Máster Universitario en Ingeniería Informática
de la Universidad de Granada

App Android: Kotlin y Jetpack Compose

BrushNBid: Subasta tu Arte

Gestión de Información en
Dispositivos Móviles (GIDM)

Autora

Marina Jun Carranza Sánchez

Índice

Índice de figuras

Índice de cuadros

1. Introducción

La presente aplicación tiene como objetivo proporcionar una plataforma eficiente y accesible para artistas noveles y compradores interesados en el mundo del arte. En la actualidad, el mercado de arte enfrenta varios retos, especialmente para los artistas emergentes que buscan una forma efectiva de llegar a un público más amplio, y para los compradores que desean encontrar obras originales de calidad a precios accesibles. A través de esta aplicación, se busca crear un entorno digital que facilite las subastas de obras de arte, promoviendo tanto a los artistas como a los compradores de una manera fácil y segura.

1.1. Motivación

La motivación principal para desarrollar esta aplicación radica en la necesidad de democratizar el acceso al arte, dando a los artistas noveles la oportunidad de mostrar y vender sus obras en un entorno de subasta en línea. A menudo, los artistas más jóvenes enfrentan dificultades para ingresar al mercado del arte debido a la falta de visibilidad y recursos. Al ofrecer una plataforma accesible y fácil de usar, se busca brindarles una oportunidad justa para destacar su trabajo y conectar con potenciales compradores.

Por otro lado, la app también responde a la demanda creciente de los compradores por plataformas digitales donde puedan acceder a una variedad de obras de arte en tiempo real, participar en subastas y adquirir piezas únicas sin las barreras físicas de las galerías tradicionales. Además, se pretende proporcionar una experiencia interactiva e inmersiva que incentive la participación activa en las subastas y promueva la compra de arte en un formato dinámico y atractivo.

1.2. Objetivos

El objetivo fundamental de esta aplicación es ofrecer una solución eficiente para las subastas de arte digital. Los usuarios podrán crear, gestionar y participar en subastas de manera fácil e intuitiva. Entre los principales objetivos se incluyen:

- **Facilitar la creación de subastas:** los artistas podrán subir sus obras de manera sencilla y personalizar las subastas según sus preferencias (precio de compra inmediata, incremento de puja, duración, etc.).
- **Acceso a obras de arte de artistas noveles:** los compradores podrán explorar una variedad de obras y participar en subastas activas de artistas emergentes.

- **Sistemas de pujas automatizadas y recomendadas:** la aplicación proporcionará recomendaciones sobre las pujas, asegurando que los compradores puedan tomar decisiones informadas al participar en las subastas.
- **Seguridad y fiabilidad:** la plataforma ofrecerá un sistema seguro para gestionar las transacciones y proteger la información personal y financiera de los usuarios.

1.3. Tareas principales de la aplicación

La aplicación ha sido diseñada para ser intuitiva y fácil de navegar. Entre las funcionalidades más relevantes, se encuentran:

- Registrarse e iniciar sesión con su cuenta de usuario.
- Subir sus obras y crear subastas personalizadas para venderlas.
- Realizar y seguir pujas en subastas activas.
- Consultar el historial de pujas y ver las subastas seguidas.
- Adjudicar las obras a los mejores postores cuando la subasta termine.

La interfaz ha sido optimizada para garantizar que tanto artistas como compradores puedan interactuar con la plataforma de manera fluida, sin complicaciones innecesarias.

1.4. Impacto esperado

Con esta aplicación, se espera poder mejorar el acceso de los artistas noveles al mercado del arte, ayudándolos a obtener mayor visibilidad y oportunidades de ventas. Al mismo tiempo, la plataforma proporcionará a los compradores una experiencia moderna y accesible para adquirir obras de arte de calidad. Este enfoque innovador tiene el potencial de cambiar la dinámica de la compra y venta de arte, permitiendo que más personas participen en el mundo del arte digital y presencial.

En resumen, esta app no solo facilita las transacciones artísticas, sino que también empodera a los artistas emergentes y crea un nuevo espacio para que el arte se descubra y se disfrute de manera dinámica e incluso divertida dado ese toque de competitividad propio de un sistema de subastas.

2. Análisis del problema

Este capítulo abarca desde la definición de historias de usuario y requisitos, hasta el desarrollo de casos de uso y sus diagramas correspondientes.

2.1. Historias de usuario y requisitos

Como se detalla en (Martin, 2011), las historias de usuario son imprescindibles en el desarrollo ágil de software, pero también resultan valiosas en cualquier otra metodología de desarrollo, ya sea tradicional o no. Pues a través de un formato sencillo, recogen de forma clara las necesidades y expectativas de los usuarios y clientes, facilitando la comunicación entre integrantes del equipo de desarrollo y clientes. Un ambiente colaborativo donde las ideas pueden fluctuar a medida que se avanza aumenta las probabilidades de éxito de cualquier proyecto.

El formato típico de una historia de usuario se basa en los tres elementos siguientes:

Como [rol del usuario], **quiero** [objetivo], **para poder** [beneficio].

Figura 1: Patrón general de una historia de usuario

Esta estructura facilita empatizar con la persona usuaria, entender qué pretende lograr sin entrar en detalles del cómo y el valor que aporta al producto, conocido como beneficio. Así, además de sintetizar en gran medida la información, permite cierta flexibilidad y la adaptación a cambios e integración de nuevas ideas durante el proceso de desarrollo.

Las historias de usuario ofrecen una perspectiva más centrada en los usuarios finales, mientras que los requisitos funcionales se limitan a describir el comportamiento del sistema. Entonces, merece la pena considerar ambos para que el producto final no solo cumpla con las especificaciones técnicas, sino que también otorgue valor real a quienes va dirigido.

2.1.1. Historias de usuario

Las siguientes historias de usuario establecen los requisitos de la aplicación de subastas.

HU01: Registrar nuevo usuario	
Como	nuevo usuario/a
Quiero	poder registrarme en la aplicación proporcionando mis datos personales
Para	crear una cuenta y poder acceder a las funcionalidades de la plataforma

Cuadro 1: Historia de usuario nº 1

HU02: Iniciar sesión	
Como	artista o comprador/a
Quiero	poder iniciar sesión una sola vez al abrir la app, sin tener que volver a hacerlo al cerrar y abrir la app
Para	no tener que volver a autenticarme cada vez que abro la aplicación

Cuadro 2: Historia de usuario nº 2

HU03: Cerrar sesión	
Como	artista o comprador/a
Quiero	poder cerrar sesión en la aplicación
Para	proteger mi cuenta y mis datos cuando termine de usar la aplicación

Cuadro 3: Historia de usuario nº 3

HU04: Buscar subasta concreta	
Como	artista o comprador/a
Quiero	poder buscar una subasta concreta por el nombre de la obra o del autor/a
Para	encontrar fácilmente la subasta de mi interés

Cuadro 4: Historia de usuario nº 4

HU05: Ver detalles de una subasta	
Como	comprador/a o artista
Quiero	poder ver los detalles de una subasta
Para	conocer más sobre la subasta, como las obras y el historial de pujas

Cuadro 5: Historia de usuario nº 5

HU06: Crear subasta	
Como	artista
Quiero	poder crear una subasta seleccionando una de mis obras disponibles ya creadas, y personalizarla con valores como incremento, precio de compra inmediata y duración
Para	poder poner a la venta mis obras en la plataforma y gestionar las condiciones de la subasta

