

Proyecto Final DAW

Zplayer



Noel Barcia Almagro

IES Arcipreste de Hita

04/04/2024

Índice

1. Definición del proyecto.....	5
1.1 Tipos de Empresas. Sectores Productivos.....	5
1.2 Descripción del Problema.....	6
1.3 Objetivos. Especificación de Requisitos.....	6
<u>1.3.1 Requerimientos Funcionales.....</u>	8
<u>1.3.2 Requerimientos No Funcionales.....</u>	9
<u>1.3.3 Requerimientos Técnicos.....</u>	10
1.4 Herramientas de diseño.....	10
<u>1.4.1 Node.js.....</u>	11
<u>1.4.2 Express.js.....</u>	11
<u>1.4.3 Angular.....</u>	11
<u>1.4.4 Tailwind css.....</u>	11
<u>1.4.5 MongoDB.....</u>	11
<u>1.4.6 Font awesome.....</u>	11
<u>1.4.7 Angular Material.....</u>	12
1.5 Estudio de viabilidad.....	12
<u>1.5.1 Viabilidad técnica.....</u>	12
<u>1.5.2 Viabilidad económica.....</u>	12
<u>1.5.3 Viabilidad del mercado.....</u>	13
<u>1.5.4 Análisis DAFO.....</u>	13
<u>1.5.5 Conclusión.....</u>	14
2. Planificación del proyecto.....	14
2.1 Secuenciación de las fases del proyecto.....	14
<u>2.1.1 Análisis y Definición.....</u>	14
<u>2.1.2 Diseño y Planificación.....</u>	14
<u>2.1.3 Codificación.....</u>	15
<u>2.1.4 Pruebas.....</u>	15
<u>2.1.5 Documentación.....</u>	15
<u>2.1.6 Despliegue.....</u>	15
<u>2.1.7 Mantenimiento y Evolución.....</u>	15
2.2 Planificación de recursos y tiempo.....	15
2.3 Prevención de riesgos en el desarrollo.....	16



3. Análisis del Proyecto.....	17
3.1 Casos de Uso.....	17
3.2 Modelos de datos.....	17
<u>3.2.1 Esquema Entidad / Relación.....</u>	<u>17</u>
<u>3.2.2 Modelo relacional y normalización.....</u>	<u>18</u>
<u>3.2.3 Diccionario de datos.....</u>	<u>19</u>
4. Diseño del Proyecto.....	20
4.1 Diseño físico de la BD y estructura de los ficheros.....	20
4.2 Diseño de la estructura de clases y librerías (diagrama de clases).....	23
4.3 Diseño de la interfaz gráfica.....	24
5. Desarrollo y realización del proyecto.....	37
5.1. Indicadores de calidad.....	37
5.2 Elaboración de una batería de pruebas para detectar errores.....	37
<u>5.2.1 Pruebas funcionales.....</u>	<u>37</u>
<u>5.2.2 Otras Pruebas.....</u>	<u>43</u>
5.3 Evaluación y solución de incidencias.....	43
5.4- Evaluación y seguimiento de las actividades.....	44
6- Implantación del Proyecto.....	44
6.1- Plan de implantación.....	44
6.2 Manual de instalación.....	44
6.3 Manual de usuario.....	45
7. Conclusiones Finales.....	51
7.1. Evaluación del proyecto: objetivos conseguidos y no alcanzados.....	51
7.2 Propuestas de mejora.....	51



1. Definición del proyecto

El mundo de los videojuegos ha experimentado un crecimiento exponencial en los últimos años, convirtiéndose en una gran industria en todo el mundo con millones de jugadores. Aunque, a pesar de esta popularidad y la gran cantidad de jugadores, muchos de estos aún enfrentan dificultades para encontrar a otros con quienes compartir su pasión y pasar grandes momentos en compañía, aquí es donde surge la idea de Zplayer, una red social diseñada para la comunidad de gamers.

1.1 Tipos de Empresas. Sectores Productivos

Zplayer no solo se presenta como una red social para gamers, sino también como un impulsor del sector, sobre todo en el ámbito del entretenimiento y el ocio. Al facilitar la conexión entre los jugadores y las comunidades de videojuegos, esta aplicación contribuye a:

- ❖ **Aumento del uso de juegos sin mucha popularidad:** La aplicación puede servir como un escaparate a nuevos lanzamientos de empresas indies, con las que podría incentivar a los jugadores a adquirir nuevos títulos que a lo mejor nunca hubieran descubierto.
- ❖ **Desarrollo de nuevas oportunidades de negocio:** La aplicación podría ofrecer oportunidades de negocio para empresas relacionadas con el mundo de los videojuegos, como desarrolladores, marcas de hardware, publishers, agentes, etc.
- ❖ **Crecimiento en comunidades de juegos:** La aplicación puede servir para que las empresas de videojuegos consigan a nuevos jugadores publicitándose con usuarios de la aplicación.



1.2 Descripción del Problema

La comunidad de jugadores enfrenta diversas dificultades para conectar entre sí, encontrar compañeros de juego adecuados y descubrir nuevos títulos que se ajusten a sus gustos e intereses.

Esto puede ser por la falta de plataformas específicas para el descubrimiento de jugadores.

Estos problemas tienen consecuencias negativas para la comunidad, como:

- ❖ **Disminución de disfrute:** La experiencia de algunos juegos puede verse afectada negativamente por la falta de compañeros de juego adecuados
- ❖ **Pérdida de experiencias:** Hay jugadores que pierden la oportunidad de jugar a grandes títulos, por el hecho de desconocerlos o no tener feedback de estos.

1.3 Objetivos. Especificación de Requisitos

La finalidad de **Zplayer** es resolver los problemas de la comunidad, como la falta de herramientas efectivas de conexión y descubrimiento dentro de la comunidad de jugadores, creando una red social para que estos conecten entre sí, descubran nuevos juegos y compartan su pasión por los videojuegos.

Teniendo esto en cuenta, se pueden definir los siguientes requerimientos:

- ❖ **Registro y gestión de perfiles:** Los usuarios podrán crear perfiles completos que incluyan información personal de sí mismos, en el que se registran todos sus posts y mensajes privados.
- ❖ **Búsqueda de jugadores:** Los jugadores podrán buscar y descubrir a otros jugadores para seguir o chatear con ellos.



- ❖ **Sistema de posts:** Los usuarios podrán poner posts públicos sobre los juegos a los que juegan actualmente, buscando gente para jugar, etc.
- ❖ **Chat y mensajería:** Zplayer ofrece herramientas de chat y mensajería instantánea para que los usuarios puedan comunicarse entre sí de forma privada.
- ❖ **Usabilidad:** Zplayer debe ser fácil de usar, intuitiva y accesible para todo tipo de usuarios, independientemente de su nivel de experiencia técnica.
- ❖ **Rendimiento:** Zplayer debe ser una plataforma robusta y escalable que pueda soportar un gran número de usuarios y tráfico simultáneo.
- ❖ **Seguridad:** La plataforma debe implementar medidas de seguridad adecuadas para proteger los datos de los usuarios evitando robos de información y accesos no deseados.
- ❖ **Confiabilidad:** Zplayer debe ser una plataforma confiable que funcione de manera constante, sin interrupciones, y sin pérdidas de datos
- ❖ **Accesibilidad:** La plataforma debe ser accesible para personas con discapacidades, siguiendo las pautas de accesibilidad.
- ❖ **Mantenimiento:** Zplayer debe contar con un plan de mantenimiento adecuado para garantizar su correcto funcionamiento, la actualización de sus funcionalidades y la implementación de nuevas funcionalidades.
- ❖ **Estabilidad:** Zplayer no debe presentar errores ni fallos frecuentes inesperados durante su ejecución

Estos requisitos los dividiremos posteriormente en distintos tipos de requisitos



1.3.1 Requerimientos Funcionales

Nº REQ	REQUERIMIENTOS	DESCRIPCIÓN DEL REQUERIMIENTO
RF1	Registro usuario	La aplicación debe permitir el registro de nuevos usuarios en esta
RF3	Login usuario	La aplicación debe permitir que un usuario anteriormente registrado pueda acceder a la aplicación con sus datos
RF2	Publicar posts	La aplicación debe permitir que los usuarios publiquen posts
RF3	Borrar post	La aplicación debe permitir que los usuarios puedan borrar un post propio
RF4	Ver listado posts	La aplicación debe permitir ver el listado de los posts existentes
RF5	Descubrir y buscar jugadores	La aplicación debe permitir que un usuario pueda buscar y descubrir a otros
RF6	Editar perfil	La aplicación debe permitir que un usuario edite su perfil cambiando datos
RF7	Distinción de posts	La aplicación debe distinguir los posts por comunidades
RF8	Mensajes privados	La aplicación debe permitir manejar mensajes instantáneos privados entre usuarios



1.3.2 Requerimientos No Funcionales

Nº REQ	REQUERIMIENTOS	DESCRIPCIÓN DEL REQUERIMIENTO
RF1	Usabilidad	El usuario debe poder saber usar la aplicación de manera rápida y sin dificultades
RF3	Rendimiento	La aplicación debe funcionar eficientemente, realizando la carga de datos y las demás funciones de manera rápida y efectiva.
RF2	Seguridad	La aplicación debe ser robusta y escalable, que pueda soportar un gran número de usuarios y tráfico simultáneo
RF3	Confiabilidad	La aplicación debe mantener la integridad de los datos de la aplicación, evitando pérdidas de datos inesperados durante su ejecución o la inestabilidad de los datos
RF4	Accesibilidad	La plataforma debe ser accesible para personas con discapacidades, siguiendo las pautas de accesibilidad
RF5	Mantenimiento	La aplicación debe tener su código estructurado y bien refactorizado siguiendo los principios básicos, para facilitar posibles mejoras y modificaciones futuras
RF6	Estabilidad	La aplicación debe no presentar errores ni fallos frecuentes inesperados durante su ejecución



1.3.3 Requerimientos Técnicos

- **Node 20.11:** Como lenguaje de programación del lado del servidor.
- **Angular 17.3:** Framework de desarrollo frontend para construir la interfaz de usuario.
- **TypeScript:** Como lenguaje de programación del lado del cliente.
- **HTML, CSS:** Para el desarrollo de la interfaz de usuario.
- **MongoDB:** Como base de datos NoSQL para almacenar datos de usuarios y posts.
- **Express:** Framework de Node.js para construir APIs y manejar rutas.
- **Git:** Para controlar y subir los cambios efectuados en el proyecto de manera automática y rápida contra un servidor en la nube que en este caso será **Github**.

1.4 Herramientas de diseño

El desarrollo de una aplicación web moderna implica la creación de interfaces de usuario atractivas y experiencias de usuario fluidas, además de una base de código sólida y escalable. Para lograr esto, es fundamental contar con las herramientas adecuadas que faciliten el proceso de diseño, desarrollo y colaboración.

A continuación, enumeraremos estas herramientas y explicaremos su funcionalidad dentro de la aplicación.

1.4.1 Node.js

Es un entorno de ejecución de JavaScript de código abierto que permite ejecutar código JavaScript en el lado del servidor. Es una plataforma popular para el desarrollo de aplicaciones web modernas debido a su velocidad, escalabilidad y flexibilidad. Este será utilizado para el desarrollo del lado del servidor utilizando el framework de Express.js

1.4.2 Express.js

Es un framework web Node.js minimalista y flexible que te permite crear aplicaciones web robustas y escalables. Es uno de los marcos de Express.js más populares y ampliamente utilizados para el desarrollo de aplicaciones web Node.js.



1.4.3 Angular

Es un framework web front-end basado en JavaScript. Puedes utilizar Node.js y Express.js para crear un servidor web que sirva el código Angular a los clientes web. El lenguaje de programación utilizado será Typescript.

1.4.4 Tailwind css

Es un framework CSS de bajo nivel que te permite crear diseños de interfaz de usuario personalizados con facilidad. lo usaremos para darle estilos al código, y hacer la aplicación responsive.

1.4.5 MongoDB

Es una base de datos NoSQL que almacena datos en documentos JSON, proporcionando un modelo de datos flexible y escalable, lo usaremos para almacenar los datos de los usuarios, los posts o los mensajes directos.

1.4.6 Font awesome

Usaremos los recursos que nos ofrece font awesome, como los iconos

1.4.7 Angular Material

Es una biblioteca de componentes de interfaz de usuario para angular que implementa los principios de Material Design de Google., usaremos algunos de los componentes de angular material

Otras herramientas que utilizo son la IA de bing para crear los logotipos, banners y las imágenes de comunidades

1.5 Estudio de viabilidad

En este estudio se analizará la viabilidad del proyecto **Zplayer**, desde tres perspectivas: técnica, económica y de mercado. El objetivo es determinar si la aplicación tiene el potencial de ser exitosa y sostenible en el tiempo.



1.5.1 Viabilidad técnica

Zplayer plantea un desarrollo con funcionalidades complejas, como la creación de usuarios, la búsqueda de jugadores, la creación de posts y mensajes directos, sin embargo, las tecnologías actuales permiten desarrollar aplicaciones con estas características. El único inconveniente en el desarrollo de la aplicación es el desconocimiento previo de los lenguajes y frameworks utilizados.

1.5.2 Viabilidad económica

Zplayer podría generar ingresos económicos a través de diferentes modelos de negocios, como por ejemplo:

- ❖ **Publicidad:** Dentro de nuestra plataforma se podrían mostrar anuncios de diferentes marcas, publicitando videojuegos o otros productos relacionados con el mundo gamer
- ❖ **Suscripciones o mejoras de usuario:** Se pueden ofrecer suscripciones premium o mejoras de usuario que permitan acceso a funcionalidades adicionales.
- ❖ **Integraciones con desarrolladores:** Se puede establecer pactos con desarrolladores de videojuegos para ofrecer contenido exclusivo o integraciones dentro de la aplicación

Además la inversión inicial sería baja, ya que en lo único en lo que se destinaría el dinero es en la compra de un buen alojamiento web, y en marketing de la plataforma.

1.5.3 Viabilidad del mercado

El mercado de los videojuegos es uno de los más grandes y dinámicos del mundo. Se espera que este mercado continúe creciendo en los próximos años, impulsado por el desarrollo de nuevas tecnologías y la mejora de algunas ya creadas (como las VR), y por la creciente popularidad del mundo competitivo, los eSports.

Zplayer se dirige a un público muy amplio, compuesto por jugadores de todas las edades y niveles de experiencia.



Existen muchas plataformas y redes sociales en las que existen comunidades dentro de ellas dirigidas a los videojuegos como twitter, discord, o muchas otras , pero hay pocas que su enfoque principal sea la conexión entre jugadores y el descubrimiento de juegos nuevos, por eso se busca la diferencia en **Zplayer**, enfocando su funcionamiento en la conexión entre jugadores.

