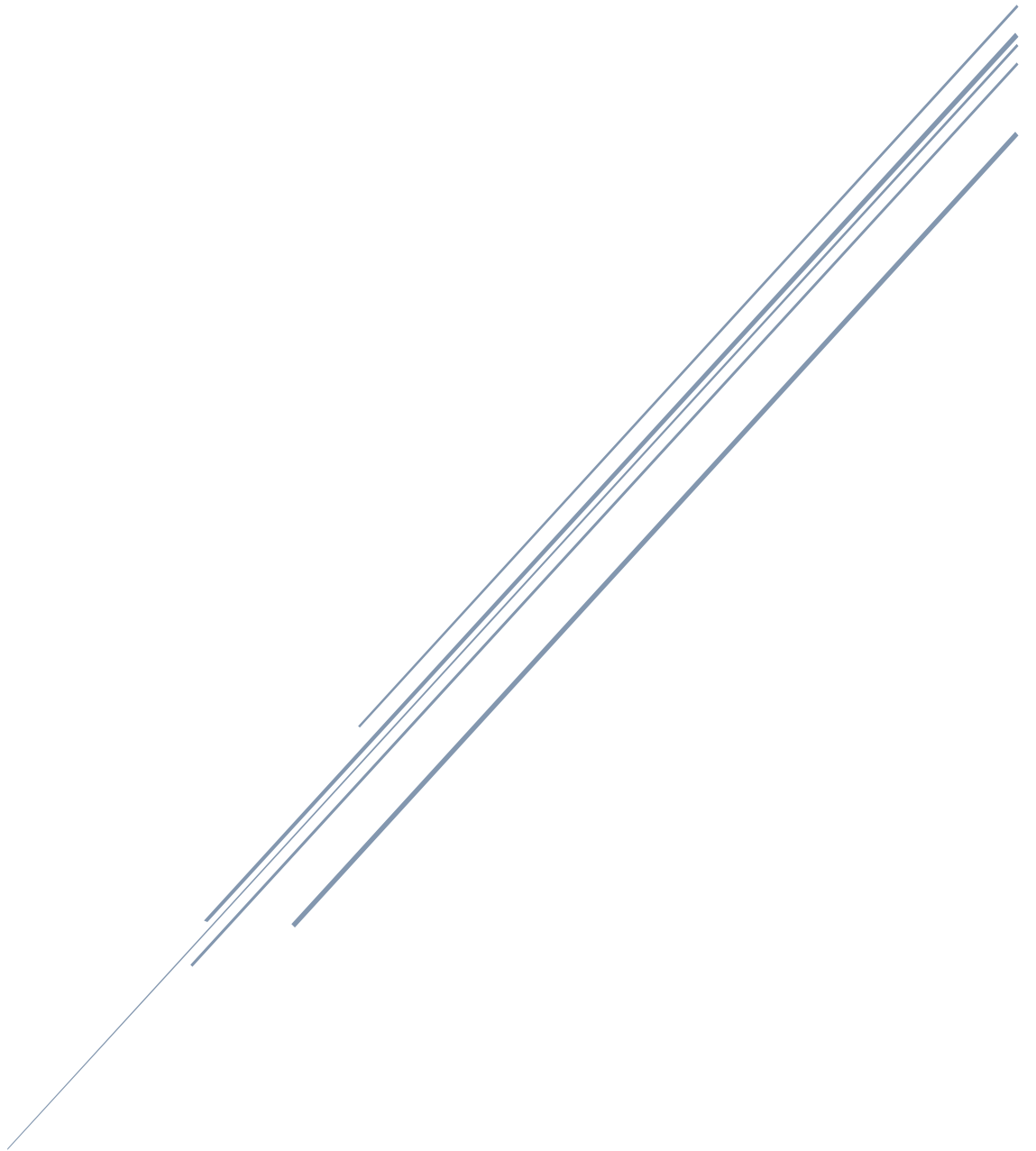


METAL PUNCH

Planificación y análisis del desarrollo de un videojuego
VR para Android controlado externamente por guantes
diseñados con Arduino



Índice de contenido

FASE I: Identificación del proyecto.....	3
FASE II: Diseño del Proyecto	4
1. Objetivo y viabilidad técnica y legal del proyecto	4
2. Metodología	4
2.1 Organización de la carga de trabajo	4
3. Planificación de los recursos	5
3.1 Software	5
3. 2 Hardware.....	6
4. Actividades	7
4.1 Obtención de los recursos.....	7
5. Secuenciación	8
5.1 Pasos a seguir	8
6. Evaluación	9
6.1 Herramientas y métodos empleados para evaluar el funcionamiento del proyecto	9
FASE III: Desarrollo del proyecto.....	10
1. Desarrollo del servidor web	10
1.1 Creación de la máquina virtual en Azure	10
1.2 Conexión a la máquina virtual de Azure	11
1.3 Creación de servidor y base de datos.....	12
1.4 Creación contenedor Docker	17
2. Creación de la página web.....	21
2.1 Configuración e inicio del proyecto	22
2.2 Desarrollo de los componentes de la aplicación	22
2.3 Creación de las vistas de la aplicación	23
2.4 Funciones relacionadas con la API	25
2.5 Creación de contenedor Docker para almacenar la aplicación.....	31
2.6 Resultado final	33
3. Desarrollo del videojuego.....	33
4. Diseño de los guantes	36

4.1 Componentes físicos	37
4.2 Software	40
FASE IV: Evaluación y conclusiones finales	46
1. Complicaciones encontradas durante el desarrollo del proyecto	46
2. Evaluación y conclusiones	48
Referencias	50

FASE I: Identificación del proyecto

La realidad virtual (RV) permite la inmersión del usuario en entornos simulados que pueden utilizarse para la adquisición o mejora de nuevas competencias, puesto que permite representar escenarios que en la vida real no podrían ser usados para practicar las habilidades del sujeto, ejemplos de ello sería el uso de entornos de realidad virtual para representar un incendio o una operación compleja. Asimismo, también es un importante recurso de ocio para usuarios de todas las edades.

En este contexto, que muestra la importancia y necesidad de la existencia de programadores con conocimientos de desarrollo de realidad virtual, se plantea la creación de un videojuego de boxeo que emplee la RV, en la que los movimientos de los jugadores serán capturados por unos guantes de creación específica para el juego, que a su vez se comunicarán con el videojuego para transmitir dicha información.

Como apoyo al videojuego, se creará un servidor web basado en REST para el almacenamiento de los datos de los jugadores, la autenticación de usuarios y la transmisión y recepción de información.

Asimismo, para facilitar el registro de los usuarios, así como para informar sobre el videojuego y sus objetivos, se creará una aplicación web. Esta además tendrá la capacidad de elaborar una tabla con el ranking de los usuarios basado en sus puntuaciones.

FASE II: Diseño del Proyecto

1. Objetivo y viabilidad técnica y legal del proyecto

El objetivo es ser capaz de desarrollar de forma simultánea tanto un videojuego de RV para android como unos guantes que sirvan como controlador externo.

Tras analizar las diferentes herramientas y soluciones a disposición de los desarrolladores, se ha llegado a la conclusión de que existen actualmente gran cantidad de métodos que permiten llevar a cabo las tareas planteadas para el proyecto sin que ello conlleve una dificultad técnica lo suficientemente excesiva como para que el mismo no sea viable. Esto es gracias a que actualmente existen muchos proyectos open-source que proporcionan gran cantidad de recursos tanto a programadores novatos como a expertos, lo cual posibilita un aprendizaje más rápido y sencillo.

El propósito de este proyecto es adquirir la capacidad de trabajar con algunas de las tecnologías con mayor demanda en el mercado, así como poner en práctica las competencias adquiridas a lo largo del grado superior y aumentar los conocimientos de las mismas.

Legalmente no existe ningún impedimento para el desarrollo del mismo, dado que todos los recursos de software empleados son open source y no están protegidos contra su uso ni modificación, a excepción de Unity, cuya licencia gratuita no puede emplearse si con el juego desarrollado se obtienen ingresos o financiaciones mayores a 120,000 USD

2. Metodología

2.1 Organización de la carga de trabajo

Para organizar las diferentes actividades a desarrollar para completar el proyecto, se ha optado por aplicar la metodología Waterfall o 'en cascada'.

La secuencia que sigue este método está compuesta de las siguientes fases:

- Captura y documentación de requisitos
- Diseño
- Desarrollo

- Test
- UATs
- Corrección de errores y ajustes finales
- Puesta en producción

El motivo de esta elección es que, al tratarse de un proyecto no colaborativo, y al no estar orientado a la comercialización del producto, se ha considerado preferible optar por una estructura lineal de las fases del desarrollo, lo que permite realizar el seguimiento del progreso del proyecto de forma más simple, dado que el alcance se conoce de antemano, asimismo, dado que se deben diseñar dos componentes claramente diferenciados (el videojuego y los guantes de control), un desarrollo en paralelo facilitará la detección temprana de errores y requisitos que no se hayan podido reconocer en las fases de diseño, lo cual proporcionará más espacio a realizar los posibles cambios necesarios.

3. Planificación de los recursos

3.1 Software

Se creará un servidor web basado en REST para almacenar los datos de los jugadores, permitir el registro y la autenticación de usuarios y manejar la información enviada por el juego para modificar las puntuaciones de los usuarios. Así mismo permitirá acceder a la información de los jugadores y al ranking de los mismos.

Este servidor web se creará haciendo uso de Python y de los framework Django y Django REST.

Para la creación del proyecto de Django, se recomienda crear un entorno virtual de desarrollo, para así aislar los ajustes y que, en caso de estar trabajando en varios proyectos al mismo tiempo, cada cual con sus diferentes configuraciones, no haya ningún conflicto.

Para esto se ha hecho uso de Anaconda. Anaconda Distribution incluye Conda, un sistema open-source de administración de entornos y paquetes, entre sus funcionalidades se encuentran la rápida instalación de paquetes y sus dependencias, así como la facilidad para la creación y el uso de nuevos entornos virtuales, herramienta que se empleará en este proyecto.

Para la creación de la aplicación web se hará uso de Javascript y React JS.

ReactJS está basado en un paradigma llamado programación orientada a componentes en el que cada componente es una pieza con la que el usuario puede interactuar. Estas piezas se crean usando una sintaxis llamada JSX permitiendo escribir HTML (y opcionalmente CSS) dentro de objetos JavaScript, lo cual facilita al desarrollador medio la creación de nuevos proyectos. Algunas de las principales ventajas de React JS es que es de código abierto, tiene la capacidad de acelerar el desarrollo de aplicaciones, emplea función de Recarga Activa (Hot Reloading), que permite al desarrollador implementar cambios en un código y ver en tiempo real cómo afectan a la aplicación.

El desarrollo de la aplicación web se realizará mediante Visual Studio Code.

Se creará un contenedor haciendo uso de Docker que permita correr el servidor en una máquina virtual en Azure que se creará para este proyecto. Esta hará uso de una imagen de Ubuntu Server y contará con una dirección IP reservada para acceder al servidor.

