UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN

Biblioteca
UNIVERSITAS Miguel Hernández

"IMPLEMENTACIÓN DE UN VIDEOJUEGO ONLINE MULTIJUGADOR PARA NAVEGADOR WEB"

TRABAJO FIN DE GRADO

Septiembre - 2022

AUTOR: Joan Marc Quiles Arronis
DIRECTOR/ES: Jesús Javier Rodríguez Sala

Biblioteca
UNIVERSITAS Miguel Hernández

RESUMEN

El proyecto consiste en la implementación de un videojuego online multijugador ejecutado en navegador. Dado que el juego no necesita instalación y funciona en cualquier navegador moderno (Chrome, Firefox, Safari…), se puede jugar desde cualquier dispositivo que soporte estos programas a excepción de dispositivos móviles.

La principal herramienta que he utilizado para el desarrollo es Phaser 3. Este framework de Javascript está diseñado específicamente para la creación de videojuegos ejecutados en navegador. Además de renderizar los sprites y controlar la lógica del juego, Phaser ofrece un motor de físicas que simplifica la programación del movimiento y las colisiones de los objetos.

Por otro lado, dado que el juego es multijugador en tiempo real, para garantizar una buena experiencia, la latencia debe ser mínima. Socket.io, un framework de Javascript, permite que se cree un canal directo entre cliente y servidor y, así, puedan transmitir cualquier tipo de información, ya sean valores simples o estructuras más complejas como arrays, en todo momento.

El juego está diseñado en 2D y, por tanto, carece de profundidad. Las partidas se desarrollan en una pantalla fija y con vista cenital. El apartado artístico, inspirado en la estética retro y los juegos arcade, está formado por sprites creados utilizando la técnica de Pixel Art. Este método de diseño se centra en resaltar los píxeles del dibujo y no suavizar las imágenes.

El aspecto jugable se centra en el combate entre dos equipos de 3 usuarios cada uno. Cada jugador toma el control de un tanque y lucha para destruir los vehículos enemigos, ganando el bando que primero lo consiga. Al terminar la batalla, cada participante obtiene cierta cantidad de recursos que puede utilizar para desbloquear más tanques con los que jugar.

Para terminar, quisiera mencionar que el proyecto está pensado para que su desarrollo no finalice nunca. Actualmente hay siete tanques y tres escenarios, además de las pantallas de inicio, carga y el menú, pero, en caso de que se comercializara, se iría añadiendo más contenido y funcionalidad con cada actualización.

Biblioteca
UNIVERSITAS Miguel Hernández

ÍNDICE GENERAL

Biblioteca
UNIVERSITAS Miguel Hernández

Biblioteca UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

ÍNDICE DE TABLAS

Biblioteca Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

ÍNDICE DE FIGURAS

Biblioteca Universidad Miguel Hernández

Capítulo 1
Introducción

1.1.- LA INDUSTRIA DE LOS VIDEOJUEGOS

1.1.1.- Crecimiento de la industria de los videojuegos

La industria de los videojuegos ha sufrido un gran crecimiento en los últimos años. En una década, el sector ha pasado de facturar setenta mil millones de dólares en 2012 a, se calcula, unos ciento noventa y seis mil millones de dólares en 2022. Este desarrollo es especialmente pronunciado en el periodo entre 2019 y 2022, donde la facturación se incrementó en sesenta mil millones, tan sólo ocho mil millones menos que el crecimiento entre 2012 y 2018 [1].

Siguiendo esta tendencia, se calcula que en 2027 el mercado del videojuego ascenderá a casi trescientos mil millones de dólares, dejando una tasa de crecimiento anual del 8,94%. El aumento en el sector se ha visto favorecido por factores como la mejora computacional de

ordenadores, móviles, videoconsolas; la gran disponibilidad de fibra óptica y redes móviles con gran ancho de banda; y, por último, el confinamiento impuesto ante la enfermedad por coronavirus (COVID-19) durante el que muchas personas tuvieron a los videojuegos como fuente principal de entretenimiento [2].

1.1.2.- Videojuegos sociales

Según el artículo *Gaming: The Next Super Platform*, el 84% de las personas entrevistadas veía los videojuegos como una manera de conectar con gente con gustos similares y el 80% afirmaba que, además, les ayudaba a conocer a gente nueva. Así mismo, el 77% de los participantes, utilizaba los videojuegos para relacionarse con sus amigos. Por último, el 67% pensaba que es fácil encontrar buenas comunidades sobre los videojuegos online.

Esta tendencia se puede ver en las costumbres de los jugadores, ya que, a la semana, dedican 16 horas a jugar, 8 a participar en streamings sobre videojuegos y 6 horas a interactuar con otros usuarios en comunidades y foros. Tal y como se ve en el artículo previamente mencionado, el aspecto social viene de la mano de los videojuegos online y multijugador. El 76% de las personas aseguraba que, en el último año, sobre todo había jugado a videojuegos online y el 73% esperaba dedicar aún más tiempo a este tipo de juegos [3].

1.2.- JUSTIFICACIÓN DEL PROYECTO

1.2.1.- Videojuegos más jugados

Dentro del gran crecimiento de la industria del videojuego, hay una categoría que predomina sobre las demás: juegos multijugador online. Esta tendencia se puede observar en las listas de los videojuegos más jugados en ordenador y dispositivos móviles.

