FULL
Pablo Hermida Gómez

2023-2024
Antonio Otero

INSTITUTO NEBRIJA | Formación Profesional

ABSTRACT

Aplicación en la que se podrán vivir diferentes experiencias a través de portales. Dicha aplicación será multijugador para la mayor diversión de los usuario al igual que contiene diferentes herramientas para la socialización dentro de la misma.

Existirá también una página web donde las empresas podrán reservar vallas y carteles publicitarios dentro del juego para poder financiar el mantenimiento de la aplicación.

Application in which you can live different experiences through portals. This application will be multiplayer for the greatest enjoyment of the users as well as containing different tools for socialization within it.

There will also be a website where companies can reserve billboards and advertising posters within the game to finance the maintenance of the application.

Índice de contenido

INTRODUCCIÓN

Mi TFG es una aplicación donde puedes vivir una segunda vida de la forma que quieras. Es una vida de fantasía, puedes esquiar por una pirámide, jugar al fútbol boca abajo o hacer lo que quieras. Mi intención es dar a la gente la oportunidad de vivir otra vida más entretenida y divertida que la suya propia. Hoy en día es mucho más fácil que antes. Se ha demostrado que las gafas VR es un dispositivo mucho más inmersivo que otros. (Edstutia, 2023)

El problema para las personas que quieren realizar cualquier tipo de sueño o actividad que no pueden hacer en la vida real es principalmente que no tienen una plataforma o un lugar para llevar a cabo estos sueños o acciones. Mi aplicación puede proporcionar ese tipo de espacio donde pueden estar y hacer lo que quieran. El mercado no tiene muchas aplicaciones como mi idea aparte de *"Second live"* sin embargo mi aplicación se basa en VR cuando *"Second live"* es sólo para 3D y se enfoca en un aspecto más realista que fantástico. Por un lado *"Second Live"* es una app donde puedes vivir como en la tuya pero virtualmente y por otro lado mi app es una forma de vivir una segunda vida aparte de la tuya pero donde puedes hacer cosas que no podrías hacer en tu vida real.

La mayoría de las personas desean hacer ciertas cosas diariamente que no hacen por miedo a lastimarse o lastimar a alguien o simplemente porque esa cosa ni siquiera es posible. Por eso la mayoría de la gente tiene sueños que nunca hará. Mi aplicación hace que tus sueños se hagan realidad sin que se rían de ti, en un espacio en el que no te pueden hacer daño ni puedes hacer daño a los demás y en el que tus sueños más irrealizables se hacen realidad. (Levoy, 2021)

Para lograr una experiencia lo más inmersiva posible, la aplicación está diseñada para ejecutarse en un casco de realidad virtual, pero dado que estos cascos son caros y sólo cierto tipo de personas con ingresos considerables pueden permitírselos, la aplicación también es compatible con plataformas de PC. El juego será gratuito, por lo que personas con cualquier tipo de ingresos podrán permitírselo. Los clientes pueden jugarlo en un PC o, si quieren una experiencia más inmersiva, pueden jugarlo en un casco de realidad virtual. La aplicación será apoyada financieramente a través del alquiler de pancartas publicitarias dentro de la aplicación.

OBJETIVOS

El objetivo principal es desarrollar un metaverso en el cual se pueden experimentar diferentes aventuras a través de portales. Además de poder socializar y comunicarse con los demás jugadores del metaverso. Se busca la posibilidad de que las empresas puedan publicitarse dentro del metaverso.

OBJETIVOS SECUNDARIOS:

- Una aplicación compatible con las gafas de real virtual
- Un sistema de comunicación por voz de proximidad
- Un sistema de contactos dentro del metaverso
- Diferentes portales con diferentes aventuras dentro de ellos
- Una página web dónde las empresas puedan meterse y reservar carteles publicitarios
- Un sistema de retransmisión en vivo para poder interactuar con los demás jugadores sin necesidad de estar conectado a la aplicación
- Un sistema de inventario para poder manejar todos los objetos del metaverso
- Unos servidores capaces de aguantar la carga de red de un juego multijugador al igual que la capacidad de almacenaje para los carteles publicitarios del mismo
- Sistema de registro e inicio de sesión de usuarios y empresas tanto en el metaverso como en la página web

REQUISITOS

Para la aplicación dependen de qué versión se quiera utilizar, en caso de querer utilizar la versión VR sea de requerir de unas gafas de realidad virtual. En caso de querer utilizar la versión de escritorio se requiere de un ordenador con conexión a internet, un procesador y una gráfica de gama media, almacenamiento de al menos 8 gigas y si se quiere utilizar el chat de voz un micrófono.

En caso de querer utilizar la versión web ya sea tanto para empresas como para ver los streams siendo usuario hace falta de un navegador que sea capaz de interpretar HTML CSS y javascript..

