DOCUMENTO FUNCIONAL DEL PROYECTO

ALUMNO/GRUPO: Joel Lozano Barbancho DAM2A

1. Introducción y contexto

El objetivo de este proyecto es desarrollar un videojuego 2D *pixel-art* del género Idle, con mecánicas de pesca y funcionalidades sociales (chat y mercado entre jugadores), accesible desde PC y dispositivos móviles.

La motivación principal surge de la falta de propuestas que integren la progresión incremental característica de los juegos Idle con la interacción en tiempo real entre usuarios, ofreciendo así una experiencia más completa y atractiva.

El público objetivo está formado por jugadores casuales que buscan un entretenimiento accesible, con la posibilidad de mantener su progreso en la nube y continuar la partida desde cualquier plataforma. Para dar respuesta a esta necesidad, se plantea la creación de un sistema que permita a los usuarios registrarse, guardar sus datos de manera segura y disfrutar de un entorno de juego que fomente tanto la progresión individual como la interacción comunitaria.

La solución propuesta consiste en un juego de pesca en el que el usuario puede capturar peces mediante un sencillo sistema de clics, coleccionarlos, venderlos para obtener recursos o intercambiarlos con otros jugadores. Todo ello se complementa con un chat global en tiempo real y un mercado común, lo que convierte la experiencia en algo más que un simple juego Idle: un espacio social y dinámico en el que cada jugador contribuye a la comunidad.

2. Análisis de requisitos

Dentro de este apartado se detallarán los requisitos, tanto funcionales como no funcionales, que el videojuego *Fish Collector: Idle* deberá ofrecer, desde el registro y guardado en la nube hasta la mecánica de pesca, el inventario, el chat y el mercado de intercambio. Finalmente se analizarán aquellas restricciones que pueden influir en el desarrollo de este proyecto.

2.1. Requisitos funcionales (RF)

Código	Descripción
RF1	El sistema permitirá registrar nuevos usuarios
RF2	Los usuarios podrán iniciar sesión y recuperar su progreso de su última sesión de juego
RF3	El jugador podrá pescar peces mediante clics/taps en la pantalla
RF4	El sistema contará con un menú que mostrará inventario, recursos y mejoras disponibles
RF5	El jugador podrá vender peces para obtener recursos
RF6	El jugador podrá almacenar peces en su inventario para coleccionismo/intercambio
RF7	El sistema incluirá un chat global en tiempo real
RF8	Los jugadores podrán realizar ofertas de intercambio en un mercado común
RF9	El sistema gestionará misiones, eventos y recompensas para desbloquear mejoras
RF10	El juego incluirá un minijuego de pesca con dificultad variable según la rareza del pez

2.2. Requisitos no funcionales (RNF)

Código	Descripción
RNF1	La interfaz será accesible desde PC y dispositivos móviles
RNF2	El guardado en la nube deberá ser seguro y confiable
RNF3	El chat deberá funcionar en tiempo real con baja latencia
RNF4	El sistema deberá ser capaz de soportar al menos 500 usuarios concurrentes
RNF5	La interfaz debe ser intuitiva y fácil de utilizar

2.3. Restricciones

El desarrollo del proyecto estará condicionado por una serie de limitaciones técnicas y organizativas que marcarán el alcance y la planificación del trabajo. En primer lugar, se establece como requisito obligatorio el uso del motor **Godot** junto con su lenguaje **GDScript**, ya que se trata de una herramienta open-source especialmente adecuada para la creación de videojuegos 2D en pixel-art. Asimismo, la gestión del servidor, la base de datos y la autenticación de usuarios se implementará mediante Firebase, lo que permitirá disponer de guardado en la nube y comunicación en tiempo real sin necesidad de mantener infraestructura propia.

A pesar de contar con multitud de recursos técnicos (documentación, tutoriales, etc.), el proyecto se desarrollará con un tiempo limitado, lo que obliga a priorizar las funcionalidades esenciales y a organizar el trabajo en fases bien definidas. Se utilizarán herramientas de apoyo como GitHub para el control de versiones, Figma para el diseño de la interfaz y Taiga para la gestión de tareas, lo que facilitará la organización y el seguimiento del progreso.

Por último, existen dependencias técnicas que condicionan el funcionamiento del sistema: será imprescindible contar con conexión a internet para la sincronización de datos y el uso del chat, y se deberá garantizar la compatibilidad multiplataforma para que el juego pueda ejecutarse tanto en ordenadores como en dispositivos móviles. Estas restricciones, aunque limitan el margen de acción, también proporcionan un marco claro que orienta el desarrollo hacia soluciones realistas y alcanzables.