En el top 10 de Steam, la plataforma de distribución digital de videojuegos más popular para Windows y Linux, encontramos que nueve de los diez juegos o son totalmente multijugador o tienen un modo que lo permite [4]. Con una colección de juegos algo distinta, Epic nos ofrece un resultado similar, teniendo ocho de los diez juegos opciones multijugador [5].

En el apartado móvil existen dos grandes plataformas, la Play Store de Android y la App Store de Apple. En la primera de ellas, siete juegos del top 10 son multijugador online y

tres de un solo jugador [6]. Por otro lado, en la tienda de Apple encontramos que, en junio de 2022, ocho de diez juegos son multijugador en línea [7].

Además de en las tiendas digitales, se puede ver la popularidad de este tipo de juegos en plataformas de streaming como Twitch. De los videojuegos con más espectadores, los veinte primeros o son completamente multijugador en línea, o tienen funciones que permiten jugar de ese modo [8].

1.2.2.- Motivación personal

Dedicarme al desarrollo de videojuegos es un deseo personal que tengo desde hace mucho tiempo. Por otro lado, en la carrera se cursan múltiples asignaturas orientadas a la creación de aplicaciones web, tanto front end como back end. Es de la fusión de ambas áreas que nace mi motivación para realizar este proyecto.

1.3.- OBJETIVOS

1.3.1.- Objetivos principales

Este proyecto tiene como objetivo principal la creación de una aplicación web que, además de tener aspectos de red social, gire en torno a un videojuego multijugador en línea. Aparte de buscar elementos intangibles como el entretenimiento de los jugadores y una comunidad de juego saludable y sin usuarios “tóxicos”, las funcionalidades técnicas que se buscan son las siguientes:

- Registrarse e iniciar y cerrar sesión.
- Buscar el perfil de otro usuario.
- Seguir/dejar de seguir a otros usuarios.
- Chatear con los usuarios a los que se sigue.
- Ver el rendimiento de otros jugadores.
- Modificar nombre de usuario, contraseña y email.
- En el juego, ver los recursos y vehículos disponibles.
- Creación de salas donde se disputen combates entre dos equipos de tres jugadores.
- Dentro de batalla, mover el tanque, girar la torreta y disparar.
- Almacenamiento de los resultados de la batalla en la base de datos.
- Eliminar y bloquear temporalmente a los jugadores.

1.3.2.- Objetivos personales

En el ámbito personal, el principal objetivo es adquirir conocimientos y práctica en la implementación de aplicaciones web y en el desarrollo de videojuegos ejecutados en navegador. Además, dado que el proyecto abarca back end y front end, otra finalidad es comenzar en la especialización como full stack web developer.

Algunas de las tecnologías usadas que sirven para cumplir estos objetivos (se verán con más detalle en el apartado 3, Hipótesis de trabajo) son:

- HTML
- CSS 3
- JavaScript
- MongoDB
- Express.js
- React Native
- Node.js
- Socket.io
- Phaser 3

1.4.- LÍMITES DEL PROYECTO

El proyecto utiliza herramientas que no se han visto en la carrera y que, por tanto, he tenido que aprender desde cero. En caso de comercializar la aplicación, el tiempo y dinero, además de personal (programadores, diseñadores, compositores para música y sonido…), necesarios para su desarrollo sobrepasa el alcance de un Trabajo Fin de Grado.

Un aspecto muy importante, que surge por la propia naturaleza online del proyecto, es la ciberseguridad. En consonancia con lo previamente dicho, debido a mi falta de conocimiento en el área y al tiempo necesario para incluir las medidas de seguridad pertinentes que exige una aplicación web, este apartado se ha dejado sin implementar.

Por último, el diseño artístico, ya sea la tipografía o los sprites, así como también el sonido, cumplen un propósito funcional y no estético. Es decir, en el supuesto de distribuir públicamente el juego, sería conveniente rediseñar el aspecto gráfico y crear sonidos y música más apropiados.

Biblioteca
UNIVERSITAS Miguel Hernández

Capítulo 2 Antecedentes y estado de la cuestión

2.1.- SITUACIÓN ACTUAL DE LOS VIDEOJUEGOS DE NAVEGADOR

Dado que el proyecto, además de tratarse de un juego de navegador, comparte muchas características con los videojuegos arcade o "retro", en este apartado se explicará en qué consiste cada género y se hará un resumen de sus orígenes y de cómo han evolucionado hasta hoy en día.

En primer lugar, hay que entender qué es un videojuego. Según la web definiciones.de, un videojuego es una aplicación interactiva orientada al entretenimiento que, a través de ciertos mandos o controles, permite simular experiencias en la pantalla de un televisor, una computadora u otro dispositivo electrónico [9].

Los videojuegos de navegador son juegos que, independientemente de la jugabilidad, narrativa, diseño artístico y calidad gráfica, su principal propiedad es que se ejecutan a través de un navegador. Normalmente están codificados con lenguajes como JavaScript, HTML5 o PHP [10].

Por otro lado, entendiendo los videojuegos arcade como un género, este tipo de juegos se caracterizan por tener un diseño sencillo, controles simples e intuitivos, niveles cortos que aumentan rápidamente en dificultad y se centran en la jugabilidad, dejando de lado la narrativa [11].