Cuadro 6: Historia de usuario nº 6

HU07: Ver subastas seguidas	
Como	comprador/a
Quiero	poder ver las subastas que sigo (en las que he participado)
Para	poder revisar rápidamente las subastas en las que estoy pujando

Cuadro 7: Historia de usuario nº 7

HU08: Ver subastas activas y filtrarlas	
Como	comprador/a o artista
Quiero	poder consultar mis subastas y filtrarlas por aquellas que están activas o finalizadas
Para	gestionar las subastas que he creado

Cuadro 8: Historia de usuario nº 8

HU09: Ver detalles de una obra	
Como	comprador/a o artista
Quiero	poder ver los detalles de una obra dentro de la subasta
Para	conocer más sobre la obra antes de tomar decisiones de puja

Cuadro 9: Historia de usuario nº 9

HU10: Adjudicar obra al mejor postor	
Como	artista
Quiero	que se adjudique la obra al mejor postor si la subasta ha terminado o si se ha pagado el precio de compra inmediata
Para	finalizar la subasta y cerrar el proceso de venta

Cuadro 10: Historia de usuario nº 10

HU11: Ver mis obras	
Como	artista
Quiero	poder ver todas mis obras subidas, pudiendo distinguir cuáles he vendido y cuáles no
Para	gestionar mis ventas y ver el estado de mis obras en la plataforma

Cuadro 11: Historia de usuario nº 11

HU12: Ver puja recomendada	
Como	comprador/a
Quiero	que se me recomiende una cantidad para pujar en una subasta
Para	hacer una puja informada y no perder la oportunidad de ganar la subasta

Cuadro 12: Historia de usuario nº 12

2.1.2. Requisitos funcionales

Esta sección define y describe las características de alto nivel (requisitos funcionales) del sistema que son necesarias para cubrir las necesidades de los usuarios. Se pueden estructurar de la siguiente manera:

RF1: Registro e inicio de sesión

- **RF1.1**: La aplicación debe permitir que los usuarios se registren proporcionando sus datos personales.
- **RF1.2**: La aplicación debe permitir que los usuarios inicien sesión utilizando sus credenciales y mantener la sesión activa durante toda la navegación.
- **RF1.3**: La aplicación debe permitir cerrar sesión para proteger la cuenta y los datos del usuario.

RF2: Creación y gestión de subastas

- **RF2.1**: La aplicación debe permitir a los artistas subir sus obras a su perfil.
- **RF2.2**: La aplicación debe permitir personalizar la subasta con parámetros como incremento, precio de compra inmediata y duración; así como seleccionar una de sus obras disponibles.
- **RF2.3**: La aplicación debe permitir a los artistas gestionar sus subastas activas y finalizadas.

RF3: Visualización y gestión de obras

- **RF3.1**: La aplicación debe permitir a los artistas ver todas las obras que han subido a la plataforma, indicando cuáles han sido vendidas y cuáles no.
- **RF3.2**: La aplicación debe permitir a los compradores ver todas las obras disponibles en las subastas activas.
- **RF3.3**: La aplicación debe permitir ver los detalles de cada obra, incluyendo su descripción, autor/a y precio.

RF4: Participación en subastas

- **RF4.1**: La aplicación debe permitir a los compradores realizar pujas en las subastas activas.
- **RF4.2**: La aplicación debe recomendar una cantidad para pujar en función de la puja más alta y el incremento de la subasta.
- **RF4.3**: La aplicación debe permitir a los compradores seguir las subastas en las que están participando.

RF5: Consultas sobre subastas y obras

- **RF5.1**: La aplicación debe permitir buscar subastas concretas por el nombre de la obra o el autor/a.
- **RF5.2**: La aplicación debe permitir ver las subastas seguidas por el usuario/a, que incluye subastas activas en las que ha participado.

RF6: Finalización de subastas

- **RF6.1**: La aplicación debe permitir adjudicar la obra al mejor postor si la subasta ha terminado o si se ha alcanzado el precio de compra inmediata.
- **RF6.2**: La aplicación debe notificar al usuario/a cuando se haya adjudicado una obra.

2.1.3. Requisitos no funcionales

Esta sección define las cualidades no relacionadas con el comportamiento directo del sistema.

RNF1: Accesibilidad y usabilidad

- **RNF1.1**: La aplicación debe ser fácil de usar y con una navegación intuitiva.
- **RNF1.2**: Los mensajes y las interfaces deben ser claros y concisos.
- **RNF1.3**: Los elementos de la interfaz deben tener un tamaño adecuado para facilitar su lectura y comprensión.
- **RNF1.4**: La aplicación debe ser responsive, adaptándose a diferentes tamaños de pantalla y dispositivos Android.

RNF2: Rendimiento

- **RNF2.1**: La aplicación debe tener tiempos de respuesta rápidos, ideales por debajo de los 3 segundos en la mayoría de las operaciones.
- **RNF2.2**: La aplicación debe manejar múltiples usuarios y subastas de forma eficiente, sin fallos ni ralentizaciones.

RNF3: Seguridad

- **RNF3.1**: La aplicación debe utilizar protocolos de encriptación para proteger las credenciales del usuario y la información sensible.
- **RNF3.2**: La aplicación debe seguir las mejores prácticas de seguridad en el manejo de datos personales y de pago.
- **RNF3.3**: La autenticación debe ser segura, permitiendo la recuperación de contraseñas y la verificación en dos pasos si se considera necesario.

RNF4: Compatibilidad

- **RNF4.1**: La aplicación debe ser compatible con las versiones más recientes de Android (Android 10 o superior).
- **RNF4.2**: La aplicación debe ser compatible con dispositivos móviles y tabletas Android.

2.1.4. Requisitos de datos

Esta sección define los requisitos relacionados con los datos que maneja la aplicación.

RD1: Gestión de usuarios

- **RD1.1**: La aplicación debe almacenar los datos personales del usuario/a (nombre, email, etc.) de manera segura.
- **RD1.2**: La aplicación debe almacenar el historial de subastas y pujas de cada usuario/a.
- **RD1.3**: La aplicación debe almacenar las obras creadas por cada usuario/a, incluyendo detalles como nombre, descripción, precio y estado de la venta.

RD2: Gestión de subastas

- **RD2.1**: La aplicación debe almacenar la información de cada subasta, incluyendo la obra asociada, el precio inicial, el incremento, el precio de compra inmediata y la duración.
- **RD2.2**: La aplicación debe almacenar el historial de pujas de cada subasta, incluyendo el nombre del comprador/a, la cantidad pujada y la fecha.
- **RD2.3**: La aplicación debe almacenar el estado de cada subasta (activa o finalizada).

RD3: Gestión de pujas

- **RD3.1**: La aplicación debe almacenar la información de cada puja realizada, incluyendo el monto y el comprador/a.
- **RD3.2**: La aplicación debe registrar la puja recomendada para cada subasta según el incremento establecido.

2.2. Casos de uso y diagrama correspondiente

Una vez se han detallado las historias y usuario y los requisitos del sistema, el siguiente paso es elaborar los casos de uso, asociados a los requisitos definidos previamente e incluso otros casos de uso. Esta técnica de ingeniería de software permite establecer la forma de interactuar entre los actores, entendidos como las entidades o sistema implicados en cada funcionalidad.

