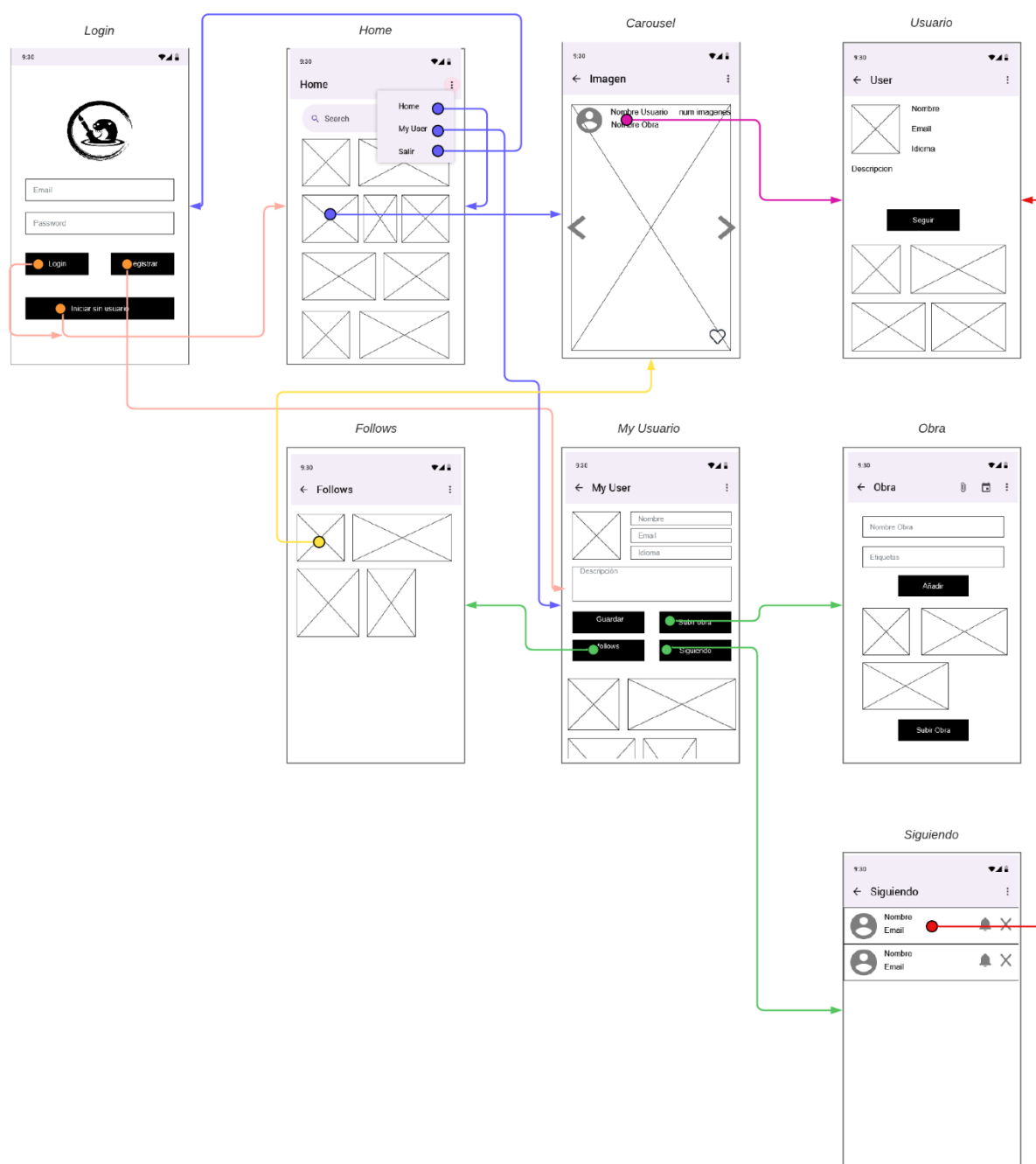


Imagen 17: Mapa de navegación

5.9 Mockups

En ArtNutria encontramos una estructura constante, una barra superior con un menú desplegable que permite la navegación dentro de los componentes de la aplicación.

El espacio que queda, representado en gris en las imágenes de diseño de la página a la derecha es el que es mutable en la aplicación y muestra un contenido distinto en función de la vista que se esté cargando, como se puede apreciar en la captura de móvil de la derecha.

Dentro de la aplicación ArtNutria encontraremos elementos básicos de la aplicación que mantendrán un estilo simple, estos serán compuestos en su mayoría por cuadros de texto, por otra parte los botones tendrán un estilo redondeado con el color verde característico de la aplicación, otro de los elementos que encontraremos son las imágenes pertenecientes a los usuarios las encontraremos siempre como una imagen circular, por último, encontramos las imágenes o videos de las obras, estas ocuparan el suficiente espacio como para que quepan tres en una sola fila, pero, al abrirlas nos mostrara las imágenes de la obra ocupando todo el espacio posible.

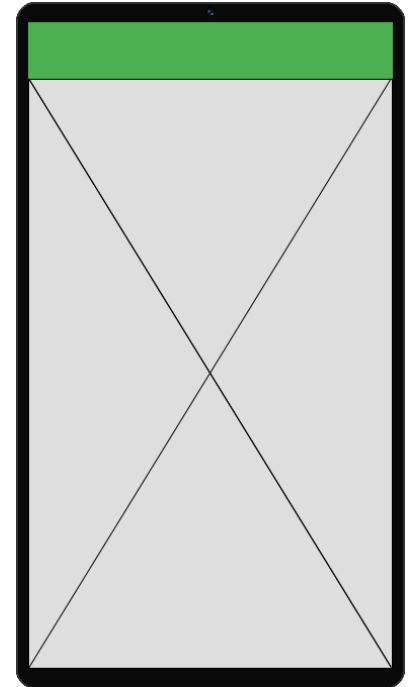


Imagen 18: Mockup base

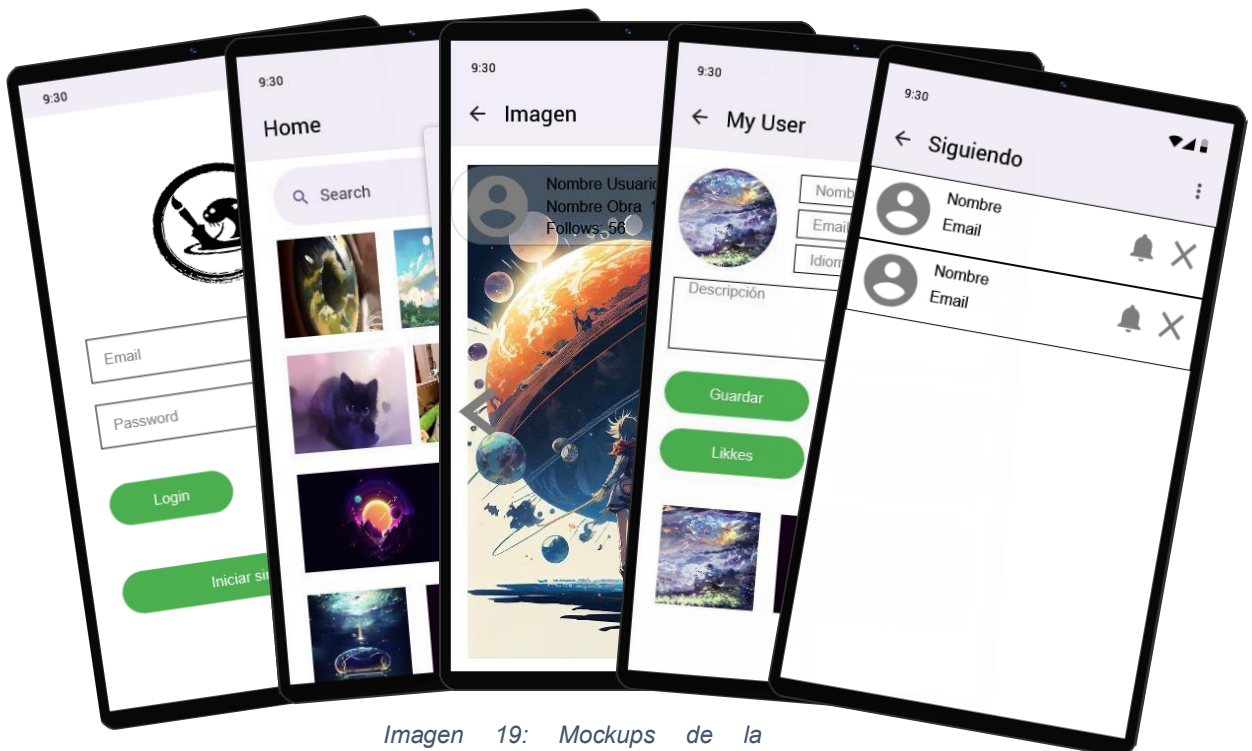


Imagen 19: Mockups de la aplicación

6 Codificación

La fase de codificación (la más larga del proyecto) empezó a mitad del primer mes de desarrollo y continúa abierta, a pesar de haber llegado ya a los objetivos planteados debido a que la aplicación no estaría terminada por completo, ya que la aplicación seguirá evolucionando a lo largo del tiempo para adaptarse y mejorar entorno a las necesidades de los usuarios.

6.1 Diseño grafico

6.1.1 Selección de plantilla

La selección de plantilla fue sencilla, decidimos que la aplicación se viese simple, intuitiva y sin mucho menú, para cumplir esto decidimos utilizar la plantilla vacía, modificándola posteriormente nosotros para adaptarla a las necesidades de la aplicación.

6.1.2 Gama de colores

El siguiente paso de la configuración del aspecto de ArtNutria fue la elección de colores. Se analizaron diferentes colores para la elección de los colores base de la aplicación, se priorizo representa la esperanza del usuario de encontrar un arte que le guste, por lo tanto, se escogió el color verde (representando la esperanza) como color principal de la aplicación, acompañándolo del color predefinido del fondo, blanco o gris claro.

6.2 Proceso de codificación

La codificación de la aplicación resulto ser un proceso complejo y largo, que por motivos prácticos se desordenaba en el tiempo, es decir, no se programaba una tarea detrás de otra, si no, que se programaban varias tareas a la vez.

ArtNutria consta básicamente de operaciones CRUD, como ya se ha mencionado varias veces a lo largo del trabajo, estas operaciones se ejecutan por medio de peticiones a Firebase.

Por lo tanto, los datos devueltos por las peticiones del usuario serán cargarán en la vista para ser mostrados, en el caso de las imágenes utilizaremos Picasso para mostrarlas.

6.2.1 Ejemplos de código interesantes

Durante el desarrollo de ArtNutria se han desarrollado y utilizado muchas funciones interesantes. Por ejemplo, a la hora de consultar datos a Firebase, este trata las consultas como trabajos, provocando que los datos solo estén disponibles mientras el trabajo está siendo realizado, para poder sacar los datos y poder mostrarlos, se ha tenido que crear una corrutina que nos pudiera devolver los datos consultado

```
suspend fun getAllObras(): MutableList<Obra> = withContext(Dispatchers.IO){
    var listaObra = mutableListOf<Obra>()

    db.collection("obras").get().await().forEach {resultado ->
        val obra = resultado.toObject(Obra::class.java)
        listaObra.add(obra)
    }
    listaObra
}
```

Otra función interesante, es la manera en la que subimos los videos u obras al apartado de Firebase dedicado para guardar ficheros (Firestore) y sacamos su url para añadirla posteriormente a las imágenes de la obra.

```
val reference = ds.reference.child("artstorage").child(id_obra + "_obra_" +
contextResolve.getType(image))

reference.putFile(image).await()

val url = reference.downloadUrl.await().toString()
urls.add(url)
```

6.3 Documentación del código

6.3.1 Interna

Android Studio permite documentar el código mediante comentarios, estos se introducen manualmente por el usuario con claves específicas y pueden ser insertado en casi cualquier parte del código, además podemos vincularlo a GitHub, controlando la autoría y fecha de las modificaciones.

Así quedan descritos los ficheros y funciones que forman parte de la aplicación. Teniendo en cuenta que GitHub nos dará la autoría y la fecha de las modificaciones al proyecto.

6.3.2 Externa

Se ha creado un manual de uso para los usuarios, que contiene la instalación y manejo básico de la aplicación.

De cara al futuro se pretende hacer un video explicativo de las funciones básicas de ArtNutria, enseñando al usuario a utilizar la aplicación de manera fácil y sencilla, facilitando el entendimiento de la aplicación.

6.3.2.1 *Manual de usuario*

Para instalar nuestra aplicación abra que descargar nuestra apk desde GitHub en nuestro dispositivo móvil para eso utilizaremos el siguiente enlace:

<https://github.com/AGN20/proyecto/blob/b0f6ed548c60813dec3c125496a1f371379a57b2/ArtNutriaAPK.apk>

Una vez descargada en nuestro dispositivo, abriremos la aplicación, si es la primera vez que la abrimos, apareceremos en la pantalla de Login.