2.1.1- Historia de los videojuegos de navegador

En la década de los noventa, debido a la baja velocidad de Internet y, por tanto, los largos tiempos de carga de las webs, un grupo de ingenieros de FutureWave Software decide modificar un programa diseñado para dibujar en tablets, convirtiéndolo en una herramienta para animar gráficos vectoriales que, además, permitía la transmisión de dichas animaciones a través de las líneas telefónicas [12]. Este software se denominó originalmente FutureSplash Animator, aunque más adelante pasaría a llamarse Macromedia Flash cuando, en diciembre de 1996 Macromedia adquiere FutureWave. Posteriormente, el 18 de abril de 2005, Adobe absorbe Macromedia, volviendo a cambiar de nombre, esta vez a Adobe Flash Player [13].

Esta tecnología comenzó a hacerse popular en la industria del videojuego a partir del año 2000, cuando Macromedia publicó la versión 5 de Flash Player. Esta actualización trajo consigo un lenguaje de scripting llamado Action Scripting. Los primeros juegos para navegador realizados con este lenguaje eran copias de los juegos arcade clásicos como Pac-Man o Tetris. Gracias a la sencillez de Action Scripting y a la versatilidad y libertad creativa que proporcionaba, muchos desarrolladores independientes comenzaron a crear juegos con Flash.

En 2001 aparecieron páginas web como Miniclip, Kongregate o Newgrounds, donde cualquier desarrollador podía publicar sus videojuegos. Estas plataformas contribuyeron a la popularidad de los juegos en Internet para navegador. Se consiguió que hubiera una gran cantidad de videojuegos de cada género. El año 2007 fue el auge de los juegos diseñados con Flash.

Poco después, los juegos multijugador en línea como World of Warcraft se hacían más conocidos y las personas empezaban a dedicarle más tiempo a las redes sociales, dejando este tipo de videojuegos de lado. Con algunas excepciones como FarmVille, Angry Birds o Happy Wheels, que llegaron a tener decenas de millones de jugadores, los videojuegos Flash empezaron a perder popularidad [14].

En 2017 Adobe anunció que abandonaría el desarrollo de Flash Player debido a que, además de contar con problemas de ciberseguridad, algunos estándares abiertos que ofrecía el mercado habían evolucionado lo suficiente y eran alternativas viables. El 31 de diciembre de 2020 Adobe cesó definitivamente el soporte de Flash y los navegadores dejaron de incluirlo por defecto [15].

Hoy en día los juegos para navegador han vuelto de la mano de tecnologías como HTML5, JavaScript y WebGL. Estas herramientas permiten crear videojuegos de rendimiento y calidad gráfica similares a los de escritorio. Además, siguen siendo relativamente sencillos de crear y, por tanto, hay gran variedad de juegos. Pese a todas las mejoras, el componente social es el que más ha vuelto a popularizar los videojuegos para navegador. Ya sea a través de las redes sociales o las plataformas de streaming y gracias a la fácil distribución de los juegos (sólo se necesita un link y, como mucho, registrarse en la web), las personas utilizan estos medios para relacionarse y entretenerse con amigos. Ejemplos de este nuevo "boom" son Everwing, que cuenta con 400 millones de usuarios en Facebook [16] o Geoguessr, que tiene 30 millones de jugadores[17].

2.1.2.- Historia de los videojuegos arcade

Los primeros videojuegos de los que hay constancia son el OXO, un tres en raya electrónico creado por Alexander S. Douglas en 1952, y el Tennis for Two, que simulaba una pista de tenis y una pelota cruzando de lado a lado. Este último juego fue creado por el físico William Higginbotham en 1958 a partir de un osciloscopio con la intención de entretener a los visitantes del laboratorio Brookhaven de Nueva York.

Pocos años más tarde, en 1962, un equipo de informáticos del MIT crearon el Spacewar! tras más de 200 horas de trabajo. En este juego se enfrentaban dos personas, cada una al control de una nave, en torno al pozo gravitatorio de una estrella. En el año 1971, con una jugabilidad muy parecida al videojuego anterior, el Galaxy Game se considera como el primer juego comercial ya que jugar una partida costaba 10 céntimos. Al principio, colocada en la Universidad de Standford, sólo se diseñó una máquina que costó 20.000 dólares. Posteriormente se diseñaron más y se adaptó para varias videoconsolas [18].

Ese mismo año, viendo una oportunidad de negocio tras conocer Spacewar!, Nolan Bushnell y Ted Dabney crearon Computer Space, la primera máquina arcade fabricada a gran escala. Con ella, consiguieron suficiente dinero para fundar su propia compañía, Atari. Siguiendo el modelo de pagar por cada partida, Bushnell crea PONG en 1972. La máquina tuvo un éxito enorme gracias a que tenía un manejo sencillo y todo el mundo podía jugar y divertirse desde la primera partida. El PONG popularizó los videojuegos y asentó a las máquinas arcade como un modelo de negocio rentable.

La época dorada de las máquinas arcade comenzó en 1978 con Space Invaders. A diferencia del resto de juegos, este incorporaba un aumento progresivo de la dificultad, un sistema de puntuación y una lista con los jugadores que más puntos habían conseguido. Estos factores, que incentivaban a los participantes para superarse tanto a ellos como a los demás jugadores y, por tanto, a jugar más, se convirtieron en un estándar para los arcades [19]. Los años ochenta trajeron consigo muchos de los juegos arcade más famosos: Pac-Man, Donkey Kong, Tetris, Battle City, Dragon's Lair, Commando, Ghost 'n Goblins, entre otros.

