INSTITUT
PUIG CASTELLAR

FIGHTING FURRY

CFGS Desarrollo de Aplicaciones Multiplataforma

ADRIA ABRIL
AITOR SANCHO
NAOMI HIDALGO

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Agradecimientos

Nos gustaría dedicar este proyecto a muchas personas, sobre todo a los profesores que nos han enseñado, guiado y ayudado durante estos dos años. En especial nos gustaría dedicárselo a Gerard, sin él este proyecto no habría salido adelante.

Gracias por ayudarnos y aguantarnos después de enviar emails con memes de gatitos y ocasionar errores cada vez más difíciles.

Te deseamos la mayor suerte y felicidad en tu nueva etapa.

Datos del proyecto

Datos del proyecto	
Título	*Fighting Furry*
Autores	Àdria Abril - Naomi Hidalgo - Aitor Sancho
Directores	Gerard Falcó - Daniel Martínez
Breve descripción	Videojuego multijugador de combate por turnos a través del uso de cartas.

Resumen del proyecto

La idea del proyecto es la de realizar un videojuego multijugador de combate por turnos con estética retro, muy similar a los juegos de la época de los 90.
En lugar de ser el típico juego machacabotones, se ha buscado desde *DesplazaosStudio*™, darle un toque más actual, con el uso del multijugador y toques de estrategia, ya que las batallas se disputan a través del uso de cartas.

El objetivo que se ha buscado o el fin con el que se ha creado este juego es el de revivir aquellas batallas que se realizaban con los amigos al salir del colegio y conectábamos las consolas de la época o las reuniones familiares con los primos y pasábamos horas y horas divirtiéndonos intentando derrocar al campeón del día.

Una vez se inicie el servidor, que puede iniciar cualquiera de los dos usuarios que vayan a jugar, o un usuario que no vaya a jugar y que solo le limite a inicializar el servidor, los usuarios podrán elegir personaje con el que jugar entre un plantel de 6 personajes distintos entre sí.
Una vez la elección de personajes se empezará la batalla, y a través del uso de cartas y a medida que avanzan los turnos deberán dejar a 0 la vida del rival.

Durante estos dos meses la organización se ha realizado mediante un diagrama de *Gantt*, la metodología de *SCRUM* en el espacio que nos proporciona *Trello* para la organización de los integrantes de *DesplazaosStudio*™.

El objetivo final sería llegar a conseguir un prototipo de juego funcional, el cual funcione de manera online y plantearse si la idea se puede llegar a publicar en una plataforma online de videojuegos.

- **Palabras clave:**

videojuego, multijugador, combate, cartas, turnos, estrategia

Índice

PARTE I - Gestión del proyecto

1. Introducción

Fighting Furry se desarrolla con la idea principal de crear un juego que sea atractivo visualmente con estética pixel y añadiendo algo muy importante en la actualidad, como es el juego online, el juego en la red. Por ello gran parte del trabajo realizado en este proyecto ha sido usar la tecnología que permite la conexión de los programas clientes a un servidor, para así, otorgar a los usuarios la experiencia de disfrutar de *Fighting Furry* de manera online.

Para ello se ha planteado desarrollar un programa servidor, el cual es el encargado de la comunicación entre ambos clientes, el cual hará de host para *Fighting Furry* y simplemente con la dirección IP de dicha máquina y el puerto al que conectarse ambos clientes podrán realizar la conexión al mismo.

El desarrollo del mismo se ha realizado mediante lenguaje *Java* en combinación con la tecnología *WebSocket*, todo ello creado desde cero.

La comunicación entre los programas clientes y el servidor como se ha comentado, se realiza a través de *WebSocket*, pero esta comunicación consiste en él envió de mensajes entre ambos programas.
El cliente A envía un mensaje al servidor, pero este mensaje ha de ser convertido a *Json* para que el servidor entienda el mensaje, así que se han creado dos conversores para realizar dicha tarea. Un conversor convierte el mensaje enviado desde Java a *Json* y el servidor para enviar la respuesta al cliente B, envía el mensaje en *Json* y el conversor lo transforma a Java para que el cliente B reciba este mensaje y lo lea sin problemas.
El mensaje ha de contener la información necesaria, no hay porqué introducir valores que no afecten al desarrollo del juego.

El programa cliente se ha desarrollado con *Java* y *LibGDX*, un framework muy potente el cual nos permite el desarrollo de videojuegos multijugador y como valor añadido es que permite el funcionamiento del juego desarrollado en distintas plataformas, escritorio, web y móvil.
LibGDX se integra perfectamente en *Java* lo que ha permitido el uso ágil de este framework y trabajar lo más a fondo posible los conocimientos adquiridos durante este tiempo por *DesplazaosStudio*™.
Una vez creado el servidor, los usuarios que usen el programa cliente únicamente deberán introducir la dirección IP y el puerto para la conexión, una vez establecida la conexión de ambos clientes ya podrán empezar a jugar una vez elegido el personaje y, por último, el juego finaliza cuando uno de los dos usuarios acabe con la vida a 0.

1.1. Contexto

Actualmente hay una gran tendencia a la creación de juegos por parte de desarrolladoras nuevas o pequeñas que no pueden competir con grandes compañías establecidas hace años en el sector y con un alto poder monetario que les permiten realizar los tan sonados juegos triple A.
Estos juegos se denominan juegos Indie, juegos independientes, que pueden presentar sus juegos gracias a plataformas como *Steam* o *Gog* que permiten la publicación de los mismos cumpliendo una serie de requisitos.
Tras ver el éxito y gran auge en el mercado de este tipo de juegos, este proyecto se ha desarrollado con dicho fin y con la misma idea aplicando las tecnologías que se han considerado más óptimas para el desarrollo.

1.2. Justificación

El motivo principal para el desarrollo de dicho proyecto es que los integrantes son habituales jugadores de videojuegos, amantes de los gatos, de ahí que tengan gran importancia en el plantel de jugadores, así como en los ataques de los mismos, y uno de los integrantes de *DesplazaosStudio*™, vivió muy de cerca esas batallas que nunca acababan entre amigos o primos en una *Super Nintendo* o *Mega Drive*, videoconsolas de la época noventera junto a grandes juegos como *Street Fighter*, *Mortal Kombat* entre otros, de ahí llega una gran referencia para la parte visual de *Fighting Furry*.

Mucho de este proyecto se basa en comprobar la dificultad que entraña realizar un servidor dedicado y portable en cualquier plataforma para poder disfrutar de *Fighting Furry* en cualquier momento.

2. Objetivos

Para hacer posible este proyecto se han planteado una serie de objetivos para hacer posible su realización y finalización.

- Objetivos

  · Aprender a crear un videojuego desde cero con *LibGDX*
  · Crear un servidor operativo
  · Configuración de servidor
  · Aprender la metodología y el uso de *WebSockets*
  · Creación de programa cliente
  · Configuración de programa cliente
  · Estética visual atractiva

· Portabilidad total de servidor y cliente
· Manual de instrucciones vía web
· Utilizar metodologías ágiles durante el proyecto (Diagramas de *Gantt*, *SCRUM*, *Trello*)
· Finalización del proyecto en el tiempo disponible

2.1. Posibles obstáculos

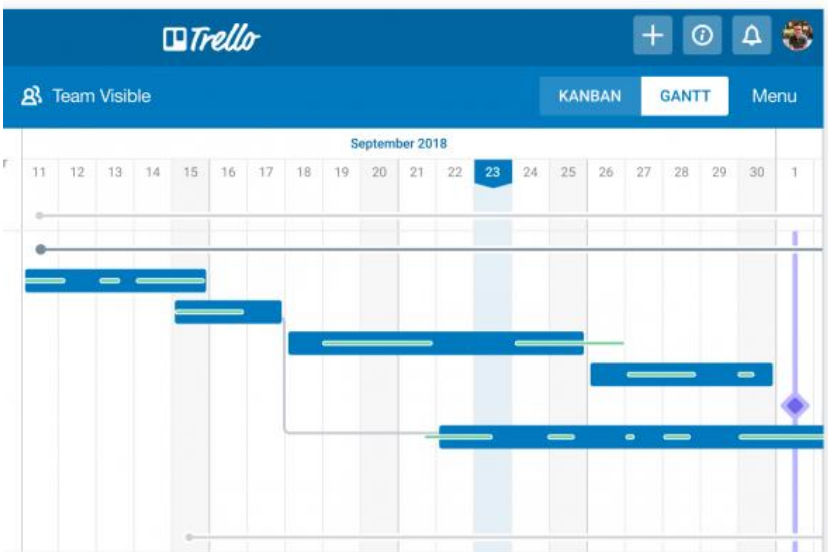
Por otro lado, *DesplazaosStudio*™ ha sopesado los posibles obstáculos a encontrar durante el desarrollo de *Fighting Furry.*

Entre ellos se encuentran los problemas que puede acarrear la implantación de un framework desconocido por parte de los desarrolladores y que supone un hándicap importante y un gran incremento del tiempo necesario para finalizar el proyecto.

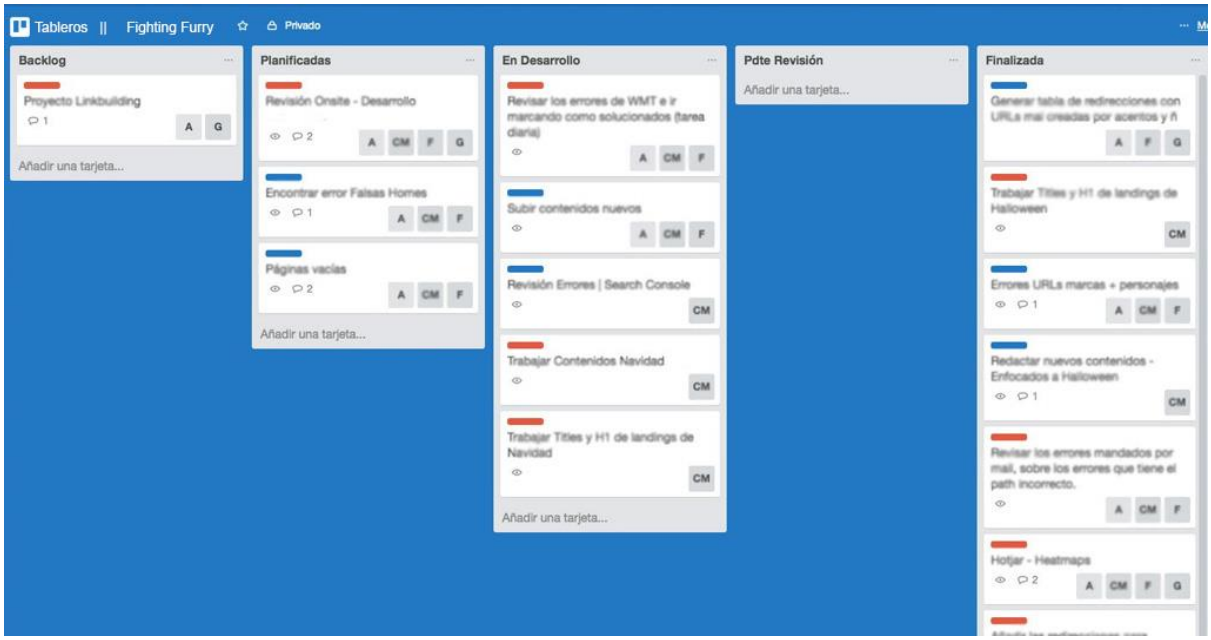
Desde *DesplazaosStudio*™ y tras un análisis de la situación se ha considerado centrarse en los objetivos principales y aparcar de manera temporal diversas funcionalidades que ya tienen una base creada para el uso futuro, como puede ser una API o BBDD.