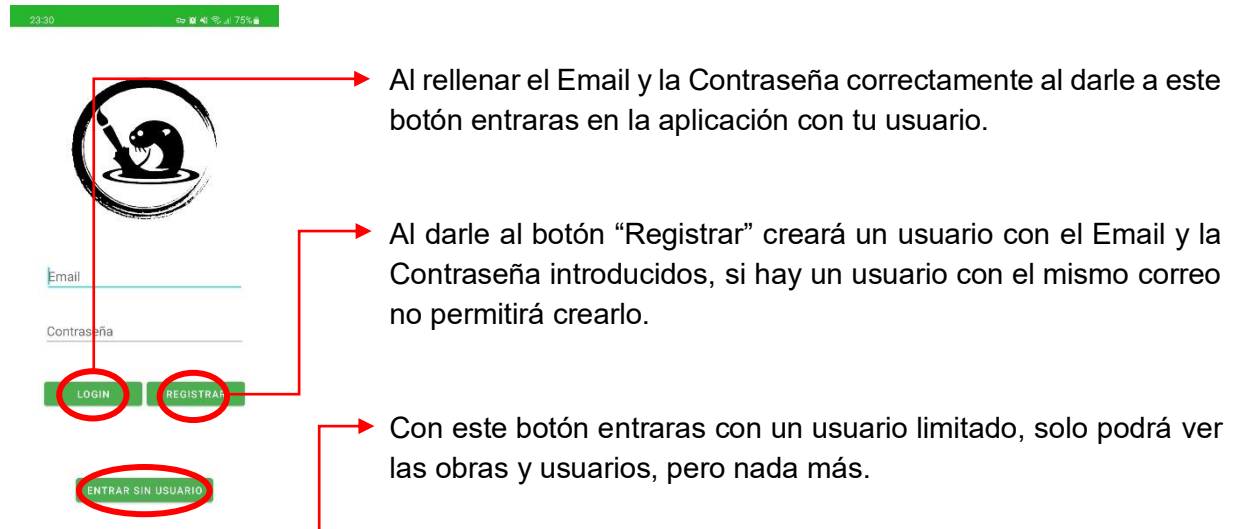


Imagen 20: Login de la aplicación

Una vez logueado podremos ver la siguiente vista, que nos mostrara todas las obras subidas por los usuarios. Los botones de arriba nos permitirán elegir el tipo de búsqueda que vamos a hacer y debajo tendremos el buscador y las imágenes que coinciden con lo que buscamos

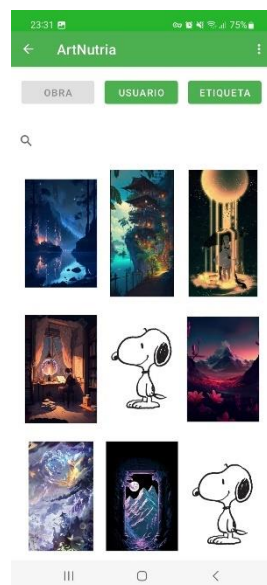


Imagen 21: Home app

Aquí tienes una muestra de cómo funcionarían las búsquedas

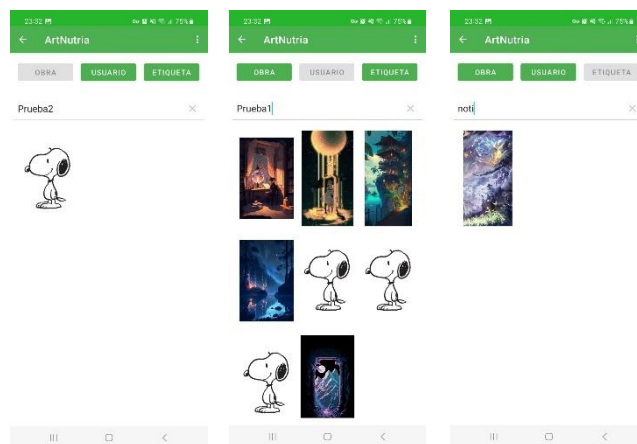



Imagen 22: Ejemplos de búsqueda

Cabe destacar, en la parte superior derecha encontraremos el menú con el símbolo de tres puntos (), al apretarlo nos abrirá un menú desplegable que tendrá tres opciones, Inicio (nos dirigirá a esta pantalla), Mí perfil (nos llevará a nuestro perfil) y salir (nos sacará de la cuenta y volveremos al login).

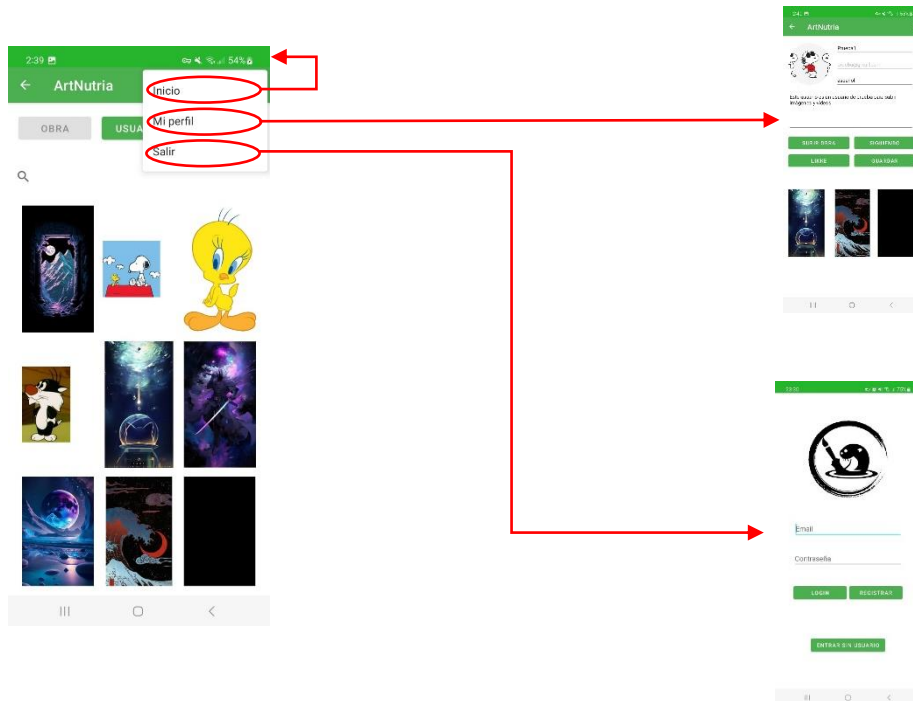


Imagen 23: Ejemplo menú

Otra de las funciones que hay en esta pantalla se encuentra al apretar cualquier imagen o video, esta nos dirigirá a una pantalla donde se nos mostrará todas las imágenes o videos de esa obra con algo de información del usuario y de la obra, donde podremos seguir al usuario y darle *like* a la obra. También podemos dirigirnos a los datos del usuario que ha subido esa obra, dando en el banner donde sale la información general de la obra.

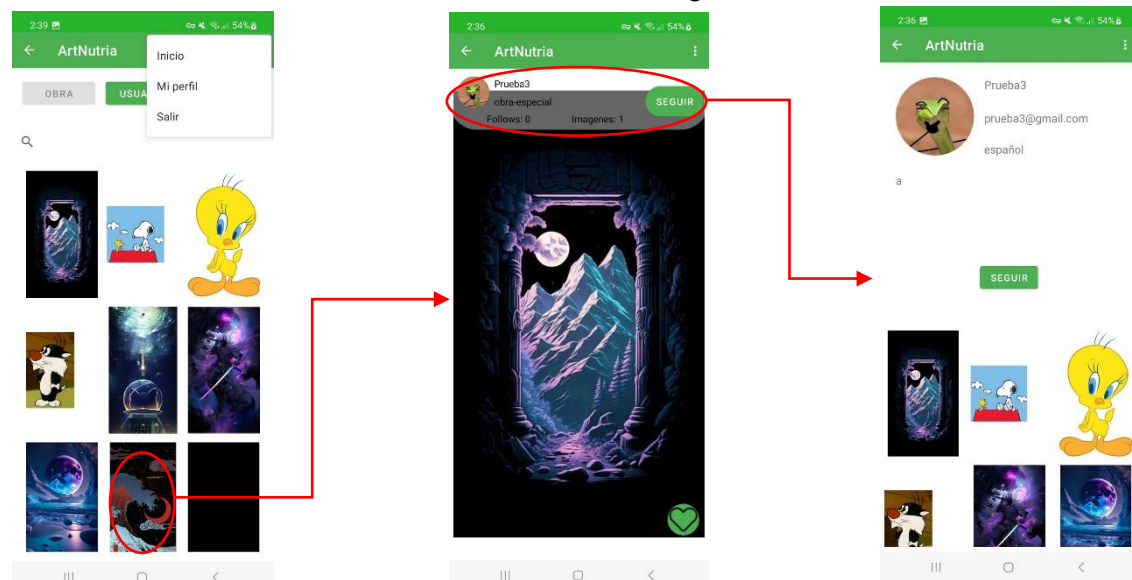


Imagen 24: Ejemplo carrusel

Si nos dirigimos a “*mi usuario*” veremos una vista que nos muestra nuestra información, las obras subidas y cuatro botones, “*guardar*” nos guardará los datos de nuestro usuario, “*subir obra*” nos dirigirá a una pantalla donde podremos crear o modificar una obra, “*Likkes*” nos dirigirá a una lista de obras a las que hemos dado *Likke*, “*siguiendo*” nos dirigirá a una lista de usuarios a los que seguimos, también podremos cambiar la imagen de la foto de perfil cliqueando en esta.

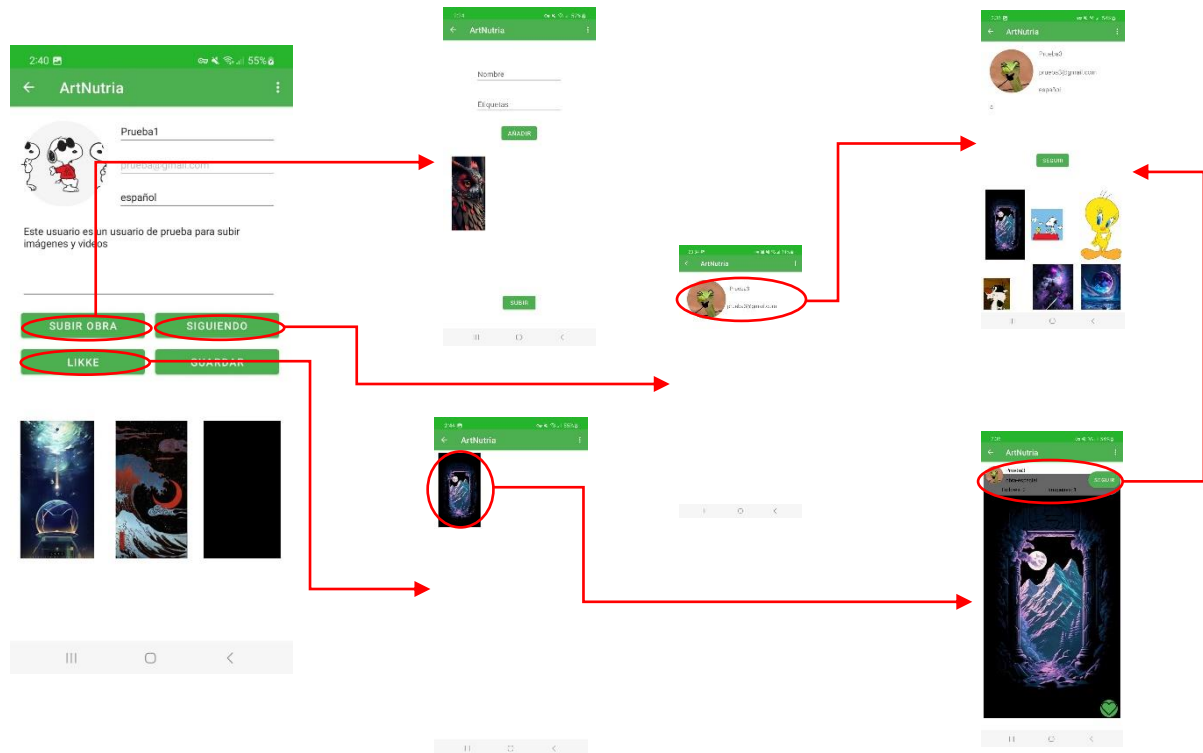


Imagen 25: Ejemplo pantalla *mi usuario*

Igual que en home, las imágenes que pertenecen a obra al cliquear encima nos dirigirán a la vista con la imagen en grande y un carrusel con todas las imágenes, y lo mismo pasa en la parte de siguiendo, que nos dirigirá a la página del usuario.

Y por último explicaremos como subir y modificar una obra, entraremos a “mi perfil” y daremos en subir obra, esto nos dirigirá a la vista, nos pedirá poner el nombre de la obra y las etiquetas, con el botón “añadir imagen” podremos añadir las imágenes o videos que queramos a la obra y con el botón “subir” subiremos la obra. Para modificar solo tendremos que poner el nombre de la obra que queramos modificar y ya estaremos modificando la obra existente

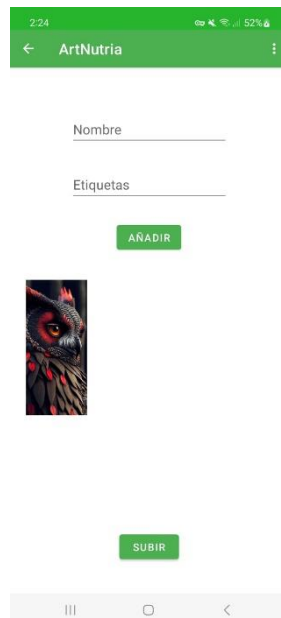


Imagen 26: Subir obra aplicación

6.4 Seguridad

Uno de los motivos por los que se eligió Firebase como base de datos fue por la cantidad de herramientas que nos brinda, una de estas herramientas es Firebase authentication, como su propio nombre indica, nos permite autenticar los usuarios que acceden a nuestra aplicación mediante el SDK de Firebase authentication para integrar de forma manual uno o más métodos de acceso.

Aunque Firebase authentication permite la autenticación de varias maneras, nosotros nos centraremos en correo y contraseña, que mediante el método [`signInWithEmailAndPassword`](#) podremos identificar si el usuario existe fácilmente o por el caso contrario con [`createUserWithEmailAndPassword`](#) podremos crear el usuario.

6.4.1 Inyecciones de código

ArtNutria se comunica con Firebase mediante consultas ya preparadas, permitiendo así el paso de parámetros de forma segura.

6.4.2 Excepciones

En cuanto al manejo de excepciones se capturaron todos los conocidos. Para evitar errores en el método *create* los elementos se crean con un modelo con variables ya predefinidas, que van cambiando por los especificados por el usuario siempre que no sean nulos, al terminar ejecuta la función sobre el modelo ya modificado. En el caso del método *update*, controla que lo pasado no sea nulo o vacío y modifica el fichero.

6.5 Compatibilidad

Durante el desarrollo de ArtNutria se ha buscado la máxima compatibilidad de la aplicación con los diferentes dispositivos móviles, pensando principalmente para su manejo en pantallas verticales.

7 Implantación

Aunque el presente proyecto haya culminado, lo cierto es que ArtNutria como aplicación en realidad aún no está preparada para hacer un despliegue al uso, por lo que la descripción que sigue es para cualquier aplicación que utilice Android Studio como IDE y Kotlin como lenguaje de programación.

7.1 Clonar el proyecto

En primer lugar, ejecutar:

```
git clone  
https://github.com/AGN20/proyecto/tree/b0f6ed548c60813dec3c125496a1f371379a57b2/ArtNutria
```

En caso de no poder clonar el proyecto, se puede descargar el zip, con:

```
git clone  
https://github.com/AGN20/proyecto/blob/b0f6ed548c60813dec3c125496a1f371379a57b2/ArtNutria.zip
```

Se requiere tener instalado Android Studio. Una vez clonado el proyecto, podremos iniciarlo sin problemas.