A medida que avanzaba la década de los noventa, los arcades fueron perdiendo jugadores ante la creciente popularidad de las videoconsolas, que ofrecían más potencia sin salir de casa, de los cibercafés y de la conexión a Internet. Pese a que aún salieron arcades muy populares como Street Fighter, Mortal Kombat, Virtua Tennis o Dance Dance Revolution, las máquinas arcade fueron desapareciendo [20].

Hoy en día se pueden rescatar muchos de los juegos arcade a través de diferentes programas. MAME (Multiple Arcade Machine Emulator) es un framework orientado a la emulación distribuido bajo la licencia de software libre GNU GPL 2.0. Dado que las máquinas arcade se diseñaron con un hardware específico y que, tarde o temprano dejarán de funcionar, este emulador nació para preservar los videojuegos de la época y que no se perdieran con el tiempo. Para evitar problemas de copyright, MAME se centra en emular el funcionamiento de las arcade y videoconsolas "vintage", pero no en los juegos en sí, siendo necesaria una copia original para poder jugar [21].

Por último, además de emuladores como MAME, existen sistemas operativos como RetroPie, se trata de una distribución basada en linux y pensada para ejecutarse en los ordenadores en miniatura como Raspberry, centrados en la ejecución de juegos de máquinas arcade, videoconsolas y los clásicos de ordenador. Es también software libre y está licenciado bajo la GNU GPL [22].

2.2.- HERRAMIENTAS DISPONIBLES EN EL MERCADO

En el desarrollo de videojuegos intervienen múltiples disciplinas (programación, diseño gráfico, diseño de niveles, de sonido, de interfaces, etc.) y, por tanto, se usan una gran variedad de herramientas. A diferencia de un juego de escritorio, dado que este proyecto es, además, una aplicación web, es necesario el uso de tecnologías para el desarrollo del front y back end.

2.2.1.- Editores de código

2.2.1.1.- Visual Studio Code

Visual Studio Code es un editor de código gratuito distribuido por Microsoft, aunque dispone de una versión open source llamada Code – OSS bajo la licencia MIT [23]. Tiene soporte para muchos lenguajes de programación como por ejemplo JavaScript, Python, Java, C y sus versiones, HTML, CSS y PHP, entre otros. Además, dispone de plugins oficiales y desarrollados por la comunidad que permiten conectar la aplicación con Github o utilizar Jupyter notebook con Python [24] (entre otras funciones).

El editor tiene un menú lateral donde se pueden abrir directorios para navegar a través de las carpetas y archivos de un proyecto. También hay una sección donde conectarse con Git y hacer control de versiones o colaborar con otros usuarios en el desarrollo. Otra opción disponible es la de depurar y ejecutar el código siguiendo los parámetros de un archivo JSON. Por último, Visual Studio Code tiene la capacidad de abrir el terminal Powershell o la consola de comandos.

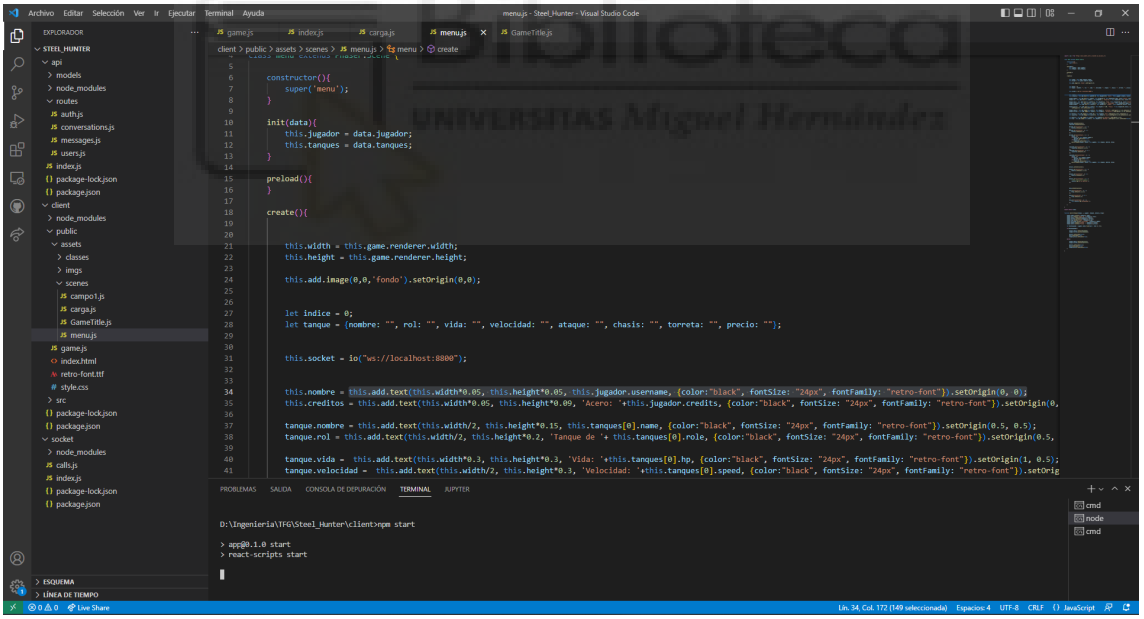

Figura 2.1.- Visual Studio Code

2.2.1.2.- Notepad++

El editor Notepad++ es gratuito, de código abierto y está licenciado bajo la GNU General Public License. Está basado en Scintilla, un componente de edición de código y está escrito en C++. Mediante la utilización de la API de Win32 y la librería estándar de C++

(STL) consigue una alta velocidad de ejecución, y además, ocupa poco espacio en el disco duro. Está disponible para sistemas operativos basados en ARM [25].