1.5.4 Análisis DAFO

Debilidades	Amenazas
<ul style="list-style-type: none"> - Nueva en el mercado - Recursos limitados - Dependencia de Vercel 	<ul style="list-style-type: none"> - Competencia - Cambios tecnológicos - Cambios en las preferencias de los usuarios
Fortalezas	Oportunidades
<ul style="list-style-type: none"> - Diseño innovador - Funciones únicas - Potencial de crecimiento 	<ul style="list-style-type: none"> - Integración con otras aplicaciones - Monetización

1.5.5 Conclusión

Zplayer será una plataforma online innovadora y atractiva, para la gestión de juegos y la interacción entre jugadores. La aplicación buscará una solución viable y escalable que cumpla con los requisitos funcionales y no funcionales antes comentados.

Existe algunas dificultades a la hora de el desarrollo de la aplicación que son el desconocimiento previo de los lenguajes y frameworks utilizados, y la dificultad de conseguir capital para implementar mejoras o comprar un buen servidor web en el que alojar la aplicación, aunque existan este inconveniente la aplicación es completamente viable, ya que no existe una inversión inicial, solo se necesitarán horas de desarrollo.



2. Planificación del proyecto

2.1 Secuenciación de las fases del proyecto

El proyecto se puede dividir en las siguientes fases:

2.1.1 Análisis y Definición

- ❖ **Análisis de mercado:** Recopilar y analizar información sobre el mercado de videojuegos, la competencia y el público objetivo.
- ❖ **Definición de requisitos:** Especificación detalladas de las funcionalidades y características que debe tener **Zplayer**, con un resumen inicial.

2.1.2 Diseño y Planificación

- ❖ **Diseño de la arquitectura de la aplicación:** Definición de la infraestructura necesaria para el desarrollo de la aplicación.
- ❖ **Diseño de la interfaz de usuario:** Creación de diseños como sketches, mockups y prototipos de la interfaz de usuario de la aplicación.
- ❖ **Diseño de la experiencia de usuario:** Definición del flujo de navegación, las interacciones, y la usabilidad de la aplicación.

2.1.3 Codificación

- ❖ **Desarrollo del front-end:** Creación de la interfaz de usuario de la aplicación.
- ❖ **Desarrollo del back-end:** Implementación de la lógica de la aplicación con la base de datos .

2.1.4 Pruebas

Realización de pruebas para comprobar y verificar el correcto funcionamiento del software creado, utilizando distintos tipos de pruebas, probando el 100% de funcionalidades.



2.1.5 Documentación

Elaboración de la documentación de todo el proyecto, creando una memoria y manuales de uso e implantación

2.1.6 Despliegue

Se intentará desplegar la aplicación en una plataforma gratuita, para ponerla en funcionamiento.

2.1.7 Mantenimiento y Evolución

- ❖ **Corrección de errores:** Solución de errores y problemas técnicos que puedan surgir en la aplicación.
- ❖ **Actualizaciones de contenido:** Agregación de nuevas funcionalidades, juegos y contenido a la aplicación
- ❖ **Mantenimiento de la infraestructura:** Actualizando y manteniendo la infraestructura para garantizar el buen funcionamiento de la aplicación

2.2 Planificación de recursos y tiempo

Para la realización de este proyecto se requerirán pocos **recursos económicos**, dentro de los **recursos humanos** estaré solo yo, que seré el que realice todas las partes del proyecto y los **recursos materiales** serán un ordenador y un monitor que serán con los haré el desarrollo y las pruebas de la aplicación.

Para observar la planificación del tiempo la representará el diagrama de Gantt a continuación

Tareas	Marzo	Abril	Mayo	Junio
(1) Análisis y Definición	(1)			
(2) Diseño y Planificación		(2)		
(3) Codificación			(3)	
(4) Pruebas				(4)
(5) Documentación		(5)		
(6) Despliegue				(6)
(7) Mantenimiento y Evolución				(7)



2.3 Prevención de riesgos en el desarrollo

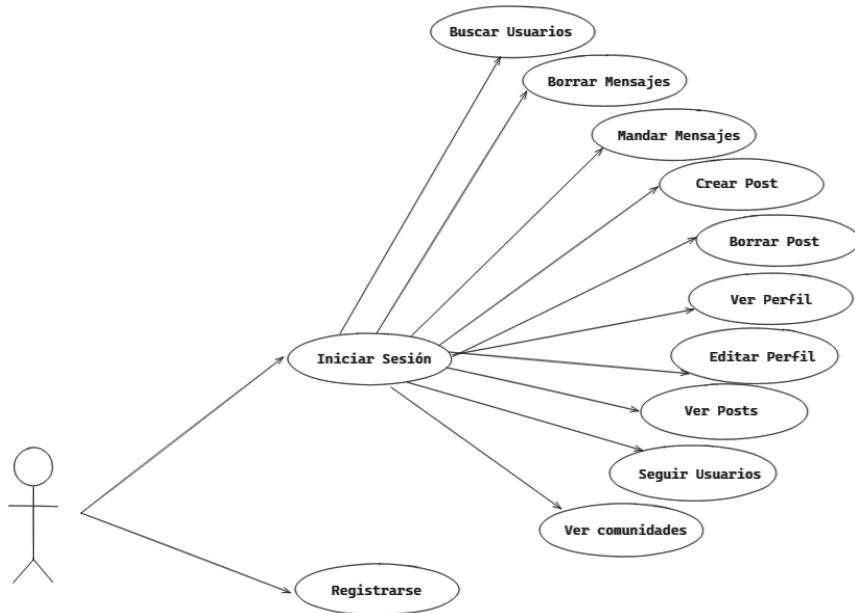
La gestión de riesgos es un aspecto fundamental para el éxito de cualquier proyecto. En el caso del desarrollo de la aplicación Zplayer, es importante identificar, analizar y prevenir los posibles riesgos que puedan surgir durante el proceso, estos riesgos se dividen en los siguientes tipos:

- ❖ **Riesgos técnicos:** Relacionados con la complejidad del desarrollo, la tecnología utilizada, la infraestructura y las herramientas.
- ❖ **Riesgos de planificación:** Relacionados con la estimación del tiempo, los recursos y el presupuesto.
- ❖ **Riesgos de gestión:** Relacionados con la toma de decisiones de la aplicación.

3. Análisis del Proyecto

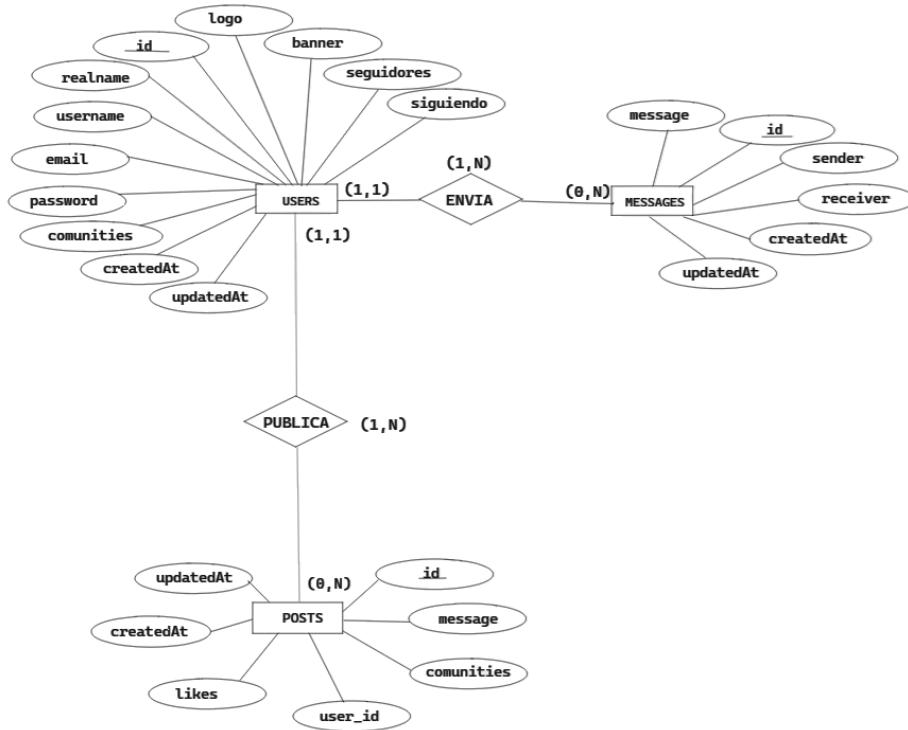
3.1 Casos de Uso

Los casos de uso son las acciones que un usuario puede realizar dentro de la aplicación, un usuario puede iniciar sesión y si no tiene cuenta registrarse, cuando el usuario haya iniciado sesión dispondrá de las siguientes acciones.



3.2 Modelos de datos

3.2.1 Esquema Entidad / Relación



3.2.2 Modelo relacional y normalización

MESSAGES-> (ID, SENDER(FK),RECEIVER(FK),MESSAGE, CREATEDAT,UPDATEDAT)

- Clave Primaria: ID
- Clave Foránea: RECEIVER(USER)
- Clave Foránea: SENDER(USER)

POSTS-> (ID,MESSAGE,COMUNITIES,USER_ID,LIKES,CREATEDAT,UPDATEDAT)

- Clave Primaria: ID
- Clave Foránea: USER_ID(USER)

USERS-> (ID, REALNAME, USERNAME, EMAIL, PASSWORD,COMUNITIES, LOGO, BANNER, SEGUIDORES, SIGUIENDO, CREATEDAT,UPDATEDAT)

- Clave Primaria: ID



MESSAGES-> Cumple la 1FN ya que todos sus valores son atómicos, también cumple la 2FN ya que todos sus atributos dependen funcionalmente de forma completa de la clave primaria, por último cumple la 3FN porque los atributos no dependen transitivamente de la clave primaria.

POSTS -> Cumple la 1FN ya que todos sus valores son atómicos, también cumple la 2FN ya que todos sus atributos dependen funcionalmente de forma completa de la clave primaria, por último cumple la 3FN porque los atributos no dependen transitivamente de la clave primaria.

USERS -> Cumple la 1FN ya que todos sus valores son atómicos, también cumple la 2FN ya que todos sus atributos dependen funcionalmente de forma completa de la clave primaria, por último cumple la 3FN porque los atributos no dependen transitivamente de la clave primaria.

3.2.3 Diccionario de datos

TABLA	CAMPO	TIPO	DESCRIPCIÓN	NULL
MESSAGES	ID	STRING	PK	NOT NULL
	SENDER	STRING	FK USER	NOT NULL
	RECEIVER	STRING	FK USER	NOT NULL
	MESSAGE	STRING		NOT NULL
	CREATEDAT	DATE		NULL
	UPDATEDAT	DATE		NULL

TABLA	CAMPO	TIPO	DESCRIPCIÓN	NULL
COMUNICACIONES	ID	STRING	PK	NOT NULL
	MESSAGE	STRING		NOT NULL
	COMUNITIES	STRING		NULL



POSTS	USER_ID	STRING	FK USER	NOT NULL
	LIKES	STRING[]		NULL
	CREATEDAT	DATE		NULL
	UPDATEDAT	DATE		NULL

TABLA	CAMPO	TIPO	DESCRIPCIÓN	NULL
USERS	ID	STRING	PK	NOT NULL
	REALNAME	STRING		NOT NULL, UNIQUE
	USERNAME	STRING		NOT NULL
	EMAIL	STRING		NOT NULL
	PASSWORD	STRING		NOT NULL
	COMUNITIES	STRING[]		NOT NULL
	LOGO	STRING		NULL
	BANNER	STRING		NULL
	SEGUIDORES	STRING[]	CONJUNTO DE USERS	NULL
	SIGUIENDO	STRING[]	CONJUNTO DE USERS	NULL
	CREATEDAT	DATE		NULL
	UPDATEDAT	DATE		NULL



4. Diseño del Proyecto

4.1 Diseño físico de la BD y estructura de los ficheros

Para crear las tablas crearemos modelos en la base de datos, con las que automáticamente crearán esas tablas en mongoDB, si no estaban creadas de anteriormente:

user.model.js ->

```
zplayer-BackEnd > models > user.model.js > ...
1  const mongoose = require("mongoose");
2
3  const userSchema = new mongoose.Schema(
4    {
5      realname: {
6        type: String,
7        required: true,
8      },
9      username: {
10        type: String,
11        required: true,
12        trim: true,
13        unique: true,
14      },
15      email: {
16        type: String,
17        required: true,
18        unique: true,
19      },
20      password: {
21        type: String,
22        required: true,
23      },
24      communities: {
25        type: Array,
26        required: true,
27      },
28      logo: {
29        type: String,
30        default: '/uploads/profile_pictures/avatardefault.png'
31      },
32      banner: {
33        type: String,
34        default: '/uploads/banners/bannerdefault.jpg'
35      },
36      seguidores: [
37        {
38          type: mongoose.Schema.Types.ObjectId,
39          ref: 'User'
40        }
41      ],
42      siguiendo: [
43        {
44          type: mongoose.Schema.Types.ObjectId,
45          ref: 'User'
46        }
47      ],
48      {
49        timestamps: true,
50      }
51    }
52  );
53
54  module.exports = mongoose.model("User", userSchema);
```



post.model.js ->

```
zplayer-BackEnd > models > post.model.js > postSchema
1  const mongoose = require("mongoose");
2
3  const postSchema = new mongoose.Schema(
4    {
5      message: {
6        type: String,
7        required: true,
8      },
9      communities: {
10        type: String,
11      },
12      user_id: {
13        type: mongoose.Schema.Types.ObjectId,
14        ref: 'User',
15        required: true
16      },
17      likes: [
18        {
19          type: mongoose.Schema.Types.ObjectId,
20          ref: 'User'
21        }
22      ],
23      {
24        timestamps: true,
25      }
26    );
27
28 module.exports = mongoose.model("Post", postSchema);
```

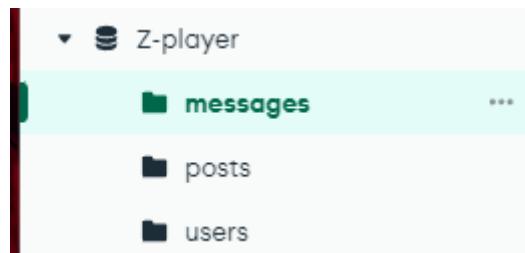
message.model.js ->

```
zplayer-BackEnd > models > message.model.js > ...
1  const mongoose = require("mongoose");
2
3  const messageSchema = new mongoose.Schema(
4    {
5      sender: {
6        type: mongoose.Schema.Types.ObjectId,
7        ref: 'User',
8        required: true,
9      },
10     receiver: {
11       type: mongoose.Schema.Types.ObjectId,
12       ref: 'User',
13       required: true
14     },
15     message: {
16       type: String,
17       required: true
18     }
19   },
20   {
21     timestamps: true,
22   }
23 );
24
25 module.exports = mongoose.model("Message", messageSchema);
```



Zplayer

Después de ejecutar node.js, comprobamos que se han creado las tablas necesarias para la aplicación, utilizando MongoDB Compass:



La estructura de ficheros será distinta dependiendo si nos situamos en el lado del cliente o en el lado del servidor.