2.2. Metodología y herramientas de seguimiento

La planificación será mediante un diagrama de Gantt el cual hará de guía para el desarrollo de *Fighting Furry*, para finalizar dicho proyecto en el tiempo estipulado y llevando una organización correcta y ordenada.



La metodología utilizada será *SCRUM*, mediante el servicio que nos proporciona *Trello*, un importante gestor de proyectos online que entre muchas de las funciones que aporta, una muy importante es una pizarra la cual permite una organización más eficiente y visible para todos los integrantes del proyecto.



Esto unido a la planificación creada con el diagrama de *Gantt*, ayuda a mantener un nivel de trabajo adecuado cumpliendo con fechas y ver de una manera clara los puntos más laboriosos o problemáticos que nos hemos encontrado a lo largo de todo el desarrollo.

3. Presupuesto

Este apartado tiene como objetivo proporcionar una medida de los costes de la realización del proyecto. Esta estimación servirá para valorar las necesidades y oportunidades de la realización del proyecto y estimar los recursos económicos necesarios en el término de realización del proyecto.

3.1. Elementos de la estimación inicial

Para llevar a cabo una estimación inicial se han considerado una serie de elementos para el desarrollo del proyecto, asimismo, la estimación es flexible, debido a posibles actualizaciones, cambios durante el desarrollo, costes variables incalculables a la hora de realizar la estimación inicial y obstáculos que se puedan producir y se han de tomar en cuenta.

Los elementos considerados para la estimación inicial son los siguientes:

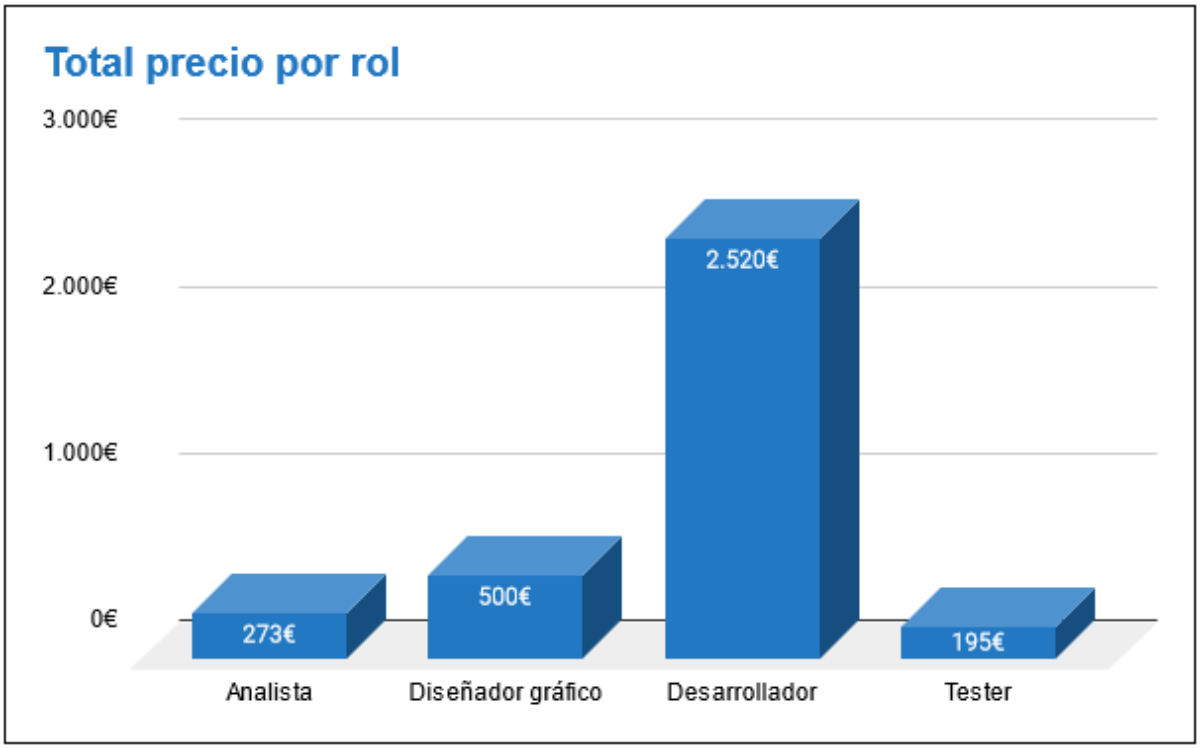
- Empleados:
  - 3x Analista junior
  - 2x Diseñador gráfico junior
  - 3x Desarrollador junior
  - 3x Tester

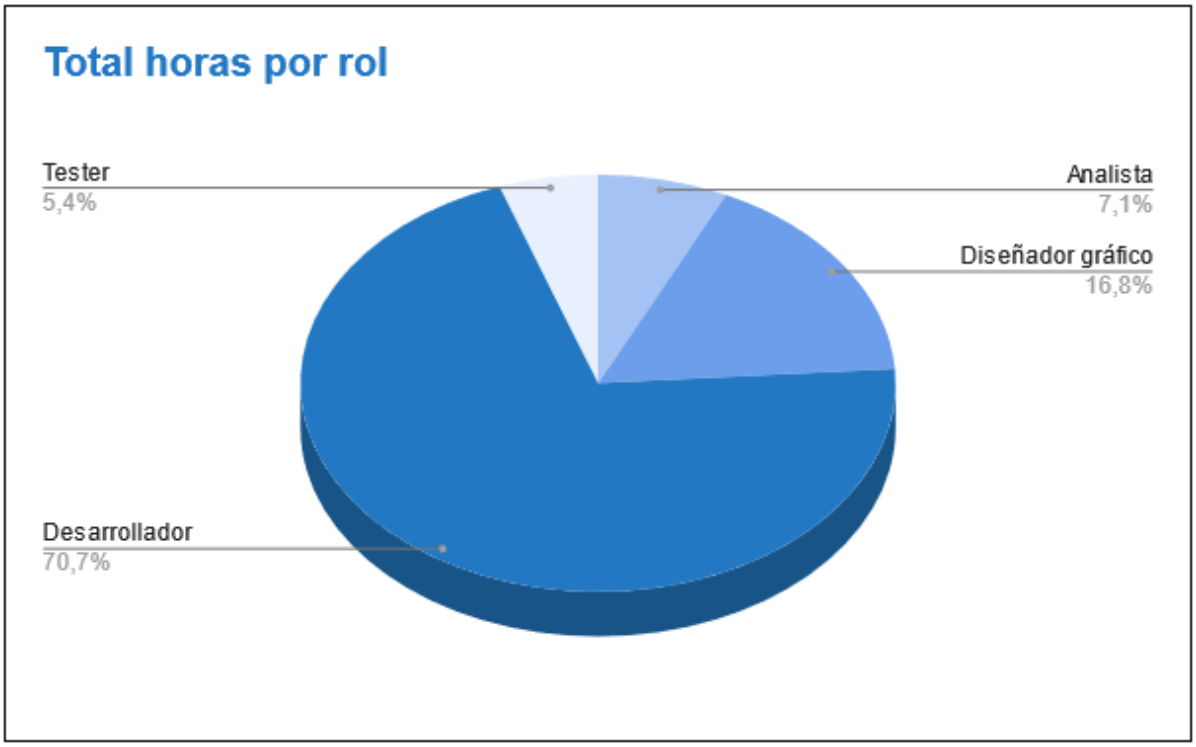
- Otros gastos:
  - Publicación en plataforma digital (*Opcional)

- Presupuesto:

Presupuesto estimado			
	Horas	**Precio €/h**	**Total**
Analista junior A	7h	13€/h	91€
Analista junior B	7h	13€/h	91€
Analista junior C	7h	13€/h	91€
Diseñador gráfico junior A	25h	10€/h	250€
Diseñador gráfico junior B	25h	10€/h	250€
Desarrollador junior A	70h	12€/h	840€
Desarrollador junior B	70h	12€/h	840€
Desarrollador junior C	70h	12€/h	840€
Tester A	5h	13€/h	65€
Tester B	5h	13€/h	65€
Tester C	5h	13€/h	65€
TOTAL	**297h**		**3.488€**
*Opcional: Publicación			100€
TOTAL + PUBLICACIÓN			**3.588€**

*En dicho presupuesto se incluye el equipo y licencias necesarias para el desarrollo



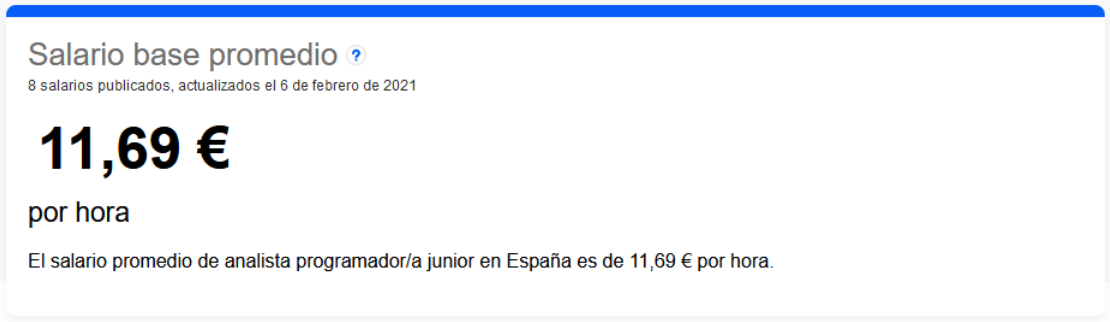


3.2. Justificación de los costes estimados

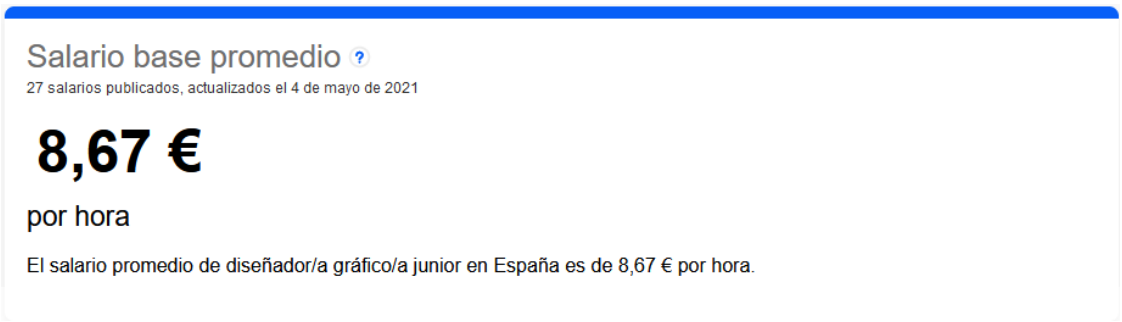
El salario de los diferentes roles del proyecto, se ha asignado tras contrastar diferentes páginas webs que aportan este tipo de información y como coste opcional se ha consultado la publicación del proyecto una vez finalizado en una plataforma de juegos.

indeed es uno de los muchos portales que nos facilitan el promedio base de los distintos roles necesarios para llevar a cabo este proyecto.