La conexión con la máquina virtual se hará mediante SSH y se empleará para estos efectos la herramienta MobaXterm.

Para el desarrollo del videojuego se utilizará Unity, dados los recursos gratuitos que ofrece (tutoriales interactivos, licencia gratuita), la posibilidad de desarrollar juegos para Android de realidad virtual sin necesidad de utilizar plataformas adicionales y la cantidad de opciones para el desarrollo, animación y modelado del que dispondremos.

Para programar las placas de Arduino que nos permitirán utilizar los guantes para manejar los movimientos de los jugadores, se empleará el lenguaje C++, y se utilizará Arduino IDE como entorno de desarrollo.

3. 2 Hardware

En cuanto a los recursos de Hardware que se emplearán, serán los siguientes:

-Gafas de realidad virtual para Android compatibles con GoogleCardboard.

Google Cardboard funciona colocando el teléfono a una distancia óptima de las lentes y, usando aplicaciones compatibles, las lentes crean un efecto 3D. Empleándolas puedes incluso mover la cabeza y las imágenes responderán al movimiento, dando la impresión al jugador de que está inmerso en el juego. El motivo de decidir usar Google Cardboard es su fácil configuración, su precio económico y su funcionalidad. Si bien Google las retiró del mercado, todavía existen tiendas que venden gafas VR compatibles.

-Placa Arduino Uno . Se ha escogido Arduino por ser un microcontrolador con mucho soporte por parte de su comunidad (existe gran cantidad de documentación y proyectos de ejemplo que ayudan al desarrollador a la hora de tomar decisiones en el diseño del proyecto), así como por tener un precio económico y por todas las posibilidades que ofrece.

En un primer momento, se propone emplear 1 placa Arduino Uno por cada guante, un módulo MPU-6050 (este módulo permite un alto rendimiento con un bajo consumo de energía y coste; combina un giroscopio de 3 ejes y un acelerómetro de 3 ejes y se empleará para la captura de movimiento) y el módulo Bluetooth HC-06 para realizar la conexión con Unity.

4. Actividades

4.1 Obtención de los recursos

Para los recursos relacionados con el software, al ser todos salvo Azure de uso gratuito (inclusive Unity, siempre que se emplee una licencia personal, MobaXterm, siempre que se use la Home Edition y Anaconda, siempre que se emplee la Distribution Edition), bastará con dirigirse a la página de descarga de los mismos.

UNITY: <https://unity.com/download>

ARDUINO IDE: <https://www.arduino.cc/en/software>

PYTHON: <https://www.python.org/getit/>

MOBAXTERM: <https://mobaxterm.mobatek.net/download.html>

ANACONDA: <https://www.anaconda.com/products/distribution>

VISUAL STUDIO CODE: <https://code.visualstudio.com/download>

Para la creación de la máquina virtual de Azure, se indicarán los pasos en el apartado [Creación de la máquina virtual de Azure](#) de esta memoria. Cabe destacar que, si bien no es gratuito, [Azure Students](#) ofrece 100\$ de crédito inicial, que son los que se han empleado para este proyecto.

Para la obtención de React, se indicarán los pasos a seguir en el apartado [Creación de la página web](#) .

Para la obtención de los recursos relacionados con el hardware, se ha optado por diversas tiendas, atendiendo a la calidad ofertada y a los precios de los productos:

GAFAS VR: <https://www.amazon.es/Gafas-realidad-virtual-Google-Cardboard>

PLACA ARDUINO UNO: <http://store.arduino.cc/products/arduino-uno-rev3>

MÓDULO MPU-6050: <https://es.aliexpress.com/item/1734534460>

MÓDULO HC-06: <https://www.amazon.es/Modulo Bluetooth>

PROTOBOARD: <https://www.amazon.es/Breadboard-Arduino>

CABLES DE CONEXIÓN: <https://www.amazon.es/Jumper-Wire-Cables>

5. Secuenciación

5.1 Pasos a seguir

Los pasos a seguir vienen determinados por la metodología escogida, sin embargo con algunas modificaciones, dado que el proyecto no va a ser puesto en producción, y tampoco se necesitarán los UAT's , dado que no existe un cliente final en este caso, y, dada la fecha límite del proyecto, se ha prescindido de los tests; así pues las últimas fases serían la recopilación de los resultados obtenidos y la preparación para la defensa del proyecto, de forma que el proyecto se desarrollaría en las siguientes fases:

- Captura y documentación de requisitos técnicos y materiales
- Diseño de la estructura del proyecto
- Desarrollo
- Desarrollo de la base de datos
- Desarrollo simultáneo de los diferentes componentes del proyecto
- Corrección de errores y ajustes finales
- Recopilación y evaluación de los resultados obtenidos
- Preparación de la defensa del proyecto

6. Evaluación

6.1 Herramientas y métodos empleados para evaluar el funcionamiento del proyecto

Para evaluar el resultado final del proyecto, se tendrá en cuenta si se han cumplido los objetivos marcados, siendo estos la creación del servidor, la creación de la página web, el desarrollo de videojuego y la creación de los guantes.

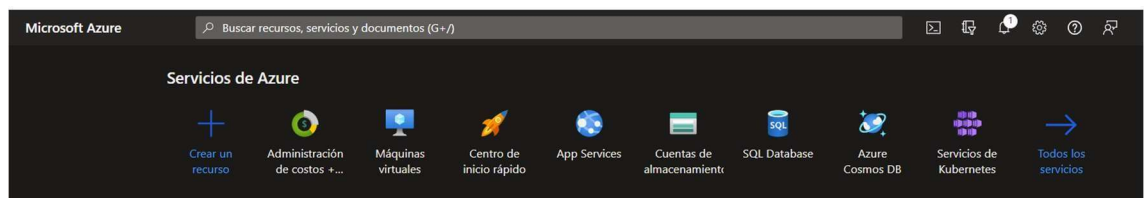
Se ha considerado importante pues, teniendo en cuenta los plazos previstos de entrega del proyecto, alcanzar los objetivos mencionados anteriormente, ampliar los conocimientos referentes a los programas, herramientas y lenguajes empleados y adquirir la capacidad de realizar la propuesta, planteación y el análisis de forma eficiente.

FASE III: Desarrollo del proyecto

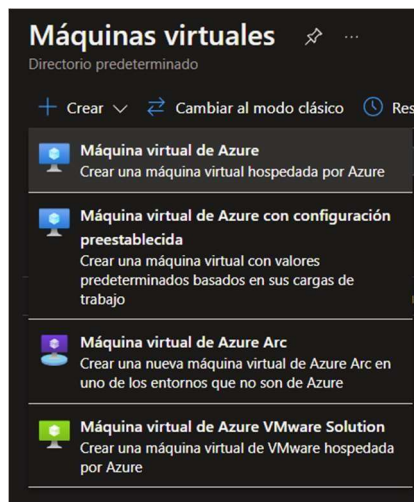
1. Desarrollo del servidor web

1.1 Creación de la máquina virtual en Azure

En primer lugar se debe acceder a la página principal del portal de Azure <https://portal.azure.com/#home>. Una vez allí, en el apartado de servicios de Azure, se selecciona la opción de 'Máquinas virtuales':



Una vez en el directorio de máquinas virtuales, se selecciona la opción crear y, en el desplegable que aparecerá, la opción 'Máquina virtual de Azure':



En el formulario de creación de la máquina virtual se deberán rellenar los datos básicos de la nueva máquina virtual. Se ha escogido en esta ocasión una imagen de Ubuntu Server 20.04 LTS, pero existen diversas opciones válidas que podrían seleccionarse.

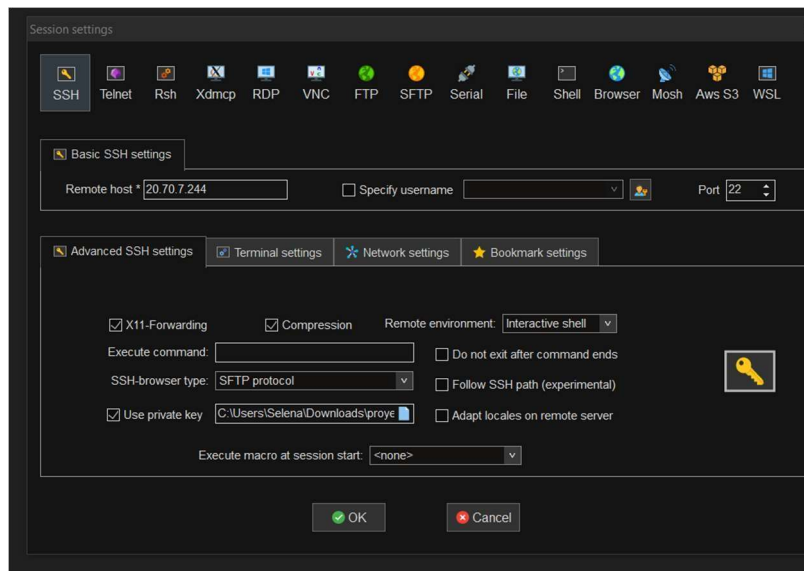
Es preciso, para la posterior conexión con la máquina, que en 'Tipo de autenticación' se escoja Clave Pública SSH.

Una vez finalizada la creación de la máquina, se debe descargar la clave privada SSH

1.2 Conexión a la máquina virtual de Azure

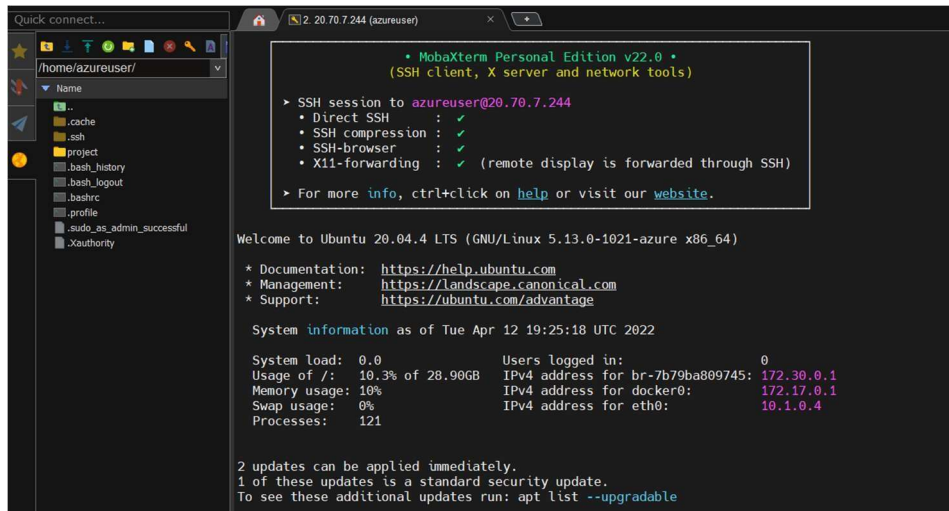
Para realizar la conexión con la máquina virtual previamente creada, se debe acceder a la aplicación MobaXterm.