❖ **Front-end :**

src: contiene todo el código fuente front de la aplicación, este contiene la carpeta **app y assets**, la carpeta app además de las siguientes carpetas contendrá los ficheros principales de la aplicación.

→ **components:** se guardarán todos los componentes de la aplicación junto con sus ficheros (HTML,CSS, TS).

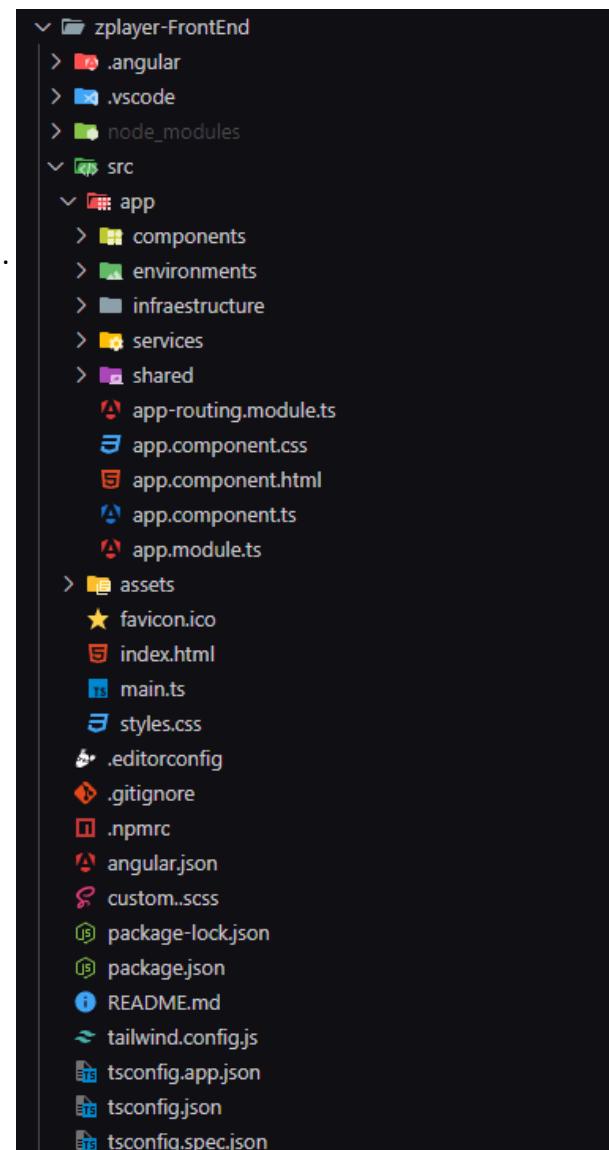
→ **environments:** se guardarán los ficheros con las configuraciones para conectarse a la API.

→ **infrastructure:** se guardarán los ficheros con las interfaces de datos.

→ **services:** se guardarán los ficheros con los servicios de la aplicación, que estos se encargan de realizar las peticiones al back, mediante sus funciones.

→ **shared:** aquí se guardarán todos los componentes, servicios, ..., que sean compartidos.

→ **assets:** se guardarán las imágenes y



recursos estáticos de la aplicación.

❖ Back-end :

En el raiz del backend se encuentran los ficheros de configuración que se encargan del arranque del back, y la configuración de la bd.

→ **controllers:** contiene los ficheros controladores, que estos disponen de las funciones que se conectan a la base de datos y los devuelven.

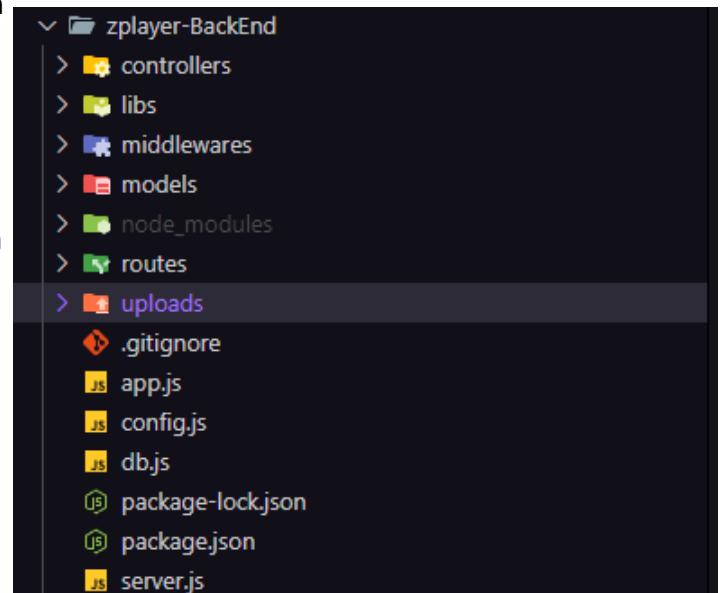
→ **libs:** contiene el fichero para crear el jwt.

→ **middlewares:** contiene el fichero para configurar el CORS.

→ **models:** contiene los ficheros con los modelos de datos de la bd

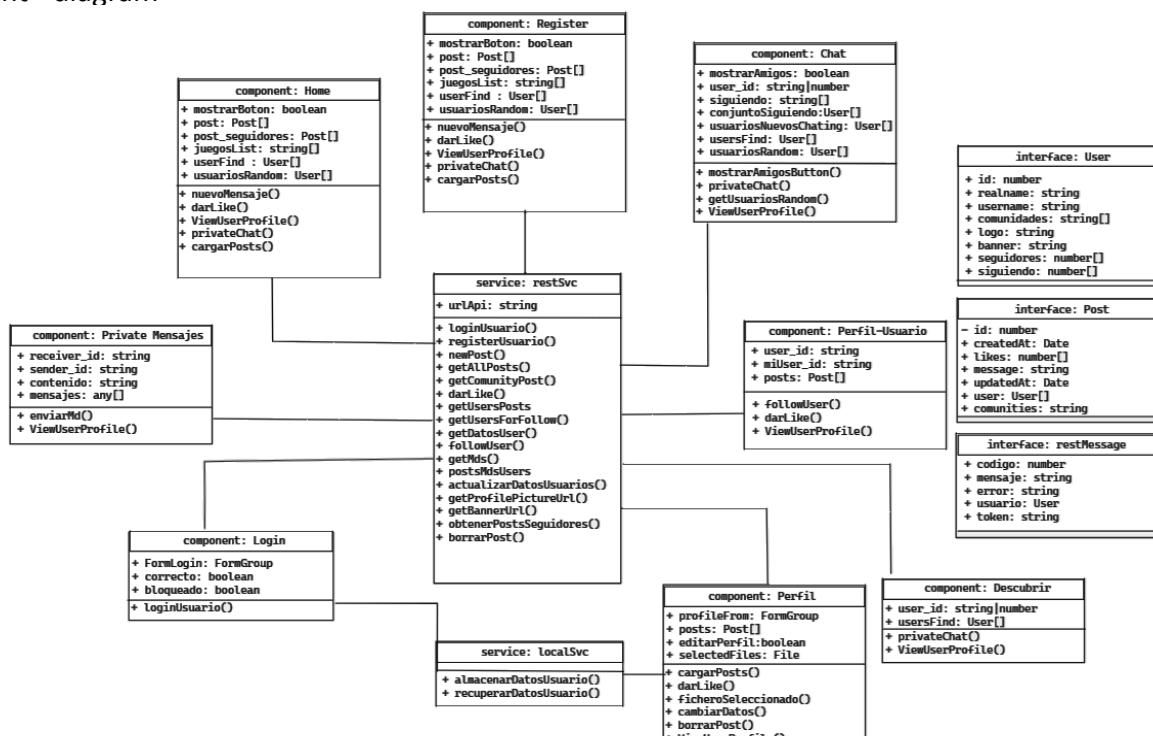
→ **routes:** contiene los ficheros con las rutas de la aplicación.

→ **uploads:** contiene las imágenes que vienen de el front al cambiar una imagen o un banner.

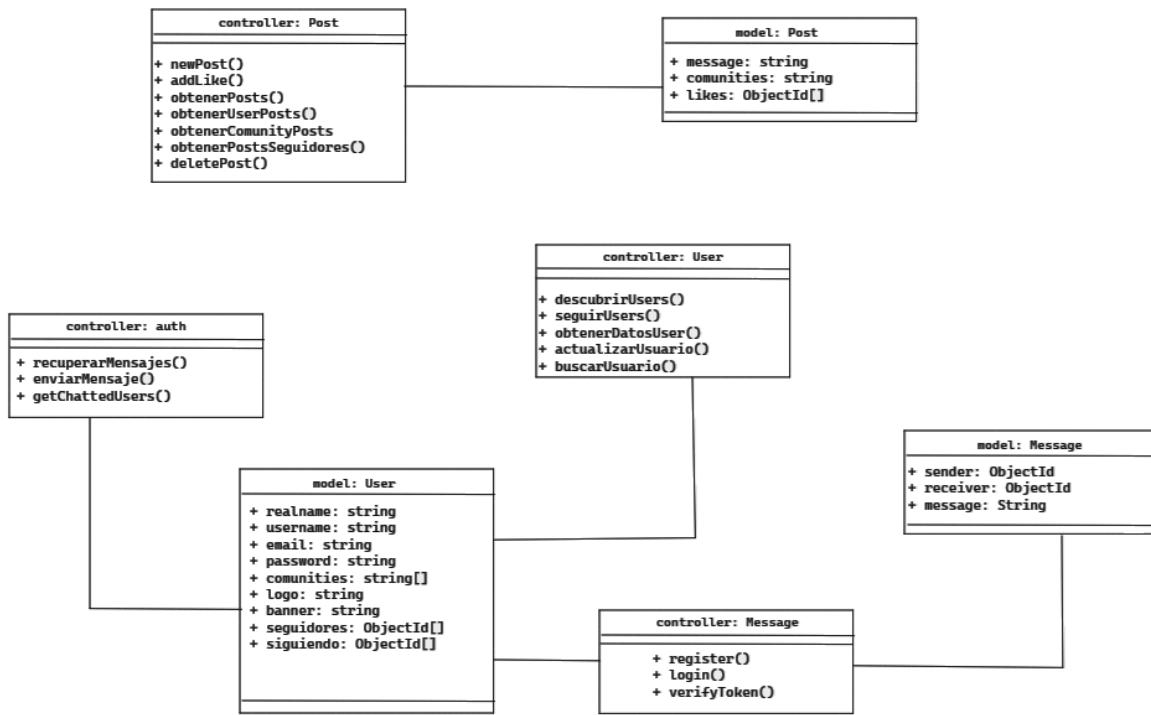


4.2 Diseño de la estructura de clases y librerías (diagrama de clases)

Front - diagram



Backend - diagram



4.3 Diseño de la interfaz gráfica

Para el diseño de la interfaz gráfica se busca un estilo moderno, sencillo y minimalista, con poca variedad de colores para no generar una saturación excesiva, la idea principal de la aplicación es diseñada para ordenadores y tablets , pero los dispositivos móviles también podrán utilizarla al completo ya que será completamente responsive

- ❖ Sketch:

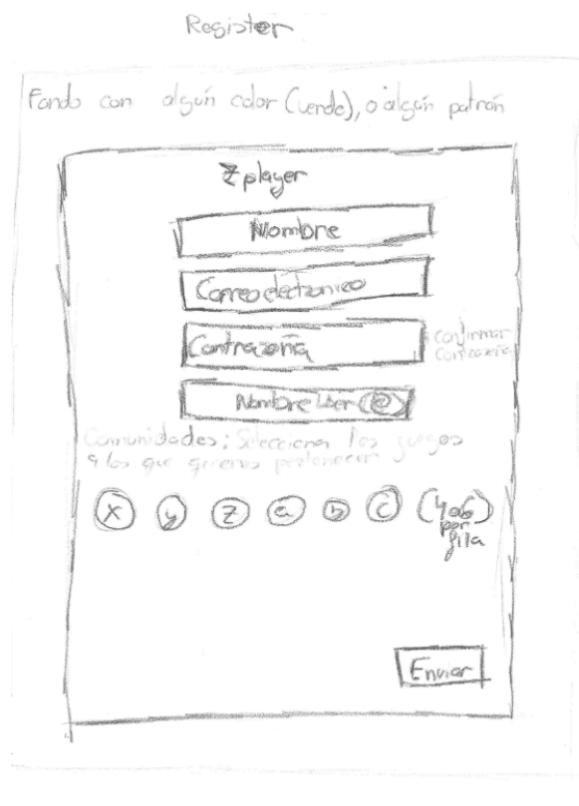
Los sketch se hicieron en el inicio del desarrollo de la aplicación es posible que existan muchos cambios



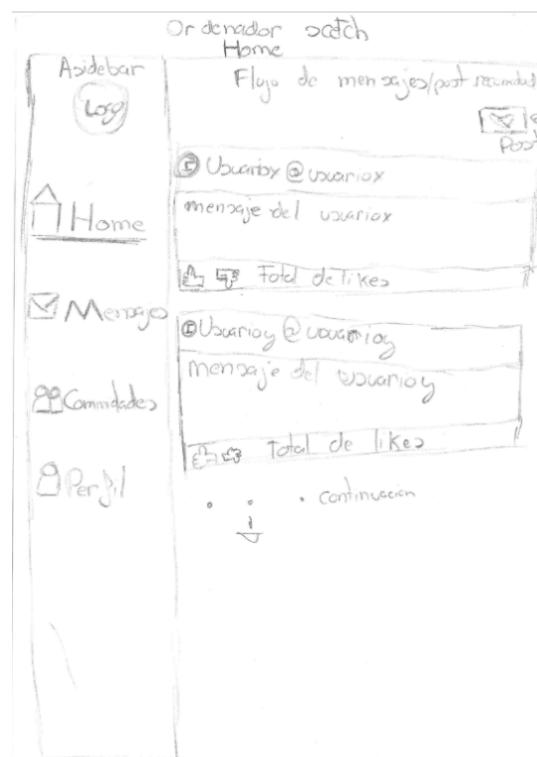
→ Login:



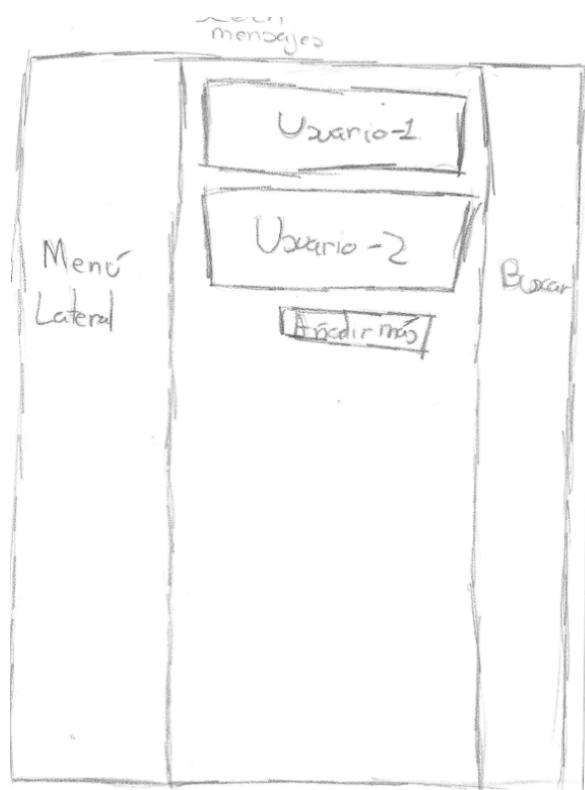
→ Register:



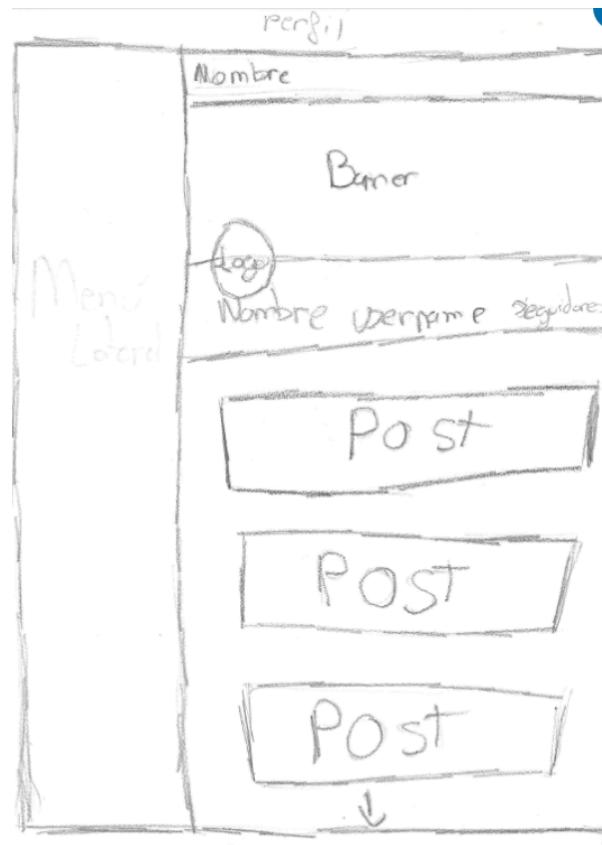
→ Home:



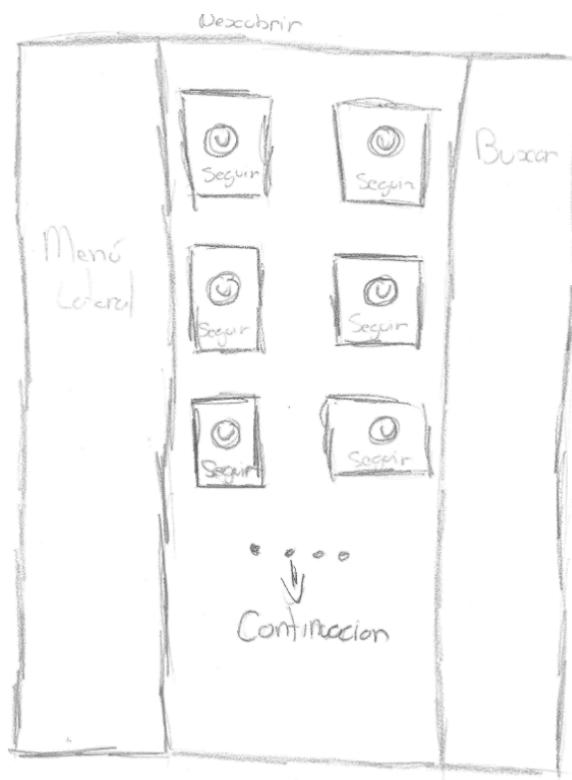
→ Mensajes:



→ Perfil:



- Descubrir:



- Comunidades:



❖ Wireframe:

→ Login:

COLOR DE FONDO O ALGO ANIMADO

LOGOTIPO

FRASE DE BIENVENIDA

NOMBRE USUARIO

CONTRASEÑA

ENVIAR

→ Register:

COLOR O ALGO ANIMADO DE FONDO

REGISTRATE

NOMBRE REAL

CORREO ELECTRONICO

CONTRASEÑA

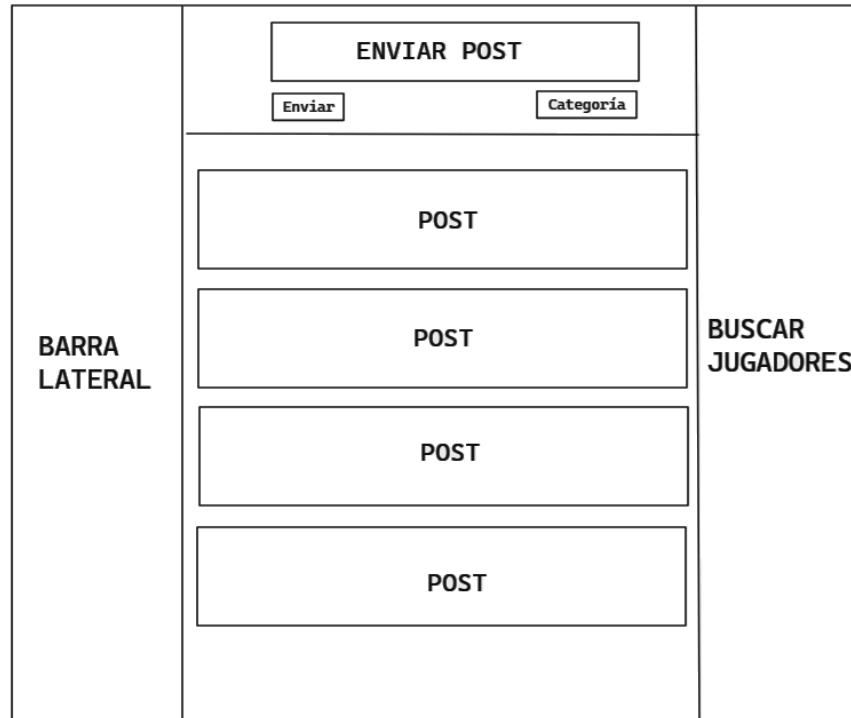
NOMBRE DE USUARIO

ESTILOS VIDEOJUEGOS

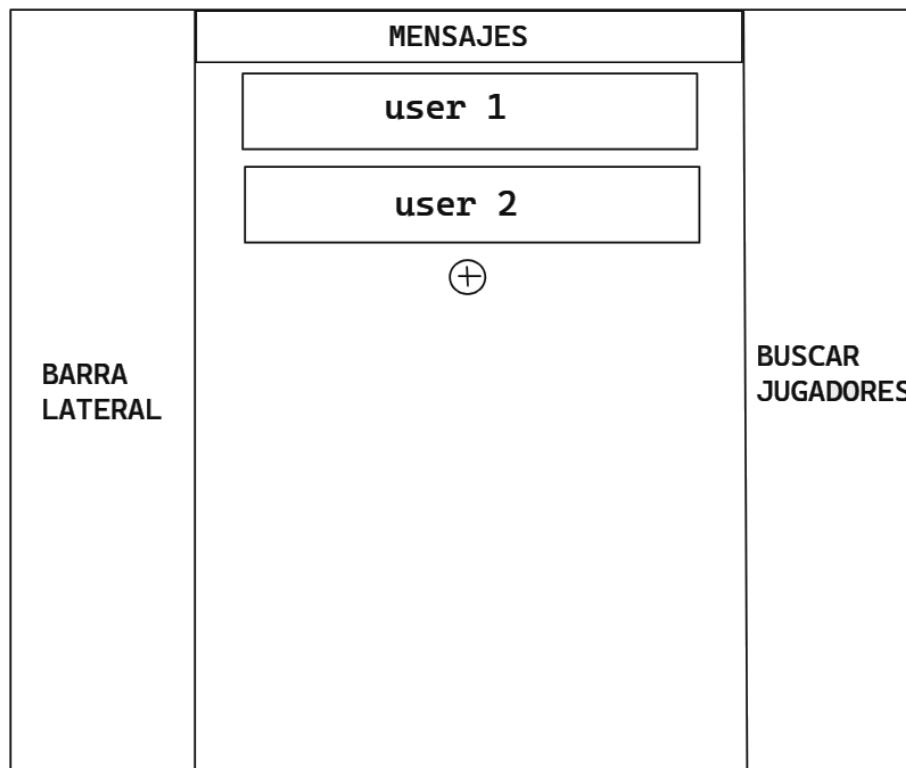
ENVIAR



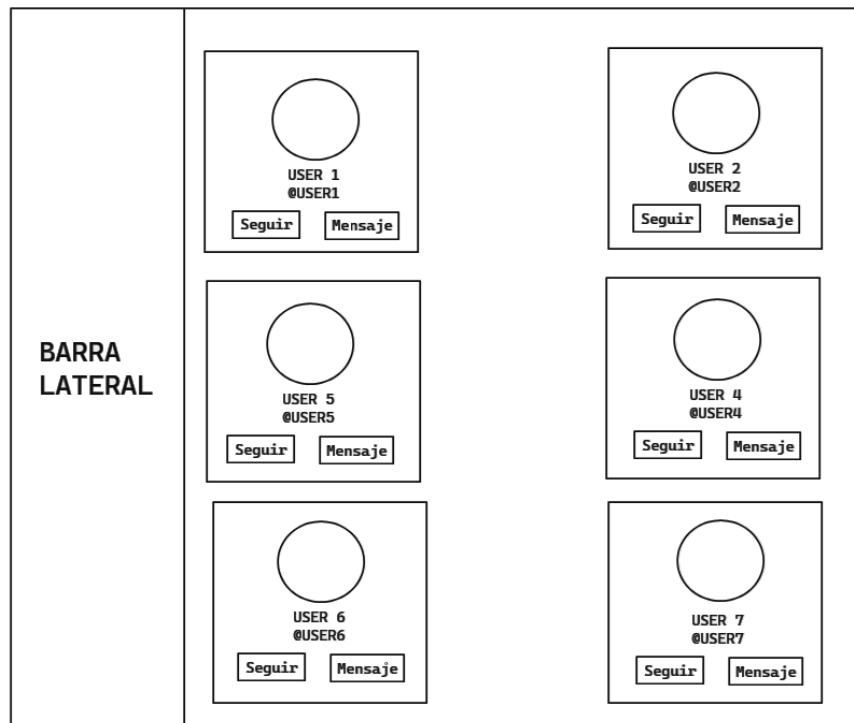
→ Home:



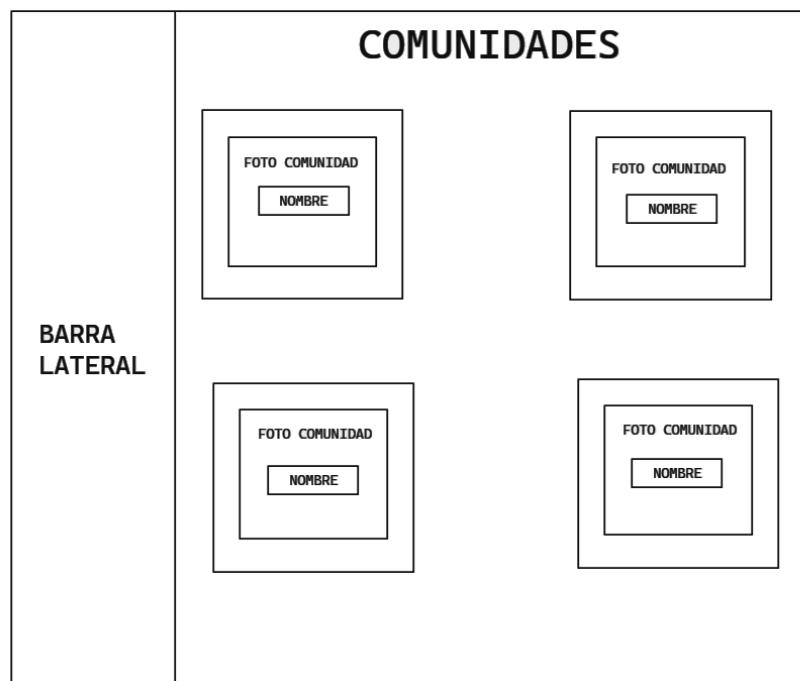
→ Mensajes:



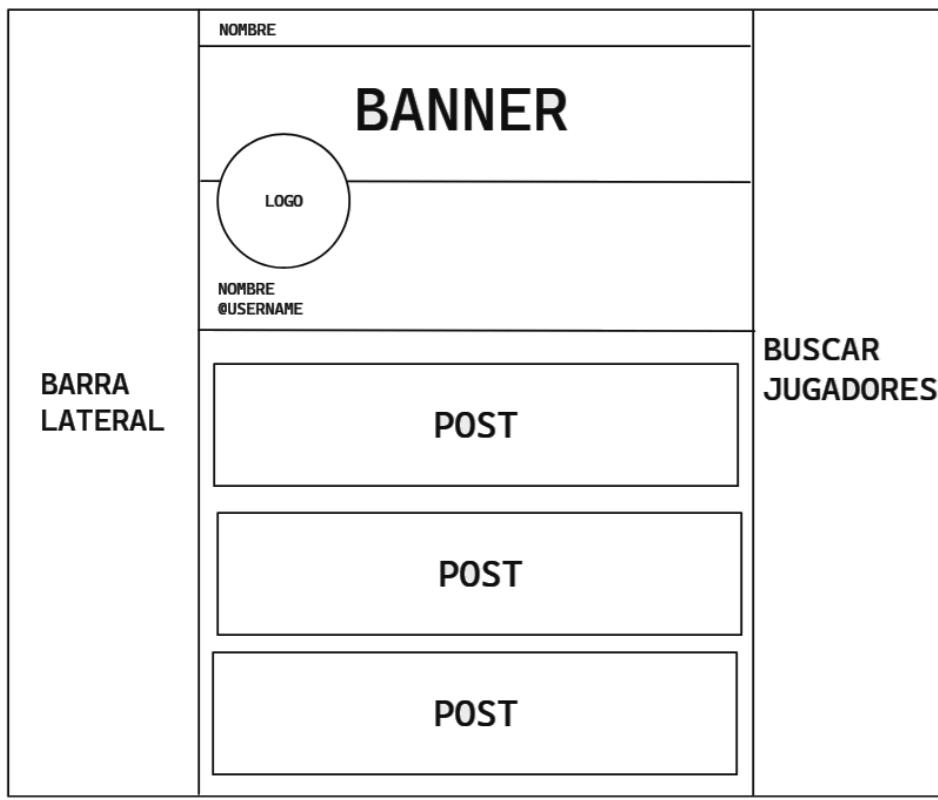
→ Descubrir:



→ Comunidades:

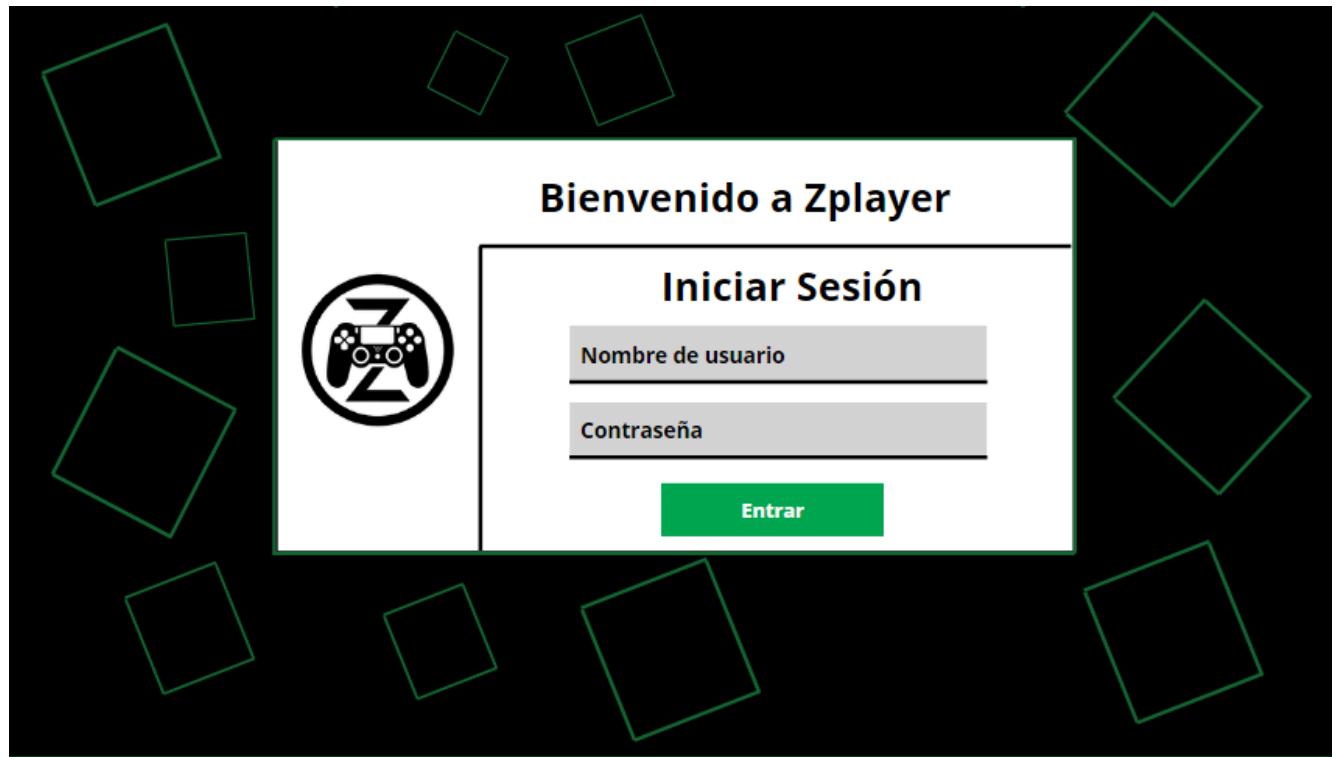


→ Perfil:

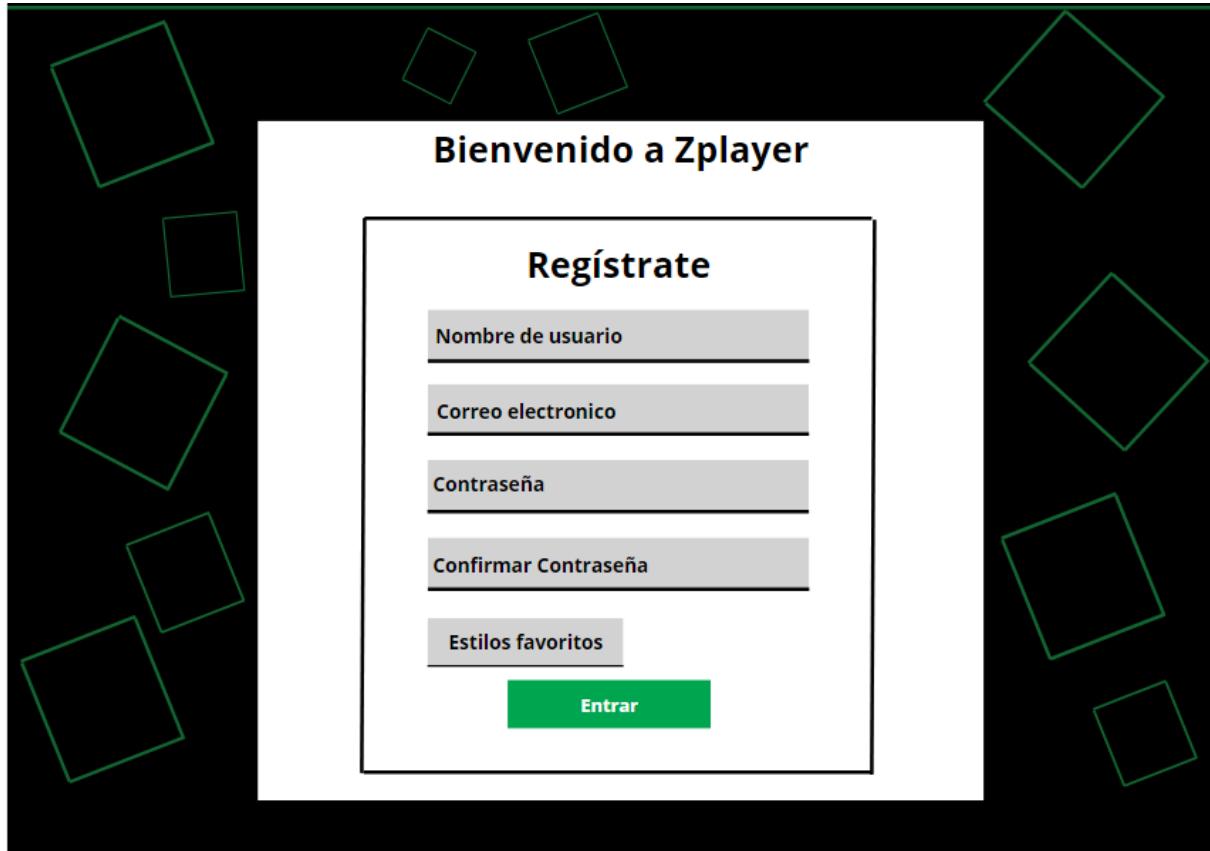


❖ Mockup:

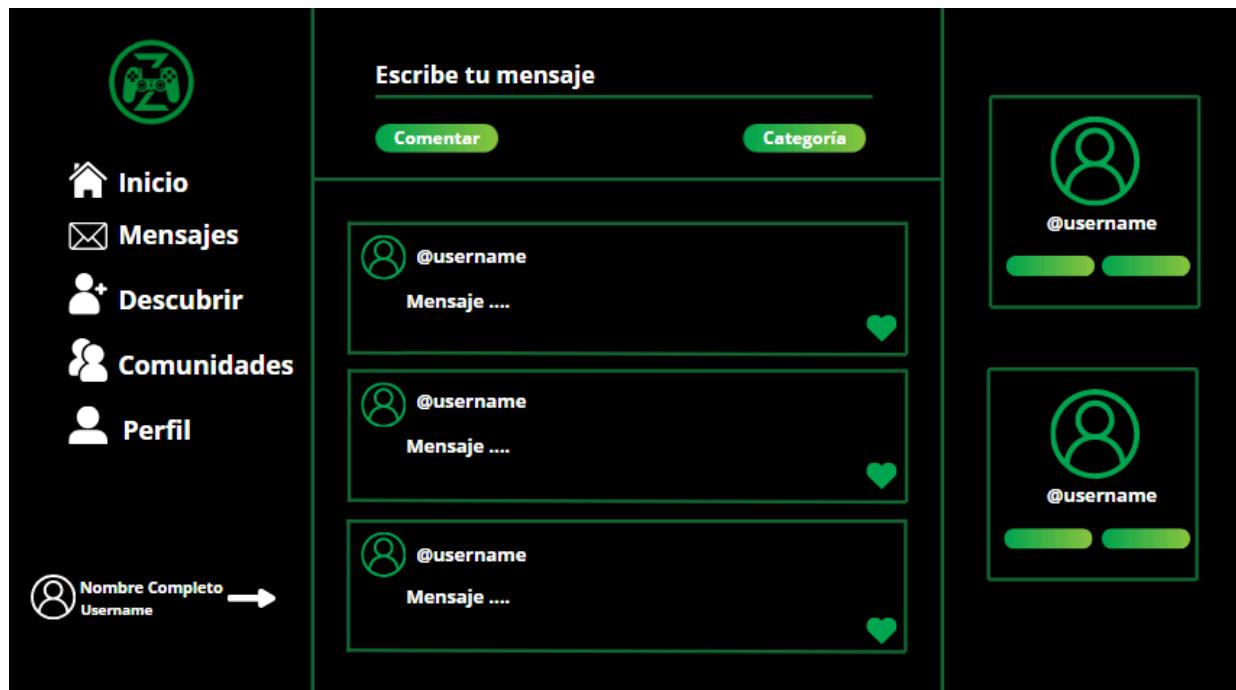
→ Login:



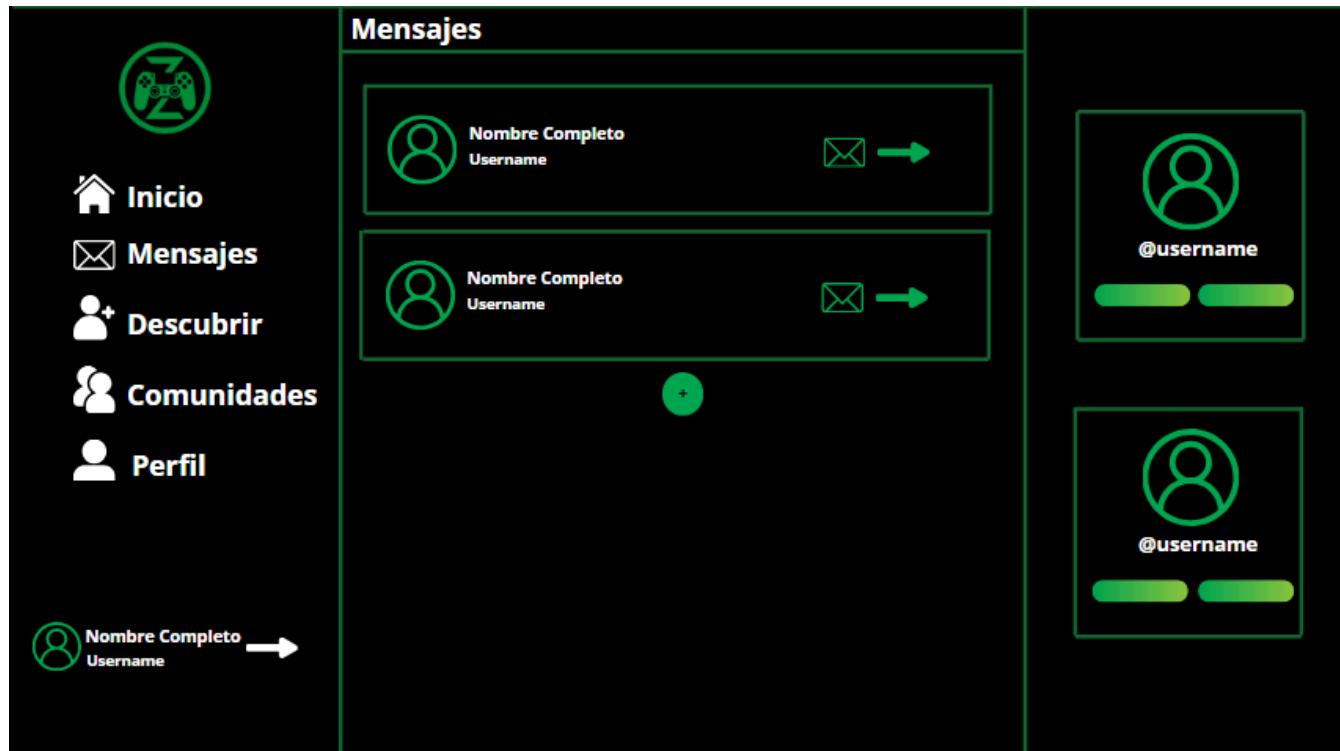
→ Register



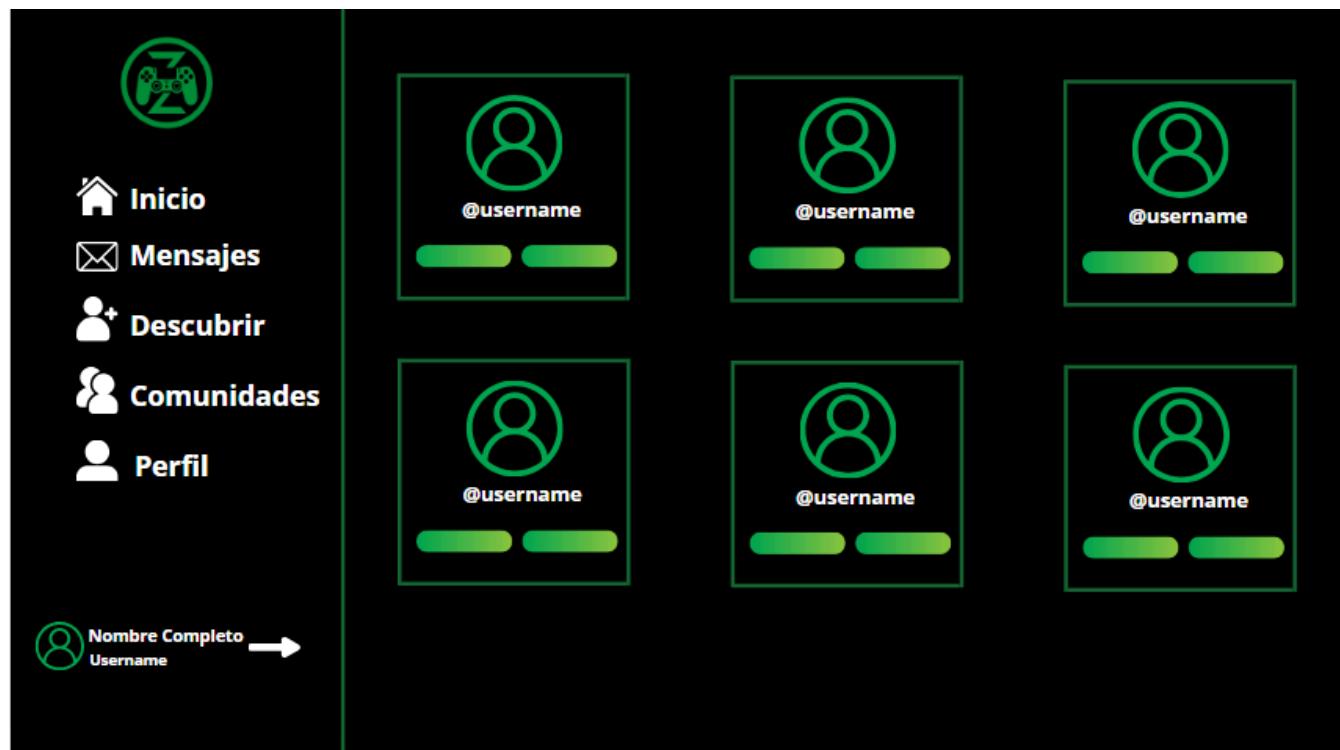
→ Home



→ Mensajes



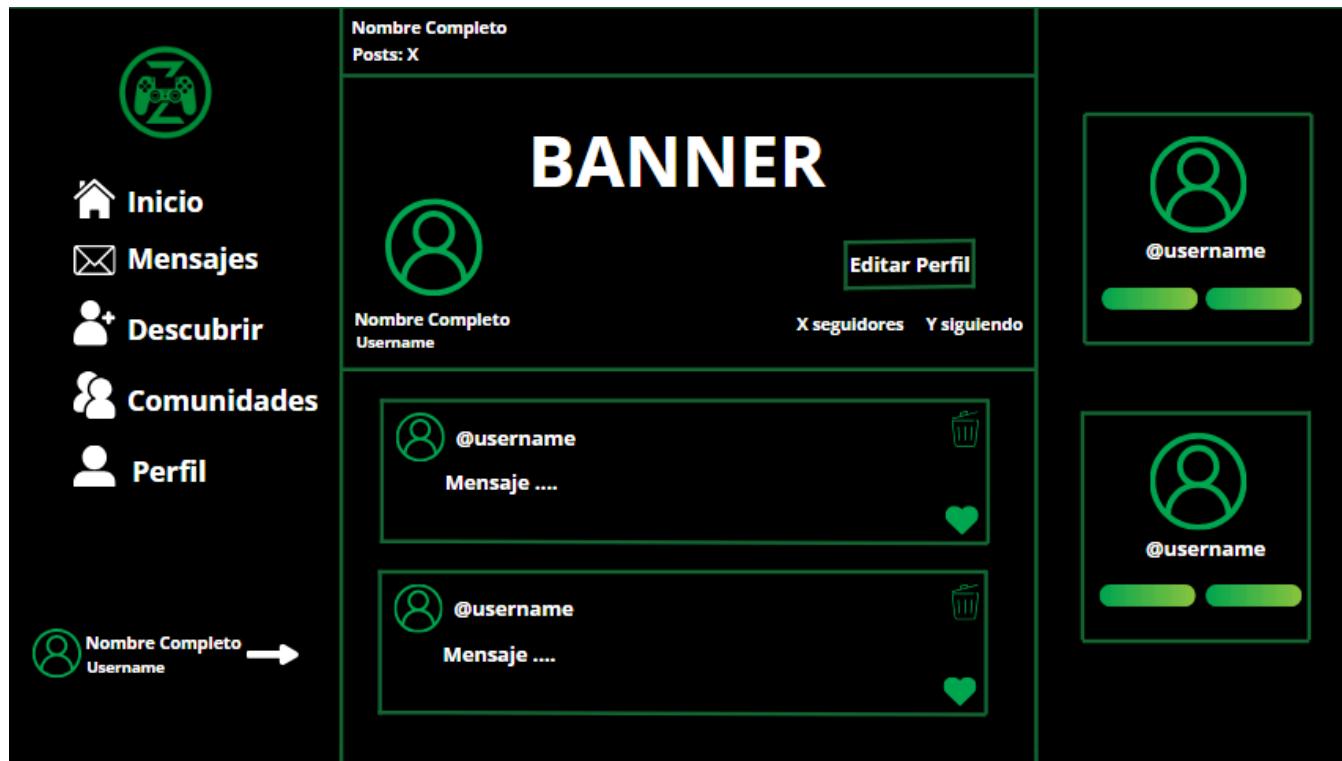
→ Descubrir



→ Comunidades



→ Perfil



5. Desarrollo y realización del proyecto

5.1. Indicadores de calidad

Para comprobar la calidad de la aplicación he considerado algunas claves, como:

- **Tasa de errores y fallos:** Medir y controlar la frecuencia y gravedad de errores en el sistema
- **Tiempo de respuesta:** Medir el tiempo promedio que tarda el sistema en responder a las solicitudes del usuario
- **Facilidad de uso:** Evaluar la usabilidad a través de pruebas de usuario y un análisis del funcionamiento de la aplicación a través de un usuario.
- **Efectividad de la autenticación:** Realizar un manejo adecuado de los mecanismos de autenticación y asegurar la seguridad de los usuarios
- **Tiempos de carga:** Medir los tiempos de carga de la aplicación

5.2 Elaboración de una batería de pruebas para detectar errores

5.2.1 Pruebas funcionales

El objetivo de estas pruebas es comprobar desde la interfaz el correcto funcionamiento de la aplicación, y comprobando la veracidad de los datos que obtenemos a lo largo de sus funcionalidades.

Se realizarán las siguientes pruebas manuales desde la interfaz:

Registro

- Accedemos a la pantalla de Registro, enviamos el formulario sin llenar ningún campo.

Resultado: alert avisando que rellene todos los campos adecuadamente.



- Accedemos a la pantalla de Registro, añadimos un correo electrónico no válido
Resultado: mensaje en rojo avisando que no es de tipo correo
- Accedemos a la pantalla de Registro, añadimos una contraseña de menos de 7 caracteres
Resultado: mensaje en rojo avisando que la contraseña no puede tener menos de 7 caracteres
- Accedemos a la pantalla de Registro, hacemos click sobre uno de los campos pero no lo rellenamos.
Resultado: mensaje en rojo avisando que ese campo es obligatorio rellenarlo
- Accedemos a la pantalla de Registro, tratamos de crear un usuario con un correo electrónico anteriormente registrado
Resultado: alert avisando de que ese correo ya está vinculado a otra cuenta anteriormente registrada
- Accedemos a la pantalla de Registro, tratamos de registrar un usuario poniendo valores distintos en los campos de contraseña y confirmar contraseña
Resultado: mensaje en rojo avisando que las contraseñas no coinciden
- Accedemos a la pantalla de Registro, rellenando correctamente los campos de registro y teniendo éxito insertándose a la base de datos
Resultado: redirección a la página de login para poder iniciar sesión con el usuario anteriormente creado

Login

- Accedemos a la pantalla de Login, enviamos el formulario sin llenar ningún campo.
Resultado: alert avisando que rellene todos los campos adecuadamente.
- Accedemos a la pantalla de Login, y ponemos un usuario o una contraseña que no correspondan a un usuario anteriormente creado
Resultado: alert avisando que las credenciales son incorrectas



- Accedemos a la pantalla de Login, rellenando correctamente los campos de Login y teniendo éxito comprobando los datos en la base de datos

Resultado: redirect al componente Home

Logout

- Accedemos a un componente, después de haber realizado el login correctamente, y pulsamos al botón de logout.

Resultado: redirect al login y borrado de sessionStorage

Home

- Acceso a la ruta, sin haber iniciado sesión

Resultado: redirect al login

- Acceso al componente home , una vez iniciado sesión

Resultado: carga correcta de los posts, carga correcta de los usuarios a la derecha par buscar, carga correcta de los datos de el usuario (logo, nombre, username)

- Acceso al componente home , envío de post sin añadir una categoría

Resultado: envio correctamente del post con la categoría “general”, y carga automática del post en su contenedor

- Acceso al componente home , dar like a un post anteriormente subido

Resultado: se suma 1 a la cantidad de likes del post y se rellena el icono

- Acceso al componente home , dar like a un post anteriormente subido y que este usuario ya le había dado like

Resultado: se resta 1 a la cantidad de likes del post y se vacía el icono

- Acceso al componente home , y entrar en la pestaña de para ti, habiendo seguido anteriormente a un usuario

Resultado: vemos los posts de las personas que seguimos

- Acceso al componente home , y entrar en la pestaña de para ti, sin seguir a nadie

Resultado: vemos un mensaje diciendo que no ha encontrado posts asociados a mis seguidores y que siga a gente



- Acceso al componente home , scrolleo 20 post y compruebo la funcionalidad de la carga progresiva o scroll infinito
Resultado: vemos que al scrolllear 20 post cargan los siguientes 20
- Acceso al componente home , usando el panel de la derecha para entrar en el perfil de un usuario que no sigo actualmente
Resultado: vemos correctamente la página del usuario
- Acceso al componente home , usando el panel de la derecha para seguir a un usuario
Resultado: seguimos al usuario correctamente y desaparece del panel de la derecha
- Acceso al componente home , usando el panel de la derecha para mandar un mensaje a un usuario
Resultado: Accedemos correctamente al componente de mensajes privados del usuario
- Acceso al componente home , scrolleamos hacia abajo en los post y comprobamos que podemos volver arriba con el botón flecha
Resultado: Volvemos al principio de los posts
- Acceso al componente home , enviamos un post con cualquier otra categoría
Resultado: Se publica correctamente el post con la categoría asignada

Mensajes

- Acceso a la ruta, sin haber iniciado sesión
Resultado: redirect al login
- Acceso al componente mensajes , anteriormente no hemos chateado con nadie
Resultado: En mensajes no nos aparecerá nadie
- Acceso al componente mensajes , pulsamos sobre el botón + para chatear con alguien
Resultado: Nos aparece la gente que seguimos y aún no hemos chateado con ellos
- Acceso al componente mensajes , y clicamos sobre un usuario.
Resultado: Nos lleva a sus mensajes privados



- Acceso al componente mensajes , y comprobamos que las funcionalidades de la derecha.

Resultado: Funcionan igual que en el componente Home

Mensajes Privados

- Acceso a la ruta, sin haber iniciado sesión

Resultado: redirect al login

- Acceso al componente mensajes privados, aún no hemos enviado nada al usuario.

Resultado: No aparecerá ningún usuario aún, arriba veremos correctamente los datos con quien estamos chateando

- Acceso al componente mensajes privados, enviamos un mensaje.

Resultado: El mensaje se enviará correctamente y se colocará en la parte derecha del chat

- Acceso al componente mensajes privados, click sobre el usuario

Resultado: Nos manda a su perfil

Descubrir

- Acceso a la ruta, sin haber iniciado sesión

Resultado: redirect al login

- Acceso al componente descubrir

Resultado: si hay usuarios en la aplicación y no los sigues aparecen correctamente

- Acceso al componente descubrir, buscas un username

Resultado: si un usuario contiene en el username el contenido que has buscado aparece correctamente

- Acceso al componente descubrir, haces click sobre el icono o nombre de un usuario

Resultado: te dirige correctamente al perfil de el usuario

- Acceso al componente descubrir, haces click sobre el botón de seguir

Resultado: sigues correctamente al usuario y desaparece del componente de descubrir.

- Acceso al componente descubrir, haces click sobre el botón de mensaje

Resultado: te redirige correctamente al componente de mensaje privado



Comunidades

- Acceso a la ruta, sin haber iniciado sesión
Resultado: redirect al login
- Acceso al componente comunidades, después de hacer login correctamente
Resultado: aparición correctamente de las comunidades de la aplicación
- Acceso al componente comunidades, hacer click sobre una comunidad
Resultado: redirección correctamente a la comunidad clicada
- Acceso al componente comunidad, sin haber iniciado sesión
Resultado: redirect al login
- Acceso al componente comunidad, dar like a un post de la comunidad anteriormente subido
Resultado: se suma 1 a la cantidad de likes del post y se rellena el icono
- Acceso al componente comunidad, dar like a un post de la comunidad anteriormente subido y que este usuario ya le había dado like
Resultado: se resta 1 a la cantidad de likes del post y se vacía el icono
- Acceso al componente comunidad, enviar un post
Resultado: se envía un post con la categoría de la comunidad en la que estés enviando un post
- Acceso al componente comunidad, barra lateral funciones básicas
Resultado: mismo funcionamiento que en otros componentes

Perfil

- Acceso a la ruta, sin haber iniciado sesión
Resultado: redirect al login
- Acceso al componente perfil, habiendo iniciado sesión
Resultado: muestra correctamente los posts del usuario, de 20 en 20 para conseguir una carga progresiva, carga de datos personales correcta.
- Acceso al componente perfil, borrar post
Resultado: borra correctamente el post seleccionado



- Acceso al componente perfil, dar like a un post
Resultado: se suma 1 a la cantidad de likes del post y se rellena el icono
- Acceso al componente perfil, dar like a un post y que ya le había dado like
Resultado: se resta 1 a la cantidad de likes del post y se vacía el icono
- Acceso al componente perfil, click botón editar perfil
Resultado: muestra correctamente el modal para cambiar los datos del usuario
- Acceso al componente perfil, cambiar campo
Resultado: cambia correctamente cualquier campo del usuario
- Acceso al componente perfil, barra lateral funciones básicas
Resultado: mismo funcionamiento que en otros componentes

5.2.2 Otras Pruebas

Mientras se desarrollaba la aplicación y se implementan nuevas funcionalidades o se modificaban las ya existentes, se han ido ejecutando pruebas para comprobar que no se han producido nuevos errores por culpa de esta nueva implementación, o también controlar que por culpa de esta nueva implementación falle otra cosa de ese componente.