7.2 Despliegue de Firebase

No hace falta la configuración de Firebase, ya está configurada en el proyecto y no sería necesario volver a configurarlo. En el caso de que quieras añadirlo a tu Firebase personal (necesitaras una cuenta en Firebase y ten en cuenta que esta aplicación ya ha sido añadida a un proyecto en Firebase, esto te dará problemas al intentar añadirla a otro proyecto en Firebase, en este caso cree una nueva aplicación y copie los elementos necesarios para el funcionamiento de la aplicación):

1. En la consola de Firebase, crearemos un nuevo proyecto.

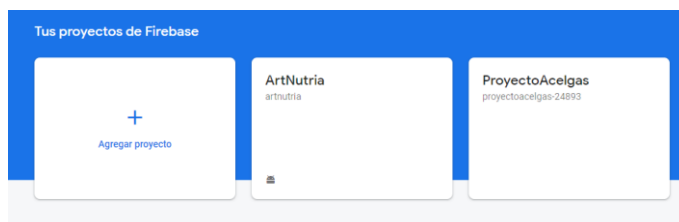


Imagen 27: Ejemplo crear proyecto 1

2. Ingresaremos el nombre de nuestro proyecto, admitiremos el Google Analytics y crearemos una cuenta nueva de Google Analytics, le decimos la ubicación donde se va a crear la app y aceptamos las condiciones.



Imagen 29: Ejemplo crear proyecto 2.1



Imagen 30: Ejemplo crear proyecto 2.2

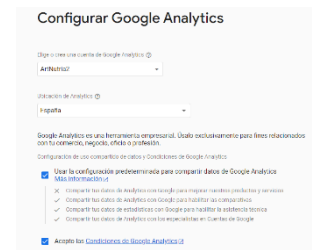


Imagen 28: Ejemplo crear proyecto 2.3

3. Una vez dentro daremos al icono de Android, y comenzaremos a registrar nuestra app, para esto último nos pedirá el nombre del paquete de Android, un sobrenombre para la app y el SHA-1 (se explicará cómo sacarlo en el paso siguiente).

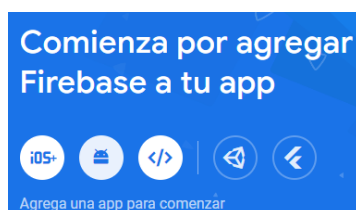


Imagen 32: Ejemplo crear proyecto 3.1

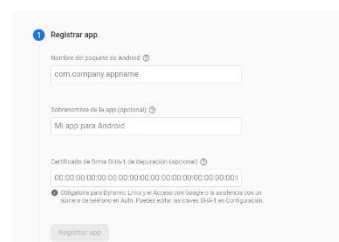


Imagen 31: Ejemplo crear proyecto 3.2

- Para sacar el SHA-1 de nuestro proyecto, nos dirigiremos a Android Studio abriendo nuestro proyecto, en la parte derecha de Android Studio encontraremos el **Gradle**, abrimos el **Execute Gradle Task** y ejecutamos el comando "**gradle signingReport**" nos dará un resultado en la parte de abajo, podremos ver hay nuestro SHA-1 (saldrá como SHA1) solo tendremos que copiarlo y añadirlo en Firebase.

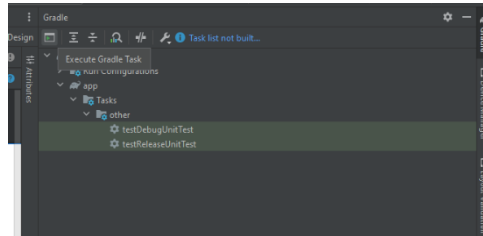


Imagen 34: Ejemplo crear proyecto 4.1

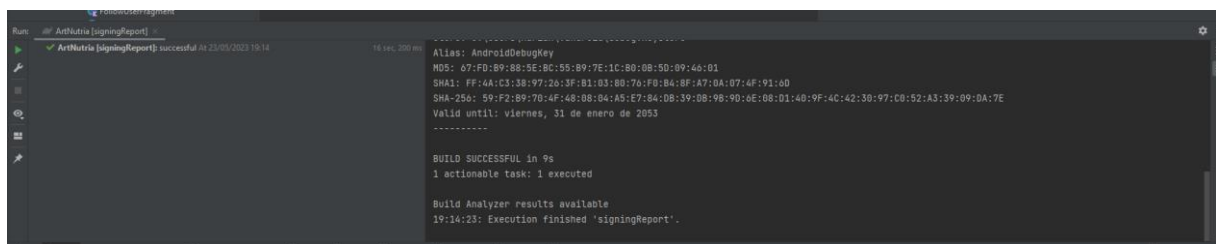


Imagen 33: Ejemplo crear proyecto 4.2

- Una vez tengamos los datos y demos a continuar nos pedirá que descarguemos un fichero y lo añadamos en la carpeta aplicación del proyecto (activa la vista de proyecto en Android Studio para verla).

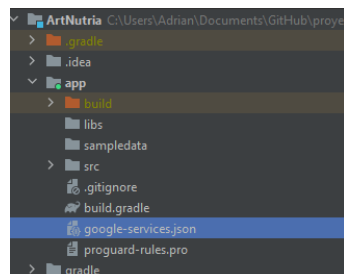
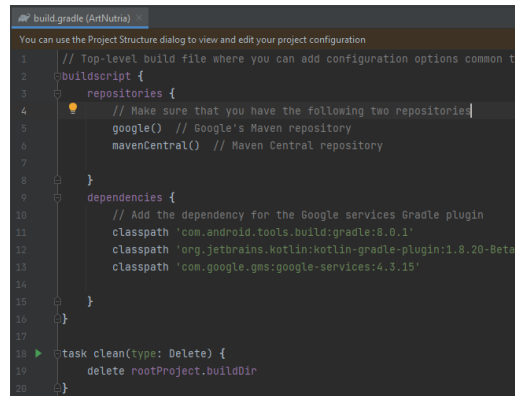


Imagen 35: Ejemplo crear proyecto 5

6. Ahora tendremos que introducir las configuraciones de Firebase, en el `build.gradle(project)`.

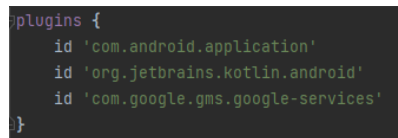


```
// build.gradle (ArtNutria)
You can use the Project Structure dialog to view and edit your project configuration
1 // Top-level build file where you can add configuration options common to
2 // subprojects
3 buildscript {
4     repositories {
5         // Make sure that you have the following two repositories
6         google() // Google's Maven repository
7         mavenCentral() // Maven Central repository
8     }
9     dependencies {
10        // Add the dependency for the Google services Gradle plugin
11        classpath 'com.android.tools.build:gradle:8.0.1'
12        classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.8.20-Beta'
13        classpath 'com.google.gms:google-services:4.3.15'
14    }
15 }
16
17 task clean(type: Delete) {
18     delete rootProject.buildDir
19 }
```

Imagen 36: Ejemplo crear proyecto 6

7. Y las implementaciones y configuraciones en el `build.gradle(app)`.

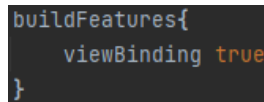
Implementamos los *plugins*:



```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'com.google.gms.google-services'
}
```

Imagen 37: Ejemplo crear proyecto 7.1

Implementamos el *Binding*:



```
buildFeatures{
    viewBinding true
}
```

Imagen 38: Ejemplo crear proyecto 7.2

Y las dependencias:

```
implementation 'androidx.core:core-ktx:1.7.0'
implementation 'androidx.appcompat:appcompat:1.6.1'
implementation 'com.google.android.material:material:1.8.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
testImplementation 'junit:junit:4.13.2'
androidTestImplementation 'androidx.test.ext:junit:1.1.5'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'

//FIREBASE
implementation platform('com.google.firebase:firebase-bom:31.2.3')
implementation 'com.google.firebase:firebase-analytics-ktx'
implementation 'com.google.firebase:firebase-auth:21.2.0'
implementation 'com.google.firebase:firebase-firestore:24.4.5'
implementation 'com.firebaseui:firebase-ui-firestore:8.0.0'
implementation 'com.firebaseui:firebase-ui-storage:8.0.0'
implementation platform('com.google.firebase:firebase-bom:31.2.3')
implementation 'com.google.firebase:firebase-storage-ktx'
implementation 'com.google.firebase:firebase-messaging'

//Picasso
implementation 'com.squareup.picasso:picasso:2.8'

//Navigation
implementation 'androidx.navigation:navigation-fragment-ktx:2.5.2'
implementation 'androidx.navigation:navigation-ui-ktx:2.5.2'
implementation 'androidx.viewpager2:viewpager2:1.0.0'
implementation 'com.google.android.material:material:1.7.0'

//Imágenes circulares
implementation 'de.hdodenhof:circleimageview:2.2.0'

//Retrofit
implementation 'com.squareup.retrofit2:retrofit:2.6.2'
implementation 'com.squareup.retrofit2:converter-gson:2.6.0'
```

Imagen 39: Ejemplo crear proyecto 7.3

8. También será necesario implementar los permisos en el *manifest*:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

Imagen 40: Ejemplo crear proyecto 8

9. Volvemos a la consola de Firebase y vamos a agregar la autenticación por correo, para esto añadimos Authentication (lo encontraremos en la parte de Compilación) y en *sign-in-method* añadimos agregamos el proveedor de correo electrónico/contraseña.

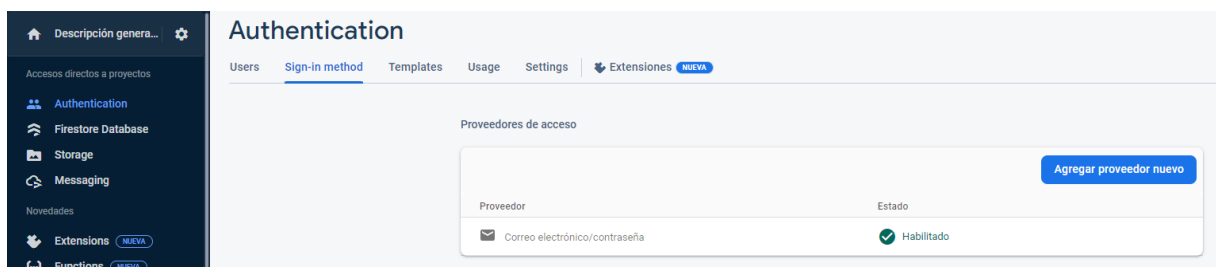


Imagen 41: Ejemplo crear proyecto 9

10. Ahora agregaremos Firestore Database y Store en la consola de Firebase, se encuentran en la parte de Compilación y habrá que modificar las reglas de ambos.

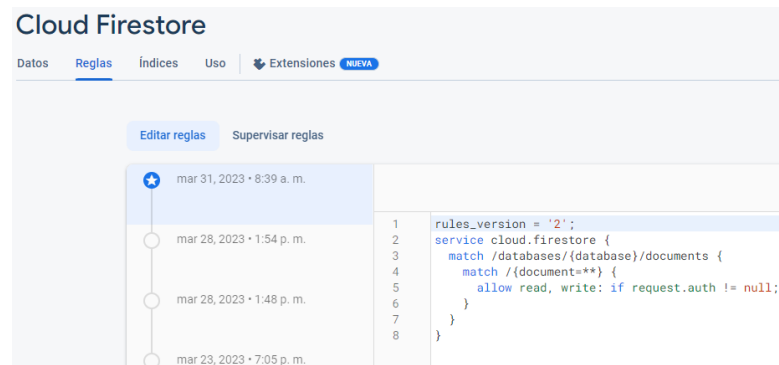


Imagen 43: Ejemplo crear proyecto 10.1



Imagen 42: Ejemplo crear proyecto 10.2

11. En el caso de store, sí que se creara la jerarquía de ficheros, aquí tienes el ejemplo de la que se utiliza en ArtNutria, dos carpetas, [artstorage](#) (guarda las imágenes de obras) y [userimage](#) (guarda las imágenes de usuario).

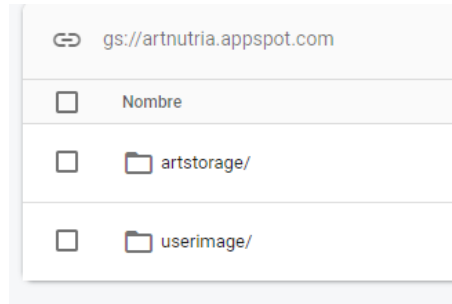


Imagen 44: Ejemplo crear proyecto 11

12. Añadiremos también en la parte de Participación añadimos el Messaging, para las notificaciones.

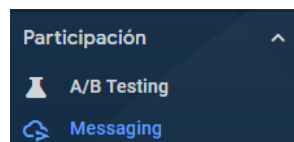


Imagen 45: Ejemplo crear proyecto 12

13. Y por último tendremos que añadir y configurar Firebase Functions, este nos servirá para controlar cuando se lanzaran las notificaciones, para esto lo añadimos desde nuestra consola de Firebase a nuestro proyecto, lo encontraremos como Functions.

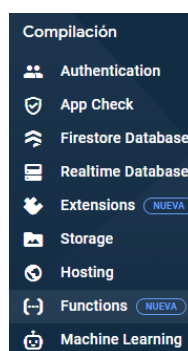


Imagen 46: Ejemplo crear proyecto 13

14. Ahora tendremos que añadir una serie de cosas en nuestro ordenador, para eso abriremos PowerShell y ejecutaremos los siguientes comandos, nos pedirá que seleccionemos el proyecto y que lenguaje utilizaremos (en nuestro caso JavaScript):
- `npm install firebase-functions@latest firebase-admin@latest --save`
 - `npm install -g firebase-tools`
 - `firebase login`
 - En el lugar que queramos que este los elementos de configuración de las funciones ejecutamos: `firebase init functions`
15. Con la última función ejecutada, se nos descargara un directorio con la configuración de las funciones, pero nos faltara añadir nuestra función buscaremos en el directorio anteriormente mencionado el fichero `index.js` y modificaremos el contenido, quedando de la siguiente manera.

```
const functions = require("firebase-functions");
const admin = require("firebase-admin");
admin.initializeApp();

exports.enviarNotificacion = functions.firestore
  .document("/obras/{obraId}")
  .onCreate(async (snapshot, context) => {
    const obra = snapshot.data();
    const usuarioId = obra.userId;

    const usuarioSnapshot = await admin.firestore()
      .doc(`/usuarios/${usuarioId}`).get();
    const usuario = usuarioSnapshot.data();

    const payload = {
      notification: {
        title: "El usuario " + usuario.nombre + " ha subido una nueva obra",
        body: "Se ha agregado la obra " + obra.nombre,
      },
    };

    return admin.messaging().sendToTopic(usuarioId, payload)
      .then((response) => {
        console.log("Notificación enviada con éxito:", response);
        return null;
      })
      .catch((error) => {
        console.error("Error al enviar la notificación:", error);
        return null;
      });
  });
```

Imagen 47: Ejemplo crear proyecto 14

16. Por último, ejecutaremos el comando: `firebase deploy --only functions`, con eso subiremos nuestra función a Firebase Functions. Es posible que la función falle, en ese caso comprueba que tienes la API y el Servicio de Firebase Cloud Messaging integrados en la consola de Google Cloud, para esto seguiremos los siguientes pasos:

- Iremos al menú desplegable en la esquina superior izquierda y dentro del desplegable de “APIs y servicios” buscaremos “APIs y servicios habilitados”.