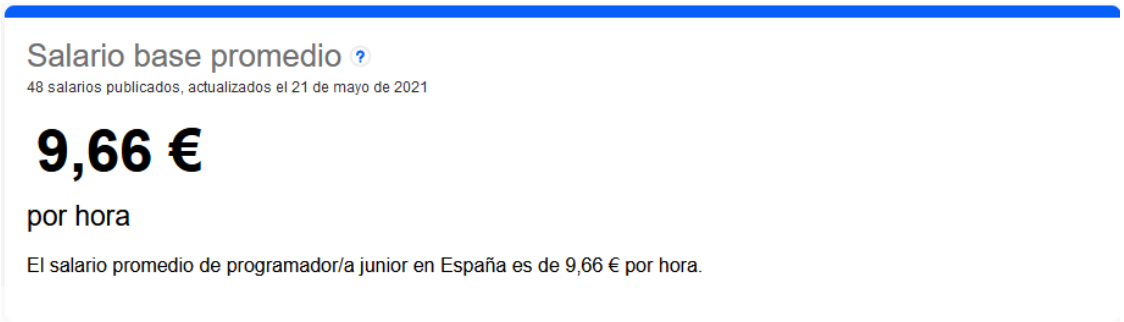
- Analista junior:



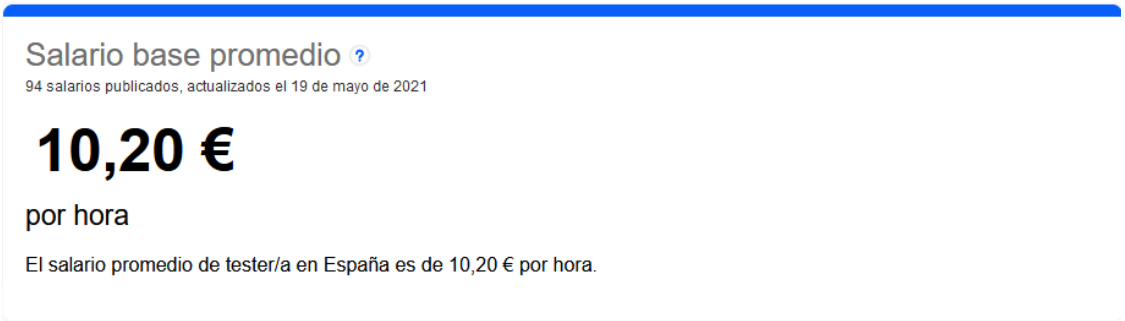
- Diseñador gráfico junior:



- Desarrollador junior:



- Tester:



- Publicación de *Fighting Furry*:

	PRECIO
Publicación	100€

** Tras la publicación del juego, la plataforma devolverá este importe de publicación inicial al alcanzar la cifra de 1000€ en ventas del juego en la propia plataforma.*

3.3. Plan de control de presupuesto

**La información de este apartado es únicamente teórica. Únicamente pretende reflejar lo que se realizaría en una situación real.*

Para llevar a cabo el control del presupuesto y los flujos de caja, se han establecido cinco fechas de control en las que se realizarán ingresos y pagos. Estas fechas son:

1) Se inicia el proyecto con el 20% del coste total del mismo.
2) Tras la segunda reunión con el cliente, debemos tener en caja la parte correspondiente al trabajo realizado hasta la fecha.
3) Reunión con el cliente y finalización de la primera fase del proyecto, donde se muestran diseños y se cobrará el trabajo realizado hasta el momento.
4) A mitad del tiempo de finalización del proyecto, habiéndonos reunido con el cliente en varias ocasiones para ultimar detalles se volverá a cobrar y se iniciará la fase de desarrollo.
5) Al finalizar el proyecto se cobrará el resto correspondiente a la parte trabajada dando por finalizado el presupuesto y el proyecto.

3.4. Valoración de la viabilidad económica-financiera

Una vez contrastado datos y estimaciones, se ha considerado que realizar un proyecto de estas características y dimensiones es únicamente viable si el equipo de desarrolladores son los que impulsan la idea y poseen los recursos necesarios para llevarla a cabo, de otra forma, para llevar a cabo este proyecto haría falta una gran inyección de capital inicial.

4. Planificación temporal

La planificación temporal describe el inicio y final del proyecto, agrupando las tareas en diferentes fases que se organizan a lo largo del tiempo e indica las decisiones tomadas respecto a las desviaciones detectadas en el punto de control.

4.1. Fases del proyecto

1- Definición del proyecto

· Analizar las necesidades a cubrir
· Analizar la viabilidad del proyecto
· Analizar e instalar el software necesario

2- Planificación y diseño del proyecto

· Elección de proyecto a desarrollar
· Desarrollar el documento y estrategia de plan de trabajo
· Desglose de tareas a realizar

3- Gestión del proyecto

· Tareas de seguimiento
· Control de actividades a realizar
· Gestión de desviaciones
· Controlar impactos en el resultado final

4- Gestión de la documentación

· Definir la gestión del proyecto de forma documental
· Producción y gestión de la documentación

5- Gestión de la calidad final del proyecto

· Testear el producto final del proyecto
· Seguimiento de resultados
· Contrastar idea final con resultado final del proyecto

4.2. Planificación inicial

En la siguiente tabla se especifica el inicio y final del proyecto, así como sus horas totales. Se indican las diferentes fases en las que se dividirá el proyecto y la propia organización.

	Horas Planificadas
Proyecto integrado	297
Bloques de Fases previas al desarrollo	9
Tareas previas a la aplicación	3
Tareas previas a la aplicación	3
Tareas previas a la aplicación	3
Aplicación	242
Análisis	12
Requisitos	3
Objetivos	4
Tecnología	5
Diseño	45
Arquitectura - Blueprint	8
Interface – Mockups	30
Persistencia - Diseño conceptual	7
Cookies	0
Desarrollo	180
Estrategia de desarrollo	5
Frontend (Gimp, LibGDX, Photoshop, Figma, Java)	60
Backend (LibGDX, WebSocket, Java)	70
BBDD (PostgreSQL)	40
Diagrama de despliegue	5
Pruebas	5
Usabilidad	5
Validación W3C (HTML, CSS, Mobile)	0
Google PageSpeed	0
Web	0

Audiovisual	7
Integración y Transición	15
Integración	15
web + app + audiovisual + marketing +...	0
Transición (del entorno de desarrollo en el entorno de producción)	0
Memoria	12
Presentación	12

4.3. Punto de control

En el punto de control se estudia el estado del proyecto y se toman decisiones en consecuencia del estado del mismo.

Todas estas decisiones o cambios quedan reflejados en la planificación final del proyecto.

4.3.1. Cambios respecto a la planificación inicial

Respecto a la planificación inicial se ha destinado mayor tiempo de horas a campos como el *backend* y *frontend* ya que está causando más problemas de los esperados en un inicio.
Este hecho está haciendo retrasar e invertir más horas de las esperadas en según qué punto del proyecto, lo que quiere decir que se tendrán que reajustar las horas planificadas y aparcar puntos del proyecto menos importantes.
Al encontrarse problemas hasta el punto actual de desarrollo también se están destinando más horas a las pruebas de código, ya que se está utilizando el *módulo en V* para testear el proyecto.

4.3.2. Consecuencias de los cambios respecto a los objetivos y desarrollo del proyecto

Las consecuencias y decisiones tomadas durante el desarrollo del proyecto han hecho que en el punto de control los integrantes hayan sopesado aparcar decisiones que se habían tomado en el inicio de la planificación.

Por parte de los desarrolladores se eliminaron de la planificación en este punto las horas destinadas a audiovisual, consistía en la realización de un trailer de promoción de *Fighting Furry*.
Otro punto que se ha dejado aparcado momentáneamente por parte de los desarrolladores es la *BBDD* y *API*, la cual, está creada la base para un posterior desarrollo, pero durante el punto de control se aparcó para designar las horas planificadas a otros menesteres como el *backend* del desarrollo del proyecto.

4.3.3. Situación del proyecto en el punto de control

Actualmente el equipo de desarrollo se encuentra dividido en varios procesos.
Los desarrolladores están encargándose de la comunicación entre cliente y servidor a través de *WebSocket*.
Los diseñadores se encargan de conceptos de cartas, interfaz, etc.
Durante todo este proyecto se está realizando la documentación del mismo.

4.4. Planificación final

En la siguiente tabla se especifica el inicio y final del proyecto, así como sus horas totales y los cambios respecto a la planificación inicial.
Se indican las diferentes fases en las que se dividirá el proyecto y la propia organización.

	Horas Planificadas	Horas Reales	Estado
Proyecto integrado	297	297	0
Bloques de Fases previas al desarrollo	9	9	0
Tareas previas a la aplicación	3	3	0
Tareas previas a la aplicación	3	3	0
Tareas previas a la aplicación	3	3	0
Aplicación	242	263	21
Análisis	12	9	-3
Requisitos	3	3	0
Objetivos	4	3	-1
Tecnología	5	3	-2
Diseño	45	44	-1
Arquitectura - Blueprint	8	7	-1
Interface – Mockups	30	30	0
Persistencia - Diseño conceptual	7	7	0
Cookies	0	0	0
Desarrollo	180	200	20
Estrategia de desarrollo	5	5	0
Frontend (Gimp, LibGDX, Photoshop, Figma, Java)	60	60	0

Backend (LibGDX, WebSocket, Java)	70	100	30
BBDD (PostgreSQL)	40	10	-30
Diagrama de despliegue	5	5	0
Pruebas	5	10	5
Usabilidad	0	0	0
Validación W3C (HTML, CSS, Mobile)	0	0	0
Google PageSpeed	0	0	0
Web	0	0	0
Audiovisual	7	0	-7
Integración y Transición	15	1	-14
Integración	15	1	-14
web + app + audivisual + marketing +...	0	0	0
Transición (del entorno de desarrollo en el entorno de producción)	0	0	0
Memoria	12	16	4
Presentación	12	8	-4

5. Leyes y normativa

Tras consultar varios sitios webs y organismos oficiales, se ha declinado desde *DesplazaosStudio*™ el uso de ley o normativa que pueda afectar al uso de *Fighting Furry.*
Las leyes y normativas consultadas sobre los juegos online están enfocadas al uso de publicidad en los mismos y la regulación de juego enfocado a casinos y sitios de la misma índole.
Cabe destacar que durante el uso de *Fighting Furry* no se almacena ninguna información sobre el usuario, por lo que no se ha incluido la LOPD (Ley Orgánica de Protección de Datos de Carácter Personal).

PARTE II - Ejecución del proyecto

1. Análisis

1.1. Especificación de requisitos

En este apartado se detalla el análisis realizado sobre las necesidades y requisitos que debe cumplir *Fighting Furry* para considerarse finalizado en su totalidad.

1.1.1. Funcionales

Fighting Furry cuenta con los siguientes requisitos funcionales:

· **Creación e inicio de servidor**
Gracias a la tecnología que proporciona *WebSocket* hace posible la creación de un servidor dedicado, y, entre otras funcionalidades, una sesión de comunicación interactiva y bidireccional entre el programa cliente y el servidor.