2.2.1. Casos de uso

CU01: Registro de usuario/a	
Descripción	El usuario/a puede registrarse en la aplicación proporcionando sus datos personales para crear una cuenta.
Precondiciones	La aplicación debe estar abierta y en la pantalla de inicio de sesión.
Postcondiciones	El usuario/a queda registrado en la plataforma y puede acceder a sus funciones (subastas, obras, pujas, etc).
Referencias	RF1.1
Flujo normal de eventos	**1.** El usuario/a accede a la pantalla de registro. **2.** El usuario/a ingresa sus datos (nombre, correo, contraseña...). **3.** El usuario confirma los datos e inicia el registro. **4.** La aplicación valida los datos y muestra un mensaje de éxito, reconduciendo a la pantalla de login. **5.** El usuario puede iniciar sesión con sus credenciales.
Flujo alterno de eventos	**2.a.** Si los datos del usuario no son válidos, la aplicación muestra un mensaje de error.

Cuadro 13: Caso de uso nº 1

CU02: Iniciar sesión	
Descripción	El usuario/a puede iniciar sesión en la aplicación con sus credenciales previamente registradas y mantenerse autenticado durante toda la navegación.
Precondiciones	El usuario/a debe estar registrado en la aplicación.
Postcondiciones	El usuario/a queda autenticado y puede acceder a todas las funcionalidades de la plataforma.
Referencias	RF1.2
Flujo normal de eventos	**1.** El usuario/a accede a la pantalla de inicio de sesión. **2.** El usuario/a ingresa su correo electrónico (o username) y contraseña. **3.** La aplicación valida las credenciales. **4.** Si son correctas, el usuario accede a la pantalla principal de la aplicación.
Flujo alterno de eventos	**2.a.** Si las credenciales no son válidas, se muestra un mensaje de error.

Cuadro 14: Caso de uso nº 2

CU03: Cerrar sesión	
Descripción	El usuario/a puede cerrar sesión para proteger su cuenta y sus datos.
Precondiciones	El usuario/a debe estar autenticado/a en la aplicación.
Postcondiciones	El usuario/a cierra la sesión y vuelve a la pantalla de inicio login.
Referencias	RF1.3
Flujo normal de eventos	**1.** El usuario/a accede a la pantalla de su perfil. **2.** El usuario/a pulsa en el botón de ajustes. **3.** El usuario/a pulsa *Cerrar sesión*. **4.** La sesión se cierra y se redirige a la pantalla de inicio de sesión.

Cuadro 15: Caso de uso nº 3

CU04: Crear subasta	
Descripción	El artista puede crear una nueva subasta seleccionando una de sus obras disponibles, y personalizarla con incremento, precio de compra inmediata, duración, etc.
Precondiciones	El artista debe estar autenticado y tener al menos una obra subida y disponible.
Postcondiciones	La subasta se publica en la plataforma.
Referencias	RF2.1, RF2.2
Flujo normal de eventos	**1.** El artista pulsa el botón de «+». **2.** Si no ha subido aún la obra, pulsa *Añadir obra*. **3.** Rellena los datos necesarios para la obra, incluyendo la imagen. **4.** Una vez confirmada la obra subida en el perfil, vuelve a pulsar el «+» y esta vez, selecciona *Añadir subasta*. **5.** Rellena los datos necesarios para la subasta, incluyendo al selección de una de sus obras disponibles (aparecerá una lista de ítems seleccionables). **6.** Una vez creada recibe un mensaje de éxito y redirige a la pantalla de *Mis subastas*

Cuadro 16: Caso de uso nº 4

CU05: Ver mis subastas	
Descripción	El artista puede consultar sus subastas y filtrarlas por activas o finalizadas.
Precondiciones	El usuario debe estar autenticado y haber creado subastas anteriormente.
Postcondiciones	Se muestra el listado filtrado de subastas.
Referencias	RF2.3
Flujo normal de eventos	**1.** El artista accede a *Mis subastas*. **2.** Selecciona un filtro: activas o finalizadas. **3.** La aplicación muestra el listado filtrado.

Cuadro 17: Caso de uso nº 5

CU06: Realizar puja (con recomendación)	
Descripción	El comprador/a puede pujar en una subasta activa. La aplicación muestra la cantidad recomendada usando técnicas de ciencia de datos.
Precondiciones	El usuario/a debe estar autenticado y la subasta debe estar activa.
Postcondiciones	La puja se registra y se actualiza el historial de subastas y puja actual.
Referencias	RF4.1, RF4.2
Flujo normal de eventos	**1.** El comprador/a accede a la subasta. **2.** Se muestra la cantidad recomendada para pujar. **3.** Introduce su puja. **4.** La aplicación la valida y la registra.

Cuadro 18: Caso de uso nº 6

CU07: Ver subastas activas y seguidas	
Descripción	El comprador/a puede ver las subastas activas y aquellas en las que ha participado.
Precondiciones	Debe estar autenticado.
Postcondiciones	Se muestran las subastas activas y las seguidas por el usuario.
Referencias	RF3.2, RF4.3, RF5.2
Flujo normal de eventos	**1.** Desde la pantalla principal, accede al filtro de subastas activas y al de seguidas. **2.** Visualiza todas las subastas activas y aquellas donde ha pujado al menos una vez.

Cuadro 19: Caso de uso nº 7

CU08: Buscar subasta	
Descripción	El usuario/a puede buscar subastas por nombre de obra o autor.
Precondiciones	Debe estar autenticado/a.
Postcondiciones	Se muestran las subastas coincidentes.
Referencias	RF5.1
Flujo normal de eventos	**1.** Introduce el texto de búsqueda. **2.** La aplicación muestra resultados coincidentes.

Cuadro 20: Caso de uso nº 8

CU09: Ver detalles de una obra	
Descripción	El usuario/a puede ver los detalles de una obra.
Precondiciones	Se puede acceder desde la pantalla de detalles de una subasta o en las obras del perfil del usuario/a.
Postcondiciones	Se muestra la descripción, tipo de obra y otros datos.
Referencias	RF3.3
Flujo normal de eventos	**1.** Accede a la obra desde una subasta o desde el perfil del usuario/a. **2.** Visualiza todos sus detalles.

Cuadro 21: Caso de uso nº 9

CU10: Ver mis obras	
Descripción	El artista puede ver todas sus obras y saber cuáles ha vendido.
Precondiciones	El usuario debe estar autenticado y haber subido obras.
Postcondiciones	Se muestra el listado con indicadores de estado.
Referencias	RF3.1
Flujo normal de eventos	**1.** Accede a *Mi perfil*. **2.** Visualiza todas sus obras con estado (disponible/vendida).

Cuadro 22: Caso de uso nº 10

CU11: Adjudicar obra	
Descripción	La aplicación adjudica la obra al mejor postor al finalizar la subasta o si se paga el precio de compra inmediata.
Precondiciones	La subasta debe haber finalizado o se ha alcanzado el precio de compra inmediata.
Postcondiciones	Se actualiza el estado y la propiedad de la obra subastada, se anuncia el/la mejor postor/a y se bloquean las pujas.
Referencias	RF6.1, RF6.2
Flujo normal de eventos	**1.** Finaliza la subasta o se alcanza el precio de compra inmediata. **2.** Se adjudica la obra al ganador/a, actualizando su estado y bloqueando las pujas.

Cuadro 23: Caso de uso nº 11

2.2.2. Matriz de cobertura de requisitos funcionales

A continuación, se comprueba que todos los casos de uso definidos en la sección anterior referencian a un requisito funcional como mínimo.