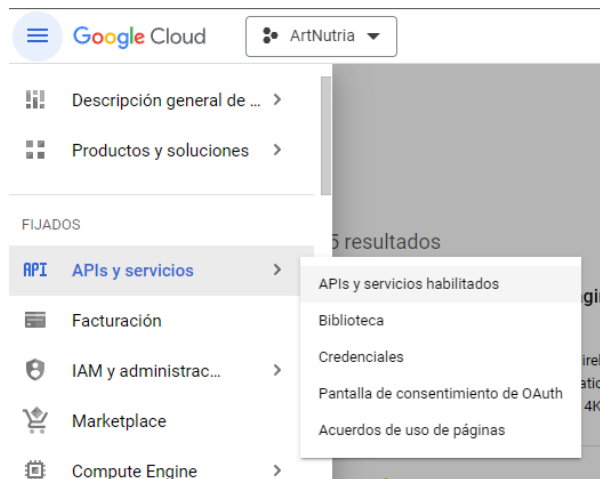


Imagen 48: Ejemplo crear proyecto 15

- Clicaremos en “HABILITAR APIS Y SERVICIOS”, en la parte superior.

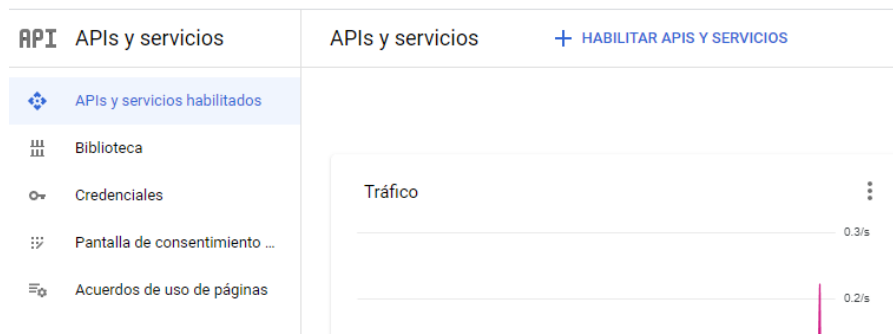


Imagen 49: Ejemplo crear proyecto 16

- Y buscamos en el buscador “Cloud Messagin” confirmaremos que tengamos “Cloud Messaging” y “Firebase Cloud Messaging API”, en el caso de que nos falte alguno, lo añadiremos.

7.3 Diagrama de despliegue

El diagrama de despliegue de ArtNutria muestra la interacción entre el software que entronca la aplicación, de forma simple. En términos generales esta aplicación serviría para cualquier aplicación que utilizara Firebase como base de datos.

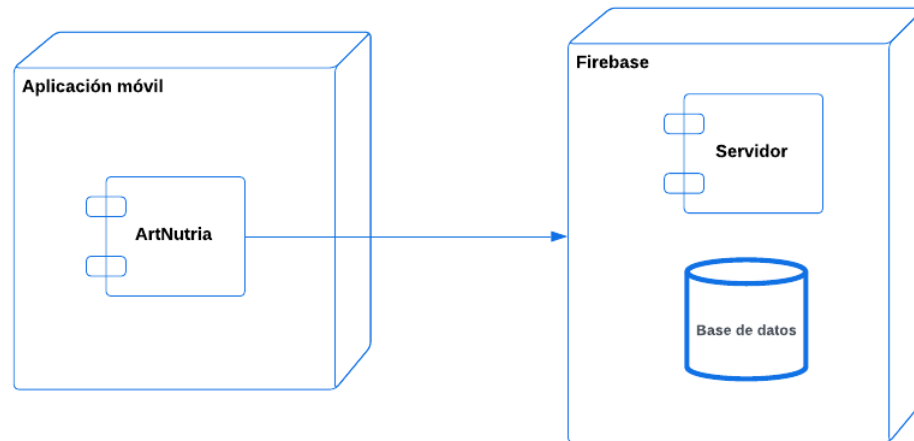


Imagen 50: Diagrama de despliegue

8 Conclusiones

El presente proyecto se planteó como idea el verano de 2022, antes de comenzar segundo de DAM. Llegar a este punto y poner en perspectiva el camino recorrido es muy interesante. Desde la premisa inicial, la aplicación ha sido sometida a variaciones, dado el aprendizaje y el descubrimiento de nuevas herramientas y formas de programar.

En etapas iniciales el proyecto se pretendía parecer a un escaparate de imágenes para que los artistas pudieran mostrar sus obras. Algunas de las tecnologías que se plantearon utilizar fueron SQLite (<https://www.sqlite.org/>) y Berkeley DB (<https://www.oracle.com>). Final mente, decidimos descartar esas opciones y utilizar Firebase, más pensando en la comodidad y familiaridad de los programadores en su uso.

8.1 Ampliaciones y mejoras

Si bien ya se ha mencionado varias veces a lo largo del presente trabajo ArtNutria no es una aplicación que se pueda considerar terminada y cerrada para siempre, y dentro de ella hay mucho espacio para su ampliación y mejora.

Entre las mejoras más destacables, estaría la integración de un sistema de pedidos y pagos en el que los usuarios pudieran contactar con otros para encargar o comisionar su arte, de manera fácil y sencilla.

Otra de las mejoras que se podrían implementar y que por tiempo no se pudo incluir, fue un chat entre usuarios, donde estos podrían hablar y discutir sobre las comisiones, facilitando el contacto entre ellos.

Se podría implantar como mejora la creación de un apartado en la parte de usuario, donde poder colocar los enlaces a las redes sociales del usuario, tales como twitter, patreon y TikTok, entre otros.

A demás no hay que olvidar que ArtNutria de momento no dispone de traducción a otros idiomas, por lo que sería interesante disponer de esta traducción a varios idiomas o implementar alguna librería que pudiera traducir dependiendo del idioma del usuario.

También sería interesante, que los usuarios dispongan de una gráfica de seguimiento, que les permita observar la visualización y likes de las obras y seguidores de su cuenta.

Otra de las cosas que se pueden agregar seria personalizar las obras que se muestran al usuario dependiendo de sus búsquedas anteriores.

Por último, una de las maneras en las que se puede monetizar la aplicación, seria añadiendo anuncios en banners en lugares vacíos dentro de la aplicación, o llevándose un x por ciento de los beneficios de una venta.

9 Bibliografía

9.1 Apuntes y material didáctico

[Temario del curso 2ºDAM Multimedia and Handheld Device Programming](#)

Temario del curso impartido el curso de 2022 a 2023 que recopila la como trabajar con Android Studio, programar con Kotlin e implementar diferentes tipos de librerías.

Creado por Carlos Tarazona

9.2 Páginas web y tutoriales

<https://es.stackoverflow.com>

Santo grial del programador, lugar donde poder consultar dudas (ingles).

<https://firebase.google.com/docs/android/setup?hl=es-419>

Documento oficial de Firebase que explica como enlazar la aplicación a la plataforma.

<https://youtu.be/pNleQQhVfd0>

Video explicativo de como subir imágenes a Firebase store.

https://www.youtube.com/playlist?list=PL_963TKBW5HA68EMhxMwjQctUKmQI93UM

Playlist de youtube que explica como implementar Firebase y sus componentes en una aplicación.

<https://stackoverflow.com/questions/46590155/firestore-permission-denied-missing-or-insufficient-permissions>

Solución de los permisos de escritura y lectura de Firebase.

https://firebase.google.com/docs/firestore/manage-data/add-data?hl=es-419#kotlin+ctx_9

Documentación de Firebase para la utilización de Firestore.

<https://stackoverflow.com/questions/72384736/retrieve-firebase-data-using-coroutines>

Solución para poder sacar los datos de Firebase y mostrarlos en la aplicación.

https://youtu.be/LvP_wcFMZ1Q

Video de youtube que explica cómo utilizar el SearchView

<https://es.stackoverflow.com/questions/9068/botones-redondeados-en-android>

Solución para mostrar los botones redondeados

https://youtu.be/bWlb2h_733E

Video que muestra cómo crear un Carrusel sencillo

<https://youtu.be/An-mS2oneCw>

Video que te enseña a poner las imágenes circulares en la aplicación.

<https://engineering.monstar-lab.com/en/post/2021/02/09/Use-Firebase-Cloudfunctions-To-Send-Push-Notifications/>

Pagina que te enseña a configurar Firebase Functions, para generar las notificaciones.

<https://stackoverflow.com/questions/57767439/an-error-occurred-when-trying-to-authenticate-to-the-fcm-servers>

Pagina para resolver el problema de las notificaciones, error de autenticación.

<https://youtu.be/rdFjT0bQBgs>

Video explicativo sobre los permisos.

9.3 Agradecimientos

A mis compañeros de DAM

Gracias por estos años de risas y buen rollo. Hemos compartido risas y sufrimientos a lo largo del camino, pero, sobre todo, recordare las veces que me hicisteis reír en mis peores momentos.

A mis profesores

Agradezco vuestra paciencia y vuestra ayuda a lo largo del camino, sobre todo en los peores momentos, gracias a cada uno de vosotros.

A mis amigos

Todos vosotros me habéis levantado el ánimo, ayudado a soportar los difíciles momentos, no solo a lo largo del curso, si no de la vida, gracias.

A mis amigos artistas

Solo con vuestros dibujos habéis conseguido emocionarme, y gracias a vosotros idee este proyecto, gracias por todo chicos.

A mi familia

Habéis estado todo el tiempo ayudándome y apoyándome todo lo posible, gracias por todo

Muchas gracias a todos vosotros

10 Anexo

10.1 Código

10.1.1 MainActivity

```
package com.adgonu.artnutria.ui

import android.Manifest
import android.app.AlertDialog
import android.content.Intent
import android.content.pm.ActivityInfo
import android.content.pm.PackageManager
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ProgressBar
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.adgonu.artnutria.R
import com.adgonu.artnutria.databinding.ActivityMainBinding
import com.google.firebase.auth.FirebaseAuth
import java.util.regex.Matcher
import java.util.regex.Pattern

//SHA1: FF:4A:C3:38:97:26:3F:B1:03:80:76:F0:B4:8F:A7:0A:07:4F:91:6D

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        Thread.sleep(3000)
        setTheme(R.style.Theme_ArtNutria)
        setContentView(ActivityMainBinding.inflate(layoutInflater).also {
binding = it }.root)

        requestedOrientation = ActivityInfo.SCREEN_ORIENTATION_PORTRAIT

        var view = View(this)

        if(FirebaseAuth.getInstance().currentUser != null){
            changeActivity()
        }else{
            checkPermissions()
            setup(view)
        }
    }

    private fun setup(view: View) {
        //Iniciamos los elementos del layout
        val btRegistrar: Button = binding.bttRegistrar
        val btLoguin: Button = binding.bttLogin
        val btEntrar: Button = binding.bttEntrar
        val edEmail: EditText = binding.edtEmail
        val edPassword: EditText = binding.edtPassword
    }
}
```

```

val progressBar: ProgressBar = binding.progressBar

var email = edEmail.text
var password = edPassword.text

/** REGISTRAR */
btRegistrar.setOnClickListener {
    progressBar.visibility = View.VISIBLE
    if (email.isNotEmpty() && password.isNotEmpty()){

FirebaseAuth.getInstance().createUserWithEmailAndPassword(email.toString(),
password.toString()).addOnCompleteListener {
    if(it.isSuccessful){
        progressBar.visibility = View.GONE
        changeActivity()
    }else{
        progressBar.visibility = View.GONE
        if(password.toString().length < 6){
showAlertPassword() }
        if(!validarEmail(email.toString())){
showAlertEmail() }
        }
    }.addOnFailureListener {
        progressBar.visibility = View.GONE
        showAlertUser()
    }
}
}

/** LOGUIN */
btLoguin.setOnClickListener {
    progressBar.visibility = View.VISIBLE
    if (email.isNotEmpty() && password.isNotEmpty()){

FirebaseAuth.getInstance().signInWithEmailAndPassword(email.toString(),
password.toString()).addOnCompleteListener {
    if(it.isSuccessful){
        progressBar.visibility = View.GONE
        changeActivity()
    }else{
        progressBar.visibility = View.GONE
        showAlertEntriFailed()
    }
}
}
}

/** ENTRAR */
btEntrar.setOnClickListener {
    progressBar.visibility = View.VISIBLE

FirebaseAuth.getInstance().signInWithEmailAndPassword(this.getString(R.stri