3. Análisis de usuarios y roles

Podemos dividir los roles que interactúan con el juego en dos roles diferentes: **jugador** y **visitante**. Cabe destacar la ausencia de un rol de administrador, ya que la gestión técnica se realiza fuera del ámbito del videojuego (vista de desarrollador) en lugar de formar parte de la experiencia del usuario.

Rol	Descripción	Permisos principales
Jugador	Usuario registrado que juega, colecciona y comercia.	Pescar, vender, coleccionar, chatear, intercambiar.
Visitante	Usuario no registrado	Acceso limitado al menú principal. Puede pasar a ser Jugador

4. Casos de uso

Definiremos los diferentes casos de uso como situaciones concretas que se pueden dar dentro del videojuego cuando un jugador interactúa con el sistema.

Caso de uso	Actor principal	Descripción	Resultado esperado
Registrar usuario	Visitante	El visitante puede crear una cuenta introduciendo sus datos, pasando a ser jugador y habilitando el guardado de datos en la nube	Usuario registrado correctamente en la base de datos de Firestore
Iniciar sesión	Jugador	Jugador introduce sus credenciales y accede al juego	Acceso concedido y progreso cargado
Recuperación de contraseña	Jugador	El juego envía un mensaje de recuperación de contraseña a la dirección de correo especificada por el Jugador	Contraseña actualizada exitosamente
Pescar pez	Jugador	Jugador clica la pantalla y, tras esperar un poco, realiza un pequeño minijuego para pescar	Captura de pez exitosa o fallida
Consultar Inventario	Jugador	Jugador abre su inventario para revisar sus peces capturados	Menú de inventario abierto y peces mostrados de forma correcta
Vender pez	Jugador	Jugador selecciona un pez desde el inventario y lo vende para obtener recursos	Recursos obtenidos
Comprar mejoras	Jugador	Jugador compra una mejora desde el menú de mejoras	Mejora comprada y aplicada exitosamente
Cambiar de escena	Jugador	Jugador cambia de escena, en la que podrá pescar nuevos peces	Cambio de escena exitoso
Ofrecer intercambio	Jugador	Jugador publica una oferta de intercambio	Oferta visible y transacción posible

Aceptar intercambio	Jugador	Jugador acepta una oferta de intercambio y se lleva a cabo la transacción	Transacción exitosa

5. Modelo de datos

Teniendo en cuenta el carácter de este proyecto, utilizar una base de datos relacional tradicional como **MySQL** o **PostgreSQL** no resulta la opción más adecuada. Aunque este tipo modelo de datos ofrece sus ventajas, también implica una estructura más rígida y menos eficiente para aplicaciones que requieren sincronización en tiempo real, flexibilidad de datos y escalabilidad dinámica. Por ejemplo, en una base relacional, obtener el inventario de un jugador implicaría realizar múltiples consultas: una para obtener los datos del jugador, otra para recuperar los ítems de su inventario, y una más para acceder a la información de cada pez. Este tipo de operaciones, que dependen de relaciones entre tablas, pueden ralentizar el rendimiento en entornos móviles o en juegos que requieren actualizaciones constantes y rápidas.

Firebase ofrece una arquitectura basada en documentos que permite almacenar datos embebidos, lo que significa que es posible guardar toda la información relevante de un jugador (inventario, mejoras, etc) dentro de un único documento. Esto facilita lecturas rápidas, reduce la complejidad de las consultas y mejora la experiencia del usuario al permitir actualizaciones en tiempo real. También permite modificar la estructura de los documentos sin necesidad de realizar migraciones complejas y ofrece reglas de seguridad específicas por documento o colección, lo que simplifica el control de acceso a los datos de cada jugador.

5.1. Estructura de datos

- **Usuario:**

```
{
  "username": "Joel",
  "email": "joel@example.com",
  "coins": 1200,
  "createdAt": "2025-10-17T15:30:00Z"
}
```

- **Pez:**

```
{
```

```
 "id": "salmon",
 "name": "Salmon",
 "rarity": 1,
 "value": 50,
}
```

- **Pez (capturado):**

```
{
 "playerId": "abc123",
 "fishId": "salmon",
 "rarity": "1",
 "quantity": 5
}
```

- **Mejoras:**

```
{
 "id": "rod_upgrade",
 "name": "Fishing Rod Upgrade",
 "cost": 300,
 "effect": "faster fishing"
}
```

- **Mejoras (Compradas):**

```
{
 "playerId": "abc123",
 "upgradeId": "rod_upgrade",
 "level": 2
}
```

- **Mensajes:**

```
{
```

```
 "playerId": "abc123",
 "message": "¡Hola mundo!",
 "timestamp": "2025-10-17T15:31:00Z"
}
```