Dicho de otra manera, todos los requisitos funcionales deben estar cubiertos por al menos un caso de uso, como se demuestra mediante la siguiente matriz:

RF / CU	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11
RF1.1	X										
RF1.2		X									
RF1.3			X								
RF2.1				X							
RF2.2				X							
RF2.3					X						
RF3.1										X	
RF3.2							X				
RF3.3									X		
RF4.1						X					
RF4.2						X					
RF4.3							X				
RF5.1								X			
RF5.2							X				
RF6.1											X
RF6.2											X

Cuadro 24: Matriz de cobertura de requisitos funcionales por casos de uso

2.2.3. Diagrama de casos de uso

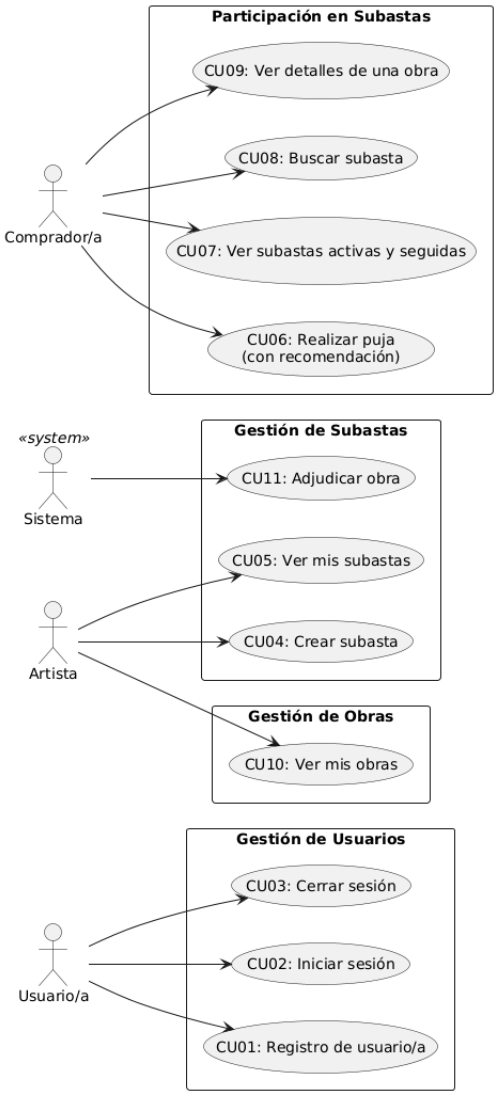

Figura 2: Diagrama de casos de uso (PlantUML contributors, 2024)

La Figura 2 muestra el diagrama de casos de uso de la aplicación de subastas. En él se representan los actores principales (Usuario/a, Artista, Comprador/a y Sistema) y los casos de uso que cada uno puede realizar, agrupados en diferentes paquetes funcionales: Gestión de Usuarios, Gestión de Obras, Gestión de Subastas y Participación en Subastas.

El caso de uso CU11 es responsabilidad del Sistema y se activa automáticamente, mientras que el resto representan funcionalidades que los usuarios/as pueden invocar desde la app, como crear subastas, realizar pujas con recomendación, consultar obras o gestionar la sesión.

2.3. Diagrama de clases

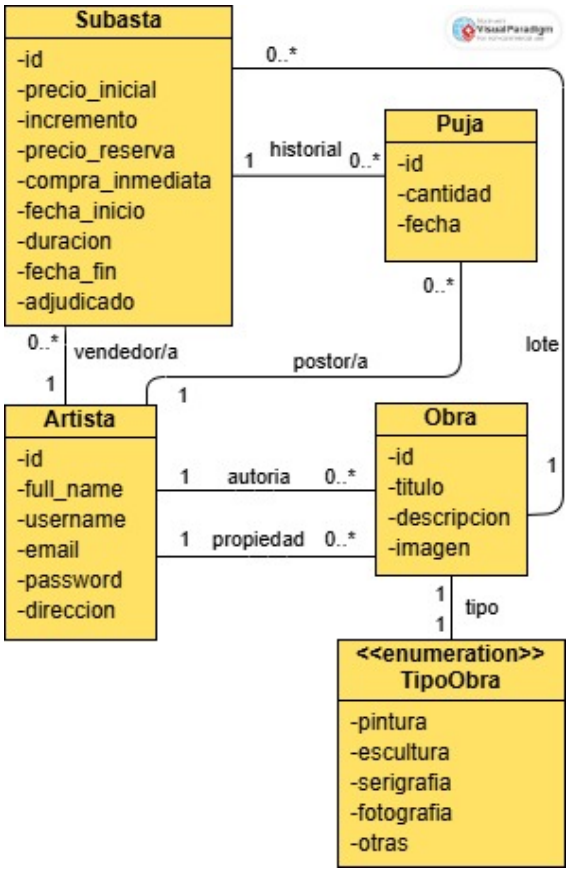

Figura 3: Diagrama de clases de la app (Visual Paradigm, 2025)

La Figura 3 representa el modelo de clases de la aplicación de subastas de arte.

- ***Subasta***, con atributos como el precio inicial, fechas, duración y el estado de adjudicación. Está asociada tanto con pujas (historial) como con artistas (vendedor/a).
- ***Puja*** almacena los detalles de cada puja, incluyendo la cantidad y la fecha, y está vinculada a una subasta específica.
- ***Artista*** representa a los usuarios/as de la plataforma que pueden ser tanto vendedores (suben obras para subastar) como compradores (realizan pujas en subastas).
- ***Obra*** representa las piezas a subastar, asociadas a un artista autor y otro propietario (que pueden o no ser el mismo) y con atributos como título, imagen y tipo de obra.
- ***TipoObra*** define los diferentes tipos de obras como pintura, escultura, etc.

3. Diseño

En esta etapa de la metodología ágil, se debe elaborar un diseño que responda a las necesidades de los usuarios/as, facilitando la interacción con la aplicación mediante un flujo de navegación claro y una interfaz intuitiva. El diseño sigue los principios de usabilidad y experiencia de usuario (UX) para asegurar que la aplicación sea fácil de usar, eficiente y agradable.

3.1. Diseño del flujo de navegación: Wireflow

El diseño del flujo de la aplicación se ha representado mediante un diagrama *Wireflow* (Figura 4), que combina la estructura de un wireframe con la navegación entre pantallas.

El diagrama Wireflow proporcionado describe de manera clara el flujo de navegación de la aplicación de subastas. En la parte superior del diagrama, se observan las opciones iniciales que permiten al usuario registrarse o iniciar sesión si ya posee una cuenta. Una vez que el usuario se autentica, tiene acceso a varias funciones clave dentro de la aplicación, como la visualización de subastas y obras.

Una de las secciones principales de la aplicación es la gestión de subastas, donde el usuario puede ver tanto las subastas activas como las que sigue, permitiéndole revisar el progreso de las mismas. Para facilitar la búsqueda de subastas, se ofrece la opción de buscar por texto o filtrar por estado. Desde la vista de detalles de una subasta, el usuario tiene la posibilidad de realizar una puja o comprar la obra si está disponible para ello.

En cuanto a la gestión de las obras, los artistas pueden añadir nuevas obras a la plataforma. Además, pueden ver los detalles de sus obras y de su propio perfil.

El flujo general de la aplicación está diseñado para ser intuitivo, facilitando a los usuarios tanto artistas como compradores el acceso rápido a las funcionalidades clave como la búsqueda de subastas, la gestión de obras, la realización de pujas, y la consulta de su historial. Este diseño asegura que los usuarios puedan realizar todas las acciones necesarias sin perderse en la navegación, lo que contribuye a una experiencia de usuario óptima.