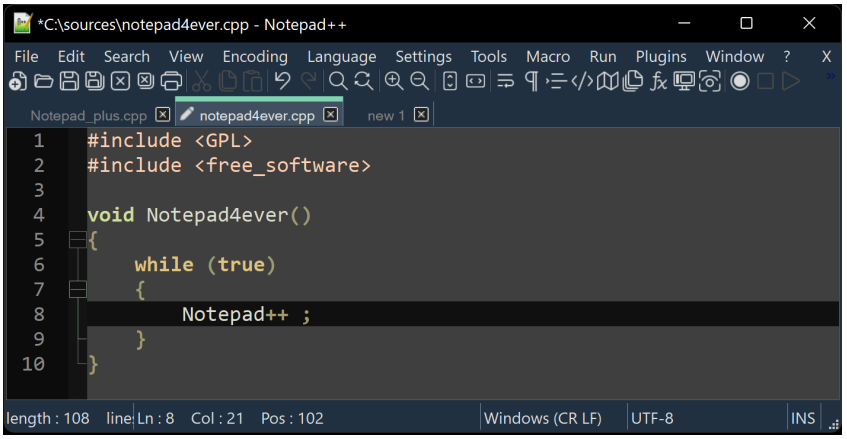

Figura 2.2.- Notepad++

2.2.1.3.- Sublime Text 4

Sublime Text es un editor de texto y de código. Nació como una extensión de Vim, un editor presente en los sistemas UNIX. Aunque se puede descargar gratuitamente, no es software libre ni código abierto. Dispone de una versión de evaluación con todas las características y sin tiempo límite, pero para su uso continuado, requiere pagar por la licencia. Es compatible con Windows, Linux, macOS, con sistemas basados en ARM64 y con dispositivos como Raspberry Pi [26].

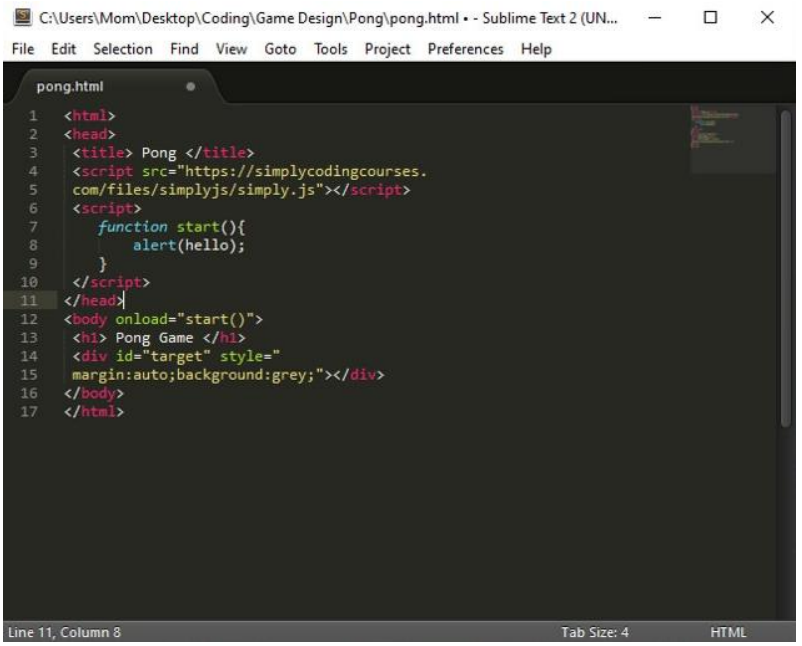

Figura 2.3.- Sublime Text

2.2.1.4.- Brackets

Es un editor de código gratuito enfocado al desarrollo web. Está escrito con los mismos lenguajes de programación que soporta: JavaScript, HTML y CSS. Tiene una interfaz sencilla y dispone de un apartado donde ver los cambios realizados al código en tiempo real. Es de código abierto y, en su comunidad de Github, señalan como punto fuerte la colaboración entre los usuarios para, según dicen, crear "*el mejor editor de código para desarrollo web*" [27].

Figura 2.4.- Brackets

2.2.2.- Herramientas para el desarrollo de videojuegos

2.2.2.1.- Phaser 3

Phaser 3 es un framework de HTML5 que se programa con JavaScript y está centrado en la creación de videojuegos para navegador y dispositivos móviles. Se distribuye bajo la licencia MIT y permite la copia, modificación y redistribución del software siempre y cuando se haga bajo la misma licencia y condiciones. Utiliza WebGL o canvas (si el primero no estuviera disponible) para el renderizado de gráficos. También gestiona la lógica del juego, los eventos (clics, pulsación de teclas, etc.) y dispone de un motor propio de físicas que permite la interacción entre los distintos elementos del videojuego [28].