- **Intercambios:**


```
{
 "senderId": "abc123",
 "receiverId": "xyz789",
 "fishId": "salmon",
 "quantity": 1,
 "rarity": 1,
 "timestamp": "2025-10-17T15:32:00Z"
}
```


- **Misión:**


```
{
 "id": "mission_001",
 "title": "Captura 5 salmones",
 "description": "Demuestra tu habilidad pescando 5 salmones.",
 "objective": {
  "type": "capture",
  "target": "salmon",
  "quantity": 5
 },
 "reward": {
  "coins": 0,
  "items": []
  "unlockUpgradeId": "rod_upgrade"
 },
 "playerId": "abc123",
 "status": "active",

}
```

6. Diseño de la interfaz

El diseño de la interfaz tiene como objetivo ofrecer una experiencia visual clara, atractiva y funcional, adaptada tanto a ordenadores como a dispositivos móviles. La estructura de navegación se ha planteado de forma intuitiva, permitiendo al jugador acceder fácilmente a las distintas secciones del juego sin perder de vista el núcleo de la experiencia: la pesca.

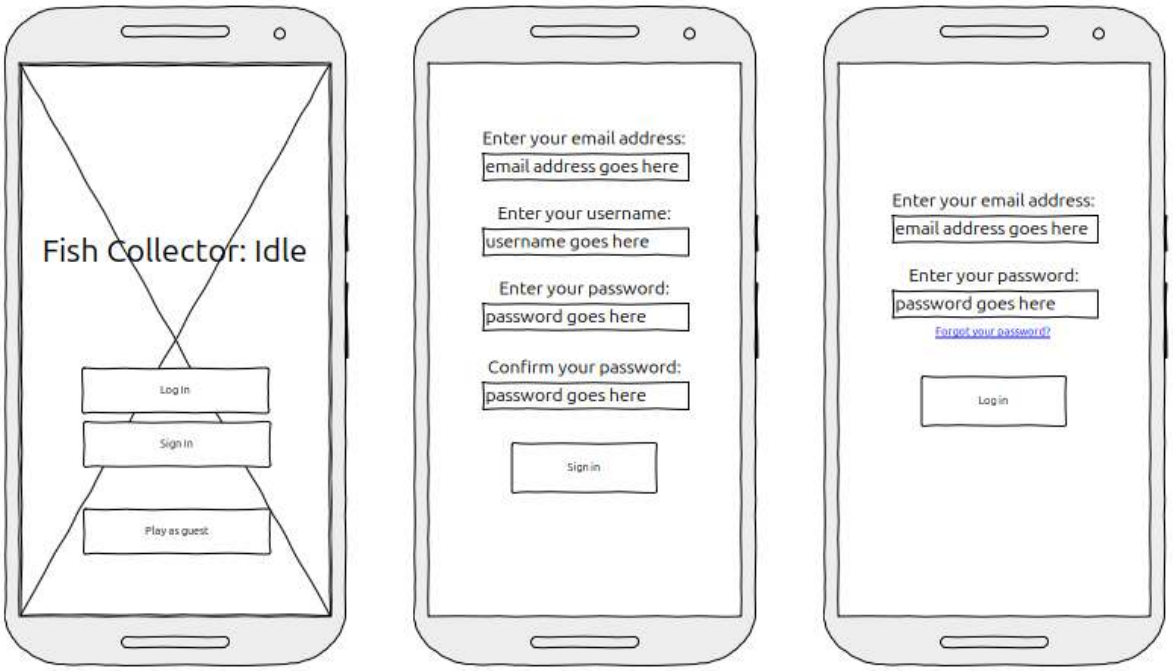
La pantalla principal será la escena de pesca, donde el jugador podrá interactuar directamente con el entorno mediante clics o toques en la pantalla. Esta escena incluirá elementos visuales que representen el mar, el personaje y los peces, así como indicadores de recursos y un inventario que muestre las mejoras disponibles y los peces capturados. Desde esta pantalla se activará el minijuego de pesca, cuya dificultad variará según la rareza del pez capturado.

El inventario contará con una interfaz organizada por categorías, permitiendo al jugador consultar las mejoras adquiridas, los peces almacenados y otros datos como estadísticas, logros, etc. Desde aquí se podrá vender peces, gestionar el progreso y revisar a las misiones activas, además de acceder al menú de opciones del juego.