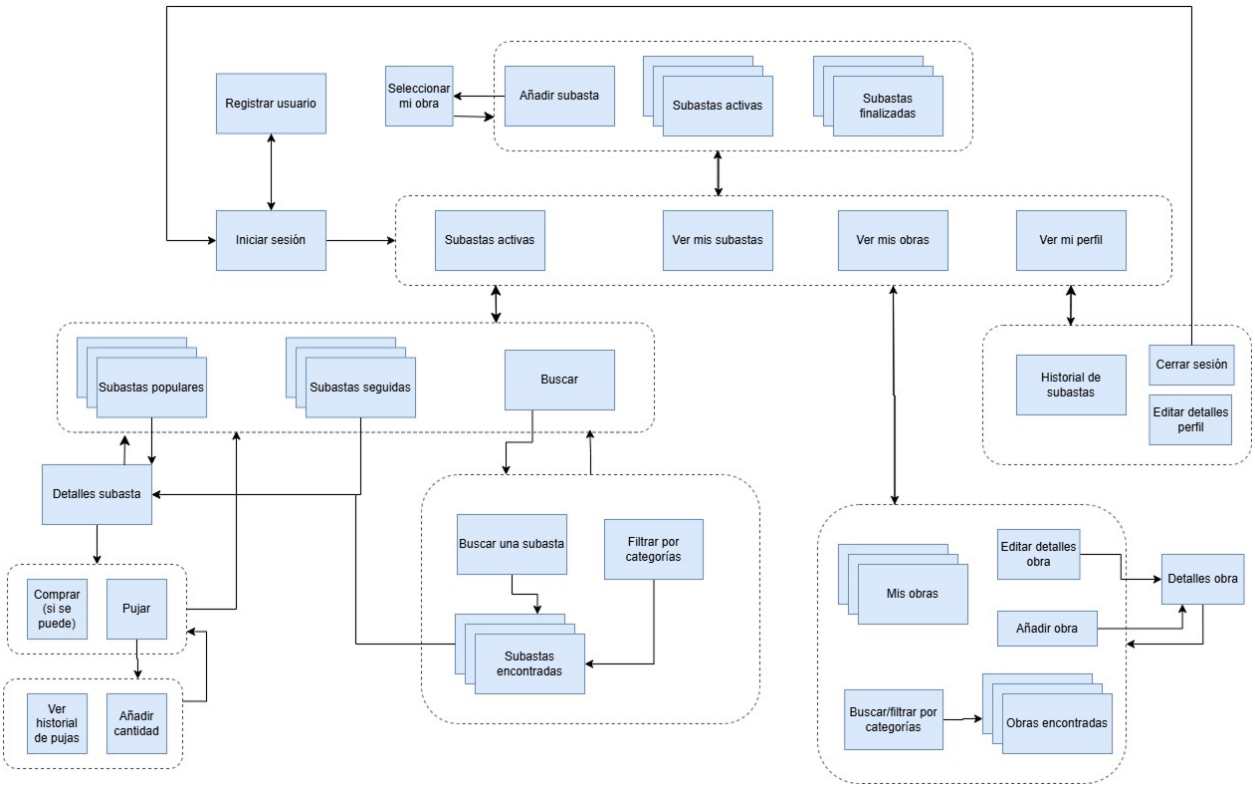

Figura 4: Diagrama Wireflow

3.2. Diseño de la interfaz: prototipo en Figma

Una vez aprobado el flujo de navegación, se ha diseñado la interfaz visual de la aplicación utilizando *Figma*, una herramienta de diseño colaborativo (Figma, 2021). El prototipo de alta fidelidad elaborado en Figma permite simular la interacción con la aplicación, mostrando cómo se verá y cómo funcionarán los elementos visuales (botones, menús, pantallas de contenido).

El diseño en Figma ha sido desarrollado teniendo en cuenta los principios de accesibilidad y facilidad de uso, lo que garantiza que la aplicación será intuitiva y fácil de navegar para todos los usuarios/as.

El proyecto completo en Figma mostrado en la Figura 5 puede ser accedido a través del siguiente enlace: **Tablero de diseño en Figma.**

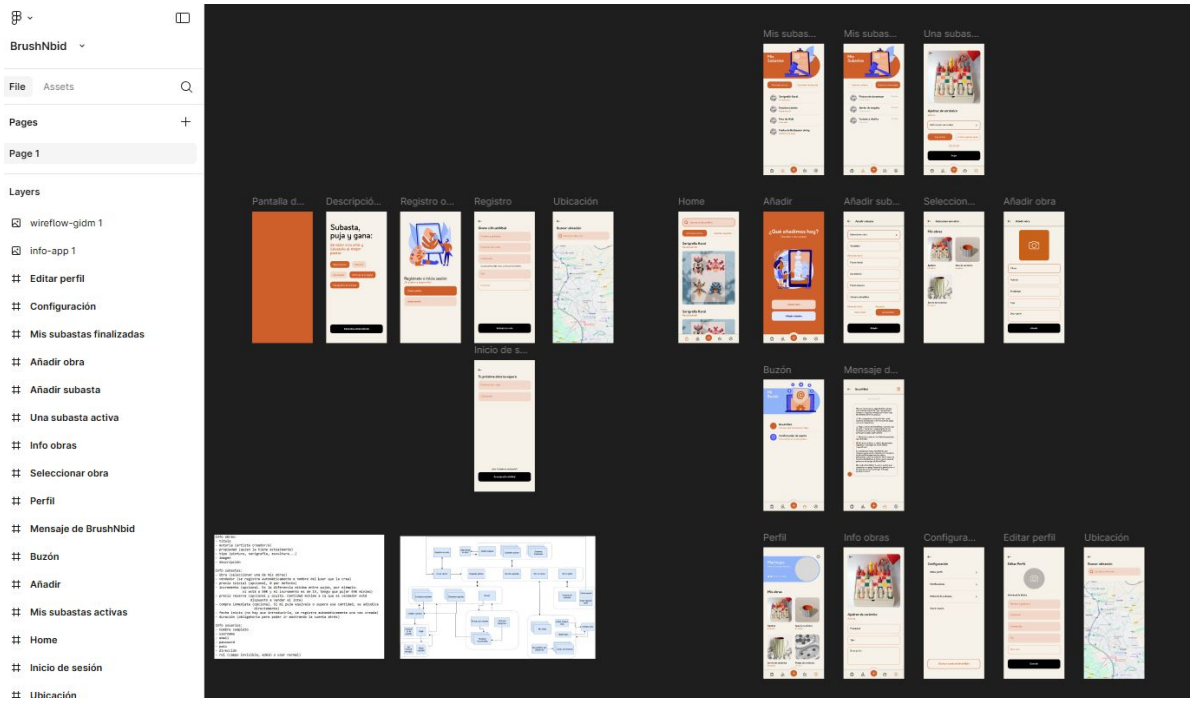

Figura 5: Prototipo de Figma

La Figura 6 muestra el diseño realizado en Figma para la pantalla principal de la app. El resto de diseños individuales para las pantallas pueden consultarse a través del enlace: **Pantallas en Figma.**

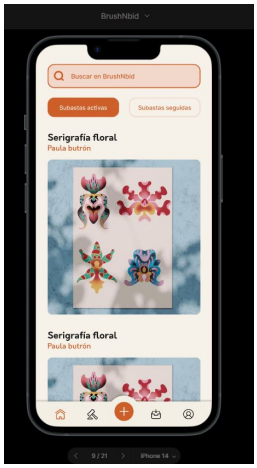

Figura 6: Ejemplo de pantalla en Figma

Este diseño no solo proporciona una estructura clara y eficiente, sino que también mejora la experiencia del usuario/a, permitiéndole realizar tareas de manera ágil y sin complicaciones. La iteración continua y la retroalimentación de usuarios/as en el prototipo aseguran que el diseño final sea completamente funcional y fácil de usar.