· **Configuración de las reglas de entrada/salida del servidor**
Mediante el lenguaje de programación Java se detallan las reglas o mensajes de entrada que ha de recibir el servidor para así generar un mensaje o salida de datos acorde a estas entradas. Estas entradas y salidas generan las reglas de juego de *Fighting Furry*.

· **Creación del programa cliente y configuración**
El programa cliente realizado con *Java*/*LibGDX* consta de una interfaz gráfica la cual informa al usuario de la conexión establecida al servidor.
A continuación, una vez conectados los dos usuarios necesarios para la experiencia multijugador, el cliente accede a una pantalla de selección de personaje y confirmación del mismo.
Por último, una vez ambos hayan elegido el personaje deseado, comenzará la partida, en la cual, habrá una comunicación directa entre el programa cliente y servidor, y mediante el uso de mensajes(reglas) se realizarán las acciones que el usuario podrá ver a través del programa cliente.

· **Sincronización de mensajes cliente-servidor**
Una vez finalizada la configuración de ambas partes, cliente y servidor, el usuario a través del programa cliente mediante la interfaz gráfica realiza unas acciones las cuales en el *BackEnd* del programa enviará unos mensajes ya configurados para que los reciba el servidor y este, leerá estos mensajes, los refuta con las reglas preestablecidas y enviará la respuesta a los programas clientes.

1.1.2. No funcionales

· Estabilidad
El servidor contiene esta cualidad, la cual permite una conexión estable entre cliente-servidor, sin desconexiones durante el juego ni errores que hagan cerrar el servidor.

· Funcionalidad
Tanto el servidor como el programa cliente, funcionan tal como estaban especificadas en su desarrollo.

· Portabilidad
Fighting Furry ha sido diseñado para ejecutarse bajo distintos S.O. así como en plataformas móviles y navegadores web.

· Compatibilidad
El servidor puede estar situado en una máquina con distinto S.O. al de los programas clientes sin problemas de compatibilidad.

· Mantenibilidad
Al estar las reglas de juego en el servidor, estas se pueden editar sin necesidad de modificar los programas cliente.

2. Diseño

2.1. Arquitectura

En esta sección se define y describe la arquitectura de *Fighting Furry,* así como la comunicación entre los distintos elementos.

2.1.1. Descripción general

Fighting Furry tiene una arquitectura cliente-servidor. Esta arquitectura reparte las tareas entre el servidor y los clientes. El funcionamiento se basa en la realización de peticiones o envíos de mensajes a través del programa cliente hasta el servidor, este último, se encarga de comprobar el mensaje recibido, lo depura y retorna la respuesta correspondiente a dicho cliente.

En *Fighting Furry* el servidor se ha de comunicar con dos clientes, pero gracias a la tecnología *WebSocket* tenemos una conexión full-dúplex, lo que permite tener dos hilos, envío y recepción, para la comunicación entre programas.

Entrando en detalle sobre la arquitectura de *Fighting Furry*, consta de tres módulos principales: cliente, mensaje y servidor.

En el módulo cliente encontramos todos los assets necesarios para el funcionamiento del juego: sonidos, sprites de personajes, elementos de HUD, fondos de escenario, etc.
Más allá de assets, nos encontramos todas las clases que forman el programa cliente, donde encontramos la configuración de todos los actores que aparecen en pantalla, navegación a través de los botones, configuración de resolución de pantalla y un largo etcétera donde encontramos toda lógica y configuración necesaria para el funcionamiento de *Fighting Furry.*

A continuación, entra en juego el módulo mensaje, este actúa como puente entre la comunicación entre el programa cliente y el programa servidor.
Para situarnos, como ejemplo, el programa cliente tiene una clase llamada “Carta” en la cual hay una serie de variables como puede ser la descripción, el tipo, o la imagen de la misma.
Todas estas variables es información que el servidor no necesita para procesar la acción de dicha carta y aquí es donde entra en uso el módulo mensaje.
Este recibe la información justa y necesaria para enviar al servidor y este procese una respuesta en consecuencia.
Se podría resumir la acción de este módulo como un conversor de un mensaje complejo que envía el cliente a un mensaje más austero o con la información necesaria para el servidor.

Por último, el módulo servidor. Este módulo consta de las clases, así como la lógica y la configuración necesaria para responder a los mensajes entrantes y dar una respuesta acorde a dicho mensaje.
Se ha desarrollado una clase especifica donde se detallan todas las respuestas para los mensajes entrantes y desde este mismo módulo se da la opción en la clase principal a editar el puerto para la conexión de los programas clientes.

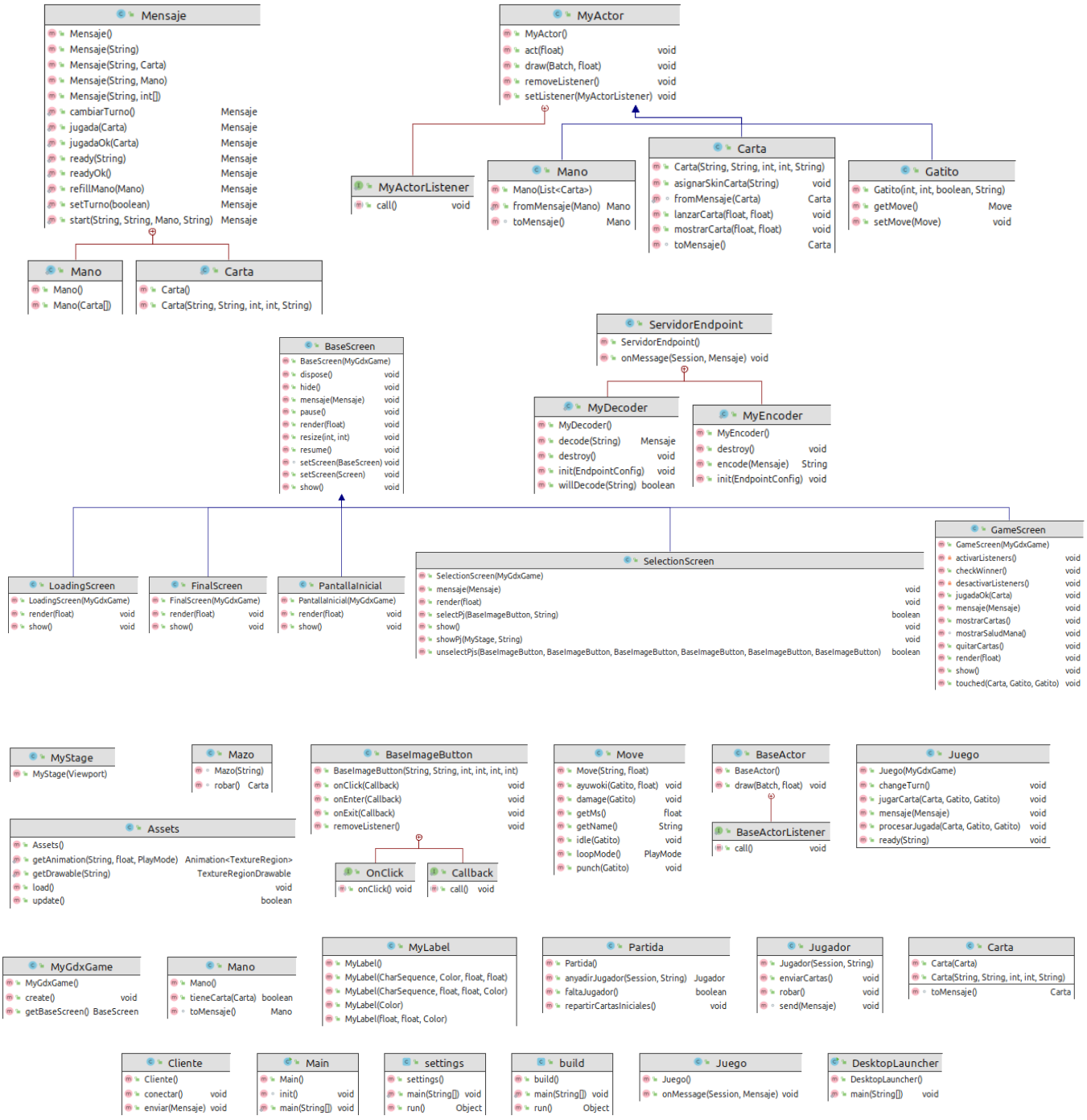
Para finalizar, la comunicación entre cliente-servidor se realiza a través de sockets, y para ello, toda la información que viaja entre ambos programas ha de hacerse en formato *Json* y para ello se han construido una serie de métodos encargados de dicha tarea.

2.1.2. Diagramas

En esta sección se describe la arquitectura del proyecto en tiempo de ejecución donde quedan reflejadas acciones, clases, métodos, etc., a través del uso de diagramas.

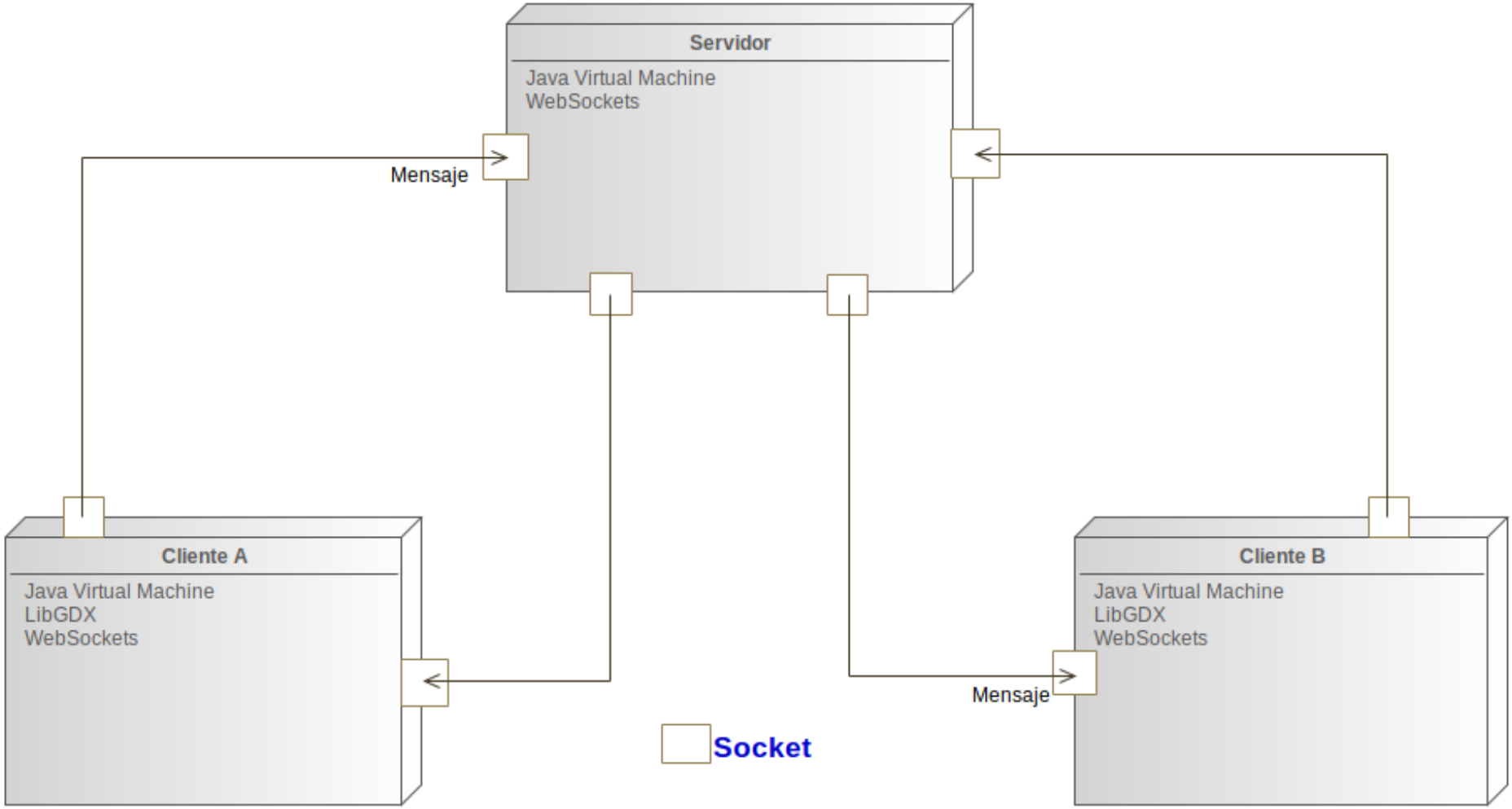
· Diagramas de clase

El diagrama de clase describe la estructura del proyecto mostrando clases, metodos y relaciones.