En el menú de herramientas, se seleccionará la pestaña 'Sesión', aparecerá un modal en el que se deberá escoger el método de conexión (en este caso SSH). Tras seleccionarlo, se deberán habilitar las herramientas avanzadas, y se rellenará el formulario de la siguiente manera:



Siendo '*Remote host*' la dirección IP de la máquina de azure y '*Port*' el número de puerto por el que se accederá. Asimismo, se seleccionará la opción '*Use private key*' y se seleccionará el archivo descargado tras la creación de la máquina virtual, el cual contiene la clave privada de acceso.

Una vez hecho esto se creará una sesión que nos permitirá visualizar, crear, modificar y eliminar elementos dentro de la máquina virtual. Aparecerá una ventana como la siguiente:



En la que la pantalla se dividirá en dos, en la parte izquierda se encontrará la vista de árbol de los elementos de la máquina virtual y en la parte derecha la terminal.

1.3 Creación de servidor y base de datos

Este punto puede realizarse de dos formas diferentes. La primera opción sería directamente en la máquina virtual de Azure, y la segunda sería en el equipo local.

Dado que, en MobaXterm pueden arrastrarse archivos desde el equipo local y situarlos directamente en las carpetas de Azure, ambas opciones darán el mismo resultado.

En este caso se ha optado por la segunda opción. En primer lugar debe verificarse que python esté instalado. Para ello, se accede a la terminal y se ejecuta el siguiente comando:

```
python --version
```

En caso de que no esté instalado, se puede descargar desde la página oficial, cuyo enlace se encuentra en el apartado de [Obtención de recursos](#) de esta memoria.

Una vez instalado, se accede al Anaconda Prompt, la consola de comandos sobre la que se trabajará en este proyecto.

Los comandos para la creación y activación del entorno serán los siguientes:

--Crear entorno con nombre 'dev' (este puede sustituirse con el nombre que se prefiera):

```
conda create --name dev
```

--Activar entorno 'dev':

```
conda activate dev
```

Una vez se ejecute el segundo comando, se entrará en el entorno creado. En este punto, se procederá a la creación del proyecto de Django, para ello, en la consola, hay que situarse en la carpeta en la que se creará el proyecto y ejecutar los siguientes comandos:

--Instalar django:

```
pip install django
```

--Creación de la estructura del proyecto:

```
django-admin startproject backend
```

--Crear aplicación para los modelos, vistas y url de los jugadores (de nombre 'players'):

```
python manage.py startapp players
```

Los módulos necesarios para el desarrollo del proyecto vienen especificados en el archivo 'requirements.txt' que se adjunta con el código del proyecto.

Para la creación de la base de datos se destacan los siguientes aspectos:

- Se ha empleado el modelo User del sistema de autenticación por defecto de Django.
- Se ha usado sqlite para la creación de la base de datos (el sistema de bases de datos que ofrece Django por defecto).
- Asimismo, se ha creado un modelo PlayerInfo, que se relaciona con User en relación 1:1.
- El modelo PlayerInfo contiene el id de usuario, que emplea para la relación, el id de jugador, la puntuación y tres métodos, uno para añadir puntuación, otro para restar puntuación, y otro para convertirlo a String.

Se ha empleado también el framework Django REST que permite una sencilla y rápida creación de API REST. Esto es lo que nos permitirá obtener la información de los jugadores.

En el proyecto se han especificado las siguientes URL's:

--Crear usuario:

```
players/create/<str:name>/<str:email>/<str:password>
```

```
# ejemplo: players/create/selena/email@email.es/admin
```

Se le deben pasar, en primer lugar, el string que corresponderá al nombre de usuario, en segundo lugar el correo electrónico y por último la contraseña.

--Iniciar sesión:

```
players/login/<str:username>/<str:password>
```

```
#ejemplo: players/login/myname/mypassword
```

En este caso se le pasan el nombre del usuario y la contraseña del mismo.

--Desconectar sesión:

```
#ejemplo: players/logout
```

```
players/logout/
```

En este caso no se le debe pasar ningún parámetro adicional, ya que lo toma de la sesión del usuario.

--Añadir/Restar puntuación:

```
players/score/<str:is_win>/<int:difficult_level>/
```

```
# ejemplo 1: /players/score/win/1
```

```
# ejemplo 2: /players/score/lost/2
```

Para determinar si se debe añadir o restar puntuación, y cuántos puntos se le deben modificar, primeramente se identifica el valor del primer parámetro (que puede ser win o lost), si el valor es 'win', es que hay que añadir puntos, y si el valor es 'lost', es que hay que restarlos.

Como segundo parámetro se le pasa el nivel de dificultad, el cual va del 1 al 3. Siendo 1 el nivel fácil, 2 el nivel normal y 3 el nivel difícil. El usuario lo determina la sesión.

La distribución de puntos es la siguiente:

Si gana :

- Nivel fácil: Se le añaden 10 puntos.
- Nivel normal: Se le añaden 20 puntos.
- Nivel difícil: Se le añaden 30 puntos.

Si pierde :

- Nivel fácil: Se le restan 5 puntos.
- Nivel normal: Se le restan 8 puntos.

- Nivel difícil: Se le restan 10 puntos.

URL's para acceder a la API:

--Visualizar todos los jugadores:

```
players/api/players
```

--Visualizar todos los usuarios:

```
players/api/users
```

--Visualizar un jugador determinado:

```
players/api/players?user_id=1
```

En este caso se le pasa como parámetro el id de usuario.

--Visualizar un usuario determinado:

```
players/api/user?username=selena
```

En este caso se le pasa como parámetro el nombre de usuario.

--Visualizar ranking de jugadores:

```
players/api/ranking
```

1.4 Creación contenedor Docker

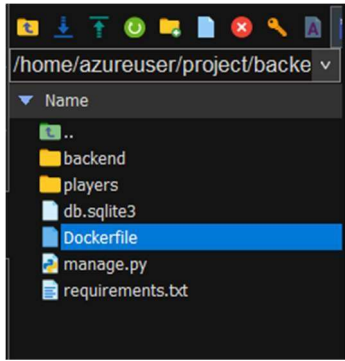
Para esta fase del proyecto se debe estar situado en la terminal de la máquina virtual creada en Azure, tal y como se indica en el apartado [Conexión a la máquina virtual de Azure](#).

Lo primero, es necesario situarse en la carpeta que almacenará el proyecto. En este caso la estructura es la siguiente:

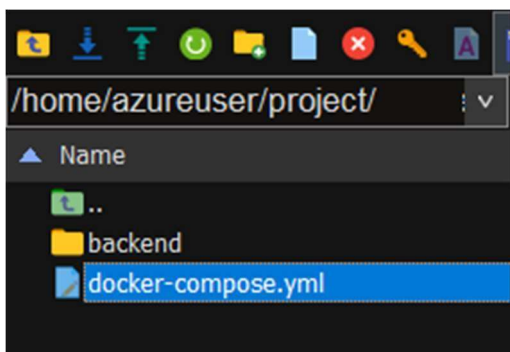
-Project

-backend

Se creará un archivo, llamado dockerfile, en la carpeta backend. De momento estará en blanco.



En la carpeta Project se creará un archivo llamado docker-compose.yml, también en blanco.



Tras ello, se debe instalar docker, para ello se hará uso de los siguientes comandos:

Primero, actualizar los paquetes existentes.

```
sudo apt update
```

Después, instalar las dependencias

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

En el siguiente comando se agregará la clave GPG

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Agregar el repositorio necesario:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

Instalar docker:

```
sudo apt install docker-ce
```

Para este proyecto, se empleará también Docker Compose. Esta es una herramienta que le permite ejecutar entornos de aplicaciones multi-contenedor basados en definiciones establecidas en un archivo YAML. Utiliza definiciones de servicio para crear entornos totalmente personalizables con múltiples contenedores que pueden compartir redes y volúmenes de datos.

En primer lugar, realizaremos la instalación de Docker Compose:

Comando para descargar la versión estable actual:

```
curl -SL https://github.com/docker/compose/releases/download/v2.2.3/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose
```

Aplicar los permisos necesarios para que el comando de docker compose sea ejecutable

```
chmod +x ~/.docker/cli-plugins/docker-compose
```

A continuación se editará el archivo docker-compose.yml tal y como se indica en la siguiente imagen.

Lo primero especificar la versión de Docker Compose que va a emplearse.

Tras ello, se creará un servicio al que se le llamará 'web', que empleará la imagen que se creará posteriormente en el archivo Dockerfile anteriormente mencionado..

Se le asignará un nombre de contenedor, que puede ser cualquiera, en este caso backend_app. Tras ello se indicará que se usará el puerto 8000.

```
* docker-compose.yml x
1 # Specifies which syntax version of Docker compose
2 version: '3'
3
4 # Build a multiservice architecture.
5 services:
6   # Create a service called web
7   web:
8     # Build an image from the files in the project root directory (Dockerfile)
9     build: ./backend
10    # Assigns a name for the container. If no name is specified,
11    # Docker will assign the container a random name
12    container_name: backend_app
13
14    # Map port 8000
15    ports:
16      - "8000:8000"
```

Ahora, en el archivo Dockerfile, se escribirá lo siguiente:

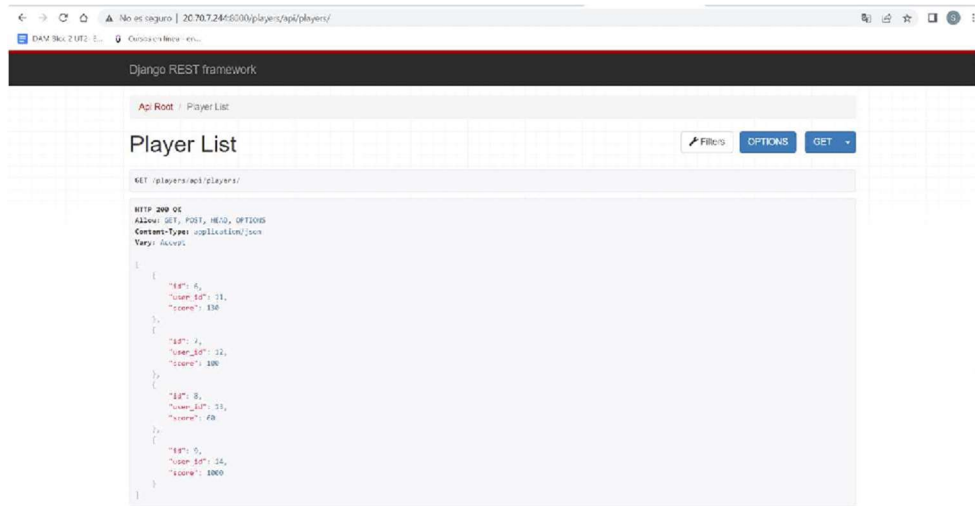
```
Dockerfile x
1 FROM python:3.9.12
2 ENV PYTHONUNBUFFERED=1
3 WORKDIR ./backend
4 COPY requirements.txt /backend/
5 RUN pip install -r requirements.txt
6 COPY . /backend/
7 CMD python manage.py makemigrations && python manage.py migrate && python manage.py runserver 0.0.0.0:8000
```

De esta forma se indica la versión de Python que va a emplearse y el directorio en el que se va a trabajar. Se indicará que se copie e instale el contenido del archivo requirements.txt, se copiará al proyecto en el que está trabajando y se lanzarán los comandos para realizar las migraciones de la base de datos y correr el servidor en el puerto 8000.