```



```
ng.NotUser),
this.getString(R.string.NotUserPassword)).addOnCompleteListener {
    if(it.isSuccessful){
        progressBar.visibility = View.GONE
        changeActivity()
    }else{
        progressBar.visibility = View.GONE
        showAlertEntriFailed()
    }
}
}

/** El usuario ya ha sido creado */
private fun showAlertUser() {
    val builder = AlertDialog.Builder(this)
    builder.setTitle(this.getString(R.string.TituloError))
    builder.setMessage(this.getString(R.string.UsuarioCreado))
    builder.setPositiveButton(this.getString(R.string.Aceptar), null)
    val dialog: AlertDialog = builder.create()
    dialog.show()
}

/** mensaje FALLO AL AUTENTIFICAR */
private fun showAlertEntriFailed() {
    val builder = AlertDialog.Builder(this)
    builder.setTitle(this.getString(R.string.TituloError))
    builder.setMessage(this.getString(R.string.ErrorEmailContraseña))
    builder.setPositiveButton(this.getString(R.string.Aceptar), null)
    val dialog: AlertDialog = builder.create()
    dialog.show()
}

/** mensaje FALLO el password no es correcto */
private fun showAlertPassword() {
    val builder = AlertDialog.Builder(this)
    builder.setTitle(this.getString(R.string.TituloError))
    builder.setMessage(this.getString(R.string.ErrorCaracteres))
    builder.setPositiveButton(this.getString(R.string.Aceptar), null)
    val dialog: AlertDialog = builder.create()
    dialog.show()
}

/** mensaje FALLO el email no es correcto */
private fun showAlertEmail() {
    val builder = AlertDialog.Builder(this)
    builder.setTitle(this.getString(R.string.TituloError))
    builder.setMessage(this.getString(R.string.ErrorFormatoGmail))
    builder.setPositiveButton(this.getString(R.string.Aceptar), null)
    val dialog: AlertDialog = builder.create()
    dialog.show()
}

/** mensaje FALLO si uno de los elementos esta vacio */
private fun showAlertNotEmty() {
    val builder = AlertDialog.Builder(this)
    builder.setTitle(this.getString(R.string.TituloError))
    builder.setMessage(this.getString(R.string.ErrorVacio))
}
```

```

        builder.setPositiveButton(this.getString(R.string.Aceptar), null)
        val dialog: AlertDialog = builder.create()
        dialog.show()
    }

    /** VALIDAR EL EMAIL */
    private fun validarEmail(texto: String): Boolean {
        val pattern = Pattern.compile("^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-\\+])*[A-Za-z0-9-]+(\\.[_A-Za-z0-9-\\+])*(\\.[A-Za-z]{2,})$")
        var comparator: Matcher = pattern.matcher(texto)
        return comparator.find()
    }

    /** CAMBIAR EL FRAGMENTO */
    fun changeActivity(){
        var intent = Intent(this, HomeActivity::class.java)
        startActivity(intent)
    }

    /** Cambio de la funcion volber atras */
    override fun onBackPressed() {
        finishAffinity()
    }

    /** Comprueba que tengamos el permiso de notificaciones activo */
    private fun checkPermissions() {
        if(ContextCompat.checkSelfPermission(this,
        Manifest.permission.POST_NOTIFICATIONS) !=
        PackageManager.PERMISSION_GRANTED){
            requestNotificationPermission()
        }
    }

    /** Pide los permisos necesarios */
    private fun requestNotificationPermission() {
        if(!ActivityCompat.shouldShowRequestPermissionRationale(this,
        Manifest.permission.POST_NOTIFICATIONS)){
            ActivityCompat.requestPermissions(this,
            arrayOf(Manifest.permission.POST_NOTIFICATIONS), 101)
        }
    }
}

```

10.1.2 HomeActivity

```

package com.adgonu.artnutria.ui

import android.content.Intent
import android.content.pm.ActivityInfo
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import androidx.navigation.NavController
import androidx.navigation.findNavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.AppBarConfiguration

```

```
import androidx.navigation.ui.NavigationUI
import com.adgonu.artnutria.R
import com.adgonu.artnutria.databinding.ActivityHomeBinding
import com.google.firebase.auth.FirebaseAuth

class HomeActivity : AppCompatActivity() {
    private lateinit var binding: ActivityHomeBinding
    private lateinit var navController: NavController

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(ActivityHomeBinding.inflate(layoutInflater).also {
binding = it }.root)

        setSupportActionBar(binding.toolbar)

        requestedOrientation = ActivityInfo.SCREEN_ORIENTATION_PORTRAIT

        val navHostFragment =
supportFragmentManager.findFragmentById(R.id.nav_host_fragment) as
NavHostFragment
        navController = navHostFragment.navController

        val appBarConfiguration = AppBarConfiguration(setOf(
            R.id.home,
            R.id.myUser,
            R.id.salir
        ))

        NavigationUI.setupWithNavController(binding.toolbar, navController,
appBarConfiguration)
    }

    // NAVIGATION
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.menu_graph, menu);
        return true;
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        return when (item.itemId) {
            R.id.home -> {
                navController.navigate(R.id.homeFragment)
                true
            }
            R.id.myUser -> {
                navController.navigate(R.id.myUserFragment)
                true
            }
            R.id.salir -> {
                FirebaseAuth.getInstance().signOut()

                var intent = Intent(this, MainActivity::class.java)
                startActivity(intent)
            }
        }
    }
}
```

```

        true
    }
    else -> super.onOptionsItemSelected(item)
}

}

override fun onBackPressed() {
    //super.onBackPressed()
    val fragmento = findNavController(R.id.nav_host_fragment)

    if (fragmento != null && fragmento.currentDestination?.id ==
R.id.homeFragment ) {

        finishAffinity()

    } else {

        super.onBackPressed()

    }

}

}

```

10.1.3 CarouselFragment

```

package com.adgonu.artnutria.ui.fragments

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.LifecycleScope
import androidx.navigation.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Adapter.ImageAdapter
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.viewmodel.FirebaseViewModel
import com.google.firebase.messaging.FirebaseMessaging
import com.squareup.picasso.Picasso
import de.hdodenhof.circleimageview.CircleImageView
import kotlinx.coroutines.launch

class CarouselFragment : Fragment() {

```

```

private lateinit var viewModel: FirebaseViewModel

private var obra: Obra? = null

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    viewModel =
        ViewModelProvider(this).get(FirebaseViewModel::class.java)
        arguments?.let {
            obra = it.getSerializable(context?.getString(R.string.obra)) as
            Obra
        }
}

override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_carousel, container,
false)
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    (requireActivity() as AppCompatActivity).supportActionBar?.show()

    var userImage: CircleImageView =
view.findViewById(R.id.ivCarUsuario)
    var nombreUser: TextView = view.findViewById(R.id.tvCarNombre)
    var nombreObra: TextView = view.findViewById(R.id.tvCarObra)
    var numFollows: TextView = view.findViewById(R.id.tvNumFollow)
    var numMedia: TextView = view.findViewById(R.id.tvNumeroImagenes)
    var botSeguir: Button = view.findViewById(R.id.bttCarSeguir)
    var followImage: CircleImageView =
view.findViewById(R.id.ivCarFollow)
    var carousel: RecyclerView = view.findViewById(R.id.carousel)
    var etiquetaUser: ConstraintLayout =
view.findViewById(R.id.clEtiqueraUsuario)

    lifecycleScope.launch{
        var user = viewModel.getUser(obra!!.uid, view.context)

        etiquetaUser.setOnClickListener {
            changeFragmentUser(user!!)
        }

        if(user!!.imagen != ""){
            Picasso.get().load(user!!.imagen).into(userImage)
        }
        nombreUser.text = user.nombre

        // Buscamos si se le a dado follow anterior mente
        var myUser = viewModel.getUser(viewModel.getIdUser(),
view.context)

```

```
        if(myUser!!.id == user.id || myUser.email ==
context?.getString(R.string.NotUser) ){
            botSeguir.isEnabled = false
            followImage.isEnabled = false
        }

        //Controlamos si ya le a dado follow
        var controllFollow = false

        for(obraUser in myUser!!.obrasLikkes){

            if(obra!!.id == obraUser){

Picasso.get().load(R.drawable.follow2).into(followImage)
                controllFollow = true
                break
            }

        }

        //al clicar cambiamos la imagen y actualizamos el follow de
Obra y el obrasFollow del usuario
        followImage.setOnClickListener {
            if(controllFollow == false) {

Picasso.get().load(R.drawable.follow2).into(followImage)
                obra!!.likkes += 1
                viewModel.updateObraFollow(obra!!.id, obra!!.likkes,
view.context)

                myUser.obrasLikkes.add(obra!!.id)
                viewModel.updateUserFollow(myUser.id,
myUser.obrasLikkes, view.context)
                numFollows.text =
context?.getString(R.string.FormLikkes) + obra!!.likkes.toString()
                controllFollow = true
                val bundle = Bundle()

bundle.putSerializable(context?.getString(R.string.obra), obra)
            }else{
                Picasso.get().load(R.drawable.follow).into(followImage)
                obra!!.likkes -= 1
                viewModel.updateObraFollow(obra!!.id, obra!!.likkes,
view.context)

                myUser.obrasLikkes.remove(obra!!.id)
                viewModel.updateUserFollow(myUser.id,
myUser.obrasLikkes, view.context)
                numFollows.text =
context?.getString(R.string.FormLikkes) + obra!!.likkes.toString()
                controllFollow = false
                val bundle = Bundle()

bundle.putSerializable(context?.getString(R.string.obra), obra)
            }
        }

        //Controlamos si ya se sigue al usuario
        var controllSeguir = false
```

```

        for(seguir in myUser.favoritos){
            if(seguir == obra!!.uid){
                botSeguir.text = context?.getString(R.string.NoSeguir)
                controllSeguir = true
                break
            }
        }

        //Cambiamos el boton dependiendo si lo seguimos o no
        botSeguir.setOnClickListener{
            if(controllSeguir == false) {
                botSeguir.text = context?.getString(R.string.NoSeguir)
                myUser.favoritos.add(obra!!.uid)

                FirebaseMessaging.getInstance().subscribeToTopic(obra!!.uid)
                viewModel.updateUserFavorite(myUser.id,
                    myUser.favoritos, view.context)
                controllSeguir = true
            }else{
                botSeguir.text = context?.getString(R.string.Seguir)
                myUser.favoritos.remove(obra!!.uid)

                FirebaseMessaging.getInstance().unsubscribeFromTopic(obra!!.uid)
                viewModel.updateUserFavorite(myUser.id,
                    myUser.favoritos, view.context)
                controllSeguir = false
            }
        }
    }

    nombreObra.text = obra!!.nombre
    numFollows.text = context?.getString(R.string.FormLikked) +
obra!!.likked
    numMedia.text = context?.getString(R.string.FormImagenes) +
obra!!.imagenes.size

    carousel.setHasFixedSize(true)
    carousel.visibility = View.VISIBLE
    carousel.layoutManager = LinearLayoutManager(requireContext(),
        LinearLayoutManager.HORIZONTAL, false)

    var imageAdapter = ImageAdapter(obra!!.imagenes)
    carousel.adapter = imageAdapter
}

fun changeFragmentUser(user: Usuario){
    val bundle = Bundle()
    bundle.putSerializable(context?.getString(R.string.user), user)

    lifecycleScope.launch {
        var myUser = viewModel.getIdUser()
        if(user.id == myUser){
            view?.findNavController()?.navigate(R.id.myUserFragment)
        }
    }
}

```

```

        }else{
            view?.findNavController()?.navigate(R.id.userFragment,
bundle)
        }
    }

}

}

}

```

10.1.4 FollowUserFragment

```

package com.adgonu.artnutria.ui.fragments

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ProgressBar
import androidx.lifecycle.LifecycleScope
import androidx.navigation.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Adapter.UserAdapter
import com.adgonu.artnutria.data.Interface.OnItemClickListenerUsuario
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.viewmodel.FirebaseViewModel
import kotlinx.coroutines.launch

class FollowUserFragment : Fragment(),.OnItemClickListenerUsuario {

    private lateinit var viewModel: FirebaseViewModel

    lateinit var rvUsuario: RecyclerView
    lateinit var userList: ArrayList<Usuario>
    lateinit var adapter: UserAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_follow_user, container,
false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val progressBar: ProgressBar =
view.findViewById(R.id.progressBarFollow)
        progressBar.visibility = View.VISIBLE

        viewModel = FirebaseViewModel()

```



```

        userList = arrayListOf<Usuario>()

        rvUsuario = view.findViewById(R.id.ryFollowUser)
        rvUsuario.setHasFixedSize(true)
        rvUsuario.visibility = View.VISIBLE
        rvUsuario.layoutManager = LinearLayoutManager(requireContext(),
        LinearLayoutManager.VERTICAL, false)

        lifecycleScope.launch{

            var myUsuario = viewModel.getUser(viewModel.getIdUser(),
            view.context)

            //Por cada uno de los elementos sacamos el usuario y lo
añadimos a la lista
            for(followUser in myUsuario!!.favoritos){
                var usuario = viewModel.getUser(followUser, view.context)
                userList.add(usuario!!)
            }

            adapter = UserAdapter(userList, this@FollowUserFragment)

            rvUsuario.adapter = adapter
            progressBar.visibility = View.GONE
        }

    }

    override fun onItemClick(user: Usuario) {
        val bundle = Bundle()
        bundle.putSerializable(context?.getString(R.string.user), user)

        view?.findNavController()?.navigate(R.id.userFragment, bundle)
    }
}

```

10.1.5 HomeFragment

```
package com.adgonu.artnutria.ui.fragments
```

```

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ProgressBar
import android.widget.SearchView
import androidx.lifecycle.LifecycleScope
import androidx.navigation.findNavController
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Adapter.ObraAdapter
import com.adgonu.artnutria.data.Interface.OnItemClickListenerObra