4. Desarrollo

En esta sección se describen las principales decisiones y tecnologías utilizadas durante el desarrollo de la aplicación.

4.1. Tecnologías utilizadas

En el desarrollo de esta aplicación se han utilizado diversas tecnologías que permiten una integración eficiente entre el backend, la aplicación móvil y la base de datos.

Para el backend, se ha utilizado *Node.js* (Node.js, 2021) con *Express* como framework para gestionar las rutas y la lógica de negocio de la aplicación (OpenJS Foundation, 2021). Además, se empleó *TypeScript* para mejorar la seguridad en el código y facilitar la detección temprana de errores mediante tipado estático (Microsoft, 2021). La API desarrollada permite la comunicación con el frontend móvil a través de llamadas HTTP, utilizando *Swagger* para la documentación y pruebas interactivas de la API (Swagger, 2021).

La base de datos utilizada es *PostgreSQL*, un sistema de gestión de bases de datos relacional de código abierto, robusto y escalable (The PostgreSQL Global Development Group, 2020), que almacena toda la información relacionada con las subastas, usuarios y obras de arte.

En el desarrollo de la aplicación móvil se utilizó *Android Studio* como entorno de desarrollo (Google, 2021), y *Kotlin* como el lenguaje principal, conocido por su simplicidad y eficiencia en el desarrollo de aplicaciones Android (JetBrains, 2021). Para la construcción de las interfaces de usuario, se empleó *Jetpack Compose*, un toolkit moderno y declarativo para la creación de interfaces nativas en Android, que permite una mayor flexibilidad y facilidad para construir UI de forma programática (Google, 2018).

En cuanto a la gestión de los datos y la comunicación con la API, se utilizó *Retrofit* junto con *Gson* para la conversión de JSON a objetos Kotlin. Retrofit se encarga de gestionar las peticiones HTTP de manera sencilla y eficiente (Square, 2021), mientras que Gson facilita la deserialización de los datos obtenidos desde el servidor (Google, 2025).

Además, se integraron varias bibliotecas para mejorar la experiencia de usuario y la funcionalidad de la app, como *Coil* para la carga y visualización de imágenes, y *Datastore* para almacenar preferencias de usuario de manera segura y eficiente. También se utilizaron varias dependencias para el manejo de la navegación dentro de la app, como *Navigation Compose*, que permite una navegación fluida y coherente entre las pantallas de la aplicación.

4.2. Arquitectura y flujo de desarrollo

La aplicación sigue una arquitectura moderna basada en el patrón *Modelo-Vista-ViewModel* (MVVM), que separa claramente la lógica de negocio de la interfaz de usuario, facilitando su mantenimiento y escalabilidad.

El backend gestiona las operaciones relacionadas con los datos, como la creación de subastas, las pujas y la autenticación de usuarios, exponiendo una API que permite la interacción con el frontend a través de Retrofit. La base de datos PostgreSQL, con un diseño relacional, garantiza la integridad de los datos y optimiza las consultas mediante índices y claves foráneas.

En el frontend, se utiliza Jetpack Compose para crear interfaces nativas y reactivas, mejorando el rendimiento y la experiencia de usuario. Este enfoque modular facilita la adaptación de la aplicación a nuevas funcionalidades sin comprometer su calidad ni rendimiento.

4.3. Estructura del proyecto

El proyecto está compuesto por tres componentes principales que interactúan entre sí: la API, la base de datos y la aplicación móvil.

4.3.1. API

La aplicación utiliza una API RESTful construida con Node.js y Express, siguiendo los principios de arquitectura REST para una gestión eficiente de los recursos. Esta API permite la interacción entre el frontend y el backend, manejando las operaciones de las principales entidades del sistema.

La API se organiza en controladores específicos para cada entidad:

- ***ObraController***: gestiona todas las operaciones relacionadas con las obras de arte.
- ***SubastaController***: se encarga de la creación y actualización de subastas, y de la gestión de las pujas.
- ***UserController***: maneja las operaciones relacionadas con los usuarios de la plataforma.
- ***MLController***: implementa la recomendación de pujas utilizando técnicas de aprendizaje automático.

Cada uno de los controladores anteriores tiene un fichero *Routes* asociado para definir las rutas de los endpoints y la documentación, generada con *Swagger*, que describe los parámetros de entrada, las posibles respuestas y los códigos de estado HTTP para cada operación.

Esta documentación interactiva permite probar los endpoints directamente desde la interfaz (Figura 7 y 8), lo que mejora la eficiencia en el proceso de desarrollo y prueba de la API.