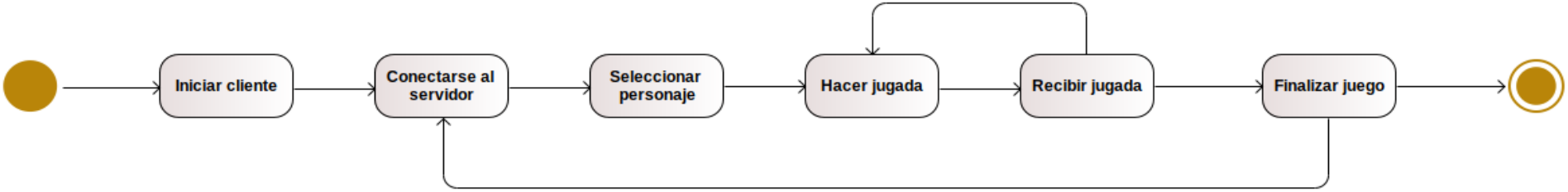
· Diagrama de despliegue

Este diagrama de despliegue muestra la arquitectura del sistema modelando la disposicion fisica de los artefactos(software) en nodos.

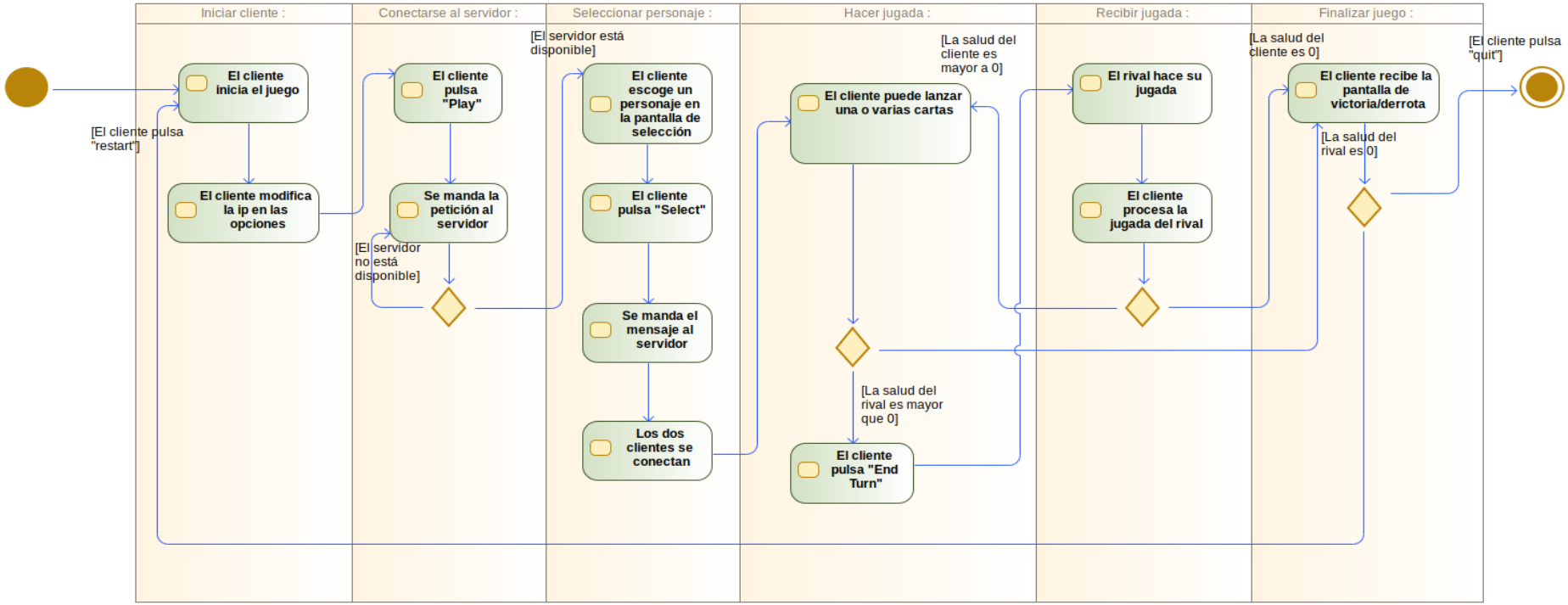


· Diagramas de estado y actividad

El siguiente diagrama de estado describe el comportamiento, posibles estados y los resultados de los eventos en el uso principal del cliente.

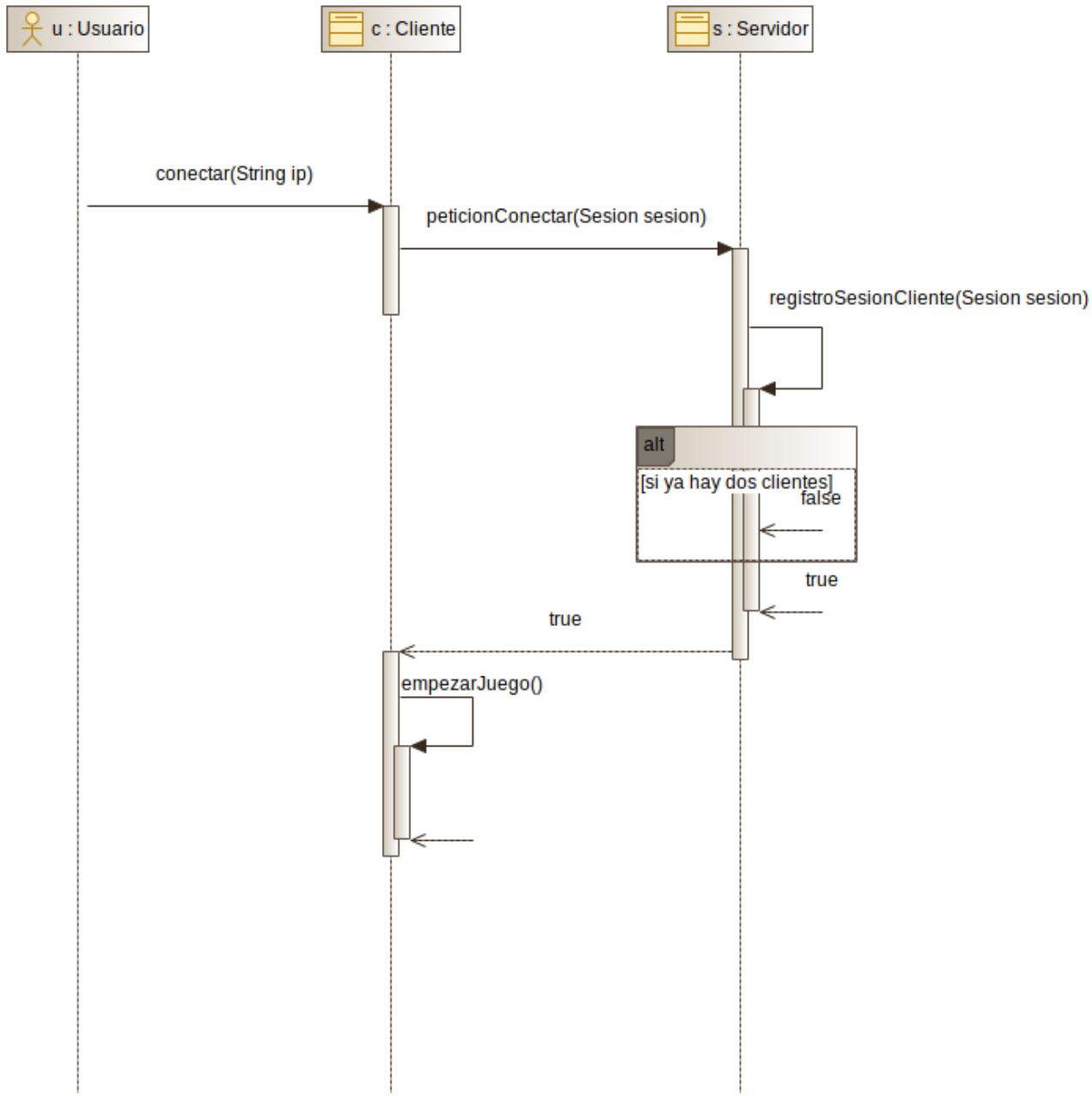


El diagrama de actividad es una espansión del diagrama de estado, donde en detalle se observan los comportamientos y eventos anteriores.