Figura 2.5.- Logotipo de Phaser

```
let batalla = this.add.image(this.width/2, this.height*0.8,'batalla').setOrigin(0.5,0.5);
let bloq = this.add.image(this.width/2, this.height*0.8,'desbloq').setOrigin(0.5,0.5).setVisible(false);

derecha.setInteractive();
izquierda.setInteractive();

derecha.on("pointerover", () => {
    derecha.setScale(2.2,2.2);
});
derecha.on("pointerout", () => {
    derecha.setScale(2,2);
});

derecha.on("pointerdown", () => {
    indice +=1;
    if(indice === this.tanques.length-1)
        derecha.setVisible(false);
    if(indice > 0)
        izquierda.setVisible(true);
    mostrarTanque(tanque, indice, this.jugador, this.tanques, batalla, bloq);
});
```

Figura 2.6.- Código de un juego hecho en Phaser 3

Figura 2.7.- Ejecución de un juego hecho con Phaser 3

2.2.2.2.- Unity

Unity es una plataforma de desarrollo de videojuegos en 2D, 3D y realidad virtual creada por la empresa Unity Technologies. Dispone de motores de renderizado de imágenes y

animaciones, de físicas en 2D y 3D y, además, tiene herramientas para el networking de los juegos multijugador y sistemas de navegación “NavMesh” para la inteligencia artificial.

Aparte de facilitar el desarrollo, Unity cuenta con sistemas para monetizar los juegos, realizar analíticas sobre el comportamiento de los jugadores y colaborar con otros desarrolladores en un mismo proyecto mientras se lleva un control de versiones. Para fomentar el uso de esta plataforma, en la web oficial hay planes para que los estudiantes de instituciones certificadas puedan aprender sobre el desarrollo, ya sea técnico o artístico [29] [30].

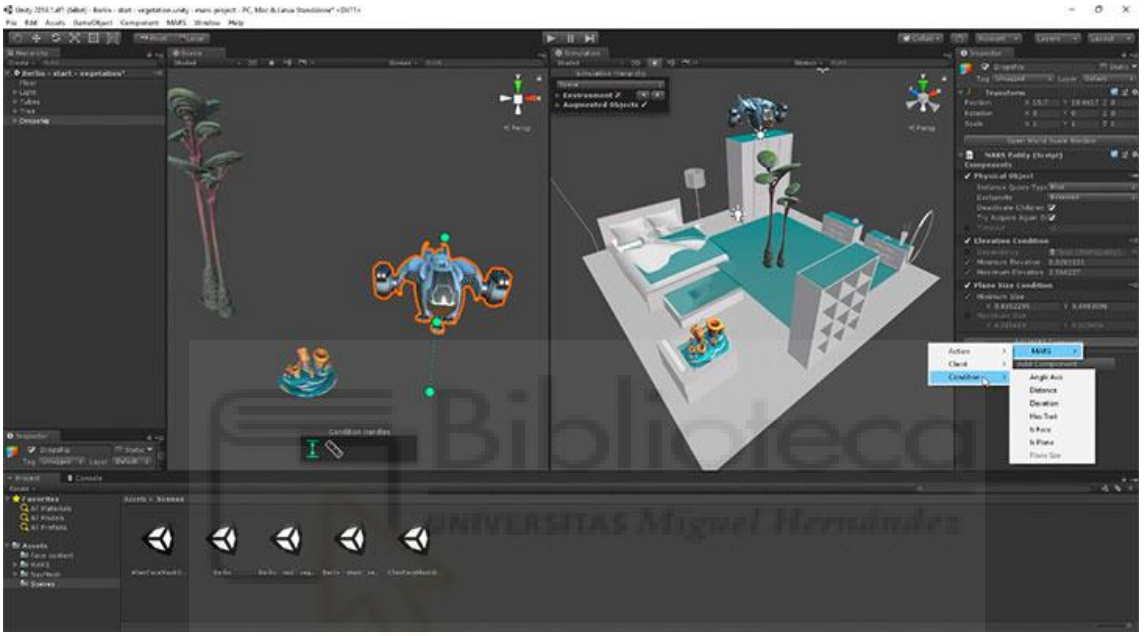

Figura 2.8.- Creación de un escenario 3D con Unity

2.2.2.3.- Unreal Engine

Desarrollado por la compañía Epic Games, Unreal Engine es una suite de herramientas para el desarrollo de videojuegos y la creación de escenarios y animaciones. Es de código abierto y su licencia permite el uso de todas las funcionalidades, pero, en caso de monetizar la aplicación desarrollada, a partir de que esta supere el millón de dólares en ingresos, Epic Games recibirá el 5% de las ganancias [31].

Unreal Engine 5 tiene herramientas para el desarrollo de videojuegos para PC, dispositivos móviles, ya sea Android o iOS, para las videoconsolas de Sony, Microsoft y Nintendo y, además, se pueden crear juegos para navegadores web utilizando la técnica del Pixel Streaming. Esta funcionalidad renderiza los *frames* en el servidor y se los envía al cliente, quien podrá interactuar con ellos [32] [33].