```

```
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.viewmodel.FirebaseViewModel
import kotlinx.coroutines.launch

class HomeFragment : Fragment(), OnItemClickListenerObra {

    private lateinit var viewModel: FirebaseViewModel

    lateinit var rvObra: RecyclerView
    lateinit var obraList: ArrayList<Obra>
    lateinit var adapter: ObraAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_home, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val progressBar: ProgressBar =
            view.findViewById(R.id.progressBarHome)
        progressBar.visibility = View.VISIBLE

        viewModel = FirebaseViewModel()

        obraList = arrayListOf<Obra>()

        rvObra = view.findViewById(R.id.rvObra)
        rvObra.setHasFixedSize(true)
        rvObra.visibility = View.VISIBLE
        rvObra.layoutManager = GridLayoutManager(requireContext(), 3)

        lifecycleScope.launch{
            obraList = viewModel.getAllObras(view.context) as
            ArrayList<Obra>
            adapter = ObraAdapter(obraList.sortedByDescending { it.fecha },
            this@HomeFragment)

            rvObra.adapter = adapter
            progressBar.visibility = View.GONE
        }

        setup(view)
        createUser()
    }

    private fun setup(view: View) {

        var btnObra: Button = view.findViewById(R.id.btnObra)
        var btnEtiqueta: Button = view.findViewById(R.id.btnEtiqueta)
        var btnUsuario: Button = view.findViewById(R.id.btnUsuario)

        var svObra: SearchView = view.findViewById(R.id.seObra)
```

```
var svEtiqueta: SearchView = view.findViewById(R.id.seEtiqueta)
var svUsuario: SearchView = view.findViewById(R.id.seUsuario)

// Dependiendo del boton que seleccionemos pondremos un buscador o
otro y desabilitaremos el boton seleccionado
btttnObra.setOnClickListener {
    svObra.visibility = View.VISIBLE
    svEtiqueta.visibility = View.GONE
    svUsuario.visibility = View.GONE
    btttnObra.isEnabled = false
    btttnEtiqueta.isEnabled = true
    btttnUsuario.isEnabled = true
}

btttnEtiqueta.setOnClickListener {
    svEtiqueta.visibility = View.VISIBLE
    svObra.visibility = View.GONE
    svUsuario.visibility = View.GONE
    btttnEtiqueta.isEnabled = false
    btttnObra.isEnabled = true
    btttnUsuario.isEnabled = true
}

btttnUsuario.setOnClickListener {
    svUsuario.visibility = View.VISIBLE
    svObra.visibility = View.GONE
    svEtiqueta.visibility = View.GONE
    btttnUsuario.isEnabled = false
    btttnObra.isEnabled = true
    btttnEtiqueta.isEnabled = true
}

// Búsqueda por Obra
svObra.setOnQueryTextListener(object :
SearchView.OnQueryTextListener{
    override fun onQueryTextSubmit(query: String): Boolean {
        if (query.isEmpty()){
            adapter.updateData(obraList)
        }else{
            filtrarNombre(query)
        }
        return false
    }

    override fun onQueryTextChange(newText: String): Boolean {
        if (newText.isEmpty()){
            adapter.updateData(obraList)
        }else{
            filtrarNombre(newText)
        }
        return false
    }
})

// Búsqueda por Etiquetas
svEtiqueta.setOnQueryTextListener(object :
SearchView.OnQueryTextListener{
```

```
        override fun onQueryTextSubmit(query: String): Boolean {
            if (query.isEmpty()){
                adapter.updateData(obraList)
            }else{
                filtrarEtiqueta(query)
            }
            return false
        }

        override fun onQueryTextChange(newText: String): Boolean {
            if (newText.isEmpty()){
                adapter.updateData(obraList)
            }else{
                filtrarEtiqueta(newText)
            }
            return false
        }
    })

    // Búsqueda por usuario
    svUsuario.setOnQueryTextListener(object :
    SearchView.OnQueryTextListener{
        override fun onQueryTextSubmit(query: String): Boolean {
            if (query.isEmpty()){
                adapter.updateData(obraList)
            }else{
                lifecycleScope.launch{
                    var user = viewModel.getIdUserName(query,
view.context) ?: null
                    if(user != null){
                        filtrarUsuario(user.id)
                    }
                }
            }
            return false
        }

        override fun onQueryTextChange(newText: String): Boolean {
            if (newText.isEmpty()){
                adapter.updateData(obraList)
            }else{
                lifecycleScope.launch{
                    var user = viewModel.getIdUserName(newText,
view.context) ?: null
                    if(user != null){
                        filtrarUsuario(user.id)
                    }
                }
            }
            return false
        }
    })
}
```

```
// Filtramos la búsqueda por nombre, si la obra de la lista tiene un
nombre parecido la añadira a la lista y actualizara la lista del
RecyclerView
private fun filtrarNombre(query: String) {

    var listaObrasFiltrada = mutableListOf<Obra>()

    for(obraData in obraList){

if(obraData.nombre.toLowerCase().contains(query.toLowerCase())){
        listaObrasFiltrada.add(obraData)
    }

    }

    adapter.updateData(listaObrasFiltrada)

}

// Filtramos la búsqueda por Etiquetas, si la obra de la lista, tiene
etiquetas que se parezcan a lo que se a escrito, la añadira a la lista y
actualizara la lista del RecyclerView
private fun filtrarEtiqueta(query: String) {

    var listaObrasFiltrada = mutableListOf<Obra>()

    for(obraData in obraList){
        for(etiqueta in obraData.etiquetas)
            if(etiqueta.contains(query.toLowerCase())){
                listaObrasFiltrada.add(obraData)
            }
    }

    adapter.updateData(listaObrasFiltrada)

}

// Filtramos la búsqueda por Usuario, si la obra de la lista, tiene un
usuario que se parezca a lo que se a escrito, la añadira a la lista y
actualizara la lista del RecyclerView
private fun filtrarUsuario(query: String) {

    var listaObrasFiltrada = mutableListOf<Obra>()

    for(obraData in obraList){
        if(obraData.uid.contains(query)){
            listaObrasFiltrada.add(obraData)
        }
    }

    adapter.updateData(listaObrasFiltrada)

}

//Si el usuario no existe, lo creamos
fun createUser() {
    viewModel = FirebaseViewModel()

    lifecycleScope.launch {
```

```

        var user = viewModel.getUser(viewModel.getIdUser(),
requireContext())

        //Comprobamos que el usuario exista
        if (user == null) {
            var usuario = Usuario(
                "",
                "",
                viewModel.getMyUserEmail(),
                "",
                "",
                "",
                ArrayList(),
                ArrayList()
            )
            viewModel.addUser(
                usuario.descripcion,
                usuario.email,
                usuario.imagen,
                usuario.nombre,
                usuario.idioma,
                usuario.favoritos,
                usuario.obrasLikked,
                requireContext()
            )
            view?.findNavController()?.navigate(R.id.myUserFragment)
        }
    }

    //Asignamos la funcion que tendra el clicar en una imagen
    override fun onItemClick(obra: Obra) {
        val bundle = Bundle()
        bundle.putSerializable(context?.getString(R.string.obra), obra)

        view?.findNavController()?.navigate(R.id.carouselFragment, bundle)
    }
}

```

10.1.6 LikkeObraFragment

```

package com.adgonu.artnutria.ui.fragments

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ProgressBar
import androidx.lifecycle.LifecycleScope
import androidx.navigation.findNavController
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Adapter.ObraAdapter
import com.adgonu.artnutria.data.Interface.OnItemClickListenerObra

```

```
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.viewmodel.FirebaseViewModel
import kotlinx.coroutines.launch

class LikkeObraFragment : Fragment(), OnItemClickListenerObra {

    private lateinit var viewModel: FirebaseViewModel

    lateinit var rvObra: RecyclerView
    lateinit var obraList: ArrayList<Obra>
    lateinit var adapter: ObraAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_likke_obra, container,
false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val progressBar: ProgressBar =
view.findViewById(R.id.progressBarLikke)
        progressBar.visibility = View.VISIBLE

        viewModel = FirebaseViewModel()

        obraList = arrayListOf<Obra>()

        rvObra = view.findViewById(R.id.rvFollowObra)
        rvObra.setHasFixedSize(true)
        rvObra.visibility = View.VISIBLE
        rvObra.layoutManager = GridLayoutManager(requireContext(), 3)

        lifecycleScope.launch{

            var myUser = viewModel.getUser(viewModel.getIdUser(),
view.context)!!
            for(idObra in myUser.obrasLikkes){
                var obra = viewModel.getObra(idObra, view.context)!!
                obraList.add(obra)
            }

            adapter = ObraAdapter(obraList.sortedByDescending { it.fecha },
this@LikkeObraFragment)

            rvObra.adapter = adapter
            progressBar.visibility = View.GONE
        }

    }

    override fun onItemClick(obra: Obra) {
```

```

        val bundle = Bundle()
        bundle.putSerializable(context?.getString(R.string.obra), obra)

        view?.findNavController()?.navigate(R.id.carouselFragment, bundle)
    }
}

```

10.1.7 MyUserFragment

```

package com.adgonu.artnutria.ui.fragments

import android.app.Activity.RESULT_OK
import android.content.Intent
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.ProgressBar
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.LifecycleScope
import androidx.navigation.findNavController
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Adapter.ObraAdapter
import com.adgonu.artnutria.data.Interface.OnItemClickListenerObra
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.viewmodel.FirebaseViewModel
import com.squareup.picasso.Picasso
import de.hdodenhof.circleimageview.CircleImageView
import kotlinx.coroutines.launch

class MyUserFragment : Fragment(),.OnItemClickListenerObra {

    private lateinit var viewModel: FirebaseViewModel
    private val GALLERY_INTENT = 1
    lateinit var obraList: ArrayList<Obra>
    lateinit var adapter: ObraAdapter
    lateinit var rvMyObra: RecyclerView

    var imagen_url: String = ""

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        viewModel =
        ViewModelProvider(this).get(FirebaseViewModel::class.java)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,

```



```

        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_my_user, container,
false)

    }

    override fun onCreateView(view: View, savedInstanceState: Bundle?) {
        super.onCreateView(view, savedInstanceState)

        val progressBar: ProgressBar = view.findViewById(R.id.progressBar)
        progressBar.visibility = View.VISIBLE

        subirImagenUsuario(view)
        subirObra(view)
        setup(view)
        followUser(view)
        guardarUsuario(view)
        followObra(view)

        obraList = arrayListOf<Obra>()

        rvMyObra = view.findViewById(R.id.rvMyObra)
        rvMyObra.setHasFixedSize(true)
        rvMyObra.visibility = View.VISIBLE
        rvMyObra.layoutManager = GridLayoutManager(requireContext(), 3)

        lifecycleScope.launch{
            obraList = viewModel.getObrasUser(viewModel.getIdUser(),
view.context) as ArrayList<Obra>

            if(obraList.size != 0){
                adapter = ObraAdapter(obraList.sortedByDescending {
it.fecha }, this@MyUserFragment)

                rvMyObra.adapter = adapter
            }

        }

        progressBar.visibility = View.GONE

    }

    // Funcion que te dirige a la lista de obras a las que as dado follow
    private fun followObra(view: View) {
        val btFollowObra: Button = view.findViewById(R.id.bttFollow)
        btFollowObra.setOnClickListener {
view?.findNavController()?.navigate(R.id.followObraFragment) }
    }

    // Funcion que guarda los datos de los usuarios
    private fun guardarUsuario(view: View) {
        val btGuardar: Button = view.findViewById(R.id.bttCambios)
        btGuardar.setOnClickListener {
            lifecycleScope.launch{

```

```

        var usuario_nuevo =
viewModel.getUser(viewModel.getIdUser(), view.context)
        crearUsuario(view, usuario_nuevo!!)
    }
}

// Funcion que crea el usuario
fun crearUsuario(view: View, user: Usuario){

    val nombre: EditText = view.findViewById(R.id.edNombre)
    val email: EditText = view.findViewById(R.id.edCorreo)
    val idioma: EditText = view.findViewById(R.id.edIdioma)
    val descripcion: EditText = view.findViewById(R.id.edDescripcion)

    viewModel.addUser(descripcion.text.toString(),
email.text.toString(), user.imagen, nombre.text.toString(),
idioma.text.toString(), ArrayList<String>(), ArrayList<String>(),
requireContext())
    view?.findNavController()?.navigate(R.id.homeFragment)

}

// Funcion que modifica la imagen del usuario, abre la galeria y puedes
cambiar la imagen
fun subirImagenUsuario(view: View){
    val imagen: ImageView = view.findViewById(R.id.ivUser)
    imagen.setOnClickListener{
        var intent: Intent = Intent(Intent.ACTION_GET_CONTENT)
        intent.type = "*/*"
        startActivityForResult(intent, GALLERY_INTENT)
    }
}

// Funcion que te dirige a crear las obras
fun subirObra(view: View){
    val btnSubir: Button = view.findViewById(R.id.btnSubir)
    btnSubir.setOnClickListener {
view?.findNavController()?.navigate(R.id.obraFragment) }
    }

// Funcion que te dirige a la lista de los usuarios seguidos
fun followUser(view: View){
    val btnSeguir: Button = view.findViewById(R.id.bttSiguiendo)
    btnSeguir.setOnClickListener {
view?.findNavController()?.navigate(R.id.followUserFragment) }
    }

fun setup(view: View){
    viewModel = FirebaseViewModel()

    val email: EditText = view.findViewById(R.id.edCorreo)
    val btGuardar: Button = view.findViewById(R.id.bttCambios)
    val btSubir: Button = view.findViewById(R.id.btnSubir)
    val btSiguiendo: Button = view.findViewById(R.id.bttSiguiendo)
    val btFollow: Button = view.findViewById(R.id.bttFollow)

    lifecycleScope.launch{

```

```

        var user = viewModel.getUser(viewModel.getIdUser(),
view.context)!!

        val nombre: EditText = view.findViewById(R.id.edNombre)
        val email: EditText = view.findViewById(R.id.edCorreo)
        val idioma: EditText = view.findViewById(R.id.edIdioma)
        val descripcion: EditText =
view.findViewById(R.id.edDescripcion)
        val imagen: CircleImageView = view.findViewById(R.id.ivUser)

        //Si el usuario es "notuser@gmail.com" no permitiremos cambiar
nada
        if(user.email == context?.getString(R.string.NotUser)){
            nombre.isEnabled = false
            email.isEnabled = false
            idioma.isEnabled = false
            descripcion.isEnabled = false
            imagen.isEnabled = false
            btGuardar.isEnabled = false
            btSubir.isEnabled = false
            btSiguiendo.isEnabled = false
            btFollow.isEnabled = false
        }

        //Recogemos los datos del usuario
        nombre.setText(user.nombre)
        email.setText(user.email)
        idioma.setText(user.idioma)
        descripcion.setText(user.descripcion)
        if(user.imagen != ""){
            Picasso.get().load(user.imagen).into(imagen)
        }else{
            Picasso.get().load(R.drawable.imagen_usuario).into(imagen)
        }
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if(requestCode == GALLERY_INTENT && resultCode == RESULT_OK){
            var imagen_uri = data?.data!!
            imagen_url = imagen_uri.toString()
            viewModel.addImage(imagen_uri!!, viewModel.getIdUser(),
requireContext())
        }
    }

    override fun onItemClick(obra: Obra) {
        val bundle = Bundle()
        bundle.putSerializable("obra", obra)

        view?.findNavController()?.navigate(R.id.carouselFragment, bundle)
    }
}