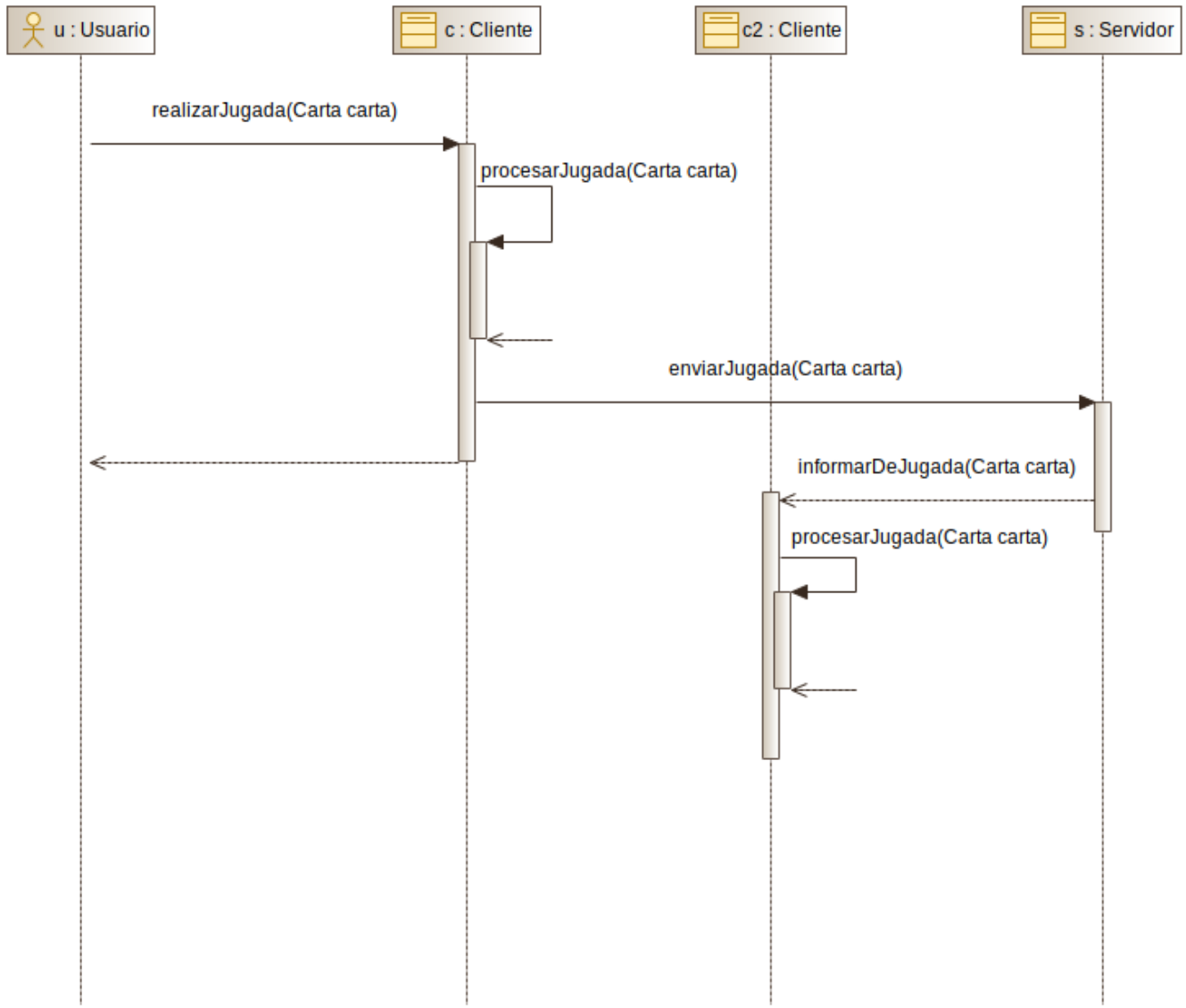


· Diagramas de secuencia

Los diagramas de secuencia muestran la interacción entre objetos.
En este primer diagrama se hace referencia a la conexión entre el programa cliente y el servidor.



En este segundo diagrama de secuencia se hace referencia a la comunicación entre cliente y servidor que se repetirá hasta la finalización de la partida.



Como podemos observar, los programadores han desarrollado la lógica de *Fighting Furry* en el programa cliente, esto se ha realizado para simplificar la comunicación entre el cliente y servidor.

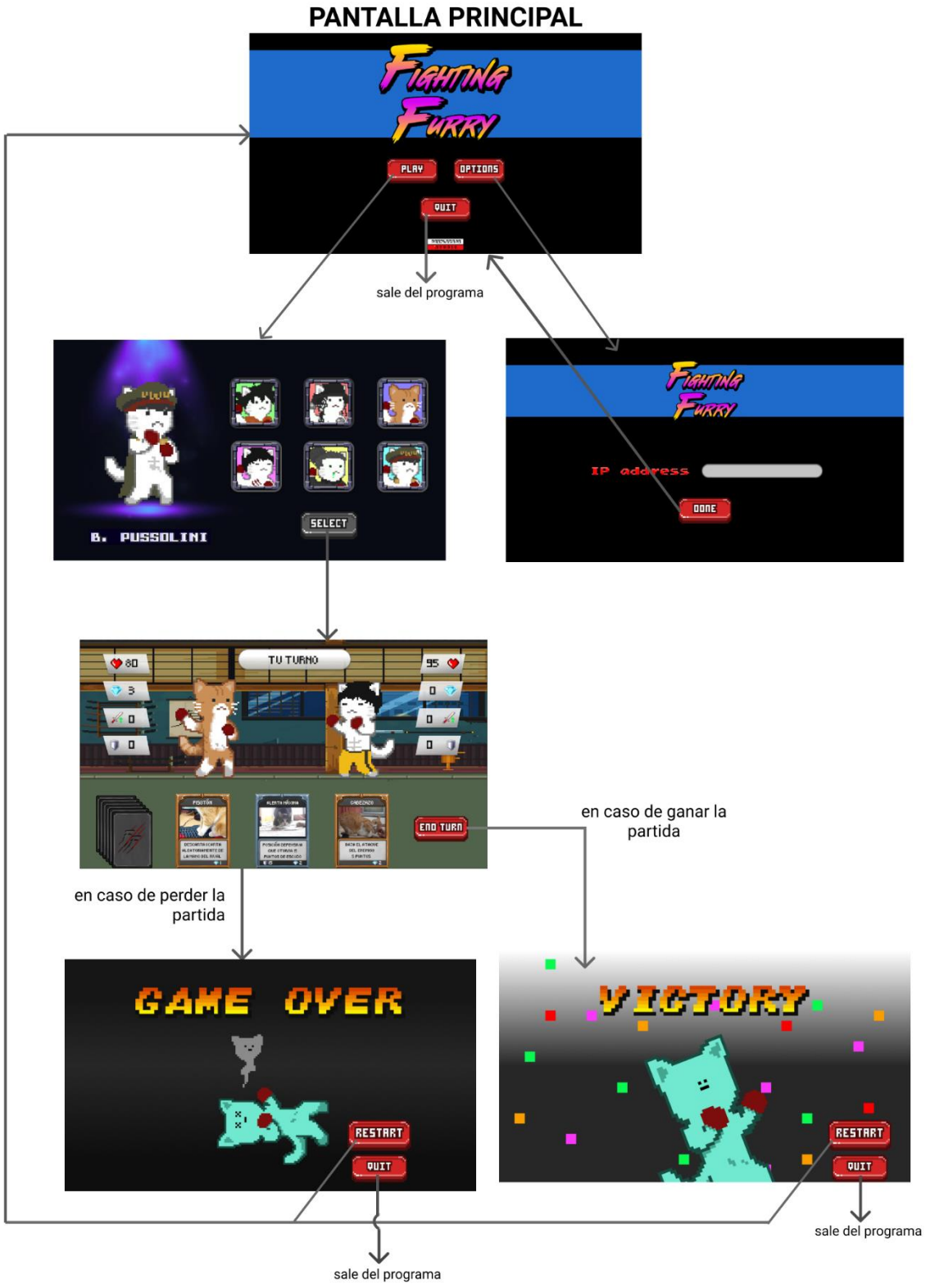
Esta elección tiene como punto positivo la simplificación de los mensajes enviados, pero, por el contrario, deja expuesta la seguridad del juego.

Pese a sacrificar el control sobre la seguridad, se ha optado por este modelo lógico por lo comentado anteriormente, la simplificación en la comunicación y el hecho de que para ejecutar y reproducir *Fighting Furry* se van a necesitar siempre un número de dos clientes.

2.2. Interfaz

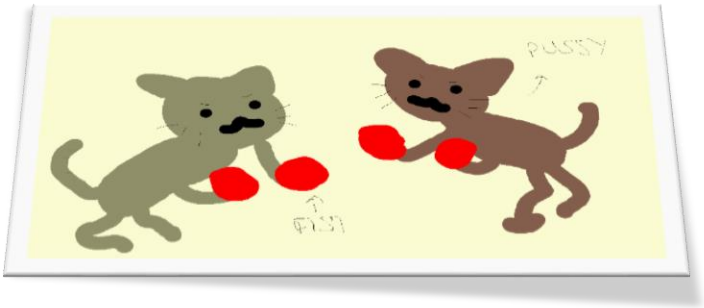
En esta sección quedan reflejados los elementos principales del diseño: El aspecto visual, la navegación, todo ello, a través de *wireframes* y *concept art*.

· Wireframe - Mockup



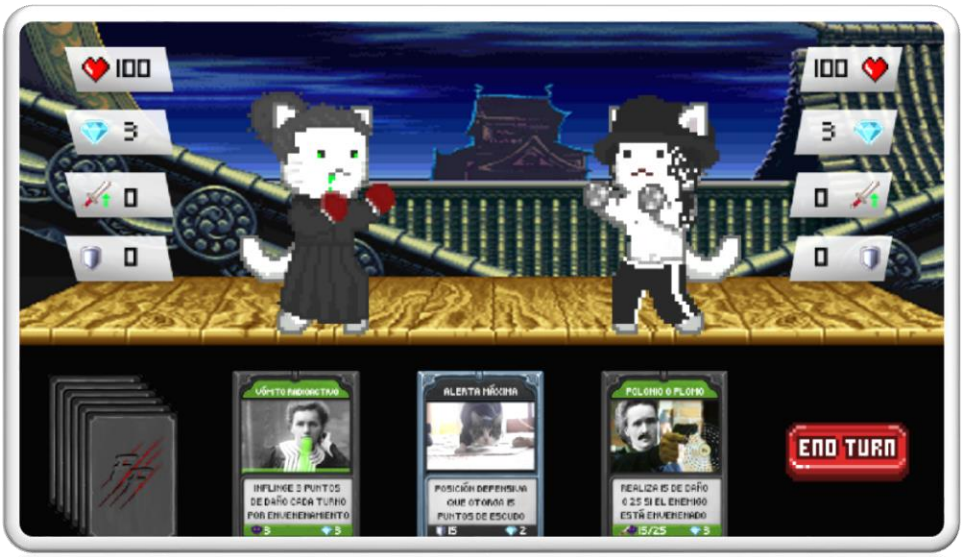
· Concept Art

- 1er diseño conceptual de *Fighting Furry*:



- Diseños orientativos durante el desarrollo:

- Diseño final de *Fighting Furry*:



2.3. Tecnología

En esta sección se detallan, especifican y justifican las tecnologías utilizadas a la hora de realizar el proyecto *Fighting Furry.*

2.3.1. Tecnología de desarrollo

· Java

Java es un lenguaje de programación de los más populares en uso, desarrollado por *Sun Microsystems* (*Oracle Corporation*).
Java es un lenguaje compilado, lo que proporciona una ejecución en cualquier máquina virtual *Java* sin importar la arquitectura del PC donde se ejecute.
El equipo de desarrollo ha elegido este lenguaje de programación ya que el objetivo de este proyecto es demostrar que se han adquirido los conocimientos a lo largo de todo el ciclo, y durante este, se ha trabajado mayoritariamente con *Java*. A su vez, al ser un lenguaje de programación conocido y familiarizado por los programadores del proyecto, estos han querido profundizar más en los conocimientos adquiridos y ponerlos en práctica.