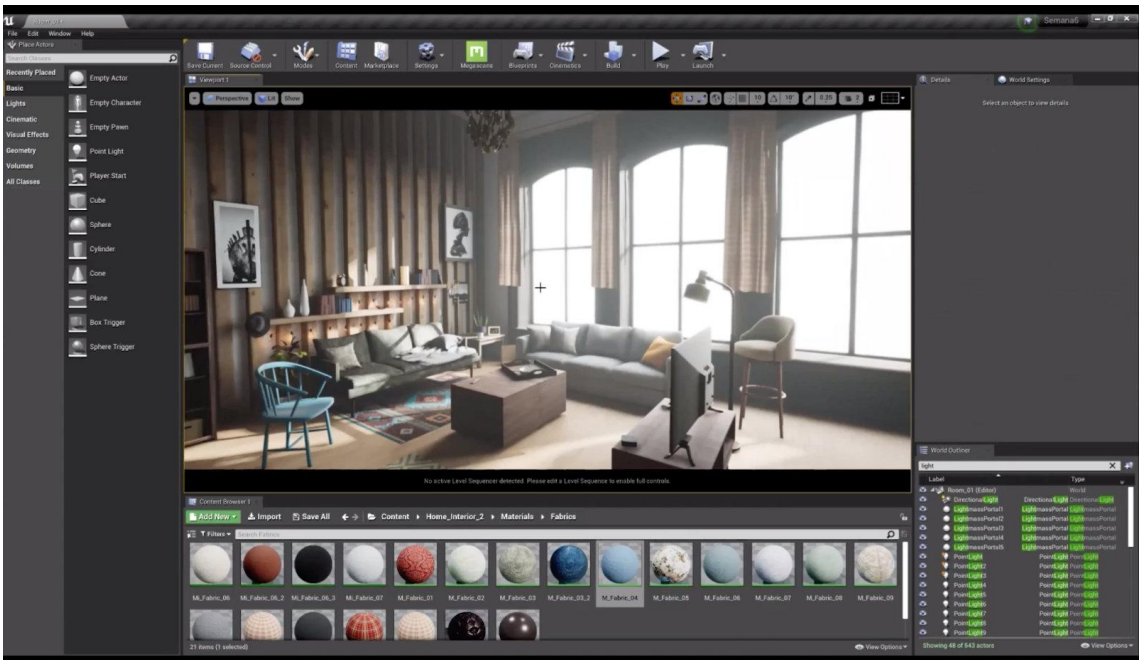

Figura 2.9.- Creación de un escenario en Unreal Engine

Figura 2.10.- Modelado 3D de un rostro con Unreal Engine 4

2.2.3.- Frameworks para la creación de interfaces

2.2.3.1.- React.js

React.js es una librería de código abierto distribuida bajo la licencia MIT y desarrollada por Facebook, Inc. Se programa con JavaScript (aunque la extensión de los archivos cambia a ".jsx") y permite, a partir de un sólo código, hacer interfaces para navegadores

usando React.js y para dispositivos móviles utilizando React Native. Al estar basado en componentes, permite modularizar mucho la interfaz y crear jerarquías para organizar los elementos de la aplicación. Por último, React determina la vista a mostrar según el estado actual, haciendo que sólo se tenga que renderizar lo pertinente en cada momento [34].

```
      try{
        await axios.post("auth/register", user);
      }catch(err){
        console.log(err);
      }

      loginCall(
        { email: email.current.value, password: password.current.value },
        dispatch
      );

    }

  }
};

return (
    <>
      <Topbar/>
      <div className="signup">

        <form className="signup-box" onSubmit={handleClick}>

          <h3 className="signup-logo">Steel Hunters</h3>

          <input
            placeholder="Usuario"
            className="signup-input"
            minLength="3"
            type="text"
            required
            ref={username}
          />
```

Figura 2.11.- Código de una vista en React.js

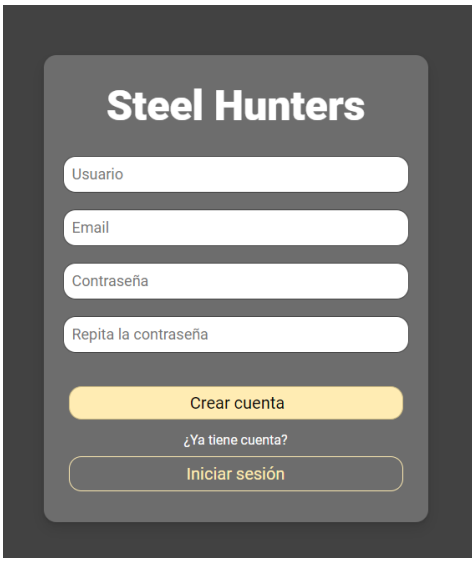

Figura 2.12.- Interfaz creada con React.js

2.2.3.2.- Vue

Vue es un framework para JavaScript de código abierto, distribuido bajo la licencia MIT, orientado a la creación de interfaces de usuario. Igual que React, al estar basado en componentes, se puede modularizar y jerarquizar la aplicación, y transmitir datos entre las diferentes partes de la interfaz. Además, como extiende la sintaxis del HTML estándar, permite que la vista cambie según el estado de JavaScript.