ANÁLISIS

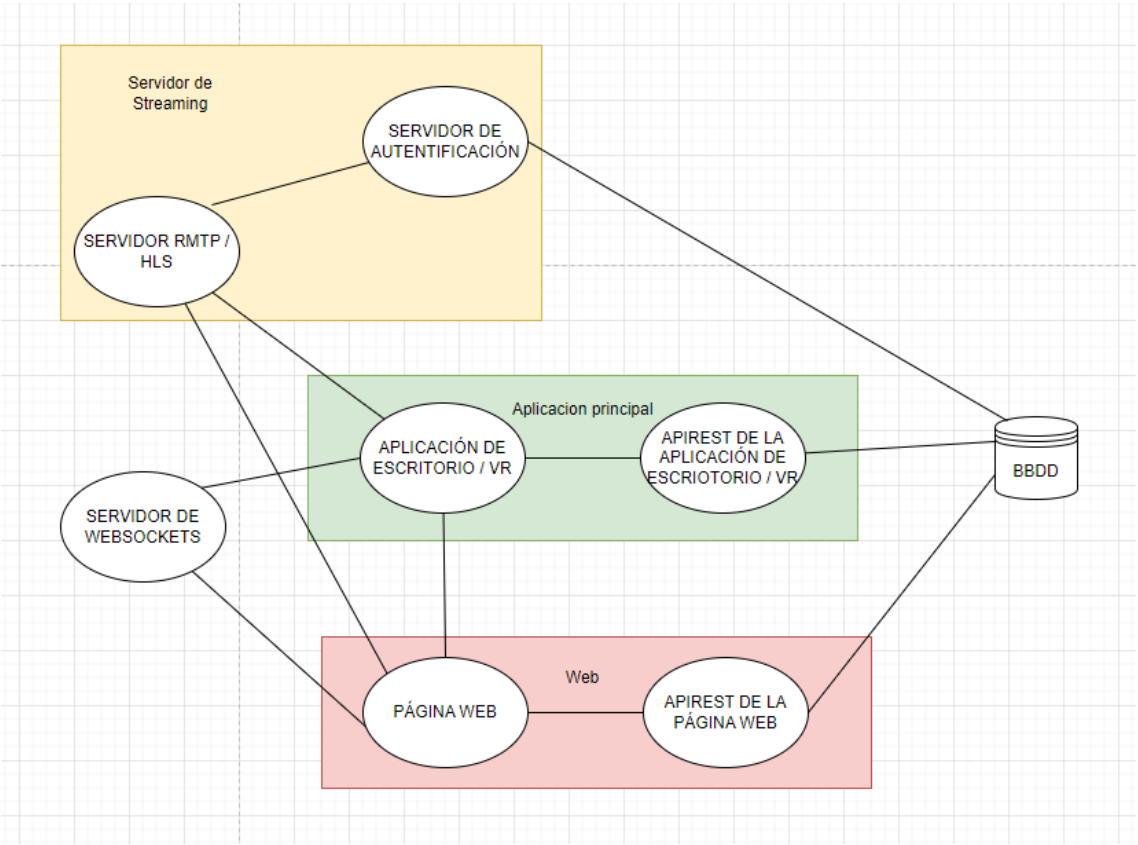
1. Servicios

El desarrollo en su conjunto está compuesto por un total de 8 servicios.

Dichos servicios son:

- **Aplicación de escritorio o aplicación para las gafas VR:** Dicha aplicación es el principal servicio del desarrollo. Es el espacio donde la gente puede conectarse y comunicarse entre ellos, viajar a diferentes mundos en los que hay características y funcionalidades únicas.
- **BackEnd de la Aplicación de escritorio o gafas de VR:** Dicho backend se encarga de manejar todos los datos que se quieren guardar modificar o borrar de la aplicación de escritorio en la base de datos.
- **Base de datos:** La base de datos es la encargada de guardar todos los datos tanto de los usuarios de la aplicación como los streamings como de las empresas de publicidad.
- **Servidor RMTP/HLS:** Es un servidor que permite a los usuarios conectarse con una aplicación de streaming y retransmitir directamente a nuestra aplicación.
- **Servidor de autorización:** Este servidor se utiliza para que la gente que intente retransmitir a nuestro servidor de streaming tengan que autenticarse antes de que la aplicación les permita retransmitir.
- **Página web:** Una aplicación web la cual permite a la gente conectarse a nuestra web para visualizar los streamings de los usuarios que estén retransmitiendo. Ademas la web servirá para que las empresas puedan reservar carteles de publicidad y poder subir carteles publicitarios.
- **BackEnd de la Web:** El backend de la web es el encargado de conectarse con la base de datos y gestionar toda la información de las empresas y de la publicidad.
- **Servidor de Websockets:** Este servidor es el encargado de proveer a todos los servicios que requieran de algún tipo de mensajería, una conexión a tiempo real.

Esquema de conexión de los diferentes servicios.



Uso de los servicios se realizaron de manera separada en vez de en la misma máquina o en el mismo entorno para si en el caso de caerse alguno de los servicios el resto seguir funcionando.

2. Arquitectura

- **Aplicación de escritorio o aplicación para las gafas VR:** Al ser una aplicación usando el motor de videojuegos “Unity”. Se usa una arquitectura por componentes.
- **BackEnd de la Aplicación de escritorio o gafas de VR:** Se utiliza la arquitectura de ApiRest por capas. Lo que nos permite un mayor control de cada capa de la aplicación
- **Servidor de autorización:** Al ser un servidor con apenas 2 endpoints para su autorización no requiere de ninguna arquitectura de desarrollo.
- **Página web:** Al usar un framework moderno se usa la arquitectura por componentes.
- **BackEnd de la Web:** Se utiliza la arquitectura de ApiRest por capas. Lo que nos permite un mayor control de cada capa de la aplicación.
- **Servidor de Websockets:** Al ser un servidor de websockets relativamente pequeño no se usó ninguna arquitectura en específico.
- **Servidor de Streaming:** Esta compuesto por 2 contenedores de Docker los cuales se lanzan usando Docker Compose.
- **BBDD:** Es una base de datos no relacional, la cual esta organizada por documentos y dentro de cada documento están los registros.

3. Tecnologías

- **Node:** Es el servidor que se utilizó tanto para el back de la aplicación de escritorio como el servidor en el que corre la página web dado que está hecha en vue y vue utiliza como servidor node. Se eligió hacer la API de la aplicación principal en node dado que la base de datos estaba hecho en una no relacional y a lo mejor cambiaba su estructura, como node permite parsear JSONS y utilizarlos como objetos sin necesidad de la creación de un modelo nos facilitaba mucho el trabajo.
- **Express:** Se eligió Express por encima de otras soluciones de desarrollo de APIS en node porque entre todas las tecnologías que hay no existe demasiada diferencia y express era una que ya había utilizado asique se me facilitaría la vida.
- **Unity:** Se eligió Unity por encima de Unreal Engine como motor de videojuegos porque Unity es una herramienta que domino y Unreal Engine no. Además, en Unreal Engine se puede hacer una aplicación entera sin la necesidad de escribir una sola línea de código (a través de los blueprints). Cosa que no atraía demasiado la atención a los profesores.
- **C#:** Se eligió C# por encima de unityScripting para programar en Unity dado que es el lenguaje más extendido para programar en Unity y en el que además de encuentra la mayoría de documentación programada.
- **Agora:** La solución de Agora se utilizó para la realización de llamadas desde dentro del juego, lo que permite a los usuarios una más fácil comunicación. Se eligió ahora sobre otras soluciones de llamadas de voz debido a los 50000 minutos gratis mensuales que nos dan.
- **Photon pun 2:** Esta solución para la realización del multijugador ante otras soluciones como podrían ser Mirror o Unity Netcode dado que esta solución nos permite utilizar un servidor externo y no tener que programar las lógicas de servidor de la aplicación, es decir en las otras soluciones populares se utiliza una comunicación P2P (peer to peer) sin embargo en esta solución se utiliza una comunicación de cliente servidor. Además, se eligió la versión Pun 2 frente a otras versiones como podrían ser la fusión debido a qué era la más apropiada para el tipo de aplicación que se quería realizar y que ya estaba familiarizado con esa tecnología.