· LibGDX

LibGDX es un framework para el desarrollo de videojuegos multiplataforma, soportando distintos S.O. tanto en PC, dispositivos móviles y web.
Se integra perfectamente con el resto de herramientas de Java y cuenta con un potente conjunto de APIs que permite mostrar imágenes y texto, construir interfaces de usuario, reproducir sonidos, el parseo de ficheros *JSON* y un largo etcétera.
En este momento el equipo de desarrollo determina que frente a las distintas opciones para el desarrollo de videojuegos multiplataforma *Fighting Furry* se desarrolla en *LibGDX* en lugar de por ejemplo *Godot*, que es un motor gráfico que también permitía el objetivo del proyecto de *Fighting Furry*.
La opción seleccionada por el equipo de desarrollo de usar LibGDX respecto a otras herramientas de desarrollo es porque se quiere profundizar en el lenguaje de programación *Java*, y, *LibGDX* es un framework establecido, que cuenta con una gran comunidad detrás, aportando soluciones y mejoras para la comunidad.

· WebSocket

WebSocket es una tecnología que proporciona una canal de comunicación bidireccional y full-dúplex, es decir, capaz de enviar y recibir mensajes e información de forma simultánea, sobre un único socket TCP.
Está diseñada para ser implementada en navegadores y servidores web, pero su versatilidad permite utilizarse por cualquier aplicación cliente/servidor.
Por parte de los desarrolladores se ha elegido esta tecnología en lugar de *Firebase,* que permite lo mismo e incluso de manera más simple que *WebSocket*, porque se ha querido profundizar en esta tecnología empleada durante este último año y ha supuesto una especie de reto poder dominar esta tecnología para el desarrollo de *Fighting Furry*.

2.3.2. Tecnología software de soporte

· Git

Git es un software de control de versiones, creado para la eficiencia, confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Esta tecnología ha permitido al equipo de desarrollo compartir el código del proyecto y la actualización del mismo.

· GitHub

GitHub es una plataforma de desarrollo colaborativo ideal para alojar proyectos utilizando el sistema de control de versiones *Git* nombrado anteriormente.

Se podría definir de forma más casual como un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código gracias a *Git*.

Gracias a la plataforma *GitHub* el equipo de desarrolladores han podido trabajar de forma segura a través de las opciones que permite este sistema, como son las bifurcaciones o ramificaciones (Branch) que permiten la división del código de forma óptima para que cada desarrollador trabaje de forma segura sin machacar o modificar el trabajo realizado por otro desarrollador, para finalmente unir todas estas ramificaciones en un código fuente principal (Merge).

Otro valor añadido que comentar de GitHub es la posibilidad de usar un sitio web gracias a *GitHub Pages* donde *DesplazaosStudio*™ ha utilizado para crear el manual de instrucciones de *Fighting Furry* en una versión más moderna y actual.

· IntelliJ IDEA

IntelliJ Idea es un entorno de desarrollo integrado (IDE) para el desarrollo de programas informáticos. Ofrece soporte para distintos lenguajes de programación, y a su vez, acepta diversas tecnologías y frameworks.

IntelliJ reúne todas las herramientas y utilidades requeridas para llevar a cabo el proyecto, y, los desarrolladores, tienen experiencia con esta herramienta usada durante todo el ciclo, permitiendo reducir el tiempo ya que conocen las funcionalidades, atajos y herramientas que aporta.

· Modelio

Modelio es una herramienta Open-Source desarrollada por *Modeliosoft* utilizada para la creación de diagramas UML.

Los desarrolladores se han decantado por esta herramienta respecto a otras como por ejemplo *Dia*, ya que están familiarizados con la misma, al trabajar con ella durante el ciclo y conocer las opciones y elementos que aporta. Durante el proyecto se observa el uso de *Modelio* en la construcción de la estructura del proyecto, así como para explicar funcionalidades y mecanismos de ejecución del mismo.

· Figma

Figma, herramienta de generación de prototipos y editor de gráficos vectorial, basado en la web, con características offline habilitadas en las aplicaciones de escritorio para S.O. *macOS* y *Windows*.

Durante este proyecto se ha enfocado el uso de *Figma* en la generación de la interfaz del usuario y el diseño de la experiencia de usuario.

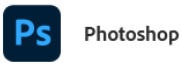
Una de las razones por las que el equipo desarrollador y de diseño ha elegido esta herramienta frente a otras similares es que permite la colaboración de los diseñadores en tiempo real.

· Gimp

GIMP es un programa Open-Source y funcional bajo la gran mayoría de sistemas operativos. Su principal funcionalidad es la creación, edición y retoque de imágenes. Otras de las grandes funcionalidades que aporta es la posibilidad de crear imágenes animadas en formato GIF.

Durante este proyecto *Gimp* se ha utilizado principalmente en la creación de los sprites de los luchadores de *Fighting Furry* y la animación de los mismos.

Por todos estos motivos comentados, *Gimp* proporciona una experiencia y herramientas apropiadas para los diseñadores para generar los sprites.



· Adobe Photoshop

Adobe Photoshop es un editor de gráficos, desarrollado por *Adobe Systems Incorporated*.

Soporta infinidad de formatos de imagen distinto, lo que le otorga una gran versatilidad combinado con todas las herramientas que posee para la edición de imágenes.

Aun siendo una herramienta de pago, y teniendo actualmente en el mercado herramientas que pueden ofrecer servicios y funcionalidades similares, se ha elegido por parte de los diseñadores del proyecto por la experiencia que tienen sobre esta herramienta usada durante años, lo que implica una reducción de tiempo a la hora de usarla.

En este proyecto se ha utilizado mayoritariamente en la creación de las cartas, escenarios, logotipos e imagen corporativa.

3. Pruebas

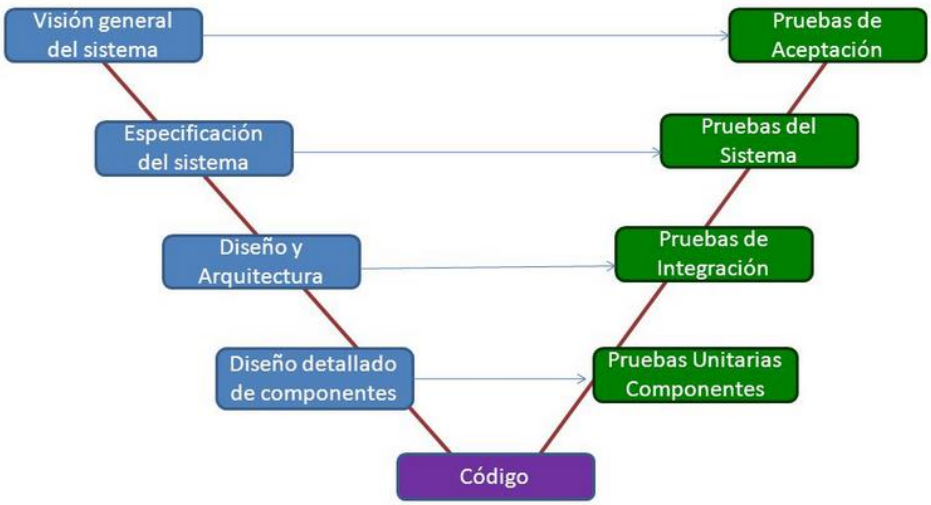
Durante el desarrollo de *Fighting Furry* se ha adoptado un *modelo en V* como estrategia de la realización de las pruebas y estas se han ido realizando durante todo el transcurso del desarrollo del proyecto.

El primer paso ha sido el desarrollo de componentes aislados y verificar y realizar pruebas unitarias a estos mismos.

Una vez se ha generado una base se pasa al siguiente paso, la integración entre componentes. En este punto se empiezan a integrar los componentes individuales creados en el primer punto para comprobar la interacción entre ellos.
En este punto del desarrollo se han encontrado problemas al desconocer el uso de *LibGDX.*

Una vez solventados los problemas encontrados durante la integración de componentes, se daría paso al tercer punto de este modelo, las pruebas de sistema donde se ponen a prueba los requisitos funcionales y no funcionales
Este se ha considerado uno de los puntos críticos del desarrollo ya que se encontraron graves problemas durante el desarrollo. La poca experiencia con *WebSockets* ha hecho paralizar el proyecto y con ello, reducir el tiempo destinado a otros elementos del mismo.
Finalmente se ha podido cumplir con estos requisitos y continuar hacia el último paso.

En este último paso se han realizo las pruebas de aceptación, estas prueban y verifican las necesidades reales de los usuarios y el fin con el cual se ha realizado este proyecto.
Una vez finalizada esta prueba el producto final, en este caso *Fighting Furry* estaría listo para su distribución.



4. Lanzamiento

En esta sección se describen las estrategias para la distribución y la publicación de *Fighting Furry* que se seguirán durante el desarrollo del proyecto, teniendo en cuenta cada uno de los componentes y sistemas externos que forman el lanzamiento del mismo.

4.1. Tareas para la distribución

Para promocionar *Fighting Furry* se ha incentivado la publicidad del mismo a través de redes sociales, pues se ha considerado que actualmente es el medio más rápido y el más accesible para todo el mundo, lo que abre una ventana bastante interesante para que conozcan nuestro proyecto.

Al tratarse de un juego *indie* no se puede competir con grandes franquicias o empresas del mundo del videojuego, de ahí que se haya descartado la idea de publicidad en spots publicitarios.

Asimismo, a día de hoy portales web y revistas que se dedican al mundo de los videojuegos son un gran escaparate publicitario para este proyecto, ya que con el auge y éxitos recientes de juegos *indie*, han dado mayor importancia a estos, otorgándoles mayor espacio en sus secciones.

Para dar a conocer nuestro proyecto a estos portales o revistas, una vez finalizado el proyecto se enviarán copias de este mismo para que estas revistas o portales puedan probar *Fighting Furry* y dar su opinión y critica.

Para la publicidad a través de redes sociales, se procederá a crear perfiles en diferentes redes sociales para informar y llegar al mayor público posible para dar a conocer *Fighting Furry*.

4.2. Publicación

Finalmente, una vez finalizado el proyecto se realiza su publicación en plataformas destinadas para ello.

En el mercado se han encontrado distintas plataformas donde publicar juegos *indie* y a continuación analizamos las opciones disponibles.

En primer lugar, se sitúa *Steam*, se podría decir que es la plataforma más conocida en cuanto se refiere a la publicación de videojuegos, uno de los principales referentes, con más de 15 años en la industria.

Se podría decir que es el mejor escaparate para juegos *indie*, pero tiene un principal problema, son sus altas comisiones y el precio de entrada.
Para dar de alta el videojuego son 100€, este capital inicial se devolvería al alcanzar la cifra de 1000€ en ventas de *Fighting Furry*, pero otro de los datos a tener en cuenta, es que, *Steam* obtiene por comisión un 30% de cada venta.

Otra plataforma importante es *GOG.com (Good Old Games)*. Bastante importante y activa en el sector, epicentro de juegos retro y juegos *indie*.
En cuanto a condiciones es muy similar a *Steam*, pero en esta ocasión, no se ha de pagar para la publicación de *Fighting Furry*.

Por último, comentar que respecto a estos juegos *indies*, se han creado plataformas recientes para poner en el mercado este tipo de juegos, como puede ser *Itch.io, GameJolt* entre otras.

En el apartado móvil, se ha consultado la publicación de *Fighting Furry* en la *Play Store,* pero ha quedado aparcada momentáneamente, ya que, aunque el juego tenga soporte para reproducirlo en teléfonos móviles, faltan detalles por realizar para la publicación de este proyecto en la plataforma de *Google*.