```

10.1.8 ObraFragment

```
package com.adgonu.artnutria.ui.fragments

import android.app.Activity
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.ProgressBar
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProvider
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Adapter.ObraImageAdapter
import com.adgonu.artnutria.viewmodel.FirebaseViewModel

class ObraFragment : Fragment() {

    private lateinit var viewModel: FirebaseViewModel
    private val GALLERY_INTENT = 1
    var imagenes: ArrayList<Uri> = ArrayList()
    lateinit var rvObraImagen: RecyclerView
    lateinit var adapter: ObraImageAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        viewModel =
            ViewModelProvider(this).get(FirebaseViewModel::class.java)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_obra, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        viewModel = FirebaseViewModel()

        rvObraImagen = view.findViewById(R.id.rvObraImagen)
        rvObraImagen.setHasFixedSize(true)
        rvObraImagen.visibility = View.VISIBLE
        rvObraImagen.layoutManager = GridLayoutManager(requireContext(), 3)

        adapter = ObraImageAdapter(imagenes)
        rvObraImagen.adapter = adapter

        subirImagenObra(view)
    }
}
```

```

        subirObra(view)
    }

    fun subirObra(view: View){
        var btnSubirObra: Button = view.findViewById(R.id.btnSubirObra)

        btnSubirObra.setOnClickListener {
            var etiquetas = ArrayList<String>()

            val progressBar: ProgressBar =
view.findViewById(R.id.progressBarObra)
            progressBar.visibility = View.VISIBLE

            val uid: String = viewModel.getIdUser()
            val nombre: EditText =
requireView().findViewById<EditText?>(R.id.edtNombreObra)
            val etiqueta: EditText =
requireView().findViewById(R.id.edtEtiquetasObra)
            val imagen = imagenes

            var etiquetasEnviar = ArrayList<String>()

            if(etiqueta.text.toString().contains(",")) {
                etiquetasEnviar =
etiqueta.text.toString().lowercase().split(",") as ArrayList<String>
            }else{
                etiquetasEnviar.add(etiqueta.text.toString().lowercase())
            }

            viewModel.addObra(uid, nombre.text.toString(), imagen,
etiquetasEnviar, requireContext())
            progressBar.visibility = View.GONE

        }
    }

    // Funcion que agrega las imagenes de la obra, abre la galeria y puedes
    añadir las imagenes
    fun subirImagenObra(view: View){
        val btnSelecImage: Button =
view.findViewById(R.id.btnSeleccionarImagen)
        btnSelecImage.setOnClickListener{

            var intent: Intent = Intent(Intent.ACTION_GET_CONTENT)
            intent.type = "*/*"
            startActivityForResult(intent, GALLERY_INTENT)

        }
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if(requestCode == GALLERY_INTENT && resultCode ==
Activity.RESULT_OK){
            var image_url = data?.data
            imagenes.add(image_url!!)
            adapter = ObraImageAdapter(imagenes)

```

```

        rvObraImagen.adapter = adapter
    }
}
}

```

10.1.9 UserFragment

```

package com.adgonu.artnutria.ui.fragments

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.lifecycleScope
import androidx.navigation.findNavController
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Adapter.ObraAdapter
import com.adgonu.artnutria.data.Interface.OnItemClickListenerObra
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.viewmodel.FirebaseViewModel
import com.google.firebase.messaging.FirebaseMessaging
import com.squareup.picasso.Picasso
import de.hdodenhof.circleimageview.CircleImageView
import kotlinx.coroutines.launch

class UserFragment : Fragment(),.OnItemClickListenerObra {

    private lateinit var viewModel: FirebaseViewModel
    private var usuario: Usuario? = null

    lateinit var obraList: ArrayList<Obra>
    lateinit var adapter: ObraAdapter
    lateinit var rvMyObra: RecyclerView
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        viewModel =
            ViewModelProvider(this).get(FirebaseViewModel::class.java)
        arguments?.let {
            usuario = it.getSerializable(context?.getString(R.string.user))
            as Usuario
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ) {

```

```

): View? {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_user, container, false)
}

override fun onCreateView(view: View, savedInstanceState: Bundle?) {
    super.onCreateView(view, savedInstanceState)

    (requireActivity() as AppCompatActivity).supportActionBar?.show()

    updateContent(view)
    setup(view)
    changeButtonSeguir(view)
}

private fun setup(view: View) {
    var nombre: TextView = view.findViewById(R.id.txNombre)
    var email: TextView = view.findViewById(R.id.txEmail)
    var idioma: TextView = view.findViewById(R.id.txIdioma)
    var descripcion: TextView = view.findViewById(R.id.txDescripcion)
    var image: CircleImageView = view.findViewById(R.id.ivUser)

    nombre.text = usuario!!.nombre
    email.text = usuario!!.email
    idioma.text = usuario!!.idioma
    descripcion.text = usuario!!.descripcion
    Picasso.get().load(usuario!!.imagen).into(image)
}

private fun updateContent(view: View) {

    obraList = arrayListOf<Obra>()

    rvMyObra = view.findViewById(R.id.rvObrasUser)
    rvMyObra.setHasFixedSize(true)
    rvMyObra.visibility = View.VISIBLE
    rvMyObra.layoutManager = GridLayoutManager(requireContext(), 3)

    lifecycleScope.launch{
        obraList = viewModel.getObrasUser(usuario!!.id, view.context)
    } as ArrayList<Obra>

    if(obraList.size != 0){
        adapter = ObraAdapter(obraList.sortedByDescending {
            it.fecha }, this@UserFragment)

        rvMyObra.adapter = adapter
    }
}

// Funcion boton seguir, añade el usuario a los seguidos y modifica el
// boton ( si ya se sigue, lo deja de seguir y modifica el boton )
private fun changeButtonSeguir(view: View) {

```

```

var btSeguir: Button = view.findViewById(R.id.btnSeguirUser)

var change = lifecycleScope.launch{

    var myUserID = viewModel.getIdUser()
    var myUser = viewModel.getUser(myUserID, view.context)

    var controllSeguir = false

    if(myUser!!.id == usuario?.id || myUser.email ==
context?.getString(R.string.NotUser) ){
        btSeguir.isEnabled = false
    }

    for(seguir in myUser!!.favoritos){
        if(seguir == usuario!!.id){
            btSeguir.text = context?.getString(R.string.NoSeguir)
            controllSeguir = true
            break
        }
    }

    //Cambiamos el boton dependiendo si lo seguimos o no
    btSeguir.setOnClickListener{
        if(controllSeguir == false) {
            btSeguir.text = context?.getString(R.string.NoSeguir)
            myUser.favoritos.add(usuario!!.id)

FirebaseMessaging.getInstance().subscribeToTopic(usuario!!.id)
            viewModel.updateUserFavorite(myUser.id,
myUser.favoritos, view.context)
            controllSeguir = true
        }else{
            btSeguir.text = context?.getString(R.string.Seguir)
            myUser.favoritos.remove(usuario!!.id)

FirebaseMessaging.getInstance().unsubscribeFromTopic(usuario!!.id)
            viewModel.updateUserFavorite(myUser.id,
myUser.favoritos, view.context)
            controllSeguir = false
        }
    }

}

}

fun setArguments(user: Usuario){
    arguments = Bundle().apply {
        putSerializable(context?.getString(R.string.user), user)
    }
}

override fun onItemClick(obra: Obra) {
    val bundle = Bundle()
    bundle.putSerializable(context?.getString(R.string.obra), obra)

```



```

        view?.findNavController()?.navigate(R.id.carouselFragment, bundle)
    }
}

```

10.1.10 FirebaseViewModel

```

package com.adgonu.artnutria.viewmodel

import android.content.Context
import android.net.Uri
import androidx.lifecycle.ViewModel
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.domain.FirestoreUseCase

class FirebaseViewModel: ViewModel() {

    val firestoreUseCase = FirestoreUseCase()

    fun addUser(descripcion: String, email: String, imagen: String, nombre:
String, idioma: String, favoritos: ArrayList<String>, obrasFollow:
ArrayList<String>, contexto: Context){

        firestoreUseCase.addUserFirestore(descripcion,email,imagen,nombre,idioma,fa
voritos, obrasFollow, contexto)
    }

    fun addImage(imagen_url: Uri, uid: String, contexto: Context){
        firestoreUseCase.addImage(imagen_url,uid,contexto)
    }

    fun getIdUser(): String{
        val uid = firestoreUseCase.getIdUser()
        return uid
    }

    suspend fun getUser(id: String, contexto: Context): Usuario? {
        val user = firestoreUseCase.getUser(id, contexto)
        return user
    }

    suspend fun getIdUserName(name: String, contexto: Context): Usuario?{
        val user = firestoreUseCase.getIdUserName(name, contexto)
        return user
    }

    fun getMyUserEmail(): String {
        val email = firestoreUseCase.getMyUserEmail()
        return email
    }

    fun updateUserFollow(id: String, dato: ArrayList<String>,contexto:
Context){
        firestoreUseCase.updateUserFollow(id, dato, contexto)
    }
}

```

```

        fun updateUserFavorite(id: String, dato: ArrayList<String>, contexto: Context){
            firestoreUseCase.updateUserFavorite(id, dato, contexto)
        }

        fun addObra(uid: String, nombre: String, imagenes: ArrayList<Uri>, etiquetas: ArrayList<String>, contexto: Context){
            val obra = firestoreUseCase.addObra(uid, nombre, imagenes, etiquetas, contexto)
            return obra
        }

        suspend fun getAllObras(contexto: Context): MutableList<Obra> {
            val listaObra = firestoreUseCase.getAllObras(contexto)
            return listaObra
        }

        suspend fun getObra(id: String, contexto: Context): Obra? {
            val obra = firestoreUseCase.getObra(id, contexto)
            return obra
        }

        suspend fun getObrasUser(uid: String, contexto: Context): MutableList<Obra>{
            val listaObra = firestoreUseCase.getObrasUser(uid, contexto)
            return listaObra
        }

        fun updateObraFollow(id: String, dato: Int, contexto: Context){
            firestoreUseCase.updateObraFollow(id, dato, contexto)
        }
    }
}

```

10.1.11 ImageViewHolder

```

package com.adgonu.artnutria.viewmodel

import android.view.View
import android.widget.MediaController
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.databinding.ImageCarouselItemBinding
import com.squareup.picasso.Picasso

class ImageViewHolder (view: View) : RecyclerView.ViewHolder(view) {
    private val binding = ImageCarouselItemBinding.bind(view)
    private val context = view.context

    fun bind (image: String){

        var extension = image.substringAfterLast("_", "")
        extension = extension.substring( 0, 5)

        if(extension == context.getString(R.string.image)){
            binding.imageCarObra.visibility = View.VISIBLE
            binding.videoCarObra.visibility = View.GONE
        }
    }
}

```

```

        Picasso.get().load(image).into(binding.imageCarObra)
    }
    else if(extension == context.getString(R.string.video)){
        val mediaController = MediaController(itemView.context)
        binding.imageCarObra.visibility = View.GONE
        binding.videoCarObra.visibility = View.VISIBLE
        mediaController.setAnchorView(binding.videoCarObra)
        binding.videoCarObra.setMediaController(mediaController)
        binding.videoCarObra.setVideoPath(image)
    }
}
}

```

10.1.12 ObraImageViewHolder

```

package com.adgonu.artnutria.viewmodel

import android.net.Uri
import android.view.View
import android.widget.MediaController
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.databinding.ImageItemBinding
import com.squareup.picasso.Picasso

class ObraImageViewHolder(view: View) : RecyclerView.ViewHolder(view) {
    private val binding = ImageItemBinding.bind(view)
    private var context = view.context

    fun bind(image: Uri){
        var contextResolve = context.contentResolver
        var extension = contextResolve.getType(image)!!.split("/")

        if(extension[0] == context.getString(R.string.image)){
            binding.imageObra.visibility = View.VISIBLE
            binding.videoObra.visibility = View.GONE
            Picasso.get().load(image).into(binding.imageObra)
        }
        else if(extension[0] == context.getString(R.string.video)){
            val mediaController = MediaController(itemView.context)
            binding.imageObra.visibility = View.GONE
            binding.videoObra.visibility = View.VISIBLE
            mediaController.setAnchorView(binding.videoObra)
            binding.videoObra.setMediaController(mediaController)
            binding.videoObra.setVideoURI(image)
        }
    }
}

```

10.1.13 ObraViewHolder

```
package com.adgonu.artnutria.viewmodel

import android.view.View
import android.widget.MediaController
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.databinding.ImageItemBinding
import com.squareup.picasso.Picasso

class ObraViewHolder(view: View): RecyclerView.ViewHolder(view) {
    private val binding = ImageItemBinding.bind(view)
    private var context = view.context

    fun bind (obra: Obra){

        var extension = obra.imagenes[0].substringAfterLast("_", "")
        extension = extension.substring( 0, 5)

        if(extension == context.getString(R.string.image)){
            binding.imageObra.visibility = View.VISIBLE
            binding.videoObra.visibility = View.GONE
            Picasso.get().load(obra.imagenes[0]).into(binding.imageObra)
        }
        else if(extension == context.getString(R.string.video)){
            val mediaController = MediaController(itemView.context)
            binding.imageObra.visibility = View.GONE
            binding.videoObra.visibility = View.VISIBLE
            mediaController.setAnchorView(binding.videoObra)
            binding.videoObra.setMediaController(mediaController)
            binding.videoObra.setVideoPath(obra.imagenes[0])
        }

    }
}
```

10.1.14 UserViewHolder

```
package com.adgonu.artnutria.viewmodel

import android.view.View
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.databinding.UsuarioItemBinding
import com.squareup.picasso.Picasso

class UserViewHolder(view: View): RecyclerView.ViewHolder(view) {
    private val binding = UsuarioItemBinding.bind(view)

    fun bind (usuario: Usuario){

        Picasso.get().load(usuario.imagen).into(binding.ivuserus)
        binding.tvnameus.text = usuario.nombre
        binding.tvemailus.text = usuario.email
    }
}
```

```

    }
}

```

10.1.15 FirebaseUserCase

```

package com.adgonu.artnutria.domain

import android.content.Context
import android.net.Uri
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.data.repo.FirebaseRepo

class FirestoreUseCase {
    val repo = FirebaseRepo()

    fun addUserFirestore(descripcion: String, email: String, imagen:
String, nombre: String, idioma: String, favoritos: ArrayList<String>,
obrasFollow: ArrayList<String>, contexto: Context){
        repo.addUser(descripcion,email,imagen,nombre, idioma, favoritos,
obrasFollow, contexto)
    }

    fun addImage(imagen_url: Uri, uid: String, contexto: Context){
        repo.addImage(imagen_url,uid,contexto)
    }

    fun getIdUser(): String{
        val uid = repo.getIdUser()
        return uid
    }

    suspend fun getUser(id: String, contexto: Context): Usuario? {
        val user = repo.getUser(id, contexto)
        return user
    }

    fun getMyUserEmail(): String {
        val email = repo.getMyUserEmail()
        return email
    }

    suspend fun getIdUserName(name: String, contexto: Context): Usuario?{
        val user = repo.getIdUserName(name, contexto)
        return user
    }

    fun updateUserFollow(id: String, dato: ArrayList<String>, contexto:
Context){
        repo.updateUserFollow(id, dato, contexto)
    }

    fun updateUserFavorite(id: String, dato: ArrayList<String>, contexto:
Context){
        repo.updateUserFavorite(id, dato, contexto)
    }
}

```

```

    fun addObra(uid: String, nombre: String, imagenes: ArrayList<Uri>,
etiquetas: ArrayList<String>, contexto: Context){
        val obra = repo.addObra(uid, nombre, imagenes, etiquetas, contexto)
        return obra
    }

suspend fun getAllObras(contexto: Context): MutableList<Obra> {
    val listaObra = repo.getAllObras(contexto)
    return listaObra
}

suspend fun getObra(id: String, contexto: Context): Obra? {
    val obra = repo.getObra(id, contexto)
    return obra
}

suspend fun getObrasUser(uid: String, contexto: Context):
MutableList<Obra>{
    val listaObra = repo.getObrasUser(uid, contexto)
    return listaObra
}

fun updateObraFollow(id: String, dato: Int, contexto: Context){
    repo.updateObraFollow(id, dato, contexto)
}

}