- **Photon voice:** Esta cuya tecnología se utiliza para el chat de proximidad que a diferencia de Agora que se utiliza para las llamadas a través de la aplicación este va modulando el volumen de la voz dependiendo de a qué distancia estás del jugador al que estás hablándole. Lo que permite una comunicación más realista.
- **Mongo:** Se esta tecnología para la base de datos. Es una base de datos no relacional lo que nos permite modificar registros de la base de datos sin afectar a la estructura del resto de registros de el mismo documento. Mongo por encima de otras bases de datos no relacionales como por ejemplo firebase o sobre otras bases de datos relacionales como puede ser MySQL debido a que la base de datos había que tenerla funcionando desde el principio para probar sus funcionalidades sin embargo podría llegar a cambiar la estructura de los registros a medida que la aplicación se hacía más grande y al usar Mongo nos permitía estos cambios sin la necesidad de desplegar de nuevo la base de datos ni tener que lanzar sentencias personalizadas para la edición de los registros.
- **Atlas:** Se usó Atlas para el hosting de la base de datos de mongo en vez de tenerlo en local dado que Atlas está en la nube y nos permitiría poder acceder a la base de datos desde cualquier API que programásemos sin la necesidad de montarnos un servidor en la nube configurar la base de datos y permitir accesos por solo esas Apis. Esto encarecería nuestros costos sin embargo gracias a Atlas podemos tenerla en la nube y de una manera muy sencilla configurarla para solo permitir acceso a la máquina EC2 de AWS.
- **UniRx:** Es una librería de programación reactiva en Unity. Se uso para los menús del juego. Se eligió esta tecnología porque era la más popular y porque 1 de nuestros assets la utilizaba.
- **Remote Config:** Remote config es una tecnología de unity que nos permite la edición de variables del juegos sin requerir de una actualización de la aplicación esto nos permite por ejemplo si un enemigo o una característica de un mundo es demasiado compleja poder editar las variables de ese enemigo o esa característica sin necesidad de lanzar una nueva actualización para la aplicación. Lo que nos daría un mayor control y tiempo de respuesta hacia los problemas que puedan tener los usuarios de nuestra aplicación.

- **Servidor RMTP:** El servidor RMTP es utilizado para que los usuarios puedan hacer retransmisiones en directo hacia nuestro servidor y nosotros poder mostrarlos en las aplicaciones que lo requieran.
- **HLS:** La tecnología HLS se configuró en el servidor de streaming debido a que los dispositivos de Apple no pueden obtener información a través de protocolos RMTP y hemos tenido que cambiarlo a HLS.
- **NGINX:** Se uso NGINX cómo servidor de streaming debido a la facilidad a la hora de configurar el RMTP y el HLS. Cuenta con dos endpoints el primero de ellos se utiliza para poder hacer las retransmisiones en directo a través de un emisor RMTP (por ejemplo, obs) y el otro endpoint se utiliza para poder obtener las estadísticas de todos los directos actualmente retransmitidos en el servidor lo que nos permite obtener un listado de dichos directos.
- **NEXPLAYER:** Debido A que ahora nuestro servidor emitía a través del protocolo HLS necesitábamos un reproductor HLS para nuestra aplicación de escritorio dado que el reproductor de vídeos de unity no permite la comunicación por HLS.
- **Docker:** Se docker para el lanzamiento de El servidor de streaming y el servidor de autorización debido a la facilidad para configurar estos lanzamientos.
- **Docker-Compose:** Dado qué teníamos que lanzar dos contenedores de docker se utilizó docker compose para la correcta configuración de más de un contenedor.
- **Websockets:** La de websockets se utilizó para la comunicación a tiempo real y bidireccional de nuestro servidor de websockets junto con todas las aplicaciones que requiriesen de este servicio. En nuestro caso se utilizó la comunicación a través del protocolo de websockets únicamente para la mensajería.
- **Socket.io:** Se eligió esta librería por encima de otras librerías para la creación de servidores de web sockets dado a su facilidad a la hora de poder crear rooms o salas y a su extensa y muy bien redactada documentación.

- **Vue 3:** Para se utilizó un framework de última tecnología cómo Vue 3. Se eligió Vue 3 por encima de otros frameworks de última tecnología como React o Angular debido a que se podía utilizar un framework de estilos como Quasar con dicha tecnología. Además, se eligió la versión 3 sobre la dos debido a sus notables diferencias y sus mejoras tanto de tipado como de rendimiento
- **Quasar framework:** Quasar es un framework de estilos que nos permite utilizar además de sus clases personalizadas para aplicarle estilos a nuestra aplicación sus componentes para darle un estilo más profesional a nuestra web.
- **Vime:** Vime es un empaquetador de paquetes para el despliegue de las aplicaciones hechas en frameworks de web y se eligió por encima de su principal competidor qué es web pack debido a que Vime está financiado y mantenido por Vue.
- **Google recaptcha v3:** Para la detección de bots en nuestra web se utilizó Google recaptcha versión 3 la cual nos permite la detección de dichos bots sin la necesidad de mostrar un cuadro de diálogo sino con llamadas HTTP al propio back de Google.
- **Google recaptcha v2:** Además se optó también por usar la versión dos debido a que como pone en la documentación de Google, utilizar ambas versiones da la mayor cobertura a posibles ataques de bots. La versión dos a diferencia de la 3 sí que muestra un cuadro de diálogo de “eres un robot”.
- **Pinia:** Pinia se utilizó para la gestión de variables globales a través de las stores dentro de nuestra web.
- **Google Login:** Nuestra página web contiene un login de usuario normal pero además permite la opción de tanto registrarse como de iniciar sesión a través de Google lo que nos permite darle una experiencia al usuario mucho más rápida y sencilla.
- **SocketIOUnity:** Dado que necesitábamos de una tecnología para la comunicación por websockets con nuestro servidor. Se eligió esta librería debido a que era la única gratuita en unity que nos permitía comunicarnos con otro servidor.