5.3 Evaluación y solución de incidencias

Lo que más problemas me ha dado problemas a la hora de desarrollar la aplicación ha sido sobre todo el cambio de imágenes de los usuarios, ya que ocurrían problemas con las rutas a la hora de cargar de vuelta la imagen del usuario, la solución fue cada vez que se cambiaba de icono o de banner un usuario, se cambiaba la ruta en la base de datos y cuando desde el front quería utilizar esa imagen , lanza una petición al back solicitándola.

Otros de los grandes problemas que he experimentado durante el desarrollo ha sido al principio de este, ya que tenía problemas a la hora de que el front se comunicara con el back, ya que ocurrían bastante errores del CORS.



5.4- Evaluación y seguimiento de las actividades

Actualmente en la aplicación se ha encontrado un pequeño bug que consiste en que cuando scrolleas hacia abajo y le das me gusta a un post , en vez de seguir donde estabas, te manda arriba, esto sucede por que al tener implementado la carga progresiva que permite cargar posts de 20 en 20, al dar like a un post se vuelven a cargar los datos, y vuelve a la página 1 (primeros 20 posts), se buscará solución a este problema lo más rápido posible ya que esto puede ser molesto para los usuarios y empeorar la experiencia dentro de la aplicación.

6- Implantación del Proyecto

6.1- Plan de implantación

Este plan de implantación describe los pasos necesarios para implementar la aplicación Zplayer en la plataforma Vercel utilizando el plan gratuito. Vercel es una plataforma de alojamiento de aplicaciones en la nube que ofrece un servicio rápido, escalable y fácil de usar, al ser un plan gratuito tiene bastantes inconvenientes:

- **Limitaciones de ancho de banda:** El plan gratuito tiene un límite de ancho de banda de 10 GB mensuales.
- **Función de caché limitada:** El plan gratuito tiene una función de caché limitada que puede afectar el rendimiento de la aplicación.
- **Soporte limitado:** El plan gratuito no incluye soporte técnico prioritario.
- **Marca de Vercel:** La marca de Vercel se muestra en la interfaz de la aplicación.

Al no tener comprado un dominio y usar también el gratuito de vercel tiene una url rara y poco reconocible, esto en un futuro debería pasar a un plan de pago si la aplicación funciona bien, aunque aunque tenga cosas negativas, nos deja alojar nuestra web gratuitamente hasta que tengamos el capital necesario.

6.2 Manual de instalación

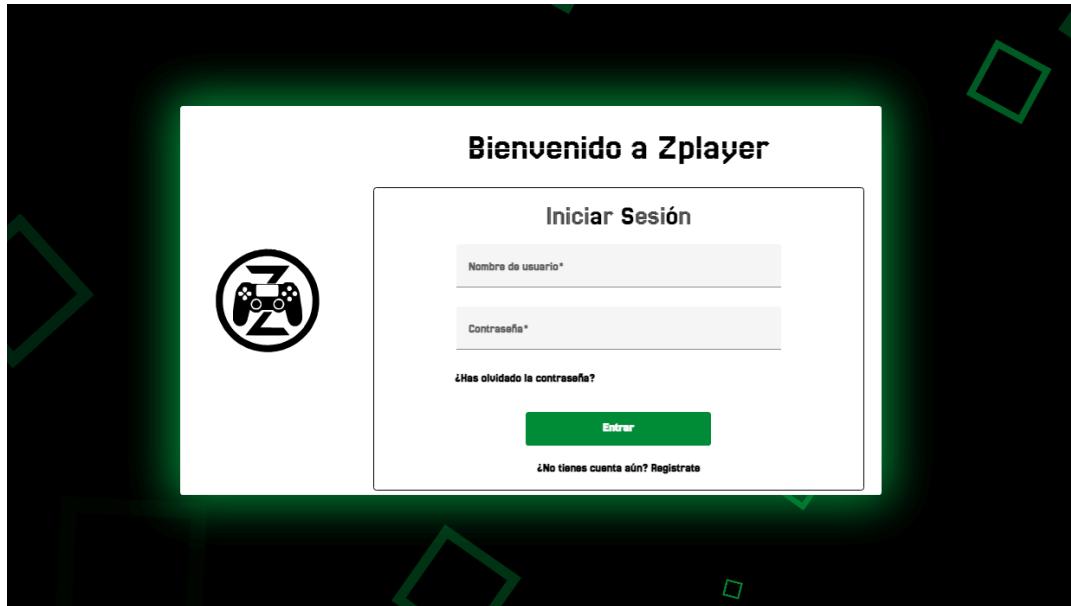
Para un usuario normal y corriente no será necesario ningún tipo de instalación ya que solo necesitarán acceder al dominio para poder usar la aplicación



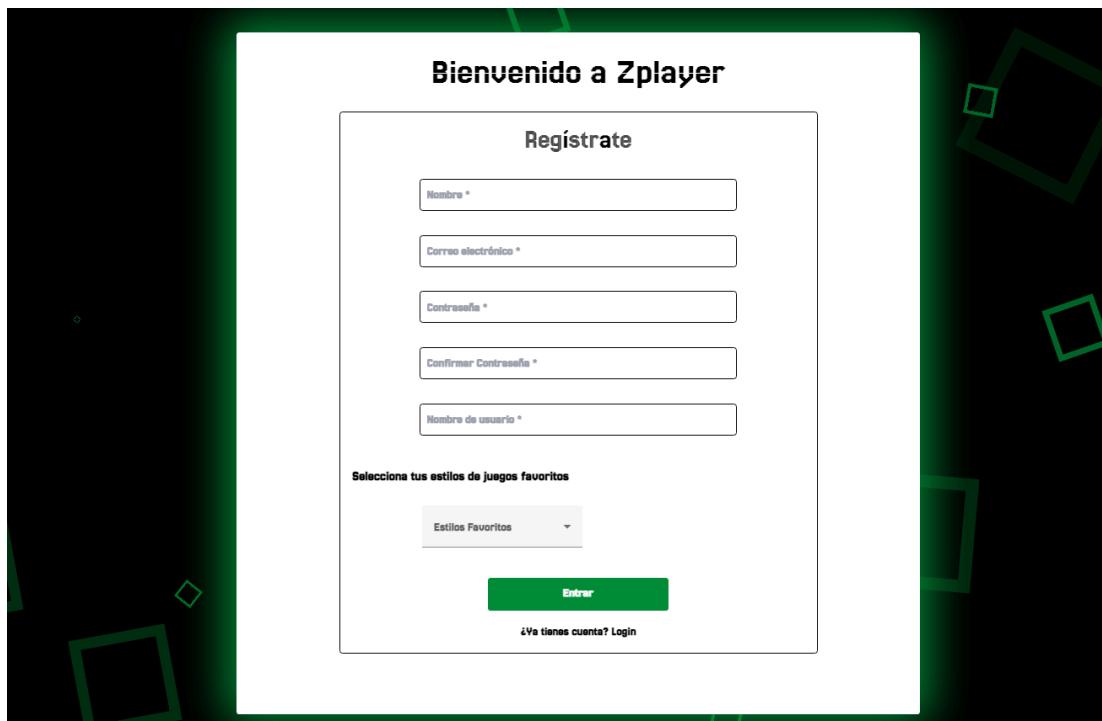
6.3 Manual de usuario

A continuación se creará un manual de usuario, para explicar visualmente el funcionamiento de la aplicación web.

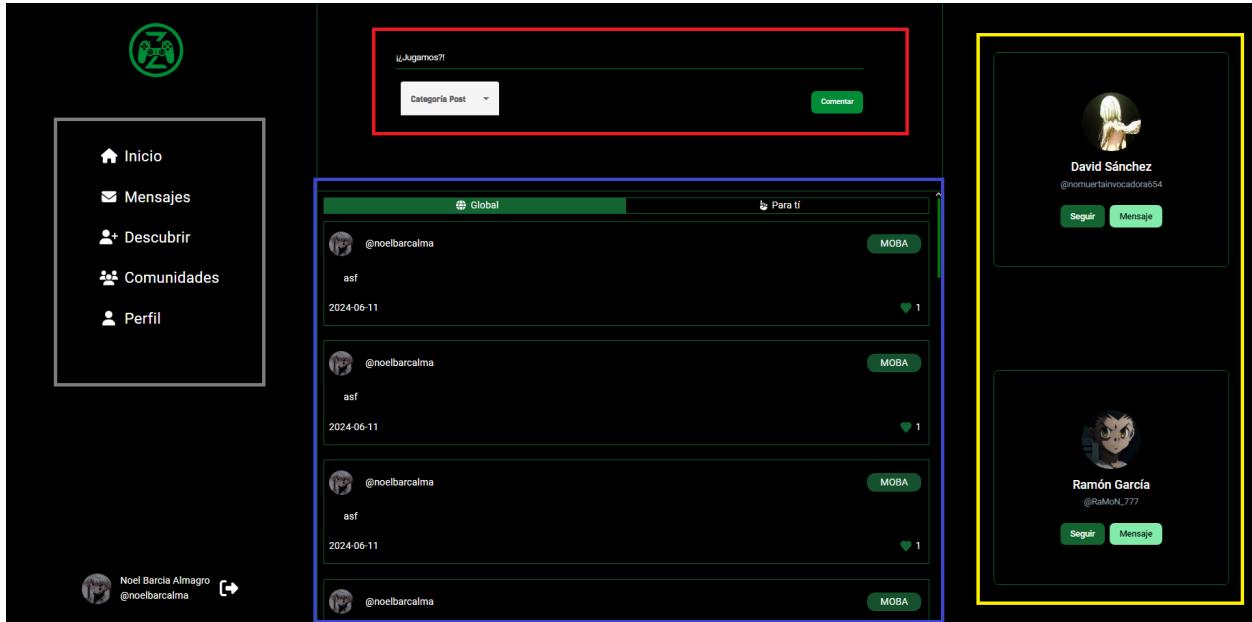
Accedemos a la web y en primera instancia nos dirigirá al componente login, donde un usuario ya creado podrá acceder a la aplicación, si es un usuario nuevo y no tiene cuenta podrá hacer click en: **¿No tienes cuenta aún? Regístrate?**.



Le llevará a una pestaña register donde podrá crearse un usuario nuevo



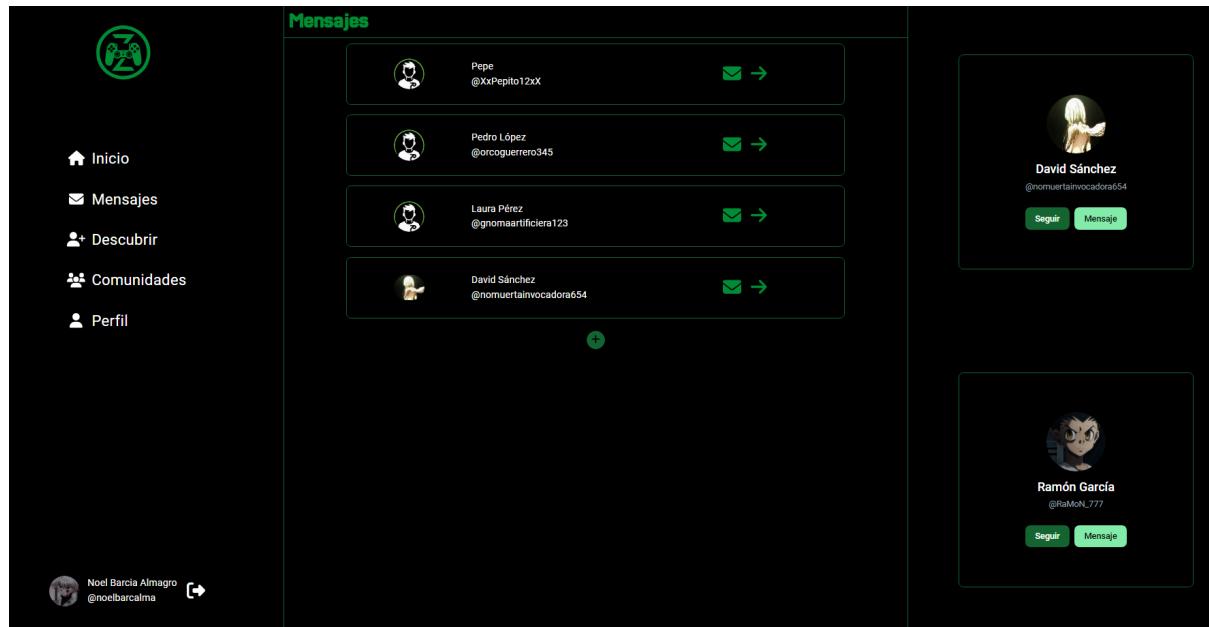
Cuando el usuario ya tenga una cuenta y haga hecho el login entrará al home en el que verá lo siguiente.