```

10.1.16 FirebaseRepo

```

package com.adgonu.artnutria.data.repo

import android.content.Context
import android.net.Uri
import android.widget.Toast
import androidx.navigation.findNavController
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.data.modelo.Usuario
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.storage.FirebaseStorage
import kotlinx.coroutines.*
import kotlinx.coroutines.tasks.await

class FirebaseRepo {
    //Base de datos de Firebase
    val db = FirebaseFirestore.getInstance()
    val aut = FirebaseAuth.getInstance()
    val ds = FirebaseStorage.getInstance()

    /** USUARIO */
    var user = Usuario("", "", "", "", "", "", ArrayList<String>(),
ArrayList<String>())

```

```

    //Si no existen un Usuario igual, se creara el Usuario con los datos,
    si existe, modifica los datos del Usuario
    fun addUser(descripcion: String, email: String, imagen: String, nombre:
String, idioma: String, favoritos: ArrayList<String>, obrasFollow:
ArrayList<String>, contexto: Context){
        var id = getIdUser()
        user.id = id
        if(descripcion.isNotEmpty()){user.descripcion = descripcion}
        if(email.isNotEmpty()){user.email = email}
        if(imagen.isNotEmpty()){user.imagen = imagen}
        if(nombre.isNotEmpty()){user.nombre = nombre}
        if(idioma.isNotEmpty()){user.idioma = idioma}
        if(favoritos.isNotEmpty()){user.favoritos = favoritos}
        if(obrasFollow.isNotEmpty()){user.obrasLikkes = obrasFollow}

        db.collection(contexto.getString(R.string.pathUsers)).document(id).set(user
).addOnCompleteListener {
            if (it.isSuccessful){
                Toast.makeText(contexto,
contexto.getString(R.string.guardado), Toast.LENGTH_SHORT).show()
            }else{
                Toast.makeText(contexto,
contexto.getString(R.string.noGuardado), Toast.LENGTH_SHORT).show()
            }
        }
    }

    fun getMyUserEmail(): String {
        var usuario = aut.currentUser
        var email = usuario!!.email
        return email!!
    }

    //Optenemos la id de nuestro usuario
    fun getIdUser(): String {
        var id = aut.currentUser?.uid.toString()
        return id
    }

    //Optenemos el id de el usuario a traves de su nombre
    suspend fun getIdUserName(name: String, contexto: Context): Usuario?{

        val consulta =
        db.collection(contexto.getString(R.string.pathUsers)).whereEqualTo(contexto
.getString(R.string.nombre), name)
        return try {
            val consultaSnap = consulta.get().await()
            if(!consultaSnap.isEmpty){
                val docuConsulta = consultaSnap.documents[0]
                docuConsulta.toObject(Usuario::class.java)
            }else{
                null
            }
        } catch (e: Exception){
            null
        }
    }

```

```

    }

    //Sacamos la informacion del usuario con el id correspondiente
    suspend fun getUser(id: String, contexto: Context): Usuario? =
    withContext(Dispatchers.IO) {

        val result =
        db.collection(contexto.getString(R.string.pathUsers)).document(id).get().await()

        if(result.exists()){
            val user = result.toObject(Usuario::class.java)
            user
        }else{
            null
        }

    }

    //Añadir la imagen del usuario
    fun addImage(imagen_url: Uri, uid: String, contexto: Context){
        val reference =
        ds.reference.child(contexto.getString(R.string.userImage)).child(uid +
        contexto.getString(R.string.coletillaUserImage) )
        reference.putFile(imagen_url!!).addOnSuccessListener {
            reference.downloadUrl.addOnSuccessListener { uri ->

        db.collection(contexto.getString(R.string.pathUsers)).document(uid).update(
        contexto.getString(R.string.imagen), uri.toString()).addOnCompleteListener
        {

            if (it.isSuccessful){
                Toast.makeText(contexto,
                contexto.getString(R.string.imagenGuardado), Toast.LENGTH_SHORT).show()
            }else{
                Toast.makeText(contexto,
                contexto.getString(R.string.imagenNoGuardado), Toast.LENGTH_SHORT).show()
            }
        }
        }
    }

    }

    //Modificar las obras a las que as dado follow
    fun updateUserFollow(id: String, dato: ArrayList<String>, contexto:
    Context){

        db.collection(contexto.getString(R.string.pathUsers)).document(id).update(c
        ontexto.getString(R.string.obrasLikkes), dato)

    }

    //Modificar los usuarios seguidos
    fun updateUserFavorite(id: String, dato: ArrayList<String>, contexto:
    Context){

```



```

db.collection(contexto.getString(R.string.pathUsers)).document(id).update(c
ontexto.getString(R.string.favoritos), dato)

    }

    /** OBRA **/

    var obra = Obra("", "", "", ArrayList<String>(), ArrayList<String>(),
    "", 0)

    //Añadir Obra
    fun addObra(uid: String, nombre: String, imagenes: ArrayList<Uri>,
    etiquetas: ArrayList<String>, contexto: Context){

        var fecha = System.currentTimeMillis().toString()
        var id = uid + nombre
        var arrayListObras = ArrayList<String>()

        db.collection(contexto.getString(R.string.pathObras)).document(id).get().ad
        dOnSuccessListener { result ->

            if (result.exists()){

                var ejecucion = GlobalScope.launch(Dispatchers.IO) {
                    addImageObra(imagenes, id, contexto)
                }

            }else{

                if(id.isNotEmpty()){obra.id = id}
                if(uid.isNotEmpty()){obra.uid = uid}
                if(fecha.isNotEmpty()){obra.fecha = fecha}
                if(nombre.isNotEmpty()){obra.nombre = nombre}
                if(etiquetas.isNotEmpty()){obra.etiquetas = etiquetas}

                db.collection(contexto.getString(R.string.pathObras)).document(id).set(obra
                ).addOnCompleteListener {
                    if (it.isSuccessful){
                        Toast.makeText(contexto,
                        contexto.getString(R.string.obraSubida), Toast.LENGTH_SHORT).show()
                    }else{
                        Toast.makeText(contexto,
                        contexto.getString(R.string.obraNoSubida), Toast.LENGTH_SHORT).show()
                    }
                }

                var ejecucion = GlobalScope.launch(Dispatchers.IO) {
                    addImageObra(imagenes, id, contexto)
                }

            }

        }
    }

```

```

    }

    //Cuando subo la imagen, hay un problema si intentas subir varias a la
    vez
    suspend fun addImageObra(imagens_url: ArrayList<Uri>, id: String,
    contexto: Context){
        //Añadimos la imagen
        var urls = mutableListOf<String>()
        var num_image = 0
        var contextResolve = contexto.contentResolver

        var obra = getObra(id, contexto)

        if(obra!!.imagenes.isNotEmpty()){
            urls = obra!!.imagenes
        }

        for(image in imagens_url){

            num_image += 1
            var id_obra = id + "_" + num_image

            val reference =
            ds.reference.child(contexto.getString(R.string.artstorage)).child(id_obra +
            contexto.getString(R.string.muletillaObra) + contextResolve.getType(image))
            reference.putFile(image).await()
            val url = reference.downloadUrl.await().toString()
            urls.add(url)
        }

        db.collection(contexto.getString(R.string.pathObras)).document(id).update(c
        ontexto.getString(R.string.imagenes), urls)

    }

    //Optener Obra por id
    suspend fun getObra(id: String, contexto: Context): Obra? =
    withContext(Dispatchers.IO){
        val result =
        db.collection(contexto.getString(R.string.pathObras)).document(id).get().aw
        ait()
        if(result.exists()){
            val obra = result.toObject(Obra::class.java)
            obra
        }else{
            null
        }
    }

    //Optener todas las obras
    suspend fun getAllObras(contexto: Context): MutableList<Obra> =
    withContext(Dispatchers.IO){
        var listaObra = mutableListOf<Obra>()

        db.collection(contexto.getString(R.string.pathObras)).get().await().forEach
        {resultado ->

```

```

        val obra = resultado.toObject(Obra::class.java)
        listaObra.add(obra)
    }
    listaObra
}

//Optener las obras de un usuario
suspend fun getObrasUser(uid: String, contexto: Context):
MutableList<Obra> = withContext(Dispatchers.IO) {
    var listaObra = mutableListOf<Obra>()

    db.collection(contexto.getString(R.string.pathObras)).whereEqualTo(contexto
.getString(R.string.uid), uid).get().await().forEach { resultado ->
        val obra = resultado.toObject(Obra::class.java)
        listaObra.add(obra)
    }
    listaObra
}

//Updatear la obra de un usuario
fun updateObraFollow(id: String, dato: Int, contexto: Context) {

    db.collection(contexto.getString(R.string.pathObras)).document(id).update(c
ontexto.getString(R.string.likkes), dato)

}
}

```

10.1.17 MyFirebaseMessagingService

```

package com.adgonu.artnutria.data.notification

import android.os.Handler
import android.os.Looper
import android.widget.Toast
import com.google.firebase.messaging.FirebaseMessagingService
import com.google.firebase.messaging.RemoteMessage

class MyFirebaseMessagingService: FirebaseMessagingService() {

    //Manejamos las notificaciones cuando la aplicación en primer plano
    override fun onMessageReceived(message: RemoteMessage) {
        Looper.prepare()

        Handler().post {
            Toast.makeText(baseContext, message.notification?.title,
            Toast.LENGTH_LONG).show()
        }
        Looper.loop()

    }

}

```

10.1.18 Usuario

```
package com.adgonu.artnutria.data.modelo

/** Data class de los usuarios */
data class Usuario(
    var id: String = "",
    var descripcion: String = "",
    var email: String = "",
    var imagen: String = "",
    var nombre: String = "",
    var idioma: String = "",
    var favoritos: ArrayList<String> = ArrayList(),
    var obrasLikkes: ArrayList<String> = ArrayList()
): java.io.Serializable
```

10.1.19 Obra

```
package com.adgonu.artnutria.data.modelo

data class Obra(
    var uid: String = "",
    var id: String = "",
    var nombre: String = "",
    var imagenes: ArrayList<String> = ArrayList(),
    var etiquetas: ArrayList<String> = ArrayList(),
    var fecha: String = "",
    var likkes: Int = 0
): java.io.Serializable
```

10.1.20 OnItemClickListenerObra

```
package com.adgonu.artnutria.data.Interface

import com.adgonu.artnutria.data.modelo.Obra

interface OnItemClickListenerObra {
    fun onItemClick(obra: Obra)
}
```

10.1.21 OnItemClickListenerUsuario

```
package com.adgonu.artnutria.data.Interface

import com.adgonu.artnutria.data.modelo.Usuario

interface OnItemClickListenerUsuario {
    fun onItemClick(user: Usuario)
}
```

10.1.22 ImageAdapter

```
package com.adgonu.artnutria.data.Adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.viewmodel.ImageViewHolder

class ImageAdapter(private var imagenList: List<String>):
    RecyclerView.Adapter<ImageViewHolder>() {
        override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
            ImageViewHolder {
                val inflater = LayoutInflater.from(parent.context)
                return
                    ImageViewHolder(inflater.inflate(R.layout.image_carusel_item, parent,
                    false))
            }

        override fun onBindViewHolder(holder: ImageViewHolder, position: Int) {
            val item = imagenList[position]
            holder.bind(item)
        }

        override fun getItemCount(): Int = imagenList.size

        fun updateData(dataList: List<String>) {
            this.imagenList = dataList
            notifyDataSetChanged()
        }
    }
```

10.1.23 ObraAdapter

```
package com.adgonu.artnutria.data.Adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Interface.OnItemClickListenerObra
import com.adgonu.artnutria.data.modelo.Obra
import com.adgonu.artnutria.viewmodel.ObraViewHolder

class ObraAdapter(private var obraList: List<Obra>, private val listener:
    OnItemClickListenerObra): RecyclerView.Adapter<ObraViewHolder>() {
        override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
            ObraViewHolder {
                val inflater = LayoutInflater.from(parent.context)
                return ObraViewHolder(inflater.inflate(R.layout.image_item,
                    parent, false))
            }

        override fun onBindViewHolder(holder: ObraViewHolder, position: Int) {
            val item = obraList[position]
            holder.bind(item)
        }
    }
```

```

        holder.itemView.setOnClickListener {
            listener.onItemClick(item)
        }
    }

    override fun getItemCount(): Int = obraList.size

    fun updateData(dataList: List<Obra>) {
        this.obraList = dataList
        notifyDataSetChanged()
    }
}

```

10.1.24 ObraImageAdapter

```

package com.adgonu.artnutria.data.Adapter

import android.net.Uri
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.viewmodel.ObraImageViewHolder

class ObraImageAdapter(private var imagenList: List<Uri>):
    RecyclerView.Adapter<ObraImageViewHolder>() {
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
        ObraImageViewHolder {
        val inflater = LayoutInflater.from(parent.context)
        return
        ObraImageViewHolder(inflater.inflate(R.layout.image_item, parent,
        false))
    }

    override fun onBindViewHolder(holder: ObraImageViewHolder, position:
    Int) {
        val item = imagenList[position]
        holder.bind(item)
    }

    override fun getItemCount(): Int = imagenList.size

    fun updateData(dataList: List<Uri>) {
        this.imagenList = dataList
        notifyDataSetChanged()
    }
}

```

10.1.25 UserAdapter

```
package com.adgonu.artnutria.data.Adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.adgonu.artnutria.R
import com.adgonu.artnutria.data.Interface.OnItemClickListenerUsuario
import com.adgonu.artnutria.data.modelo.Usuario
import com.adgonu.artnutria.viewmodel.UserViewHolder

class UserAdapter(private var userList: List<Usuario>, private val
listener:.OnItemClickListenerUsuario):
RecyclerView.Adapter<UserViewHolder>() {
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
UserViewHolder {
        val inflater = LayoutInflater.from(parent.context)
        return UserViewHolder(inflater.inflate(R.layout.usuario_item,
parent, false))
    }

    override fun onBindViewHolder(holder: UserViewHolder, position: Int) {
        val item = userList[position]
        holder.bind(item)

        holder.itemView.setOnClickListener {
            listener.onItemClick(item)
        }
    }

    override fun getItemCount(): Int = userList.size

    fun updateData(dataList: List<Usuario>) {
        this.userList = dataList
        notifyDataSetChanged()
    }
}
```