- **Ultimate Navigation AI System:** Se opto por usar la última versión del sistema de navegación de IA en unity debido a su tremenda optimización a su predecesora. Esta tecnología se utilizó para la navegación autónoma de 1 de los personajes de la aplicación.
- **Addresables:** Paquete the unity para poder comunicarnos con su Cloud Conten Delivery y poder así mostrar en los carteles publicitarios, la publicidad que las empresas nos han proporcionado a través de nuestra web.
- **Cloud Conten Delivery:** Servicio Kitty que nos permite el almacenaje y distribución a través de addresables de los contenidos proporcionados por las empresas a través de nuestra web para publicitarse en nuestra aplicación.
- **HTML, Less, JS:** Estas tecnologías se utilizaron para la maquetación desarrollo y funcionalidad de la web dentro del framework de Vue3.
- **Python:** Esta tecnología se utilizó en el back de nuestra página web para desarrollar una APIREST que permitía comunicar nuestra web con nuestra base de datos y guardar los datos necesarios para la publicidad y las empresas. Además, se eligió utilizar Python a diferencia de otras tecnologías de back debido a que contenía diversas librerías las cuales no estaban en otras tecnologías.
- **Flask:** Se cómo proveedor del framework de Python para desarrollar una API porque a diferencia de sus competidores como pueden ser FastApi, DjangoRest, tornado etc hablo con él lo que me he facilitaba el desarrollo de la aplicación además entre las diferentes posibilidades que había, las funcionalidades no cambiaban mucho de un framework a otro.
- **Trello:** Trello es una aplicación para el desarrollo de metodologías agile la cual utilicé para la organización del proyecto, así como la organización de tiempos y entregas del mismo. Se eligió tráelo por encima de otras tecnologías como por ejemplo Jira debido a que en un proyecto de una sola persona y no requería de una aplicación tan grande como Jira.
- **GitHub:** Es un gestor de repositorios basado en git el cual utilice para el control de versiones de todos los servicios que había en mi aplicación el cual me permitía en caso de tener un problema poder volver al pasado además d en caso de pérdida del ordenador o borrado de los datos poder clonar el repositorio en otro ordenador y seguir trabajando. Se usó GitHub por encima de otras soluciones como por ejemplo GitLab porque GitHub es gratuito y ya tenía una cuenta creada.

- **SourceTree:** Como aplicación para controlar todo lo que ocurría en mi repositorio utilice SourceTree que es una aplicación muy profesional. Se eligió esta aplicación por encima de otras como por ejemplo GitHub desktop o GitKraken gracias a su interfaz intuitiva.

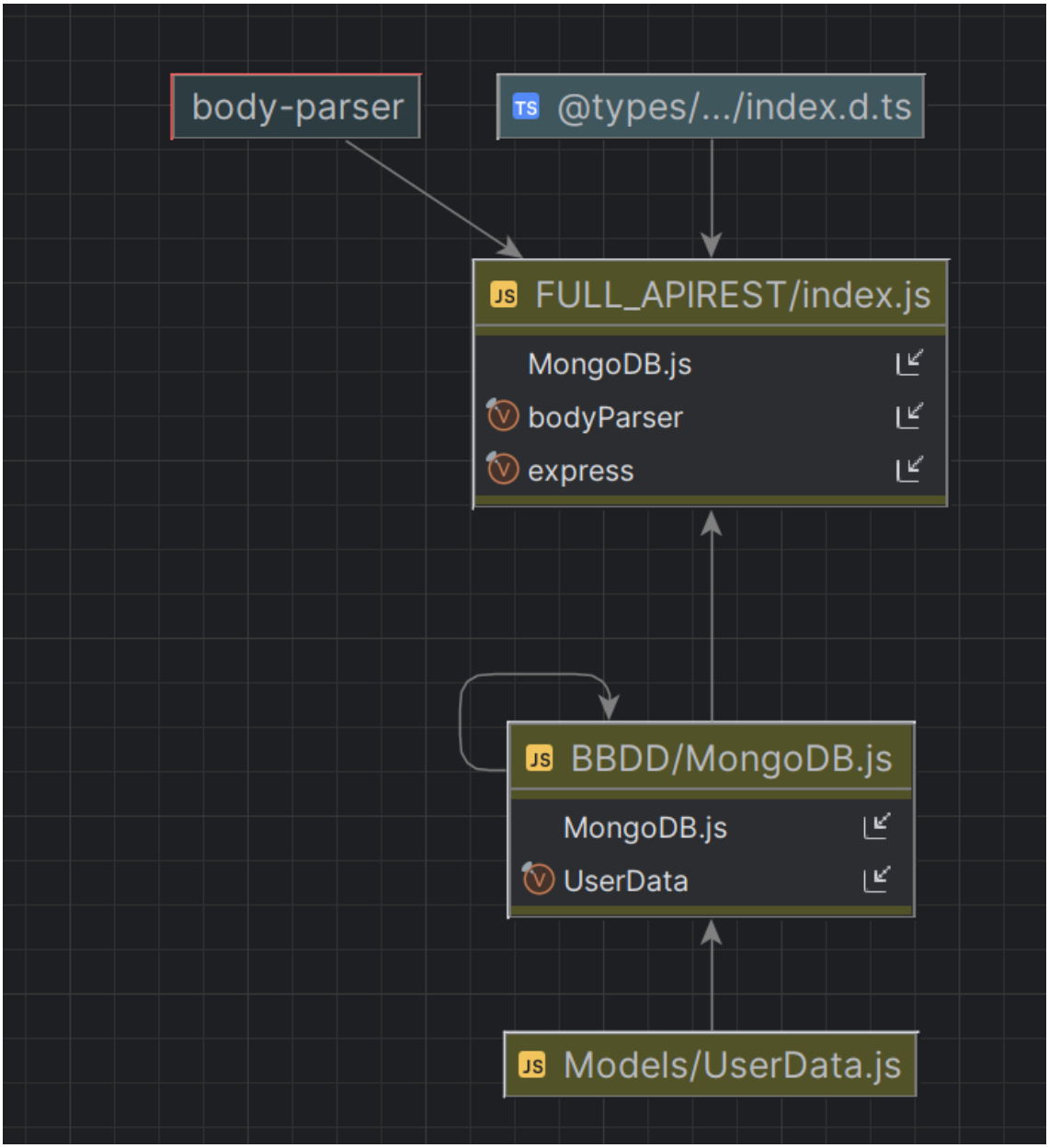
DISEÑO

1. Diagrama de clases de la aplicación de escritorio y VR

https://drive.google.com/file/d/11SeiiOPr-HAlB4Vl6VhyaNZlgzlKPz2V/view?usp=drive_link

Al ser realizada en unity se utilizó la arquitectura por componentes lo que facilita la comunicación entre ellos pero que sin embargo aumenta el número de scripts que se necesitan para el correcto funcionamiento de la aplicación. Esta aplicación es la encargada de contener a todos los jugadores del metaverso al igual que todos los escenarios. Además, también contiene los carteles publicitarios reservados por las empresas.