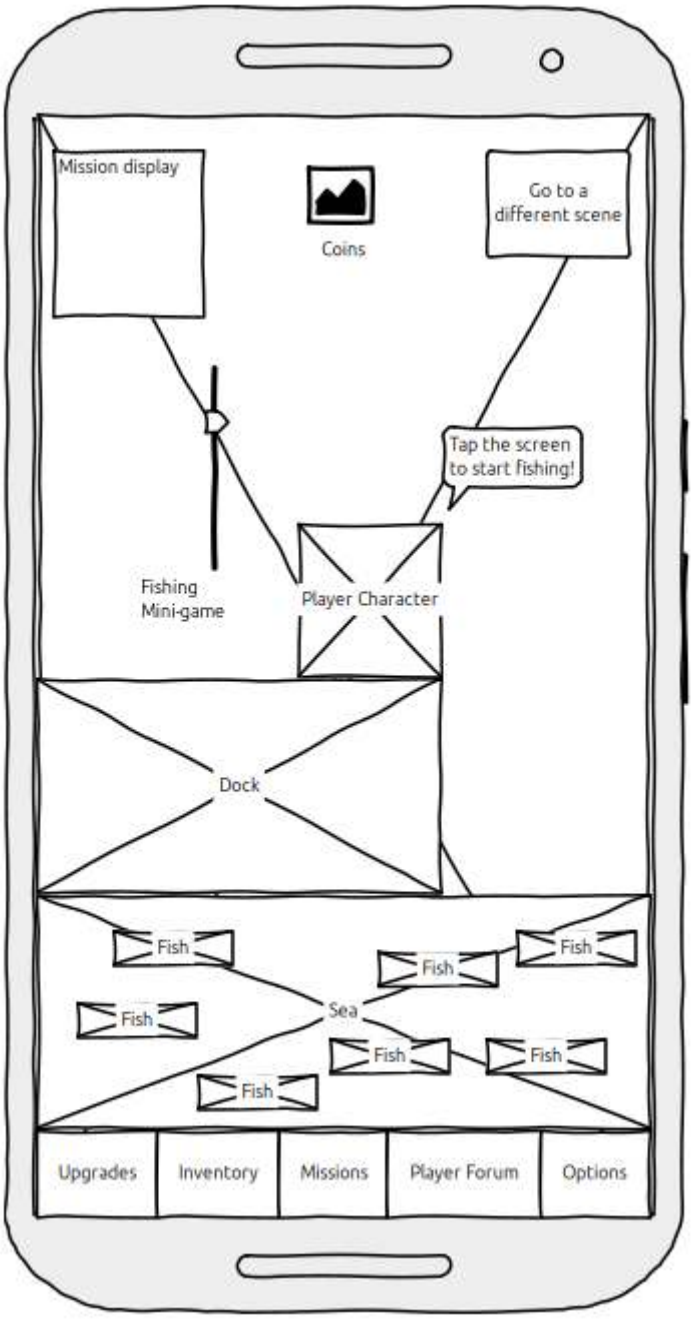
La sección de chat global estará integrada en una pantalla dedicada, con un diseño que favorezca la lectura fluida y la participación en tiempo real. Además, se incluirá una pestaña para el mercado de intercambio, donde los jugadores podrán publicar ofertas y consultar las propuestas de otros usuarios.

Cada pantalla estará vinculada a uno o varios casos de uso definidos en el análisis funcional, y se diseñará utilizando herramientas como **Figma** para garantizar coherencia visual y facilidad de navegación. El diseño buscará equilibrar la estética *pixel-art* con la funcionalidad, asegurando que el jugador pueda disfrutar de una experiencia inmersiva sin complicaciones técnicas.