Por último, hay que situarse en la carpeta Project y ejecutar el siguiente comando:

```
sudo docker-compose up -d
```

A partir de este momento el contenedor estará creado y activo. Desde el ordenador local, puede comprobarse accediendo a la IP de la máquina de Azure, junto con una de las URL especificadas en el servidor.



2. Creación de la página web

Se ha optado por crear una página web básica que permita a los usuarios visualizar la información sobre el videojuego, el ranking de jugadores del mismo, con sus respectivas puntuaciones, y el registro de nuevos usuarios.

Para ello, se ha utilizado Javascript, junto con su framework React, dada su simpleza y facilidad de uso para nuevos usuarios.

React JS facilita también la obtención de información en tiempo real, además, desde la versión 16.8, se introdujeron los Hooks.

Un Hook es una función especial que permite “conectarse” a características de React.

Esto nos permite facilitar la obtención de datos a través de la API, determinar si un usuario debe visualizar una pantalla u otra, etc.

Asimismo se ha empleado Material UI, una librería de componentes para React, que ayuda a construir las páginas web de forma más intuitiva y rápida. Los componentes creados con Material UI ofrecen al desarrollador la posibilidad de crear diseños más atractivos empleando menos código y tiempo.

Los pasos a seguir son los siguientes:

2.1 Configuración e inicio del proyecto

Para crear la web, utilizaremos Create React App, este configura tu entorno de desarrollo para poder emplear las últimas características de Javascript y optimiza tu aplicación para entornos de producción.

En primer lugar, es indispensable tener instalado Node $\geq 14.0.0$ y npm ≥ 5.6 .

Una vez cumplido este requisito, para crear un nuevo proyecto deben ejecutarse los siguientes comandos:

```
npx create-react-app app
```

```
cd app
```

```
npm start
```

Con esto el proyecto estaría creado y podríamos comenzar con el desarrollo.

2.2 Desarrollo de los componentes de la aplicación

En React, los componentes son elementos autónomos que pueden reutilizarse en la página.

De este modo, podríamos utilizar el mismo componente FormComponent para crear varios formularios diferentes, empleando así menos líneas de código y recursos, favoreciendo también posteriores cambios. En caso de que quisiera añadirse un formulario más, serían muy pocas las líneas de código que haría falta añadir, y no afectaría a los formularios empleados en otras partes de la página web.

En el caso de esta aplicación se ha decidido crear un componente Tabla, que permita la construcción de tablas de forma sencilla y rápida, así como la ordenación de las mismas según las columnas indicadas.

Para ello, se ha creado una carpeta dentro de la carpeta app/src, llamada Components, y dentro se ha creado un archivo llamado TableComponents.

Es importante en este punto destacar que, para exportar una función o componente a otra parte de la aplicación, se puede indicar de dos maneras:

1. Creando la función y, al final del archivo, añadir la fórmula `export default MyFunctionName`
2. A la hora de crear la función, añadir anteriormente las palabras `export default`, por ejemplo, `export default function MyFunctionName(){}`

En este caso, se ha creado la función de la siguiente manera:

```
export default function TableComponent(data)
```

Donde 'data' es el parámetro que se le pasará a la hora de utilizar el componente en la aplicación.

El código completo de este archivo figura dentro del anexo adjuntado a esta memoria.

2.3 Creación de las vistas de la aplicación

En la aplicación existirán 3 vistas diferentes, como ya se ha mencionado antes, la primera de ellas mostrará un pequeño resumen del juego y su objetivo, la segunda una tabla con el ranking de jugadores y la tercera un formulario de registro.

Para ello, se sobrescribirá el archivo `app/src/app.js`.

Para que el cambio entre vistas funciones, deberá crearse un hook empleando la función `React.useState()`, pasándole como parámetro 0, que sería el valor inicial de la pestaña en la que está situada el usuario.

```
const [value, setValue] = React.useState(0);
```

`useState` es un Hook que te permite añadir el estado de React a un componente de función.

El único argumento para el Hook **`useState()`** es el estado inicial. Al contrario que en las clases, el estado no tiene por qué ser un objeto.

useState devuelve una pareja de valores: el estado actual y una función que lo actualiza.

En este código se emplean varios Hooks de estado, pero únicamente necesitaremos este para determinar la vista que se muestra.

Añadiremos también una función, **handleChange**, que será llamada cuando el usuario cambie de vista. Se le pasará como parámetro el nuevo valor, y empleará la función de estado **setValue()** para modificar el valor de la vista, permitiendo así que el contenido cambie.

El contenido de cada vista permanecerá oculto hasta que el usuario seleccione la misma, para ello se creará una clase en el archivo *app/src/App.css*, llamada 'hidden', la cual tiene una única propiedad, **display**, cuyo valor es '**none**'.

Un ejemplo de ello sería en la vista del ranking de jugadores:

```
<div style={{ marginTop: '3%', marginBottom: '5%', backgroundColor: 'white', width: '80%', marginLeft: '10%', height: 300 }}  
  
  className={value == 1 ? " : 'hidden'}>  
  
    <TableComponent rows={rowsData} users={usersData} headCells={headCells} />  
  
  </div>
```

Para permitir al usuario seleccionar cada vista, se mostrará en pantalla una barra de navegación que permitirá alternar entre las tres vistas, y que llamará a la anteriormente

mencionada función `handleChange` para actualizar el nuevo valor de vista.

Este sería pues el código para alternar las vistas:

```
<Tabs value={value} onChange={handleChange} sx={{ marginTop: 3 }}>

  <Tab label="Inicio" />

  <Tab label="Ranking" />

  <Tab label="Registro" />

</Tabs>
```

Para visualizar el código completo de la aplicación, pueden acceder al anexo adjuntado.

2.4 Funciones relacionadas con la API

Para que la aplicación pueda cumplir con las funcionalidades propuestas, deberá poder tener acceso a la API.

Las funciones que obtienen y/o modifican los datos necesarios son las siguientes:

2.4.1 Función para obtener el ranking de los jugadores

Esta función devolverá una colección en formato json del ranking de jugadores., haciendo uso de la URL creada durante el desarrollo de la API Rest.

La función sería la siguiente:

```
//get the ranking data

const getRankingData = async () => {
```

```

var api_url = 'http://20.70.7.244:8000/players/api/ranking/'

const dataResponse =

  await fetch(api_url);

const data = await dataResponse.json();

//the result will be our rows for the ranking table

setRowsData(data)

}

```

Como se puede observar en la última línea, utiliza la función llamada `setRowsData()` pasándole los datos de los jugadores como parámetro. Esta es una función para actualizar el valor del Hook de estado `rows` que, como se puede comprobar en el código de la aplicación, se pasa como parámetro al invocar al componente `<TableComponent/>` en la vista de ranking de la aplicación.

2.4.2 Función para obtener los usuarios

```

//users data

const getUsers = async () => {

  var api_url = 'http://20.70.7.244:8000/players/api/users/'

  const dataResponse =

    await fetch(api_url);

  const data = await dataResponse.json();

```

```
//we will use the users in order to determinate the player's username and to make the
player registration

setUsersData(data)

}
```

Esta función se emplea para obtener los usuarios registrados, al igual que en la función anterior, actualiza un Hook de estado, en este caso users, que se le pasa como parámetro al `<TableComponent>` y que permite determinar el nombre de usuario del jugador según el `user_id` del mismo.

Se adjunta ejemplo del código para aclarar este último punto:

```
{stableSort(rows, getComparator(order, orderBy))

  .slice(page * rowsPerPage, page * rowsPerPage + rowsPerPage)

  .map((row, index) => {

    const labelId = `enhanced-table-checkbox-${index}`;

    if (users !== undefined && rows !==undefined) {

      if (users !== [""] && rows !==[""]) {

        if (users.length > 2 && rows.length>2) {

          stableSort(users, getComparator(order, "id"))

          .map((user) => {

            if (row.user_id === user.id) {

              row.username = user.username

            }

          })

        }

      }

    }

  })

}
```

```
    }  
  }  
}
```

Asimismo, el Hook users también se utiliza a la hora de registrar un nuevo jugador, para garantizar que ni el nombre de usuario ni la contraseña estén siendo utilizados.

2.4.3 Función para registrar nuevos usuarios

A esta función se le pasa como parámetro el nombre de usuario, la contraseña y el email del nuevo usuario que va a registrarse.

```
//function to register a new user, we will pass the user, email and password  
  
async function registerUser(username, password, email) {  
  
    var api_url = 'http://20.70.7.244:8000/players/create/' + username + '/' + email + '/' +  
password  
  
    const dataResponse =  
  
    await fetch(api_url);  
  
}
```

Previamente a la llamada de la función, se deben realizar las comprobaciones necesarias para garantizar que el nombre de usuario y el correo electrónico no han sido empleados previamente por otro jugador.

Para ello, se hace uso de dos funciones.

La primera de ellas, HandleSubmit, obtiene los datos del formulario, verifica que contengan los datos necesarios y, en caso de que así sea, emplea la segunda fórmula, isList, que recibe como parámetro un valor y una clave, para validar la unicidad de los datos.

En caso de que los datos del formulario sean válidos, se llamará a la función para registrar a nuevos usuarios, y se indicará que el usuario ha sido registrado para mostrar una imagen de confirmación. En caso contrario, se señalará dónde figura el error.

Función HandleSubmit :

```
//function to handle the submitted form

const handleSubmit = (event) => {

  //first we get the data from the inputs

  var username = document.getElementById("username-field").value

  var password = document.getElementById("password-field").value

  var email = document.getElementById("email-field").value

  //validate the data

  if (username != null && username != "" && password != null && password != "" && email != null && password != "") {

    //we will use our predefined function in order to know if the user or the email are already in use,

    //if is the case, we will show the error in the corresponding field

    if (isList(username, 'username')) {

      setUserUsed(true)

    } else {

      setUserUsed(false)

      if (isList(email, 'email')) {

        setUserEmail(true)

      }

    }

  }

}
```

```
        //if everything is correct we send the data to our register function and set the user
        registered state at true in order

        //to show our welcome message and let the user know that they registered without
        problems

        else {

            setUsedEmail(false)

            registerUser(username, password, email)

            setRegistered(true)

        }

    }

}
```

Función isInList:

```
//function to know if an element is in a list, we will pass the key and the value to find

function isInList(valueToFind, key) {

    //since the functions to get the data could return a promise, first we need to verify that
    there has been no errors before

    //mapping the array or it will throw us an error

    if (usersData !== null && usersData !== [] && usersData.length > 2) {

        var isInList = false

        usersData.map((user) => {

            if (user[key] == valueToFind) {

                isInList = true

            }

        })

    }

}
```

```
    })  
  
  }  
  
  return isInList  
  
}
```

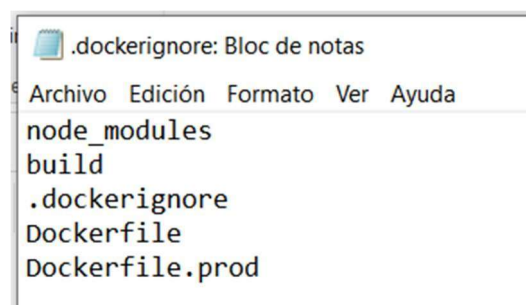
Para poder visualizar de forma más clara las funciones a las que se hace referencia, así como el resto del código de la aplicación, por favor, visualizad el anexo.