2. Diagrama de clases de la API de la aplicación de escritorio y VR

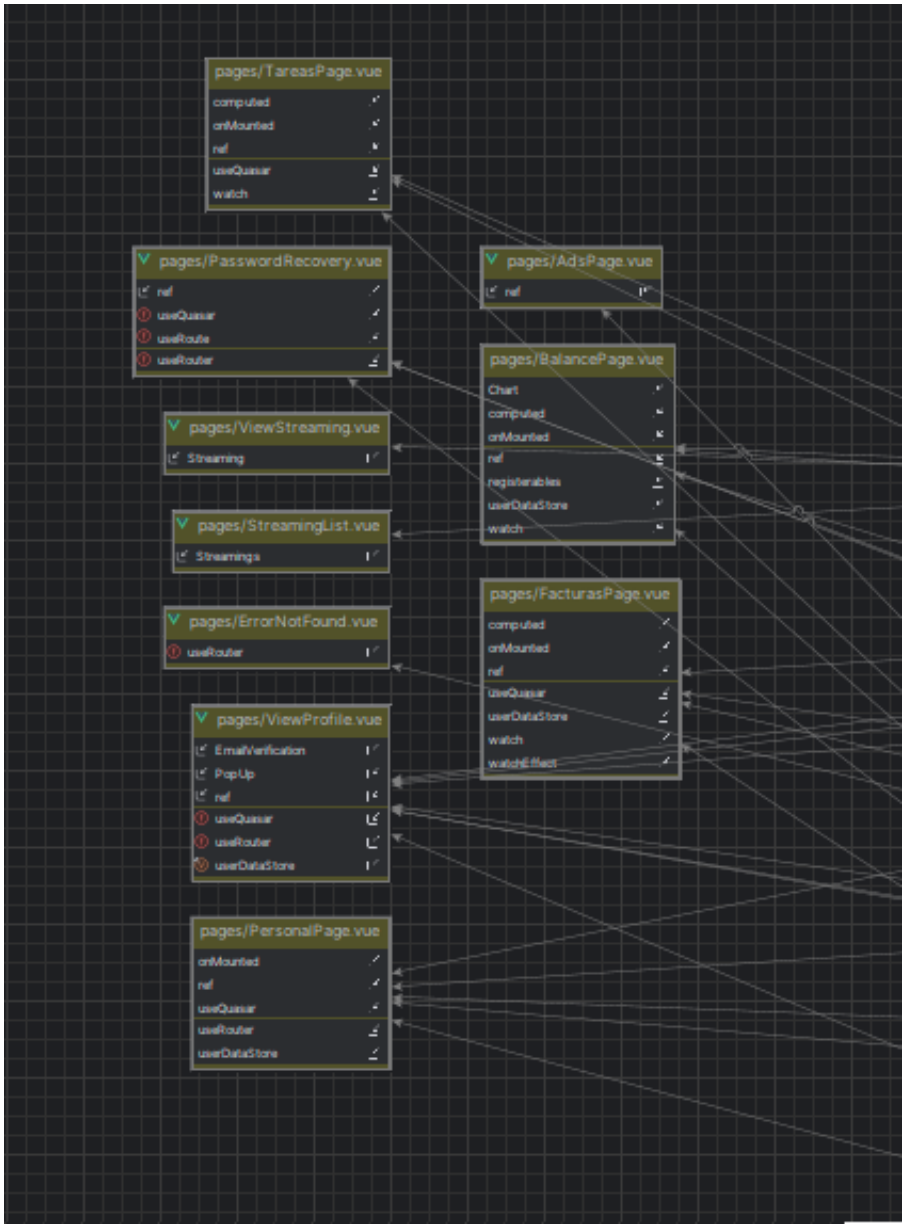


Es la encargada de comunicarse con la base de datos para guardar los datos del usuario en la aplicación, así como pueden ser sus contactos sus objetos o sus monedas. También es la encargada de que en caso de querer retransmitir un directo de la aplicación se genere una clave para la autentificación en el servidor de streaming.

3. Diagrama de clases de la web

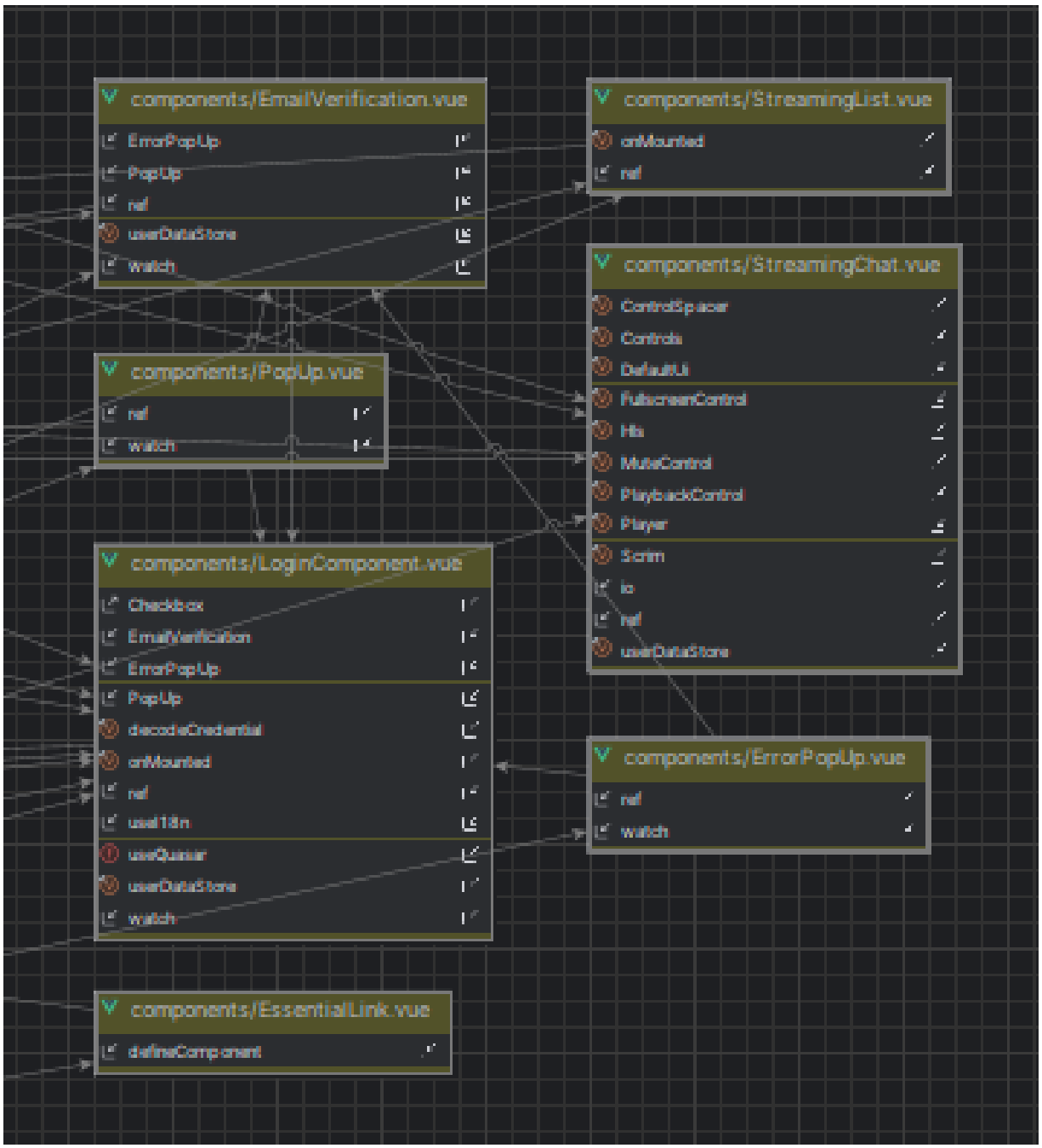
Dado que se programa por componentes no hay clase como tal por lo que se realizó el diseño sobre los componentes.