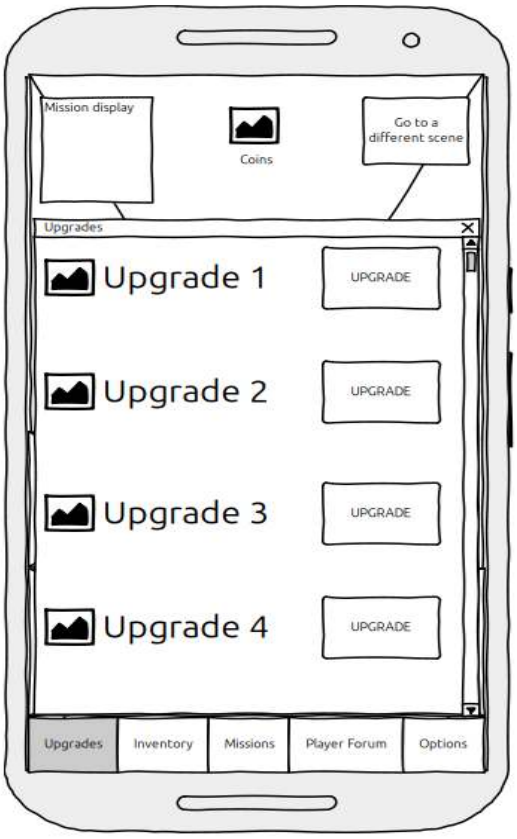
6.1. Menú principal

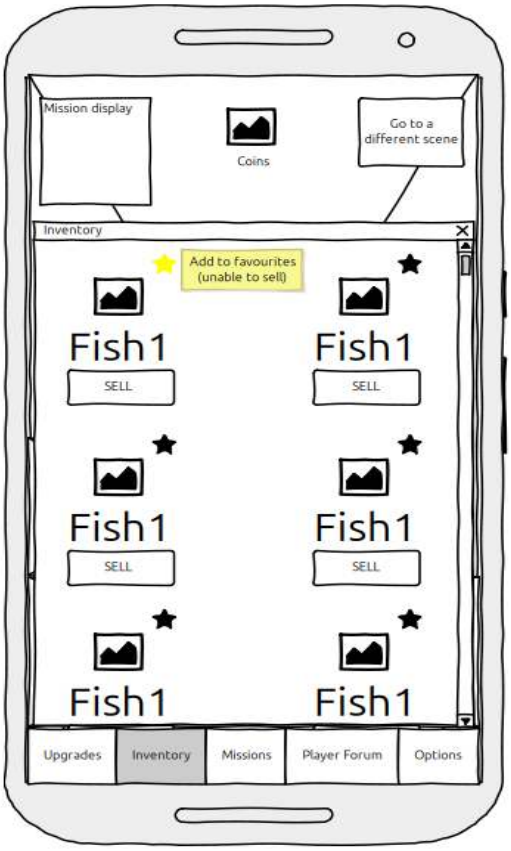


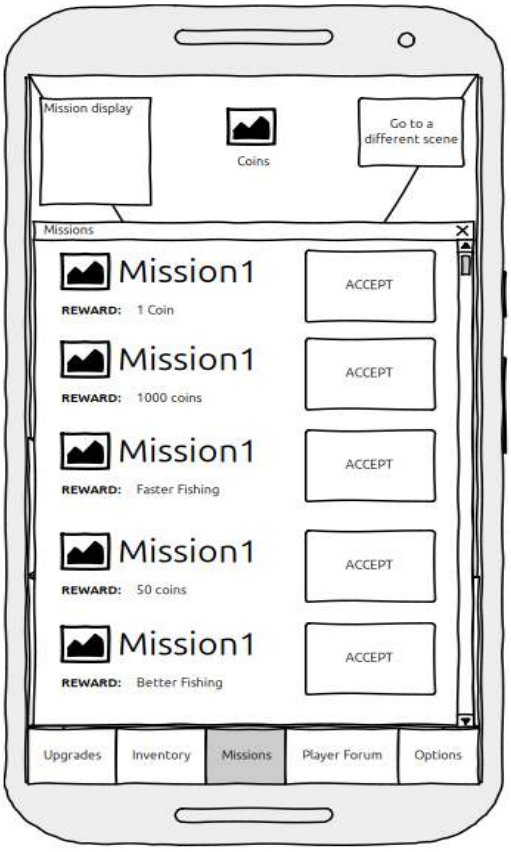
6.2. Escena principal

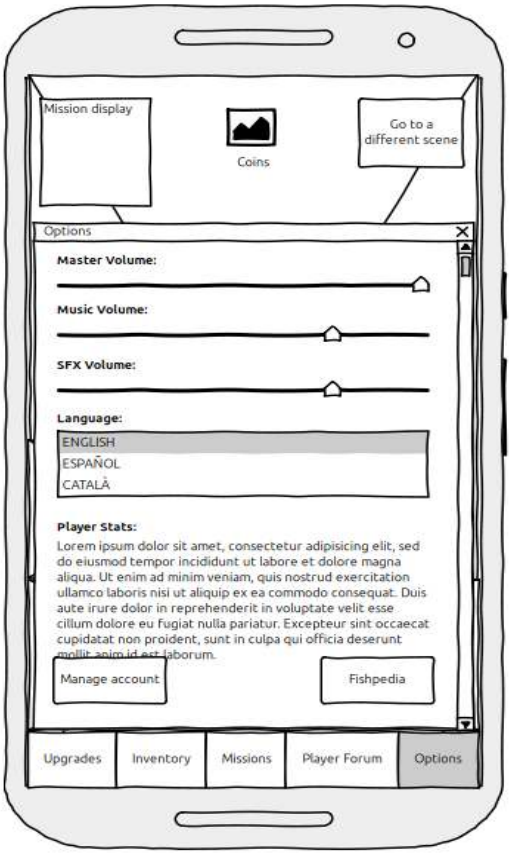


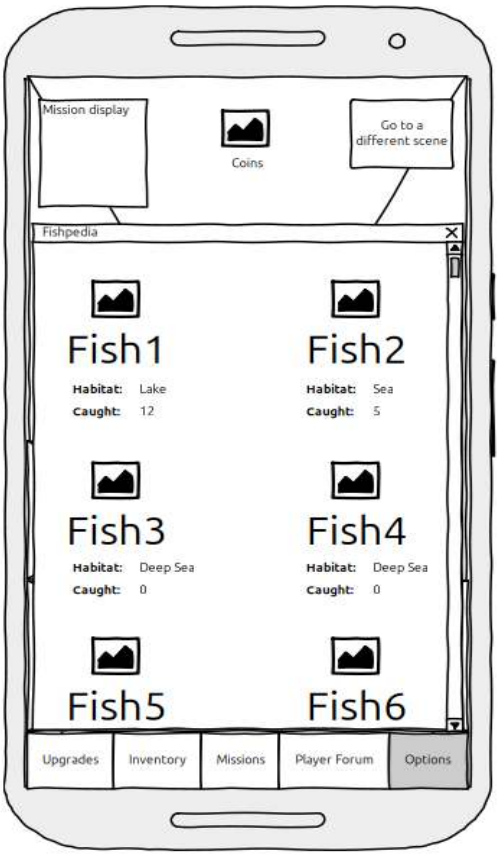
6.3. Menús de juego

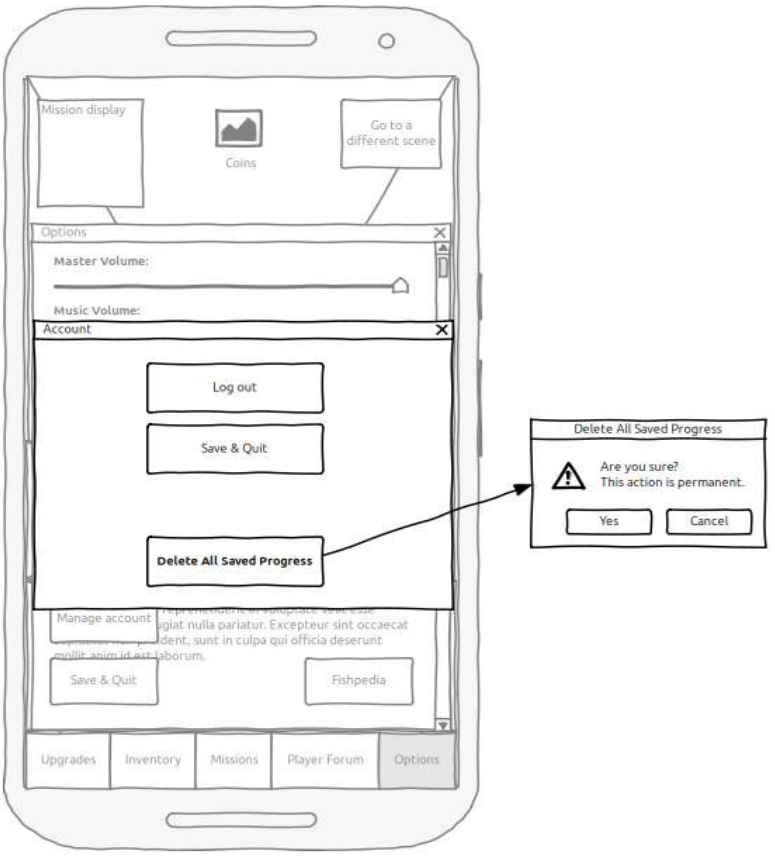




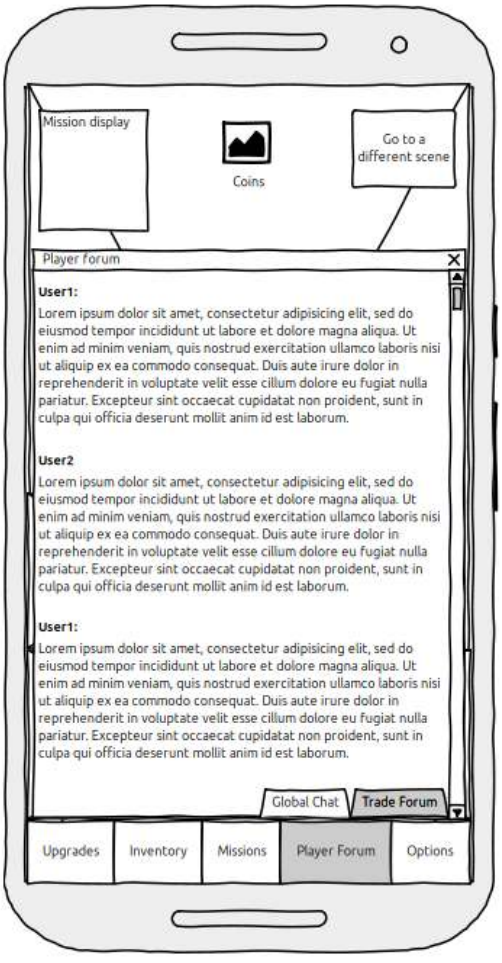


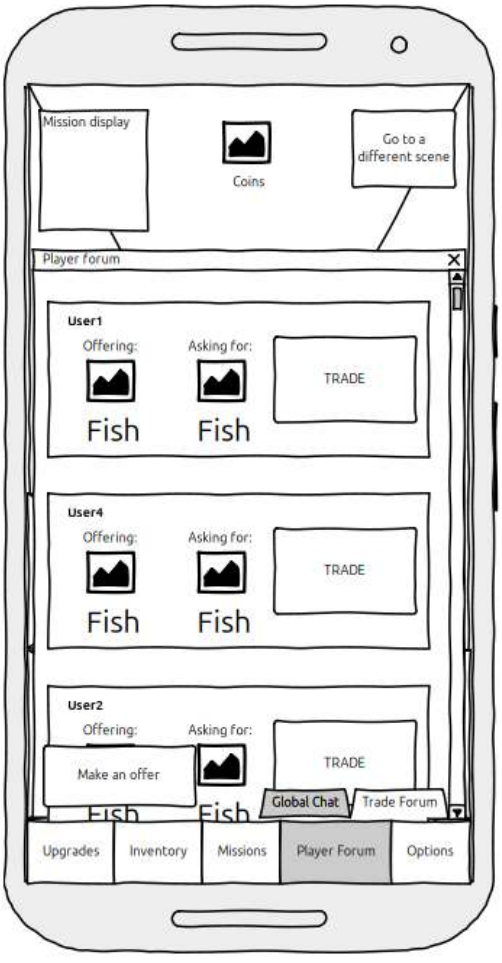