Conclusiones

En referencia al proyecto comentar que ha sido reconfortante y sorprendente ver como línea a línea de código hemos llegado a realizar *Fighting Furry*, proyecto del cual todos los integrantes estamos muy orgullosos de lo conseguido.
Partíamos con el conocimiento nulo sobre *LibGDX* y bien es cierto que costó arrancar y tuvimos momentos críticos durante el desarrollo que nos hizo plantear el uso de otras tecnologías o incluso modificar la idea principal del proyecto, gracias al profesorado que ha estado apoyándonos y ayudando en todo lo necesario nos dieron ese empujón necesario para continuar con este proyecto.
Nuestra idea era la de realizar un juego, multijugador, y lo único que habíamos hecho hasta la fecha eran juegos para mostrarlos por la terminal o en *JavaFX*, así que el cambio o salto a lo que queríamos hacer con Fighting Furry era bastante grande e importante.

Se han intentado utilizar y poner en practica todas las enseñanzas recibidas durante estos dos años en DAM, haciendo hincapié en este último año, y aplicando algún aporte de cada asignatura, de ahí que se hayan utilizado sockets, diagramas para explicar el funcionamiento, *concept art* y *mockups-wireframe* para el diseño, etc., intentar integrar todas las asignaturas, o todas las que encajaban con este proyecto de forma útil.

Hemos sido fieles a nuestras ideas, dividiéndonos el trabajo de la forma más óptima posible, apoyándonos entre nosotros cuando alguno de los integrantes o parte del desarrollo lo necesitaba.
Sí que es cierto que hemos tenido que ir aparcando ideas que nos planteamos en un inicio por falta de tiempo la mayoría de ellas, pero el resultado final se acerca mucho a lo que habíamos diseñado en un principio.
Somos conscientes que con un poco más de tiempo todas estas partes que hemos tenido que ir aparcando por falta de tiempo se hubieran conseguido, cosa que hubiera hecho que se asemejara mucho más este proyecto a la idea inicial.

Sobre ese tiempo, queríamos hacer una mención, es complicado destinar el tiempo necesario al proyecto y a la vez administrarlo con prácticas, exámenes, etc., y sí que hemos tenido momentos de bastante agobio por esto mismo, la acumulación de tareas que nos lo ha hecho pasar un poco mal, pero, vemos que con esfuerzo y dedicación todo es posible y el llegar aquí y poderos presentar este proyecto, lo demuestra.

Para finalizar comentar que no todo ha sido un camino tortuoso hasta hoy, hemos pasado momentos muy divertidos diseñando los personajes, cartas, ataques, etc., y es algo que hemos intentado transmitir a todos los que nos habéis apoyado y ayudado.

Webgrafía

· Fighting Furry

Fighting Furry - GitHub

· LibGDX

https://libgdx.com/dev/

https://github.com/libgdx/libgdx

https://libgdx.badlogicgames.com/ci/nightlies/docs/api/com/badlogic/gdx/input/GestureDetector.GestureListener.html

https://gamedev.stackexchange.com/questions/139041/how-to-create-card-layout-in-a-trading-card-game-libgdx

https://stackoverflow.com/questions/31536678/libgdx-networking-for-board-game

https://itch.io/games/made-with-libgdx/tag-card-game

https://stackoverflow.com/questions/16287759/how-to-drag-and-drop-actors-on-libgdx-scene2d/16288528

· WebSockets

https://github.com/websockets/ws

· Assets

https://opengameart.org/content/cat-fighter-sprite-sheet

https://opengameart.org/content/tcg-card-frame

https://www.pinterest.es/pin/366902700872810341/

https://www.deviantart.com/spritemight/art/Face-Set-01-Frames-RPG-Maker-VX-and-VX-Ace-550524017

https://www.fgbg.art/

https://www.101soundboards.com/boards/10441-street-fighter-ii-sounds

https://www.myinstants.com/search/?name=Street%20fighter

https://www.myinstants.com/search/?name=mortal+kombat

https://downloads.khinsider.com/game-soundtracks/album/dragon-ball-z-super-butouden-2-snes-gamerip

https://www.zophar.net/music/nintendo-snes-spc/dragon-ball-z-super-butouden-3

https://www.spriters-resource.com/snes/dragonballzsuperbutoden3/

https://www.deviantart.com/sonicmechaomega999/art/Dbz-Sprite-Backgrounds-550298135

https://mathieu-chauderlot.artstation.com/projects/qaZwP

https://www.deviantart.com/toomanypenguins/art/Playing-cards-template-Wooden-Back-589786131

· Otros

https://trello.com/es

https://www.ganttproject.biz/

- Anexos

· Manual de instrucciones vía web: [ENLACE]

Instrucciones

Objetivo

El objetivo en *Fighting Furry* es dejar al contrincante sin puntos de vida, haciendo esto formando una estrategia con las cartas que tienes disponible.

Cómo crear una partida multijugador

Para poder jugar en conexión, primero de todo, una máquina tiene que ejecutar el servidor de juego. El servidor se inicia en el puerto *12345*.

En el menú principal hay un apartado de opciones. Ahí se habrá de escribir la IP del anfitrión de la partida.

> Ej.: 192.168.10.10

Una vez se haya escrito darle al botón, que te llevará a la pantalla de inicio, y darle a *PLAY* para empezar una nueva partida. Se abrirá una pantalla de elección de personaje, donde se habrá de elegir uno de los personajes disponibles.

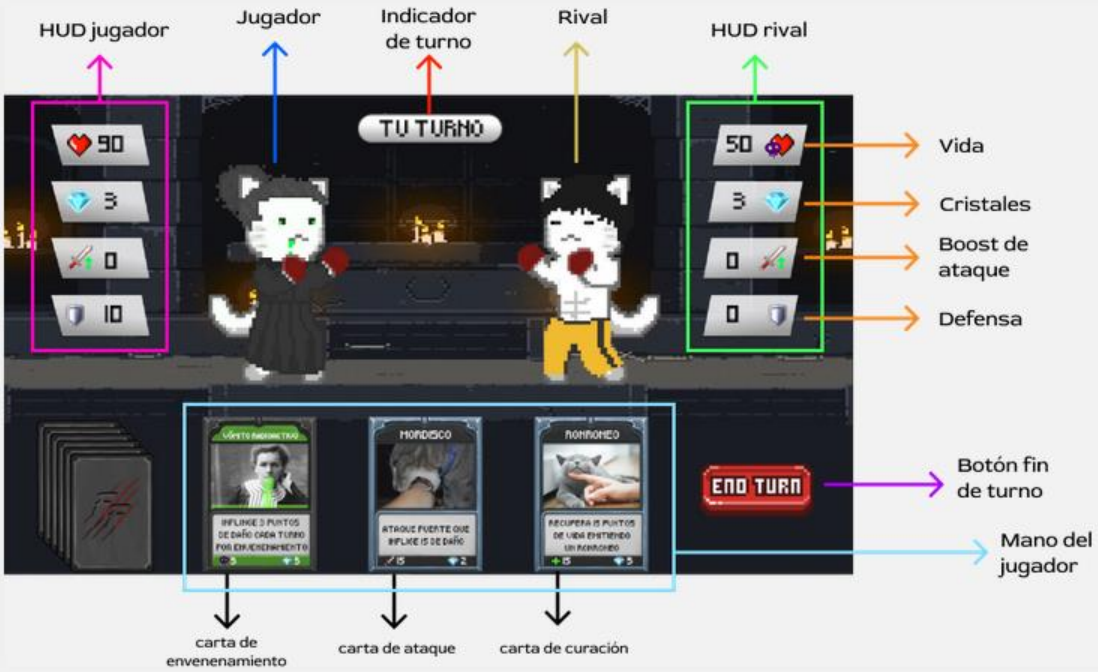
Cuando hayamos elegido a nuestro personaje darle al botón *SELECT*. La partida comenzará cuando se encuentre otro jugador contra el que jugar que haya elegido también a su personaje.

Dinámica del juego

Antes de empezar la partida eliges a uno de los seis personajes. Cada personaje tiene dos cartas especiales que se añadirán a tu mazo:

Una vez elegido tendrás 3 cartas jugables en tu mano. En la carta está señalado el nombre, la descripción de lo que hace y abajo el valor y el coste de cristales. Podrás jugar tantas cartas en tu turno como puedas comprar con tus cristales.

Al acabar tu turno darle al botón 'END TURN'.



VIDA: La salud en ese momento de la partida. Si llega a 0 se acaba el juego y gana el rival. Se puede curar usando cartas de curación pero su valor no puede sobrepasar de 100. El símbolo aparecerá superpuesto cuando el jugador está envenenado, indicando que se restará a su vida cada turno que pasa

CRISTALES: Sirven para usar cartas en cada turno. Cuando se juegue una carta, se restará el valor al que indique en ésta. Cada turno se reestablecé el valor de los cristales a 3.

BOOST DE ATAQUE: Es el ataque adicional que se suma al ataque si se usa una carta de ataque en ese turno. Se reestablece a 0 una vez usado.

DEFENSA: Es el valor de la defensa. Previene que se reste la vida por los ataques del rival. Cada vez que se reciba uno se irá restando el valor. Si el valor es 0 los ataques bajaran la salud

Cartas

Las cartas son lo que se usa en cada turno para ejecutar una acción, esta sea atacar al rival, curarse la vida, protegerse de futuros ataques, etc. Para jugar una carta sólo basta con hacer clic izquierdo con el ratón.

Este es el esquema de una carta:

Cartas normales

Ataque

Las cartas de ataque baja tanta salud o defensa al contrincante como el valor que marca en la carta

- **Zarpazo:** Inflinge 10 puntos de daño
- **Mordisco:** Inflinge 15 puntos de daño

Defensa

Las cartas de defensa suman su valor la defensa del usuario

- **Bufido:** Aporta 10 puntos de escudo
- **Alerta Máxima:** Aporta 15 puntos de escudo

Curación

Las cartas de curación suma su valor a la salud del usuario hasta llegar a 100 como máximo

- **Ronroneo:** Recupera 15 puntos de vida
- **Amasar:** Recupera 10 puntos de vida

Cartas especiales / de personaje

Paté

Cabezazo: Baja el ataque del enemigo 5 puntos.
Pisotón: Descarta una carta aleatoriamente de la mano del oponente

B. Pussolini

Autocracia: Descarta las cartas de la mano del rival.
Pacto de Acero: Empieza con 3 puntos de defensa durante 2 turnos.

Miauchel Catson

Heehee: Paraliza al rival durante un turno sin poder hacer nada.
Anti-Gravity Lean: El siguiente ataque inflige 5 puntos de daño adicionales.

Marie Furrie

Vómito radioactivo: Envenena al contrincante infligiendo 3 puntos de daño por turno.
Polonio o plomo: hace 15 de daño, si el enemigo está envenenado hace 25.

Goku

Imagen Reflejo: Esquiva las siguientes cartas que lance el rival.
Kame kame ha: Inflige 25 puntos de daño.

Miau Lee

Patada voladora: Inflinge 25 puntos de daño
Furia oriental: Hace entre 3 y 5 (aleatorio) golpes de 5 puntos de daño cada uno.

· **Diseño de lanzamiento en *Super Nintendo Entertainment System:***

· **Con la colaboración de Paté como modelo y luchador en *Fighting Furry***

*Ningún animal ha sufrido daños durante el desarrollo de este proyecto

Fighting Furry

DESPLAZADOS STUDIO

2021