2.5 Creación de contenedor Docker para almacenar la aplicación

En primer lugar, antes de comenzar a crear el resto de archivos necesarios para lanzar el contenedor de la aplicación, se deberá configurar un archivo `.dockerignore`.

El archivo `.dockerignore` tiene un funcionamiento similar al archivo `.gitignore` que se suele crear cuando se trabaja con GIT, e indica que deben ignorarse determinados archivos o carpetas.

En este caso, el contenido del archivo debe ser el siguiente:



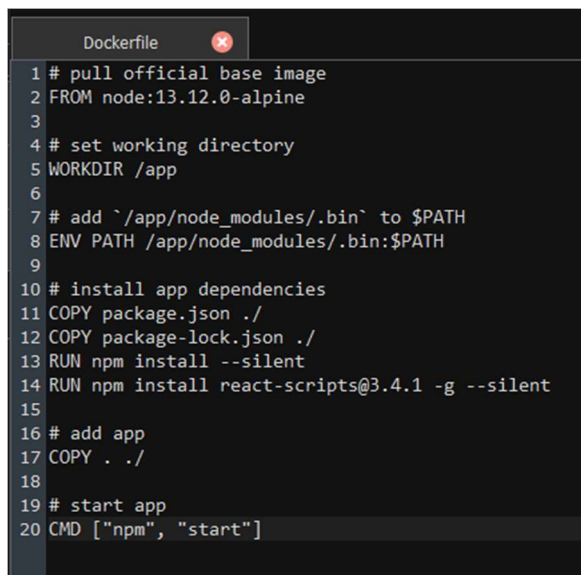
```
.dockerignore: Bloc de notas  
Archivo Edición Formato Ver Ayuda  
node_modules  
build  
.dockerignore  
Dockerfile  
Dockerfile.prod
```

La carpeta `node_modules` contiene todos los módulos empleados en la aplicación de React. Esta carpeta no es necesario adjuntarla cuando se envía o sube un proyecto con React, ya que ocupa gran cantidad de espacio de almacenamiento y ralentiza el envío o subida del proyecto. Además, al ejecutar el comando ***npm install*** se instalan los módulos necesarios definidos en los archivos ***package.json*** y ***package-lock.json***.

2.5.1 Creación de archivo Dockerfile

Al igual que durante la creación del servidor web, deberemos crear un archivo **Dockerfile** para poder crear un contenedor Docker. Este contenedor se construirá y subirá junto con el contenedor del backend, como se verá más tarde en la modificación del archivo docker-compose.

El archivo Dockerfile, situado en el path *project/app/* tendrá el siguiente contenido:

A screenshot of a code editor window titled 'Dockerfile'. The window contains 20 lines of Dockerfile instructions. The instructions are: 1. # pull official base image, 2. FROM node:13.12.0-alpine, 3. (blank), 4. # set working directory, 5. WORKDIR /app, 6. (blank), 7. # add `/app/node_modules/.bin` to \$PATH, 8. ENV PATH /app/node_modules/.bin:\$PATH, 9. (blank), 10. # install app dependencies, 11. COPY package.json ./, 12. COPY package-lock.json ./, 13. RUN npm install --silent, 14. RUN npm install react-scripts@3.4.1 -g --silent, 15. (blank), 16. # add app, 17. COPY . ./, 18. (blank), 19. # start app, 20. CMD ["npm", "start"]

```
1 # pull official base image
2 FROM node:13.12.0-alpine
3
4 # set working directory
5 WORKDIR /app
6
7 # add `/app/node_modules/.bin` to $PATH
8 ENV PATH /app/node_modules/.bin:$PATH
9
10 # install app dependencies
11 COPY package.json ./
12 COPY package-lock.json ./
13 RUN npm install --silent
14 RUN npm install react-scripts@3.4.1 -g --silent
15
16 # add app
17 COPY . ./
18
19 # start app
20 CMD ["npm", "start"]
```

2.5.2 Modificación del archivo docker-compose

Por último, faltaría modificar el contenido del archivo docker-compose.yml situado bajo la carpeta *project/*, actualizándolo de la siguiente manera:

```
1 # Specifies which syntax version of Docker compose
2 version: '3'
3
4 # Build a multiservice architecture.
5 services:
6   # Create a service called web
7   web:
8     # Build an image from the files in the project root directory (Dockerfile)
9     build: ./backend
10    # Assigns a name for the container. If no name is specified,
11    # Docker will assign the container a random name
12    container_name: backend_app
13
14    ports:
15      - "8000:8000"
16  frontend:
17
18    build: ./app
19    container_name: frontend_app
20    ports:
21      - "3000:3000"
```

2.6 Resultado final

Una vez hayamos realizado los pasos anteriores, podremos comprobar que la web funcionará y será accesible desde cualquier parte utilizando la ruta <http://20.70.7.244:3000/>.

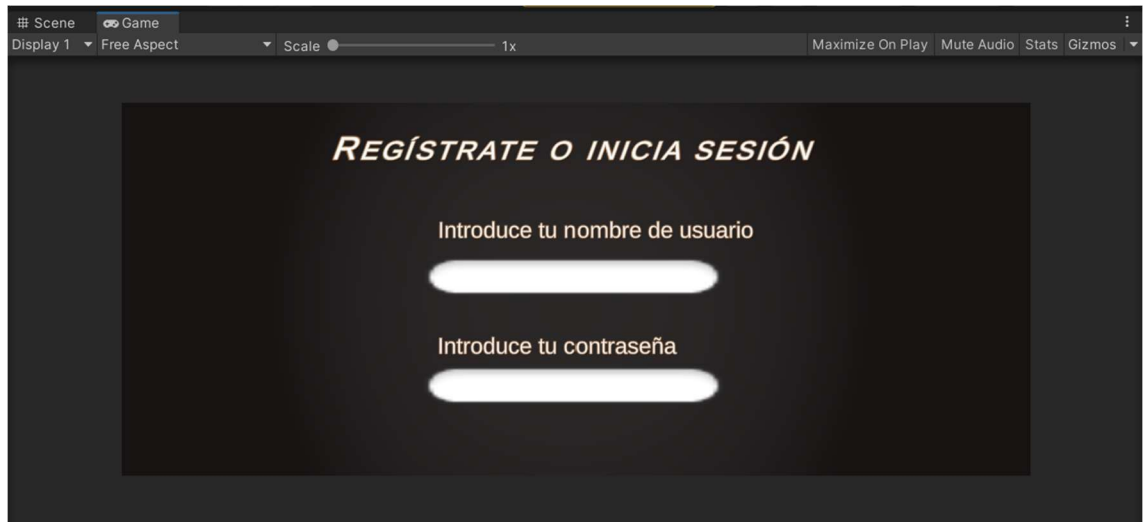
Huelga decir que la ruta será diferente dependiendo de la IP de la máquina de Azure que funcione como servidor

3. Desarrollo del videojuego

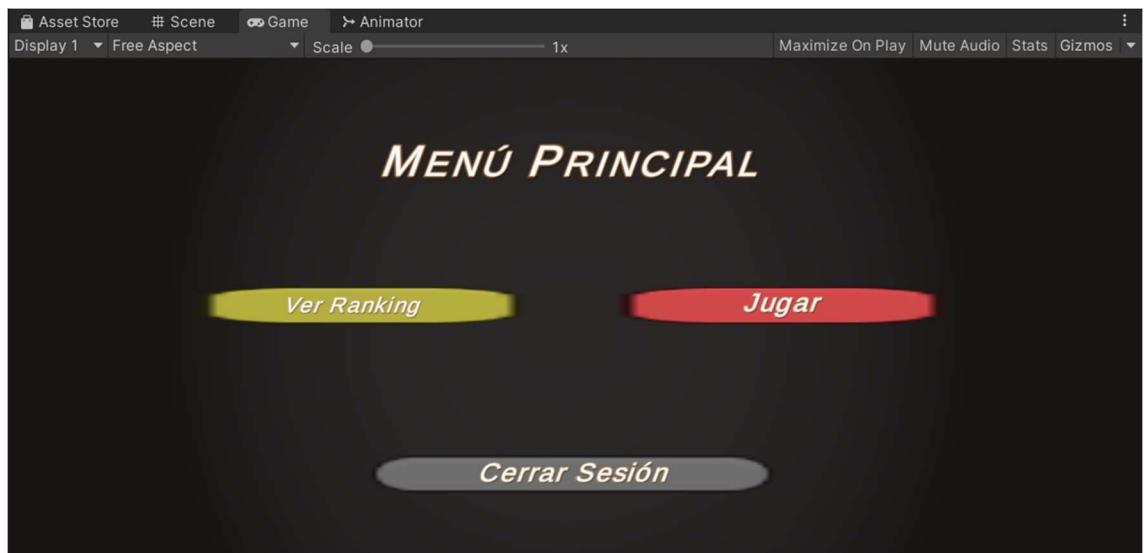
Para el desarrollo del videojuego se ha escogido emplear Unity, como se ha explicado anteriormente, proporciona una serie de facilidades al usuario para crear videojuegos de realidad virtual compatibles con Google Cardboard que se han considerado óptimas por ser sencillas, gratuitas y variadas.