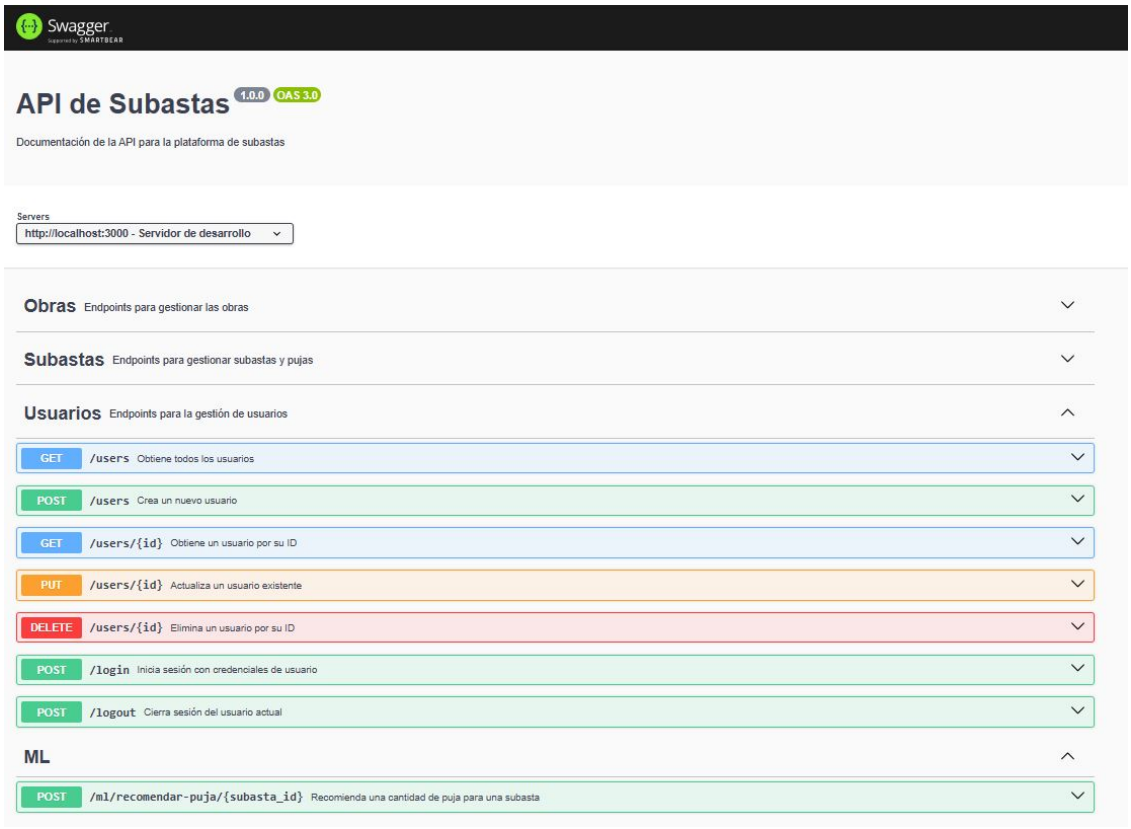

Figura 7: Interfaz de Swagger para la API REST

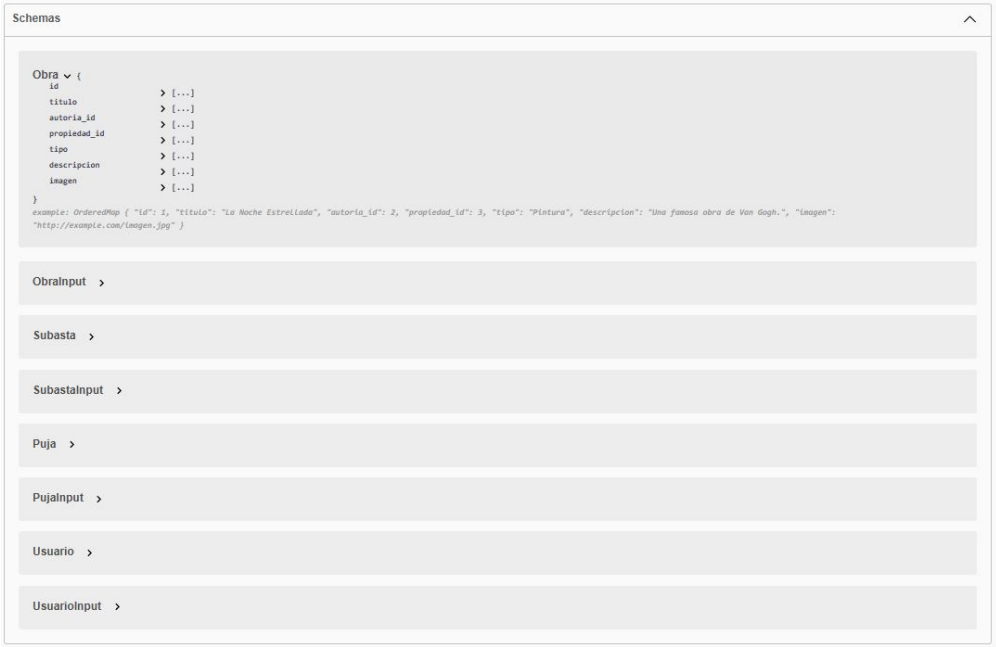

Figura 8: Esquemas definidos en Swagger

4.3.2. Base de datos

La base de datos está estructurada en un modelo relacional utilizando *PostgreSQL*. A continuación se describen las tablas principales, que pueden visualizarse en el diagrama E/R (Figura 9):

1. **Tabla artists**: almacena la información de los artistas, como su nombre, correo electrónico, contraseña, billetera digital y rol (artista o comprador).

2. **Tabla obras**: contiene las obras de arte subidas por los artistas, con detalles como título, descripción, tipo, autor/a, y la imagen asociada.

3. **Tabla subastas**: gestiona las subastas de las obras, con campos como el precio inicial, incremento, precio de reserva, duración, fecha de inicio y fin, y si la subasta ha sido adjudicada.

4. **Tabla pujas**: registra las pujas realizadas en las subastas, asociando a cada puja el monto, la fecha y el usuario que la realizó.

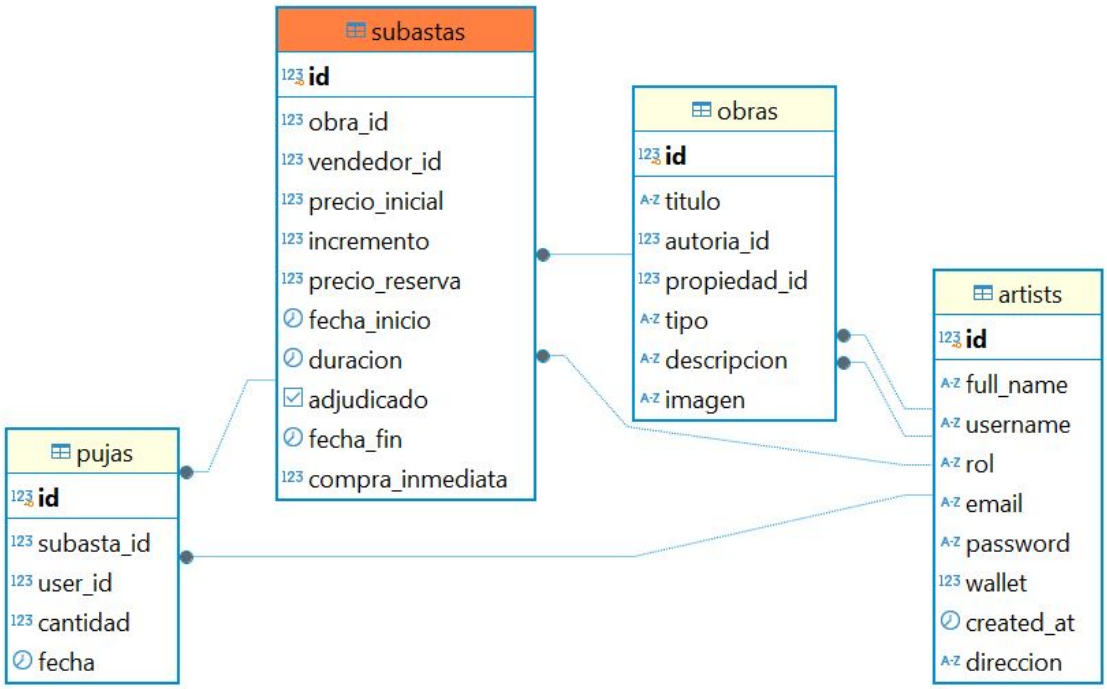

Figura 9: Diagrama Entidad-Relación en PostgreSQL

En cuanto a las relaciones entre entidades:

- *artists* y *obras*: relación de uno a muchos, donde un artista puede tener múltiples obras.
- *obras* y *subastas*: relación de uno a muchos, ya que una obra puede ser subastada varias veces (por los distintos propietarios por los que pase).
- *artists* y *subastas*: relación de uno a muchos, ya que un artista puede crear varias subastas.
- *subastas* y *pujas*: relación de uno a muchos, donde cada subasta puede recibir múltiples pujas.
- *artists* y *pujas*: relación de uno a muchos, ya que un comprador puede realizar múltiples pujas.

Este diseño garantiza la integridad de los datos y facilita la gestión eficiente de las subastas, las obras, las pujas y los usuarios dentro de la aplicación.

4.3.3. Aplicación Android

La estructura del proyecto Android sigue las mejores prácticas recomendadas para mantener el código modular, escalable y fácil de mantener. A continuación se describe la organización de las carpetas y su propósito dentro del proyecto.