6.3. Foro de jugadores





7. Planificación técnica

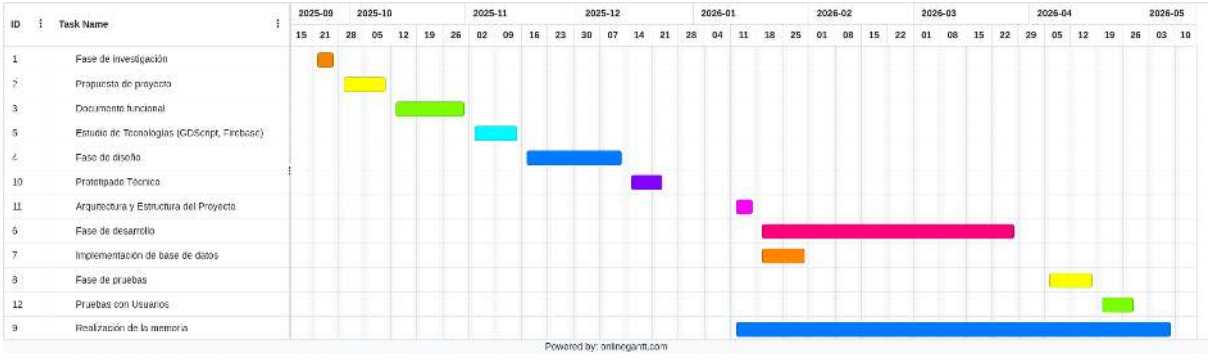
Para el desarrollo del videojuego se utilizará el motor **Godot**, una herramienta open-source que permite trabajar con gráficos en 2D y que emplea el lenguaje **GDScript**, ideal por su simplicidad y enfoque específico para videojuegos. Esta elección responde a la necesidad de crear una experiencia visual atractiva en *pixel-art*, compatible con múltiples plataformas.

En cuanto al servidor y la gestión de datos, se ha optado por **Firebase**, una solución que ofrece servicios de autenticación, almacenamiento en la nube y comunicación en tiempo real. Gracias a su integración con **Google Cloud**, se evita la necesidad de montar y mantener servidores propios, lo que facilita el desarrollo y reduce los costes operativos.