En un videojuego desarrollado con Unity, se deben proporcionar diversas escenas, cada una de las cuales sirve a un propósito específico, así pues, el primer paso ha sido realizar la creación de dichas escenas, las cuales, teniendo en cuenta los requisitos y objetivos del videojuego, son las siguientes:

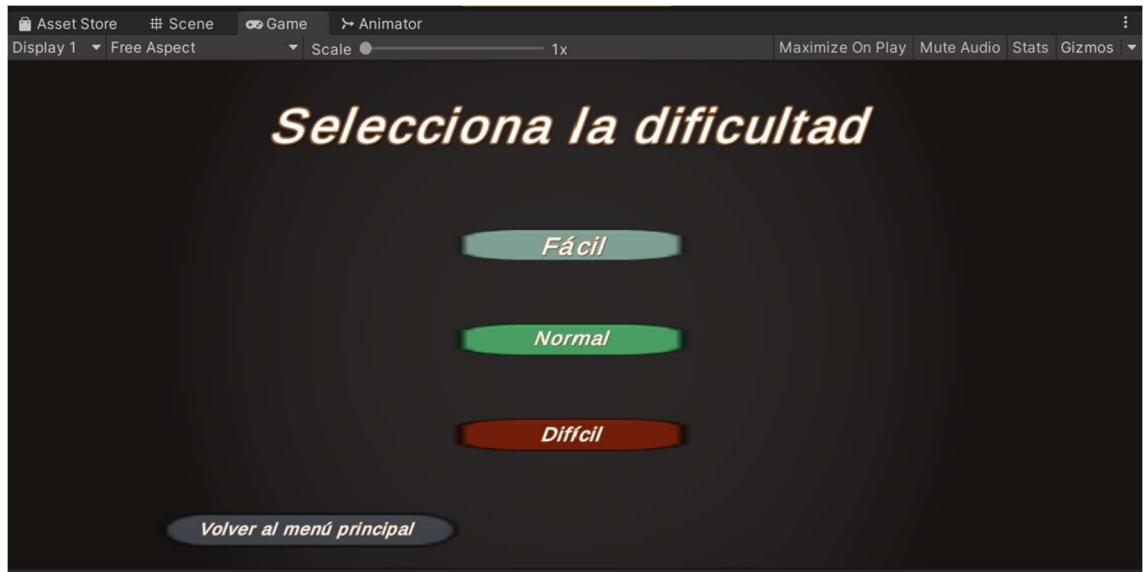
1. Pantalla inicial, en la que se realiza el Login del usuario si este ya está registrado. En caso contrario, se le redireccionará a la página web del juego para realizarlo.



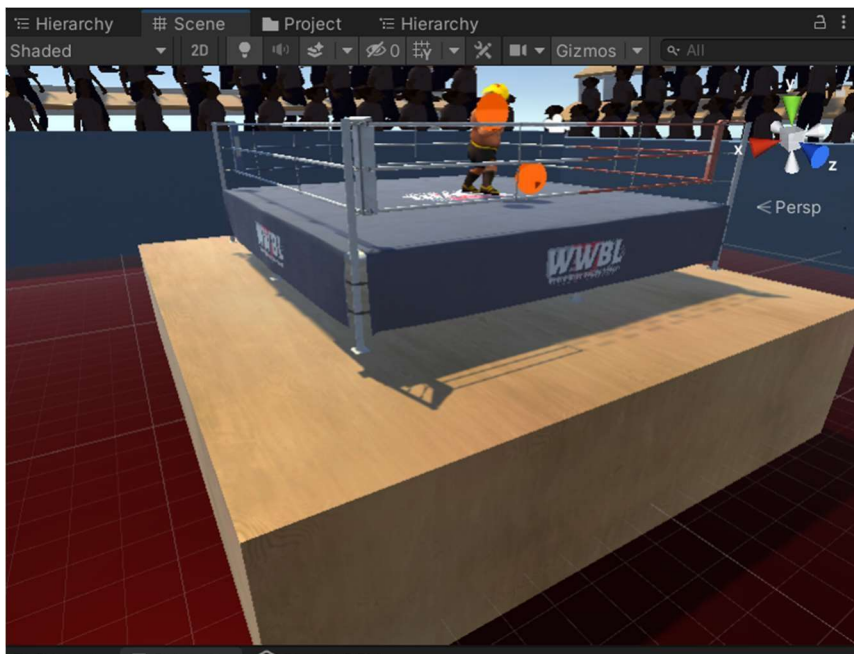
2. Pantalla de menú principal. En esta el usuario escogerá entre jugar, cerrar sesión o ver ranking. En caso de que se seleccione esta última opción, el jugador será redirigido a la página web de registro.

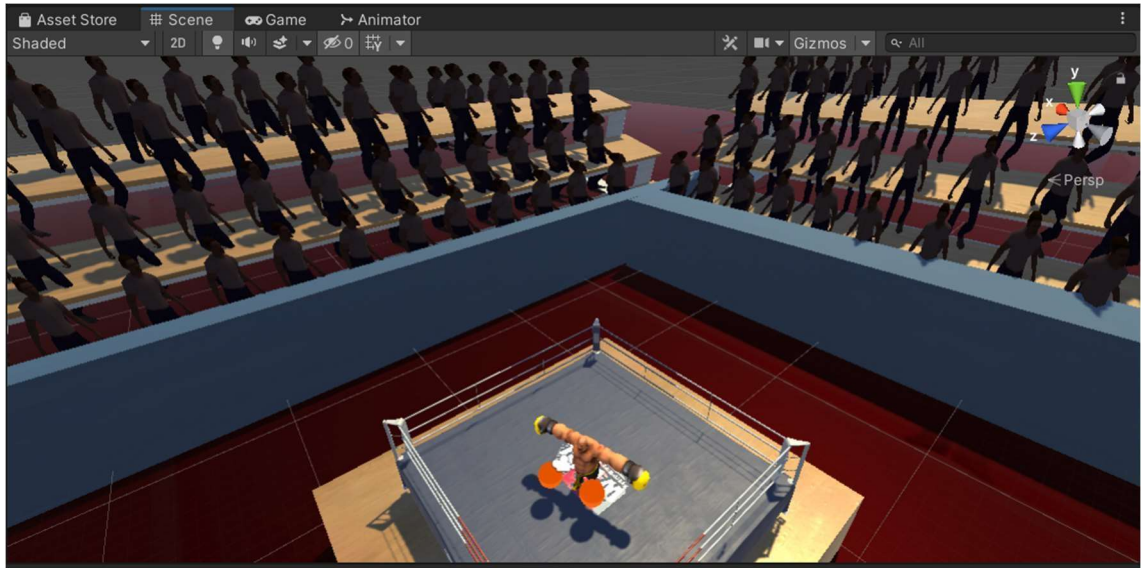


3. Pantalla de selección de dificultad. Si el usuario escoge jugar, se le redireccionará a esta pantalla, en la cual podrá seleccionar entre las tres dificultades disponibles: Fácil, Normal y Difícil.



4. Pantalla de juego. En esta pantalla será donde el usuario pueda jugar. En ella se ha pretendido la simulación de un estadio, incorporando para ello elementos físicos tales como gradas, personas, un ring de boxeo y paredes. Asimismo, se ha creado un enemigo ante el cual deberá enfrentarse el jugador, y cuya velocidad de movimiento vendrá determinada por la dificultad escogida previamente, así como el tiempo entre cambios de movimientos.





La composición de las pantallas se puede observar más detallada en el Anexo incluido en esta memoria, en la cual se especifican los objetos de los cuales se compone, así como los Scripts y animaciones empleados.

4. Diseño de los guantes

4.1 Componentes físicos

Por último faltaría únicamente el diseño de los guantes que servirán para controlar la posición del jugador y enviarle los datos de la misma a Unity.

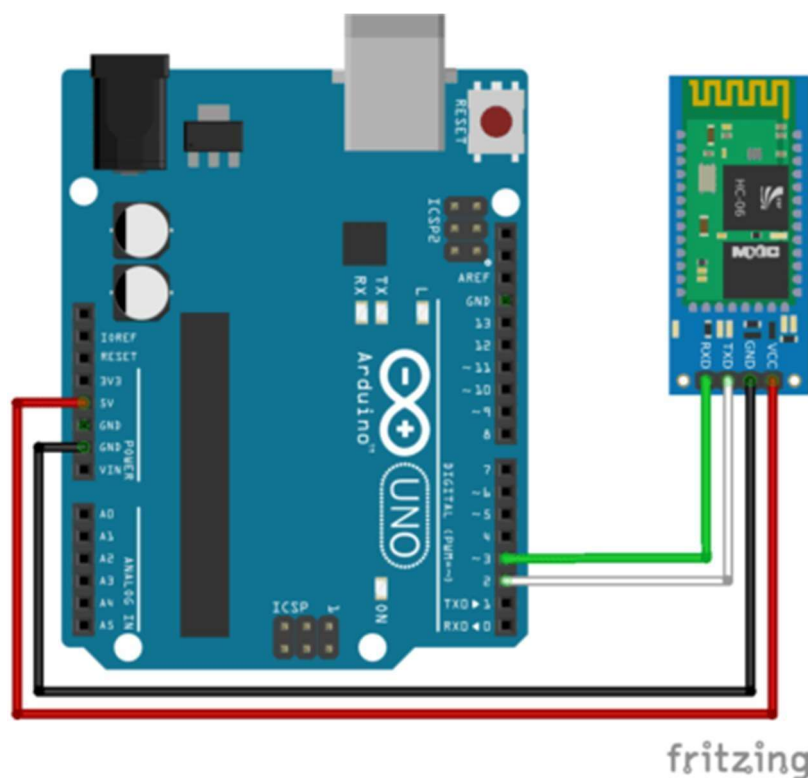
En un principio se barajaron diferentes alternativas, dado el presupuesto disponible inicialmente se optó por un microcontrolador ESP01.

Finalmente, se optó por emplear un módulo bluetooth para realizar las comunicaciones con Unity. En comparación con el microcontrolador ESP01, es más sencillo de programar, tal y como se indicará a continuación las conexiones son sencillas y rápidas. Como ventaja adicional, su reducido tamaño permite un mejor acople a los guantes.

Otras de las ventajas de emplear este módulo sería que posee buenas características de transmisión y recepción de información, así como el bajo consumo de corriente que posee tanto en funcionamiento como en modo espera.

El módulo HC-06 dispone de 4 pines, dos de ellos para la alimentación y los otros dos para las conexiones.

El esquema de conexión sería el siguiente :



HC-06

ARDUINO

VCC

5V

GND	GND
RX	TX(3)
TX	RX(2)

Por otro lado, debemos considerar también las conexiones con el módulo MPU-6050.

Este se trata de un sensor que contiene un acelerómetro MEMS y un giroscopio MEMS incorporados en un mismo chip. Un MEMS (MicroElectroMechanical Systems) funciona de forma similar a un sistema masa resorte, por lo que permite medir la aceleración.

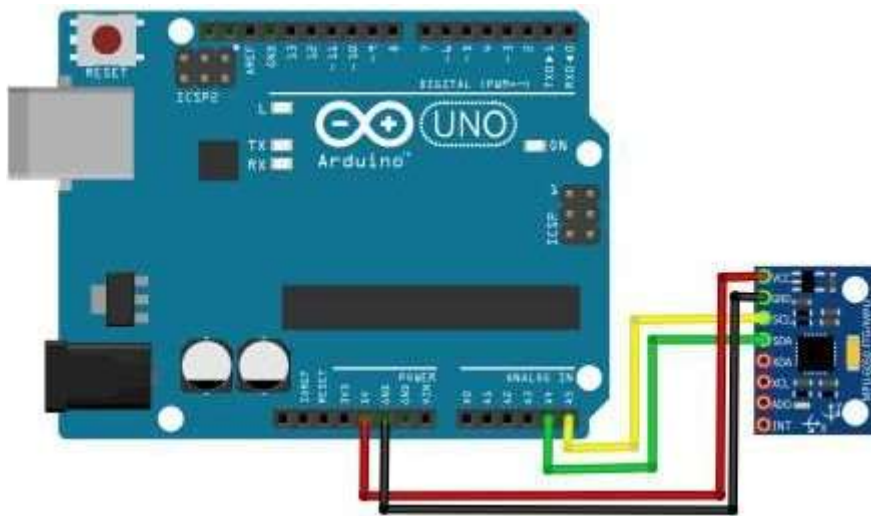
Asimismo, contiene hardware de conversión de analógico a digital de 16 bits para cada canal, lo cual permite que pueda capturar los canales 'X', 'Y', y 'Z' al mismo tiempo.