- **api**: contiene los archivos encargados de gestionar la comunicación con el backend. El *ApiClient* es responsable de establecer la configuración y las instancias necesarias para realizar peticiones HTTP, mientras que *ApiService* contiene los métodos que hacen uso de Retrofit para interactuar con la API del servidor, realizando las peticiones correspondientes a los endpoints definidos en el backend.
- **controllers**: en esta carpeta se encuentran los controladores que gestionan la lógica de negocio relacionada con las entidades de la aplicación. Los controladores para *Obra*, *User* y *Subasta* son responsables de interactuar con el *ApiService*, gestionando las llamadas a las funciones de la API. Estos controladores actúan como intermediarios entre el modelo de datos (*data*) y las vistas, manipulando la lógica de negocio antes de pasar la información a la interfaz de usuario.
- **data**: contiene todas las clases de datos, como *Puja*, *Subasta*, *User*, *Obra*, entre otras. Estas clases están diseñadas para representar la estructura de los objetos que la aplicación maneja y se utilizan para mapear los datos provenientes de la API a objetos Kotlin.

- **navigation**: contiene la lógica necesaria para gestionar la navegación entre pantallas de la aplicación. Dentro de ella se encuentra *AppNavigation*, que es el encargado de manejar las rutas y transiciones entre los distintos fragmentos o pantallas. Además, incluye el *BottomNavBar*, que proporciona una barra de navegación inferior intuitiva para que el usuario/a pueda cambiar entre las secciones principales de la aplicación rápidamente.

- **util**: contiene funciones auxiliares y utilitarias que se utilizan en diversas partes de la aplicación. Incluye funciones para gestionar los formatos de fechas, lo que permite transformar las fechas de la API en formatos fácilmente comprensibles y legibles por el usuario.

- **views**: contiene todos los archivos .kt responsables de las pantallas de la aplicación. Cada archivo de esta carpeta representa una pantalla o una sección de la interfaz de usuario, construida con *@Composable*. Las vistas definen la UI y gestionan las interacciones del usuario, y cada una está asociada a una parte específica de la aplicación, como la pantalla de subastas principales, la de perfil de usuario, o la de detalles de una obra.

- **MainActivity**: es el punto de entrada principal de la aplicación. Desde aquí, se llama a *BrushNBid*, que a su vez invoca *AppNavigation*. Esto establece la navegación inicial y permite que el usuario acceda a las diferentes secciones de la aplicación. MainActivity actúa como el contenedor general que coordina el flujo de la aplicación y mantiene la estructura básica de la navegación.

Esta organización permite una clara separación de responsabilidades, facilitando tanto el desarrollo como la escalabilidad. Al contar con una estructura modular, el código es más fácil de mantener, probar y extender.

4.3.4. Funcionalidad de la app usando ciencia de datos

Se ha integrado la ciencia de datos a través de la funcionalidad de recomendación de pujas dentro de una subasta, cuando se consultan sus detalles (Figura 10), con el fin de ayudar a las personas usuarias que puedan tener dudas acerca de cuánta cantidad pujar.

Este proceso ha consistido en emplear técnicas de aprendizaje automático para predecir/recomendar una cantidad de puja en una subasta concreta.

Figura 10: Ejemplo de recomendación de puja en emulador Android

La función, definida en la API en *MLController*, se encarga de recibir el ID de la subasta como parámetro, obteniendo los datos de la subasta actual y los datos históricos del resto de subastas que han sido adjudicadas y tienen pujas válidas.

Después, utiliza un modelo de regresión lineal multivariada, importado de *ml-regression-multivariate-linear* (ml.js, 2021), para generar una recomendación de puja. Este modelo es entrenado con los datos mencionados en el párrafo anterior, considerando factores como el tipo de obra, el autor/a el precio inicial, el incremento de la puja, el número de pujas realizadas, la puja mínima y la máxima, etc.

También se incluye una validación para que la recomendación de la puja sea al menos el valor de la puja actual más el incremento mínimo permitido por la subasta.

5. Conclusiones

A lo largo de este proyecto, se ha desarrollado una aplicación móvil destinada a facilitar las subastas de obras de arte, especialmente para artistas noveles y compradores interesados en acceder a obras originales. La aplicación integra diversas tecnologías para proporcionar una plataforma accesible y eficiente tanto para los artistas como para los compradores.

La arquitectura basada en el patrón *Modelo-Vista-ViewModel* (MVVM) ha permitido una clara separación de responsabilidades, facilitando la escalabilidad y el mantenimiento de la aplicación. El uso de tecnologías modernas como *Kotlin*, *Jetpack Compose*, y *Retrofit* para el frontend, junto con *Node.js*, *Express* y *PostgreSQL* para el backend, ha permitido crear una solución robusta y eficiente.

Además, la integración de funciones de ciencia de datos a través de un modelo de recomendación de pujas basado en aprendizaje automático ha añadido un valor significativo a la aplicación, ofreciendo a los compradores recomendaciones informadas y mejorando su experiencia de usuario.

La base de datos relacional en PostgreSQL asegura la integridad de los datos, mientras que la API RESTful expone los recursos de forma eficiente y segura. La interfaz de usuario ha sido diseñada con un enfoque en la accesibilidad y la usabilidad, garantizando una experiencia fluida para los usuarios, tanto artistas como compradores.

El sistema de subastas, la gestión de obras y pujas, y la recomendación de pujas son las funcionalidades clave que permiten a los usuarios participar activamente en el proceso de compra y venta de arte. Este enfoque ha permitido que la plataforma sea flexible y fácil de usar, cumpliendo con los objetivos de democratizar el acceso al arte y proporcionar una experiencia interactiva para todos los participantes.

Finalmente, este proyecto demuestra el potencial de las aplicaciones móviles para transformar el mercado del arte, ofreciendo nuevas oportunidades para los artistas noveles y mejorando la accesibilidad para los compradores. Las futuras iteraciones de la aplicación podrían centrarse en la mejora de la funcionalidad de la inteligencia artificial y en la expansión de la plataforma a otros tipos de arte y usuarios, lo que ampliaría aún más su impacto en el sector.

6. Bibliografía

Figma. (2021). Figma: Collaborative Interface Design Tool. https://www.figma.com/

Google. (2018). Instructivo de Jetpack Compose. *Android Developer Blog*. https://developer.android.com/develop/ui/compose/tutorial?hl=es-419

Google. (2021). Android Studio: Official IDE for Android Development. https://developer.android.com/studio

Google. (2025). Gson: A Java serialization/deserialization library to convert Java Objects into JSON and back. https://github.com/google/gson

JetBrains. (2021). Kotlin Programming Language. https://kotlinlang.org/

Martin, R. C. (2011). *Clean Code: A Handbook of Agile Software Craftsmanship* (1st). Prentice Hall.

Microsoft. (2021). TypeScript Documentation. https://www.typescriptlang.org/docs/

ml.js. (2021). Regression Multivariate Linear. https://github.com/mljs/regression-multivariate-linear

Node.js. (2021). Node.js Documentation. https://nodejs.org/en/docs/

OpenJS Foundation. (2021). Express - Node.js web application framework. https://expressjs.com/

PlantUML contributors. (2024). PlantUML: Open-source tool to draw UML diagrams. https://plantuml.com/

Square. (2021). Retrofit: Type-safe HTTP client for Android and Java. https://square.github.io/retrofit/

Swagger. (2021). *Swagger: OpenAPI Specification*. https://swagger.io/specification/

The PostgreSQL Global Development Group. (2020). PostgreSQL: The World's Most Advanced Open Source Relational Database. https://www.postgresql.org/

Visual Paradigm. (2025). Visual Paradigm: Herramienta de modelado y diseño para desarrollo de software. https://www.visual-paradigm.com/