1. Pages



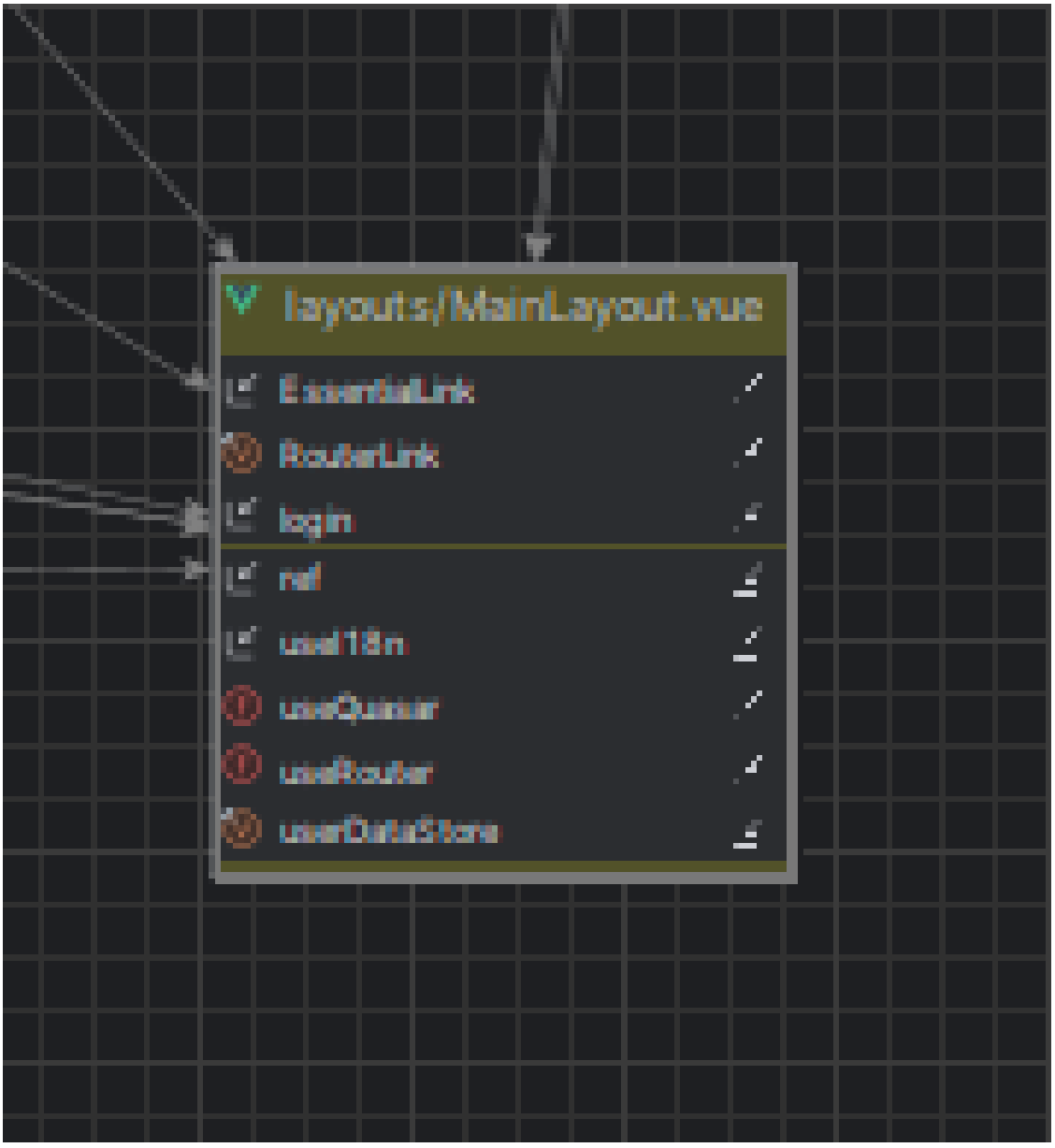
Las pages son las encargadas de contener varios componentes y mostrarle al usuario diferentes páginas dependiendo de a dónde navegue.

2. Components



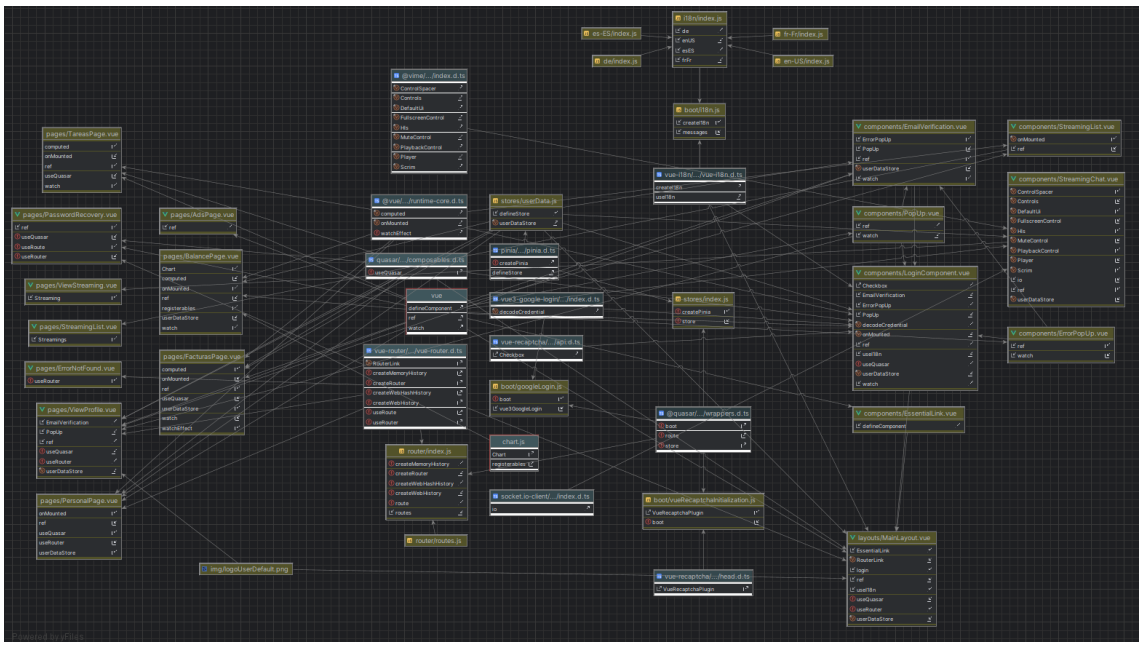
Los Componentes son los que constituyen las pages cada 1 de ellos puede estar contenido en más de una page así como su comportamiento aunque similar puede variar levemente dependiendo de los parámetros que se le pase.