La comunicación de este módulo es por I2C, esto le permite trabajar con la mayoría de microcontroladores.

El motivo de la elección de este módulo es que, al incorporar al mismo tiempo un giroscopio de tres ejes y un acelerómetro triaxial, se trata de una gran elección a la hora de desarrollar aplicaciones y videojuegos que incorporen realidad virtual.

En este caso, la conexión con Arduino debe realizarse de la siguiente manera:

MPU-6050	ARDUINO
SCL	A5
SDA	A4
GND	GND
VCC	3.3V



En este punto es importante mencionar las protoboard.

Una protoboard es una placa de pruebas, contiene numerosos orificios en los que se pueden insertar cables y otros elementos electrónicos para la realización de circuitos provisionales. La ventaja de este dispositivo es que no requiere soldar sus componentes para tener un circuito operativo.

Una protoboard se encuentra construida por las siguientes partes esenciales.

- Canal central, ubicado en la parte media del tablero y donde van conectados los circuitos integrados para lograr el aislamiento de los pines de los dos lados.
- Buses: localizados a los lados de la protoboard, están identificados por franjas de color negro o azul que indican el bus de tierra; pero también, por franjas rojas que denotan el bus de voltaje positivo.

Las protoboard específicas para Arduino generalmente vienen ya montadas, por lo que no es necesario soldar.

Sobre la placa, se montan los circuitos auxiliares y se procede a la prueba de los mismos.

En este caso, al haberse realizado diversas pruebas, se ha considerado necesario adquirir una. Sin embargo, deben tenerse en cuenta las diversas desventajas derivadas de su uso, sobre todo en circuitos más complejos:

- Solo son útiles a frecuencias bajas, no mayores a 20MHz.
- No puede utilizarse con corrientes mayores a 5A.

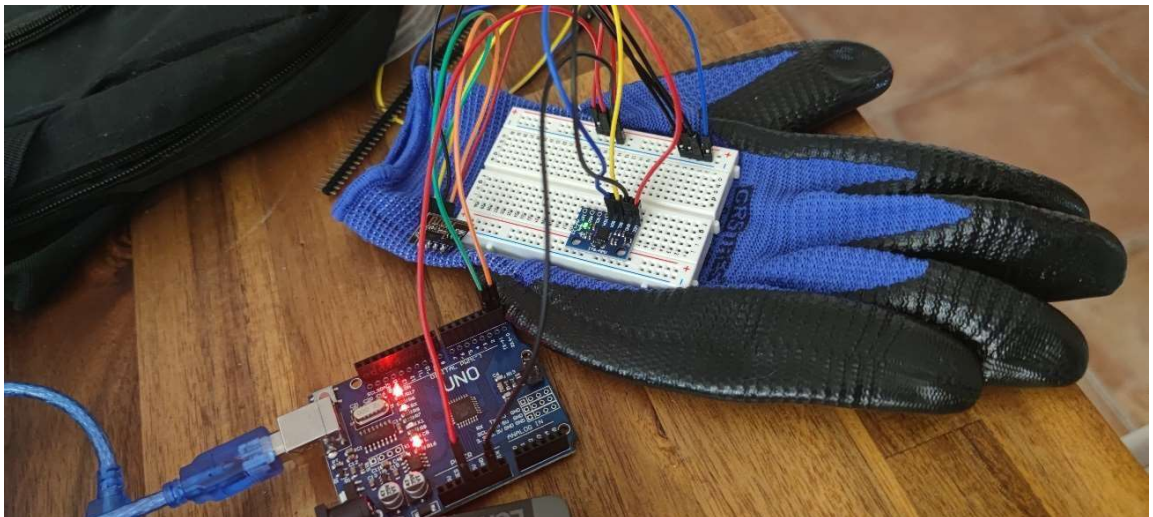
- Es peligroso a la hora de trabajar con altos voltajes.
- Pueden establecerse falsos contactos.
- En ocasiones el cableado se puede complicar, sobre todo cuando hay muchos componentes interconectados.

4.2 Software

A la hora de desarrollar el software que permita la comunicación de los guantes con el videojuego, debe tenerse, por un lado, el código que se subirá a Arduino y, por otro, el que emplearemos en los scripts de Unity para obtener la información.

En un primer lugar, indicar que, para el código de Arduino, es recomendable emplear Arduino IDE, puesto que provee de útiles y variadas funcionalidades a la hora de trabajar con estos componentes.

Para poder subir el código a Arduino, se ha empleado un cable B011 para conectar la placa al ordenador.



1. Imagen tomada durante la prueba de conexiones.

De esta forma, además, se le provee de alimentación, puesto que el proyecto no cuenta con alimentación externa, dado que se ha tratado de obtener el mayor rendimiento con el menor uso de recursos posible.

Una vez abierto Arduino IDE se debe escoger el tipo de placa que se está utilizando desde la pestaña de herramientas, y seleccionar el puerto serie al que está conectada.

Para la comunicación con el módulo Bluetooth, debemos asegurarnos de incluir la librería `SoftwareSerial`, que permite la comunicación serial en otros pines digitales de la placa de Arduino, usando el software para replicar la funcionalidad.

Para la conexión con Arduino, se debe definir el puerto Bluetooth, haciendo uso de la librería mencionada, y pasarle como argumentos los pines de transmisión y de recepción, siendo en este caso:

`SoftwareSerial BlueSerial (2,3);`

El pin 2 y 3 indican, respectivamente, el pin RX y el pin TX en el microcontrolador.

Para iniciar la comunicación deberemos abrir el puerto serie e indicar la velocidad de transmisión (*baud rate*) a la que queremos trabajar. Mencionar que es de suma importancia que todos los dispositivos que interfieran en la comunicación se encuentren en la misma velocidad.

Para ello, utilizaríamos la siguiente función:

`BlueSerial.begin(9600); //Establecemos la velocidad de transmisión en 9600`

Para enviar los datos vía puerto serie se debe utilizar la función `Serial.print()`, por ejemplo:

`BlueSerial.print("Comunicando por Bluetooth...");`

Se ha de tener en cuenta que se está programando utilizando C++, por lo que el código estará sujeto a las funciones y características de dicho lenguaje de programación.

Para la recepción de los datos obtenidos con el módulo MPU-6050, debemos tener en cuenta que, con el acelerómetro que incorpora, podemos realizar mediciones indirectas.

Por ejemplo, si se integra la aceleración en el tiempo se obtiene la velocidad, y si se integra nuevamente, se obtendrá el desplazamiento, necesitando en ambos casos la velocidad y la posición inicial respectivamente.

$$Xf = 1/4 (Af + Ao) t^2 + Vo t + Xo$$

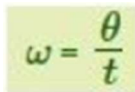
Donde **Xf** es la distancia final en metros, **Af** es la aceleración CORRIENTE final en m/s², **Ao** es la aceleración previa del último conjunto de datos en m/s², **t** es el CAMBIO en el tiempo ENTRE los conjuntos **Af** y **Ao** de datos en SEGUNDOS, **Vo** es la velocidad instantánea del último conjunto de datos en m/s, y **Xo** es la distancia final del último conjunto de datos o la suma de todas las distancias anteriores en metros.

Vo debe calcularse utilizando la aceleración anterior y la aceleración de dos conjuntos de datos anteriores o **Ao-1** utilizando la siguiente ecuación cinemática:

$$Vo = 1/2 (Ao + Ao-1) * t + Vo-1$$

Donde **Vo** es la velocidad instantánea previa en m/s, **Ao** es la aceleración previa en m/s², **Ao-1** es la aceleración de dos conjuntos de datos en m/s², **t** es el cambio en el tiempo entre **Ao** y Conjuntos de datos **Ao-1** en SEGUNDOS, y **Vo-1** es la velocidad instantánea en m/s del conjunto de datos **Ao-1** o de dos conjuntos de datos.

Por otra parte, con el giroscopio, se puede medir la velocidad angular, y, si se integra la velocidad angular con respecto al tiempo, obtener el desplazamiento angular (posición angular si se sabe dónde se inició el giro).

A small yellow rectangular box containing the mathematical formula for angular velocity: \omega = \frac{\theta}{t}.

Se deberán incluir las siguientes librerías para trabajar en este proyecto, además de la anteriormente mencionada:

```
#include "I2Cdev.h"
```

```
#include "MPU6050.h"
```

```
#include "Wire.h"
```

El pin ADDR internamente en el módulo tiene una resistencia a GND, por lo que si no se conecta, la dirección por defecto será 0x68. Esto será importante a la hora de crear la variable u objeto para el MPU6050, ya que, en caso de no conectarse este pin, podría crearse de la siguiente manera:

```
MPU6050 sensor;
```

Pero, en caso de querer trabajar con otra dirección, debería modificarse de la siguiente manera:

```
MPU6050 sensor(0x69); //El MPU6050 únicamente trabaja en 0x68 y 0x69
```

En la función void loop() habrá que iniciar, tanto la comunicación I2C y el sensor, como la comunicación serial por bluetooth tal y como se ha descrito anteriormente:

```
BlueSerial.begin(9600); // Iniciando comunicación serial por bluetooth
```

```
Wire.begin(); //Iniciando I2C
```

```
sensor.initialize(); //Iniciando el sensor
```

Al inicializar el sensor, los rangos por defecto serán:

- acelerómetro -2g a +2g
- giroscopio: -250°/sec a +250°/sec

Teniendo en cuenta que la resolución de las lecturas es de 16 bits, el rango de lectura es de -32768 a 32767.

Siguiendo en la función loop, se deberá realizar las lecturas de la siguiente manera:

```
sensor.getAcceleration(&ax, &ay, &az);
```

```
sensor.getRotation(&gx, &gy, &gz);
```

Esto almacenará los datos en las variables pasadas como parámetro.

Para escalar las lecturas y convertir el valor leído en un valor de aceleración o velocidad angular, se deberán usar las siguientes ecuaciones:

```
float ax_m_s2 = ax * (9.81/16384.0); //Repetir con az y ay
```

```
float gx_deg_s = gx * (250.0/32768.0); //Repetir con gz y gy
```

Para calcular el ángulo de inclinación, si se tiene en cuenta que la única fuerza que actúa sobre el sensor es la fuerza de la gravedad, entonces los valores que se obtienen en

los componentes del acelerómetro corresponden a la gravedad y los ángulos de la resultante serán la inclinación del plano del sensor, puesto que la gravedad siempre es vertical.