El proyecto también se apoyará en diversas herramientas complementarias: **GitHub** se utilizará para el control de versiones y la colaboración en el código; **Figma** servirá para diseñar la interfaz gráfica del juego; **Modelio** permitirá crear diagramas UML que representen la estructura del sistema; y **Taiga** se empleará como plataforma de gestión de tareas, siguiendo una metodología ágil que favorezca la organización y el seguimiento del progreso. Además, **Google Drive** se utilizará como repositorio de documentación, apuntes y referencias.

La organización del trabajo se dividirá en etapas que abarcan desde el diseño inicial hasta la implementación y validación. En las primeras semanas se abordará el diseño de la interfaz y el modelo de datos, seguido por la programación del núcleo de juego y la integración de funcionalidades como el inventario, las mejoras y el sistema de misiones. Posteriormente se desarrollará el chat global y el mercado de intercambio entre jugadores. Finalmente, se realizarán pruebas funcionales y de rendimiento para asegurar la estabilidad y calidad del producto.

7.1. Diagrama de Gantt



8. Análisis de riesgos

8.1. Identificación de riesgos

Durante el desarrollo del proyecto, es fundamental anticipar los posibles riesgos que podrían comprometer el cumplimiento de los objetivos o afectar la calidad del producto final. Uno de los principales riesgos identificados es la falta de tiempo, especialmente considerando que se trata de un proyecto con múltiples funcionalidades y un equipo reducido. Una planificación deficiente o la acumulación de tareas pueden provocar retrasos significativos.

Otro riesgo relevante está relacionado con los problemas técnicos, como incompatibilidades entre herramientas o dificultades en la integración de Firebase con el motor Godot. Estos inconvenientes pueden surgir en cualquier fase del desarrollo y requerir soluciones rápidas para evitar bloqueos.

También se considera la posibilidad de pérdida de datos, ya sea por errores en la sincronización con la nube o por fallos en el sistema de guardado.

Por último, existe la posibilidad de que surjan cambios en los requisitos del proyecto, ya sea por nuevas ideas, ajustes técnicos o feedback recibido durante las pruebas, lo que obligaría a replantear parte del desarrollo.

8.2. Valoración y respuesta

Riesgo	Probabilidad	Impacto	Plan de prevención o contingencia
Falta de tiempo	Alta	Alta	Dividir tareas y fijar entregas intermedias
Problemas técnicos con Firebase	Media	Media	Probar integración temprana
Pérdida de datos	Baja	Alta	Control de versiones
Cambios en los requisitos	Media	Media	Usar metodología ágil y mantener documentación flexible y actualizada.

9. Validación y criterios de éxito

La validación del proyecto se llevará a cabo mediante una serie de pruebas y criterios que permitirán comprobar si el sistema cumple con los objetivos establecidos. Para considerar el proyecto como exitoso, será imprescindible que los jugadores puedan registrarse correctamente, iniciar sesión desde distintos dispositivos y recuperar su progreso gracias al sistema de guardado en la nube. Además, el núcleo de juego (centrado en la pesca, la gestión de recursos y el coleccionismo) deberá funcionar de forma estable y fluida.

Otro aspecto fundamental será la interacción entre jugadores. El sistema de chat en tiempo real y el mercado de intercambio deberán permitir una comunicación efectiva y segura, sin interrupciones ni errores. Para ello, se realizarán pruebas funcionales que verifiquen cada caso de uso, así como pruebas de rendimiento que evalúen la capacidad del sistema para soportar múltiples usuarios simultáneos.

También se contemplarán pruebas de usabilidad, tanto en ordenadores como en dispositivos móviles, con el fin de garantizar una experiencia intuitiva y accesible.

10. Conclusión

El análisis funcional realizado para *Fish Collector: Idle* permite definir con claridad los objetivos del proyecto, las funcionalidades clave y las tecnologías que se emplearán durante su desarrollo.

Se ha optado por utilizar Godot como motor de juego y Firebase como solución para el servidor y la base de datos, lo que garantiza una infraestructura sólida y escalable. El sistema incluirá mecánicas de pesca, progresión incremental, coleccionismo y comunicación entre jugadores, ofreciendo una experiencia completa y atractiva.

Este proyecto busca crear un videojuego funcional, además de servir como ejercicio de aprendizaje en el desarrollo de videojuegos 2D, la gestión de datos en la nube y la implementación de funcionalidades en tiempo real.

Éste análisis proporciona una visión clara y realista del proyecto. A partir de aquí, comienza la fase de desarrollo, cuyos próximos pasos serán proceder a preparar el entorno de desarrollo, crear la estructura inicial del repositorio y la base de datos, y comenzar la implementación de los casos de uso prioritarios.