3. Layouts



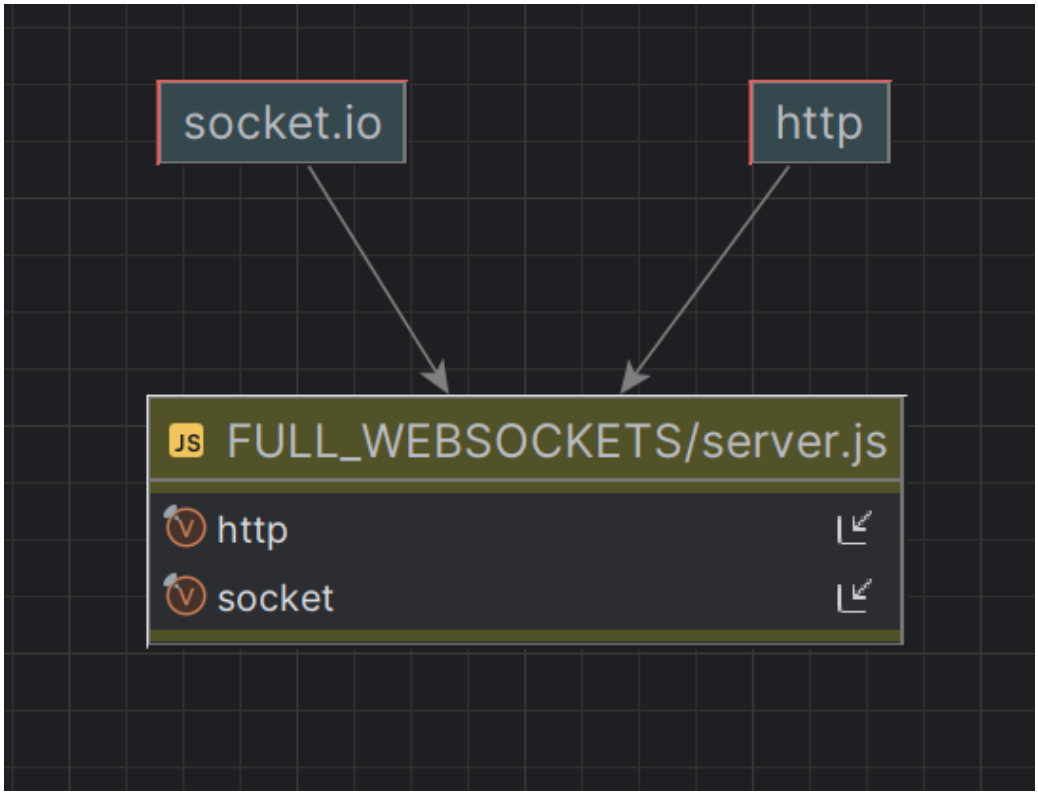
Los layouts son las plantillas qué se usan en la web para maquetarla en este caso solo se creó una plantilla.

4. Diagrama completo



A completo de la web en la que se puede apreciar tanto las páginas como los componentes como los layout estar conectados entre sí al igual que estar conectados con las librerías utilizadas en la página web.

4. Diagrama de clases del servidor de websockets



Este servidor es el encargado de retransmitir comunicación a tiempo real y bidireccional entre todos los servicios que lo usen en nuestro caso es utilizado tanto por la web como la aplicación de escritorio y VR.

5. Organización de los contenedores de Docker a través de Docker-Compose

```yaml
version: "3.9"
services:
  rtmp:
    build: ./rtmp
    ports:
      - "1935:1935"
      - "8080:8080"
    container_name: rtmp_server
    volumes:
      - ./data:/tmp/hls

  auth:
    build: ./auth
    container_name: auth_server
```

Se trata de 2 contenedores, el primero de ellos es el encargado de realizar la conexión rmtp y convertirlo a hls y utiliza tanto el puerto 1935 para esa conexión como el 8080 para la obtención del listado de estadísticas del servidor. El segundo contenedor es el servidor de autentificación y usa el puerto 8000.

BBDD

Para la base de datos se decidió utilizar MongoDB en el entorno de Atlas ya que es gratuito y funciona en la nube lo que nos facilitaba sutilización en las diversas aplicaciones. Al ser una base de datos no relacional no tiene una estructura por tablas cómo podrían tener unas bases de datos relacionales, sino que tiene una estructura por colecciones y documentos.

Estructura de colleciones:

Estructura de documentos dentro de Enterprises:

```
_id: ObjectId('66276e7dc9be3ad3de51b087')
username : "7FULL"
password : "baabbb93c698f23888c3396faba9f0bf1329af0447b2246a2afb1a1368792369"
email : "phgfull@gmail.com"
phone : "000000000"
status : 0
emailVerified : false
description : ""
token : "NlSej06d"
profile : "static/users/7FULL.png"
```

Estructura de documentos dentro de User:

```
_id: ObjectId('652fc3dae5e4962b0185b3a4')
user : "d99a176ad01de8f0971432cca7f5a10aadad6914"
▸ contacts : Array (4)
coins : 51100
▸ items : Array (3)
streamKey : "4bd5dx0qrp3d9hoio4hh2l"
```