Para calcular los ángulos de inclinación en un espacio 3D, tanto en X como en Y, deben usarse las siguientes fórmulas:

$$\theta_x = \tan^{-1} \left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}} \right)$$

$$\theta_y = \tan^{-1} \left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}} \right)$$

Así pues, para calcular los ángulos de inclinación, usaríamos lo siguiente:

```
float accel_ang_x=atan(ax/sqrt(pow(ay,2) + pow(az,2)))*(180.0/3.14);
```

```
float accel_ang_y=atan(ay/sqrt(pow(ax,2) + pow(az,2)))*(180.0/3.14);
```

Para calcular el ángulo actual se debe integrar la velocidad y conocer el ángulo actual, lo cual se realiza con la siguiente fórmula:

$$\theta_x = \theta_{x0} + \omega_x \Delta t$$

$$\theta_y = \theta_{y0} + \omega_y \Delta t$$

Así pues, en el código de Arduino sería de la siguiente forma:

```
dt = millis()-tiempo_prev;
```

```
tiempo_prev=millis();
```

```
girosc_ang_x = (gx/131)*dt/1000.0 + girosc_ang_x_prev;
```

```
girosc_ang_y = (gy/131)*dt/1000.0 + girosc_ang_y_prev;
```

```
girosc_ang_x_prev=girosc_ang_x;
```

```
girosc_ang_y_prev=girosc_ang_y;
```

FASE IV: Evaluación y conclusiones finales

1. Complicaciones encontradas durante el desarrollo del proyecto

A lo largo del desarrollo del proyecto han ido apareciendo diversas complicaciones.

En primer lugar, la falta de presupuesto no ha permitido la adquisición de los recursos óptimos, por lo cual se procedió a buscar alternativas económicas que permitiesen el correcto funcionamiento del proyecto.

Por otra parte, muchos de los conocimientos empleados se han adquirido durante el proceso, limitando así el tiempo dedicado al desarrollo.

Uno de los primeros inconvenientes que surgió fue la planteación inicial. En ella, se empleaba el microcontrolador ESP01 para permitir la comunicación entre los guantes y el videojuego.

El microcontrolador ESP8266 ESP01 es un chip de bajo coste, que tiene la capacidad de conectarse mediante WIFI y que puede emplearse como placa base o como módulo. El principal problema a la hora de emplear estos dispositivos viene dado por su difícil configuración. Se puede programar a través de un adaptador serie/USB o con el cableado adecuado, a través de Arduino. Los conectores que vienen por defecto, no permiten conectarlo a la protoboard, y teniendo en cuenta, además, el objetivo que se quiere alcanzar con el proyecto, sería difícil mantener una conexión estable con un teléfono, por no mencionar que sería poco práctico, ya que se deberían realizar diversas modificaciones cada vez que quiera cambiarse de dispositivo.

Tras probarlo y concluir que no era apto para el objetivo al que pretende llegar el proyecto, se decidió por emplear un módulo bluetooth. Sin embargo, esto ralentizó el proceso de desarrollo del proyecto.

A nivel técnico, la mayor parte de problemas se han presentado durante el desarrollo de los guantes. Uno de los principales inconvenientes fue el uso del módulo MPU-6050.

Pese a que cumple los requisitos del proyecto, y que otros sensores similares presentan el mismo problema, es necesario señalar que, dado que es altamente improbable que el sensor se encuentre en una posición 100% horizontal, conllevando posibles errores de lectura, es necesario calibrarlo, configurando OffSets y compensar de esa forma los errores que puedan recibirse.

Para solventar este inconveniente, se hicieron cambios en el código para introducir la calibración, de manera que se están modificando constantemente los offset, intentando eliminar el error con la medida real que deseamos, en este caso $a_x=0, a_y=0, a_z=1g$ y $g_x=0, g_y=0, g_z=0$. Durante la calibración, es necesario ubicar el sensor en posición horizontal y evitar moverlo, ya que esta posición será el nivel para futuras posiciones.

Durante la calibración el programa realiza las lecturas tanto del acelerómetro como del giroscopio, usando un filtro estabilizamos un poco las lecturas y cada 100 lecturas comprobamos si los valores son cercanos a los valores que deseamos leer, dependiendo de esto se aumenta o disminuye los offsets. Esto hará que las lecturas filtradas converjan a:

• *Aceleración:* $p_{ax}=0, p_{ay}=0, p_{az}=+16384$

• *Velocidad angular:* $p_{gx}=0, p_{gy}=0, p_{gz}=0$

Se debe esperar que los valores se acerquen a: 0 0 +16384 0 0 0.

Este paso entre la calibración y la obtención de los datos reales se realiza desde Unity.

Otro de los inconvenientes relacionados con el uso de sensores como el MPU-6050 es que la medida no es exacta.

Incluso cuando no se mueve el ángulo varía, o si se gira cierto ángulo y luego se regresa a la posición original el ángulo que medimos no es el inicial, esto se debe a que, al integrar la velocidad angular y sumar el ángulo inicial, hay un error producto de la mala medición del tiempo o del ruido en la lectura del MPU, el error por más pequeño que sea, se va acumulando en cada iteración y creciendo, este error es conocido como DRIFT.

Para disminuir el drift existen varios métodos, la mayoría aplica filtros para eliminar el ruido de las lecturas del sensor. También se pueden usar otros sensores como magnetómetros o acelerómetros y con los ángulos calculados con estos mejorar el cálculo del giroscopio.

Para ello en este caso se ha optado por el filtro de complemento. Este combina el ángulo calculado por el giroscopio y el ángulo calculado por el acelerómetro.

De esta forma el ángulo del acelerómetro está pasando por un filtro pasa bajos, amortiguando las variaciones bruscas de aceleración, y el ángulo calculado por el giroscopio tiene un filtro pasa altos, teniendo gran influencia cuando hay rotaciones rápidas.

//Calcular ángulo de rotación con giroscopio y filtro complemento

$ang_x = 0.98*(ang_x_prev+(gx/131)*dt) + 0.02*accel_ang_x;$

//Calcular ángulo de rotación con giroscopio y filtro complemento

ang_x = 0.98(ang_x_prev+(gx/131)*dt) + 0.02*accel_ang_x;*

2. Evaluación y conclusiones

Como se puede comprobar, se ha alcanzado el objetivo del proyecto.

Se ha creado el servidor web basado en REST, el cual se comunica con la página web creada para el registro y vista del ranking de jugadores, así como con el videojuego de realidad virtual de Unity que se ha desarrollado según lo previsto.

A su vez, los guantes que se han creado funcionan correctamente tal y como se predijo.

Si bien es cierto que, dadas las limitaciones en cuestión de recursos, no se ha obtenido un resultado óptimo, ya que, por ejemplo, las pruebas se han realizado desde el ordenador, empleando para ello el cable B011 que permite conectar la placa de Arduino al puerto USB y dotarle así de alimentación.

Sin embargo, y pese a que existan posibles mejoras, con el desarrollo de este proyecto se ha podido demostrar la gran cantidad de herramientas, programas y servicios que pueden crearse incluso con los recursos mínimos.

La realidad virtual no se limita únicamente a los videojuegos, existen muchos campos en los que su aplicación podría contribuir a realizar mejoras en los mismos. Sería el caso, por ejemplo, del desarrollo de entornos en los cuales puedan realizarse simulacros de incendios o accidentes que, por sus características, podrían no ser aptos para realizarse de forma asidua en entornos reales.

Asimismo, podría emplearse para la simulación de operaciones en las cuales las personas que se dedican a la cirugía puedan practicar de forma realista de forma previa a una operación crítica que no permita la existencia de errores.

Y, en lo referente a Arduino, si bien durante los últimos años no ha hecho más que demostrarse su versatilidad y sus cuantiosas aplicaciones, es importante seguir remarcándolo, puesto que uno de los motivos por los cuales se ha podido desarrollar hasta ese punto, y por los cuales no se prevee que vaya a dejar de hacerlo, es precisamente por su código abierto.

En general tanto durante la fase de investigación como de desarrollo se ha podido observar que aquellos programas o proyectos que cuentan con código abierto tienen una comunidad más amplia y sólida de contribuidores que permite tanto facilitar el aprendizaje a personas sin conocimientos previos al respecto como realizar mayores avances de forma más rápida, de los cuales acaban beneficiándose no sólo los desarrolladores sino toda la sociedad en conjunto, puesto que muchos de estos avances pueden ser en materia de salud, de prevención de catástrofes... en conclusión, se ha podido observar lo relevante que es el código abierto incluso más allá del ámbito de la programación.

Referencias

AranaCorp. (s.f.). *Comunicación con arduino y módulo hc-06*. Obtenido de <https://www.aranacorp.com/es/comunicacion-con-arduino-y-el-modulo-hc-06/>

Arduino. (s.f.). *Software-serial documentation*. Obtenido de <https://docs.arduino.cc/learn/built-in-libraries/software-serial>

Ardunity. (s.f.). *Getting started with MPU series*. Obtenido de <https://sites.google.com/site/ardunitydoc/getting-started/run-examples/mpuseries>

Areaciencias. (s.f.). *Desplazamiento angular*. Obtenido de <https://www.areaciencias.com/fisica/desplazamiento-angular/>

DigitalOcean. (s.f.). *How to install and use docker on ubuntu 20-04*. Obtenido de <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04>

Django Software Foundation. (s.f.). *Django documentation*. Obtenido de <https://docs.djangoproject.com/en/4.0/>

Docker. (s.f.). *Docker documentation*. Obtenido de <https://docs.docker.com/>

ElectroCrea. (s.f.). *Módulo Bluetooth HC-06*. Obtenido de <https://electrocrea.com/blogs/tutoriales/33307075-modulo-bluetooth-hc-06#:~:text=Algunas%20de%20las%20ventajas%20del,como%20en%20modo%20de%20espera.>

ElectroWings. (s.f.). *MPU-6050 Interfacing with Arduino Uno*. Obtenido de <https://www.electronicwings.com/arduino/mpu6050-interfacing-with-arduino-uno>

M Olmo, R. N. (s.f.). *Descripción del movimiento en una dimensión*. Obtenido de <http://hyperphysics.phy-astr.gsu.edu/hbasees/mot.html>

Microsoft. (s.f.). *Create Windows Virtual Machine in Azure*. Obtenido de <https://docs.microsoft.com/es-es/learn/modules/create-windows-virtual-machine-in-azure/>

Mobatek. (s.f.). *MobaXterm documentation*. Obtenido de <https://mobaxterm.mobatek.net/documentation.html>

Rice University. (s.f.). *Aceleración media e instantánea*. Obtenido de <https://openstax.org/books/f%C3%ADsica-universitaria-volumen-1/pages/3-3-aceleracion-media-e-instantanea>

Rice University. (s.f.). *Calcular la velocidad y el desplazamiento a partir de la aceleración*. Obtenido de

<https://openstax.org/books/f%C3%ADsica-universitaria-volumen-1/pages/3-6-calcularla-velocidad-y-el-desplazamiento-a-partir-de-la-aceleracion>