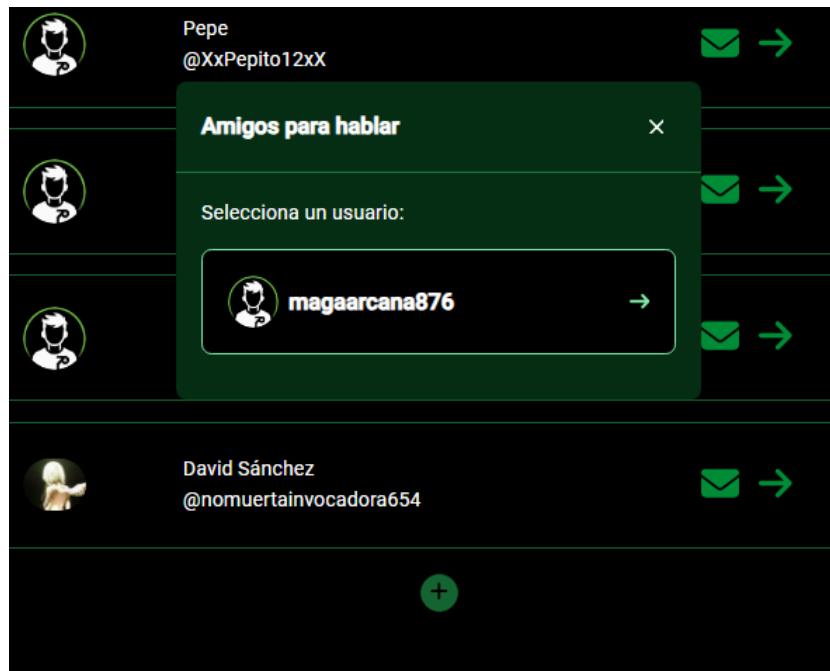
- Rojo: Apartado donde el usuario podrá poner post sobre los temas que quiera, justo debajo un select para poner la categoría del post, si no pone se guarda con la categoría “general”, y el botón de enviar
- Azul: Apartado donde cargarán los posts de la aplicación, además el usuario podrá darle me gusta en el lado derecho del post, este apartado tiene 2 opciones, el chat global donde aparecerán todos los post de todas las categorías, y el chat para ti donde aparecerán todos los post de los usuarios a los que sigues
- Amarillo: Apartado donde nos aparecerán aleatoriamente 2 usuarios que no seguimos aún, haciendo click sobre ellos nos llevarán a sus perfiles, también podremos seguirles dandole al botón de seguir, cuando sigamos al usuario desaparecerá de la barra lateral, también podremos ir a chatear con un usuario dandole al boton de mensaje
- Gris: Barra lateral por la que nos moveremos por la aplicación y como última opción podremos cerrar sesión con ese usuario

Vamos a ir a ver los mensajes dándole al botón de la barra lateral “Mensajes”

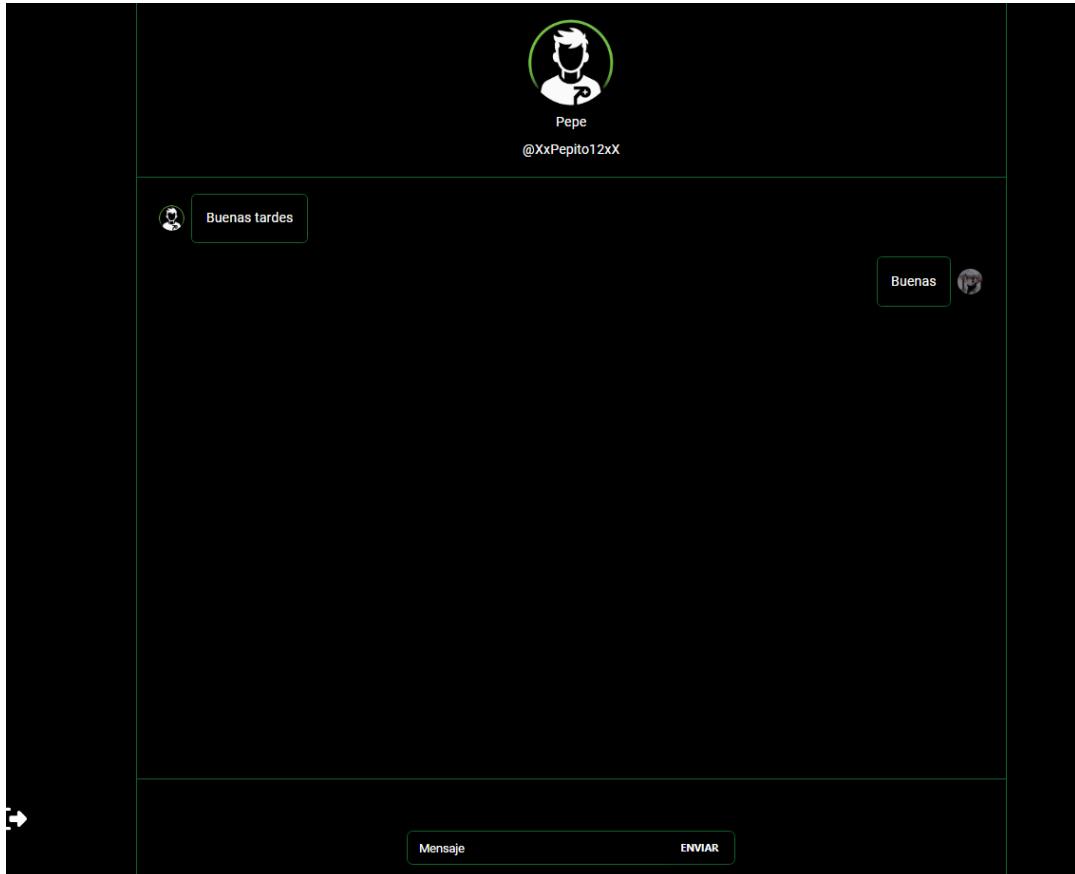




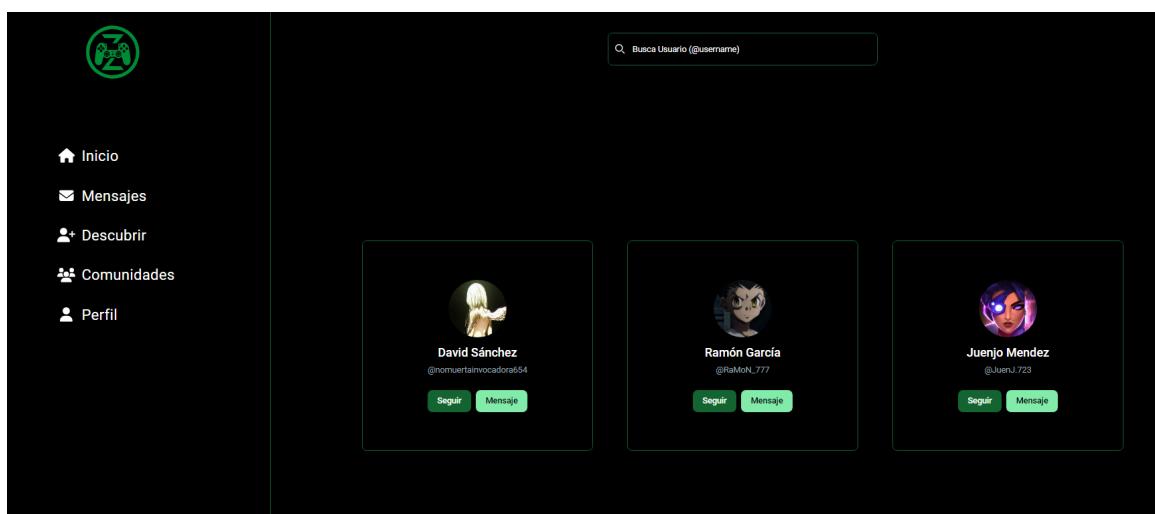
Podemos observar en el centro unos rectángulos con usuarios con los que he chateado anteriormente, pulsando sobre estos nos dirige al chat privado donde podremos chatear con ellos, pulsando sobre el botón del más, podremos seleccionar usuarios que sigues pero no has chateado con el aún.



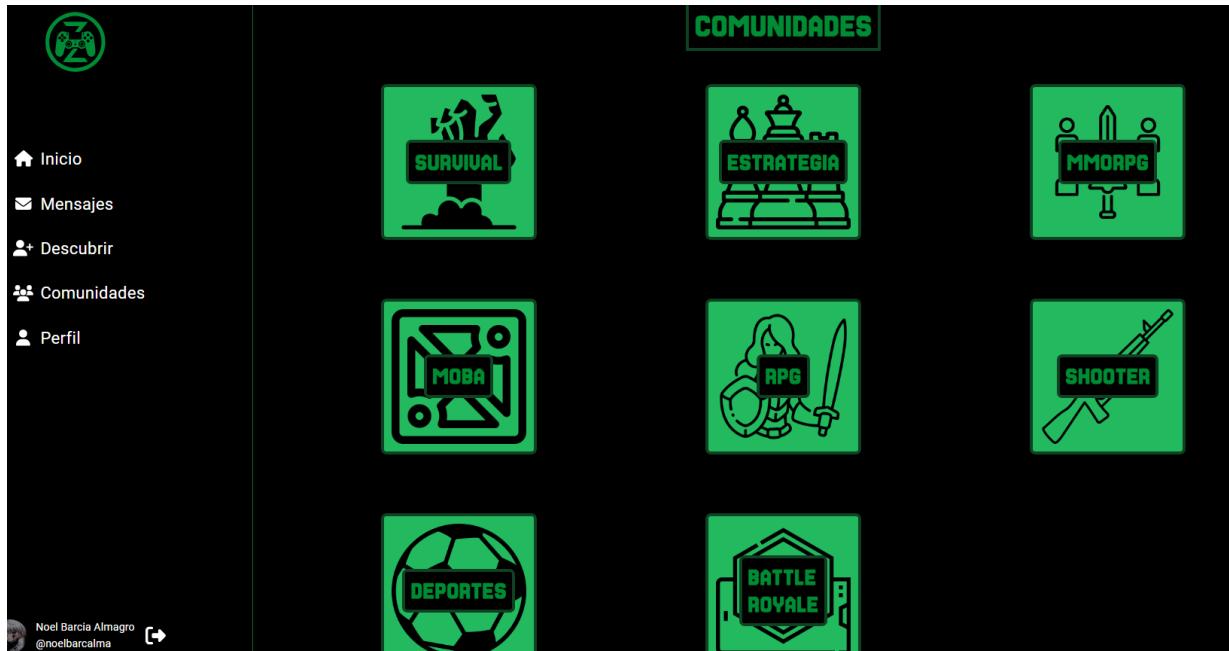
Cuando hagamos click sobre un usuario entraremos en esta pestaña donde podremos chatear con el usuario que hayamos hecho click.



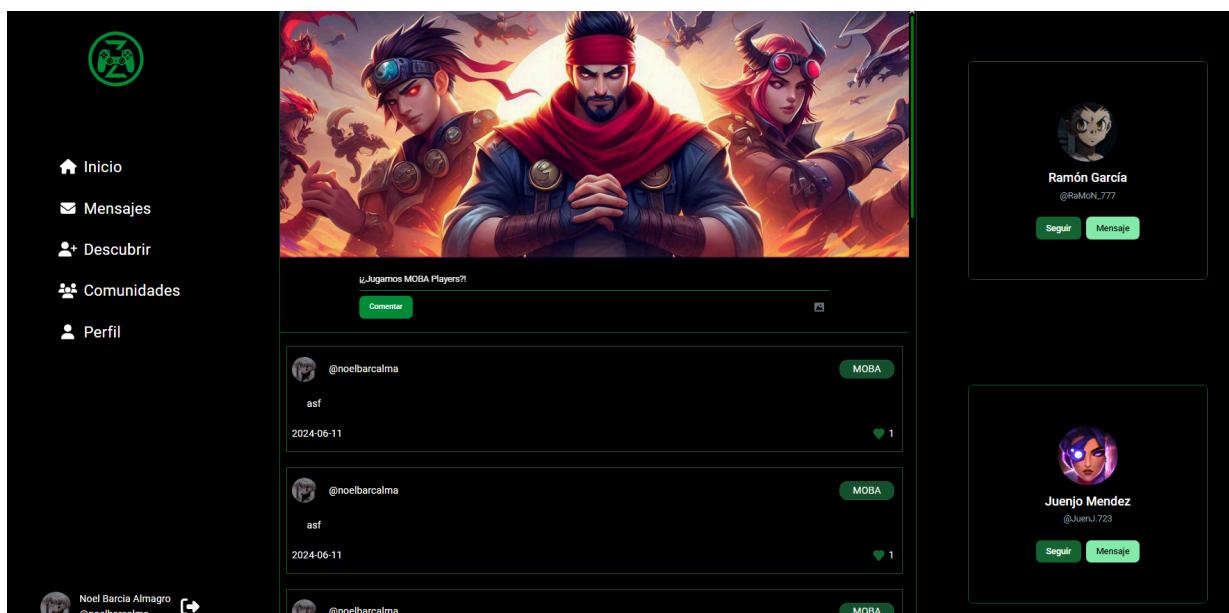
Si en el menú lateral hacemos click sobre el botón de descubrir nos llevará a la siguiente ventana, donde nos aparecerán aquellos usuarios que no seguimos, podremos seguirlos, ver su perfil y enviarles mensajes directos



Vamos a ver otra opción del menú lateral haciendo click sobre comunidades, y vemos lo siguiente, cada comunidad en un cuadrado y un icono representadolo, si hacemos click sobre uno de estos nos llevará a su componente personalizado



Por ejemplo, si vamos a la comunidad de moba por ejemplo nos lleva a la siguiente ventana, donde podremos poner posts de esta comunidad.



Y por último vamos a observar la ventana de perfil, donde el usuario podrá ver su información, como su nombre , nombre de usuario, seguidores, siguiendo, también podrá ver sus posts, borrarlos pulsando el botón de la papelera, darles me gusta, y editar su perfil.

Pulsando el botón de editar perfil nos aparecerá esta ventana donde podremos cambiar algunos datos del usuario



7. Conclusiones Finales

7.1. Evaluación del proyecto: objetivos conseguidos y no alcanzados.

En cuanto a los objetivos conseguidos, en lo personal me he defendido e incluso he conseguido un nivel medio/alto mediante la realización de este proyecto, pese a que antes de empezarlo no conocía ninguno de los lenguajes ni frameworks utilizados.

El objetivo principal de la aplicación se ha conseguido ya que se ha creado y puesto en funcionamiento la red social que tenía en mente desarrollar, creo que se le podrían haber añadido muchas más funcionalidades a la aplicación, pero el tiempo ha sido un problema a la hora del desarrollo, ya que he tenido que comprender y manejar los lenguajes, frameworks y herramientas antes de empezar el desarrollo, ya que no tenía conocimientos previos sobre estos.

7.2 Propuestas de mejora

Actualmente, la aplicación no está terminada ya que hay cosas que me hubiera gustado implementar si hubiera tenido más tiempo, y que probablemente implementaré en un futuro, algunas de estas implementaciones serían:

- **Poder enviar posts con fotos y videos:** esto podría añadir más dinamismo a la página y más opciones al usuario
- **Implementar una carga progresiva hacia arriba en los mensajes privados:** así si 2 usuarios tienen muchos mensajes entre ellos, solo cargaría una parte de ellos.
- **Implementar una carga progresiva en descubrir usuarios:** si existen muchísimos usuarios en mi aplicación, en el apartado de descubrir podrían ocurrir problemas de rendimiento a la hora de mostrarlos.
- **Mejorar la interfaz de los mensajes privados:** se podrían agregar más cosas como las horas a las que mandas los mensajes
- **Implementar mensajes de voz:** no tengo conocimientos previos de como hacer esto pero le daría más “vida” a la aplicación