IGNORAR LO DE ABAJO ES LA PLANTILLA

DATOS CENTRO DE FORMACIÓN

ALUMNO/A	
APELLIDOS: HERMIDA GÓMEZ	NOMBRE: PABLO

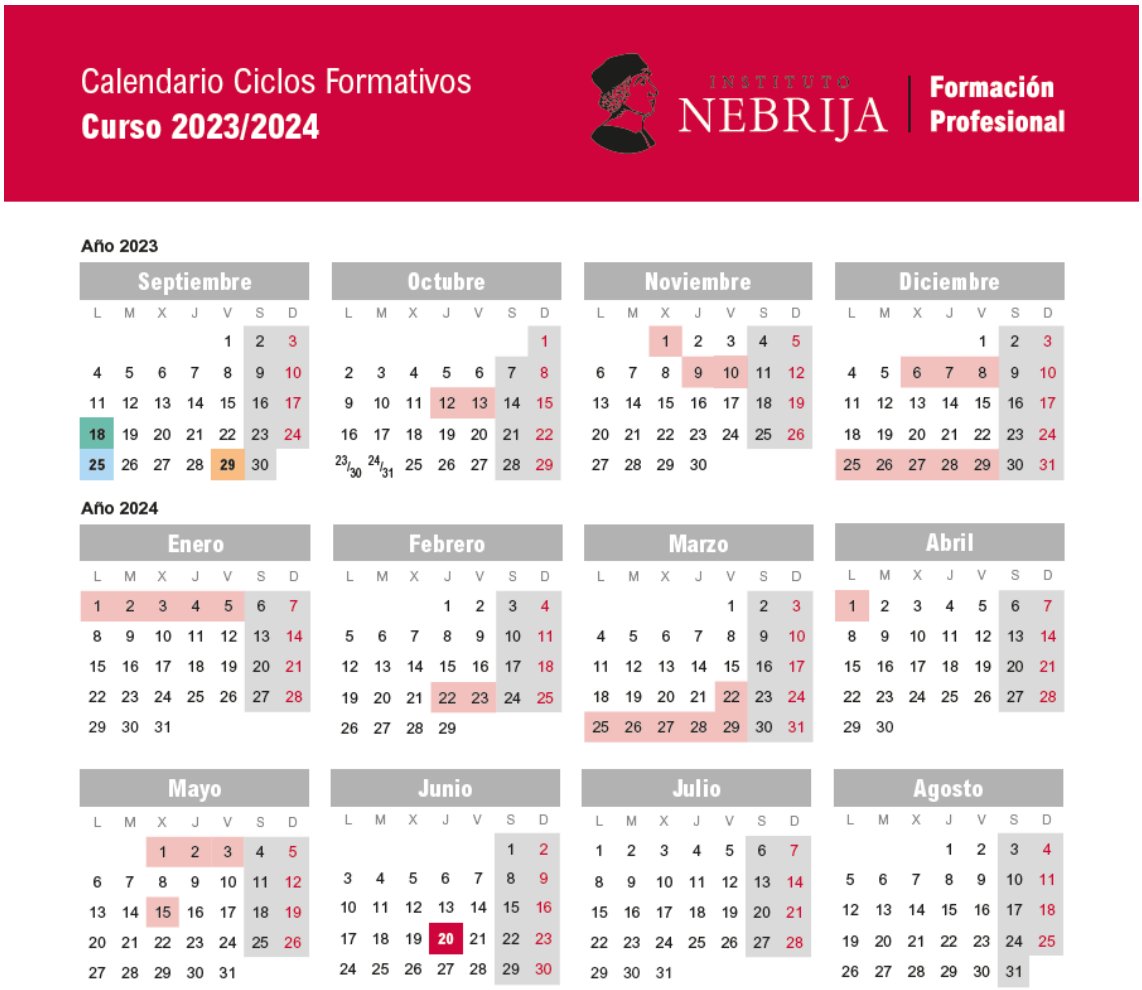
FECHA NACIMIENTO: 12/05/2004	DNI: 49155687H	TELÉFONO: 678041577
EM@IL: phermidag@alumnos.nebrija.es		
POBLACIÓN: Fuenlabrada	PROVINCIA: Madrid	CODIGO POSTAL:

ENSEÑANZA
FAMILIA PROFESIONAL: INFORMÁTICA Y COMUNICACIONES
CICLO FORMATIVO: DESARROLLO DE APLICACIONES MULTIPLATAFORMA
GRUPO: 2º

DATOS CONTACTO CENTRO DE FORMACIÓN		
DENOMINACIÓN: INSTITUTO NEBRIJA FORMACIÓN PROFESIONAL		
DIRECCIÓN: C/JUAN MARÍA LÓPEZ 60-62		
POBLACIÓN: MADRID	PROVINCIA: MADRID	CODIGO POSTAL: 28015
TELÉFONO: 91452110	CORREO ELECTRÓNICO: secretariafp@nebrija.es	
JEFE/A DE ESTUDIOS: CRISTINA RODRÍGUEZ	CORREO ELECTRÓNICO: crodriguezba@nebrija.es	
DPTO. PRÁCTICAS: ALBA FERNÁNDEZ	CORREO ELECTRÓNICO: afernandezri@nebrija.es	
CICLO FORMATIVO: DESARROLLO DE APLICACIONES MULTIPLATAFORMA		
COORDINADOR DE ÁREA: ANTONIO OTERO VEIGA	CORREO ELECTRÓNICO: aotero@nebrija.es	
TUTOR/A DOCENTE: ANTONIO OTERO VEIGA	CORREO ELECTRÓNICO: aotero@nebrija.es	
TUTOR/A DOCENTE:	CORREO ELECTRÓNICO:	
TUTOR/A DOCENTE:	CORREO ELECTRÓNICO:	
TUTOR/A DOCENTE:	CORREO ELECTRÓNICO:	
TUTOR/A DOCENTE:	CORREO ELECTRÓNICO:	

Durante el periodo de prácticas Formación en centros de trabajo, el alumnado se rige por el calendario escolar aprobado para el año académico en curso.
Os facilitamos un recorte del mismo, donde quedan recogidos los festivos y días no lectivos, publicados por Conserjería de Educación de la Comunidad de Madrid.

Cualquier duda o consulta, trasladársela al Tutor del centro de formación, para que pueda gestionarla directamente.



⇨ ANEXO 7. Ficha semanal del alumno.

- El alumno/a, deberá reflejar diariamente las tareas que realice, el puesto formativo que ocupe y las horas de la jornada.
- No debe desprenderse ninguna hoja del mismo.
- Debe firmarse y sellarse semanalmente, por el Tutor/a del centro de trabajo. (Preferiblemente con Certificado digital)
- Será revisado por el Tutor/a del centro de Formación en las tutorías obligatorias de seguimiento.

*****Una vez finalizado el periodo de prácticas el alumn@ a la mayor brevedad debe entregar al Tutor del centro de formación, el Anexo 7 impreso y debidamente firmado por el centro de trabajo.