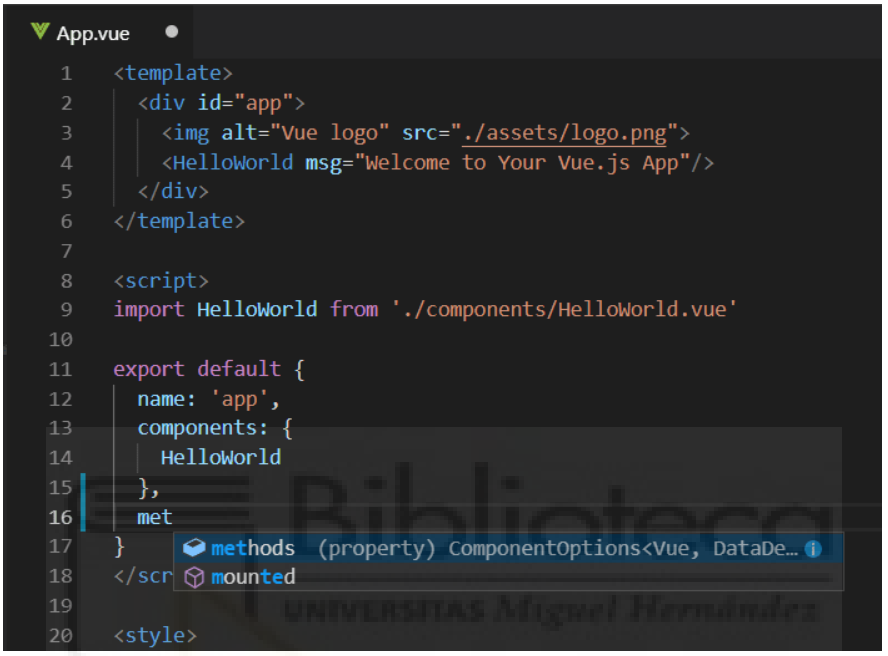

Figura 2.13.- Código de una interfaz hecha con Vue

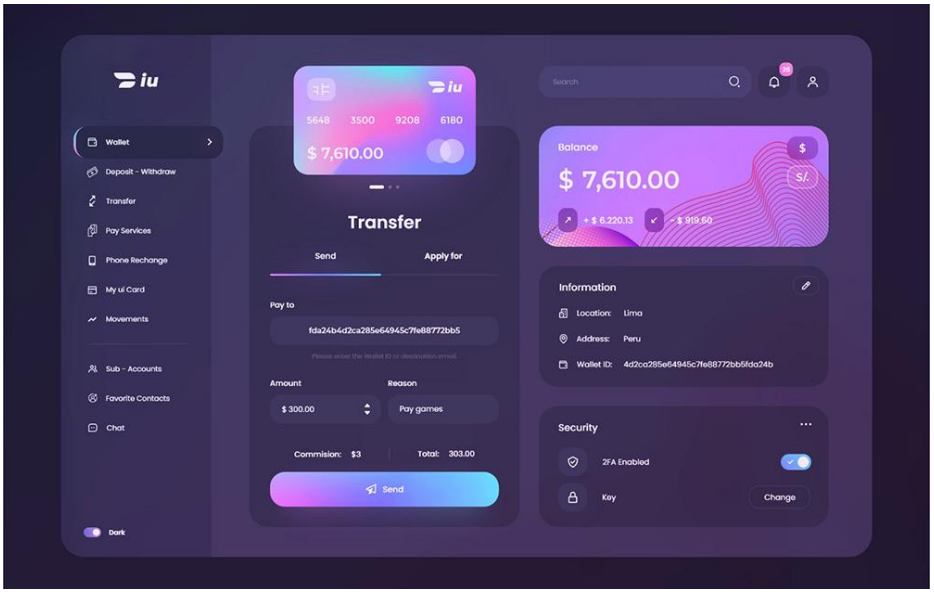

Figura 2.14.- Interfaz desarrollada con Vue

2.2.3.3.- Angular

Angular es un framework gratuito desarrollado por Google bajo la licencia MIT. Utiliza TypeScript, un superconjunto de JavaScript mantenido por Microsoft. Igual que en los dos framework anteriores, Angular es multiplataforma, ya que permite diseñar, a partir de un código, aplicaciones web para dispositivos móviles y de escritorio. También está diseñado a base de componentes con los que modularizar la interfaz. Para mejorar el rendimiento, el renderizado de la interfaz se realiza en el lado del servidor [36].

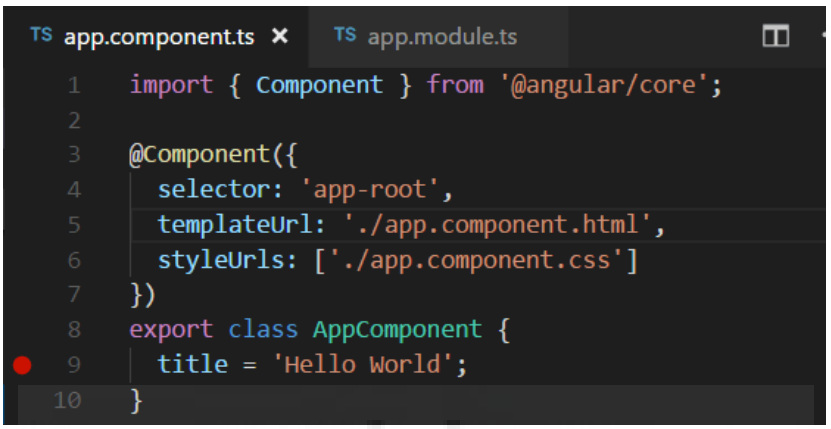

Figura 2.15.- Código de una aplicación hecha con Angular

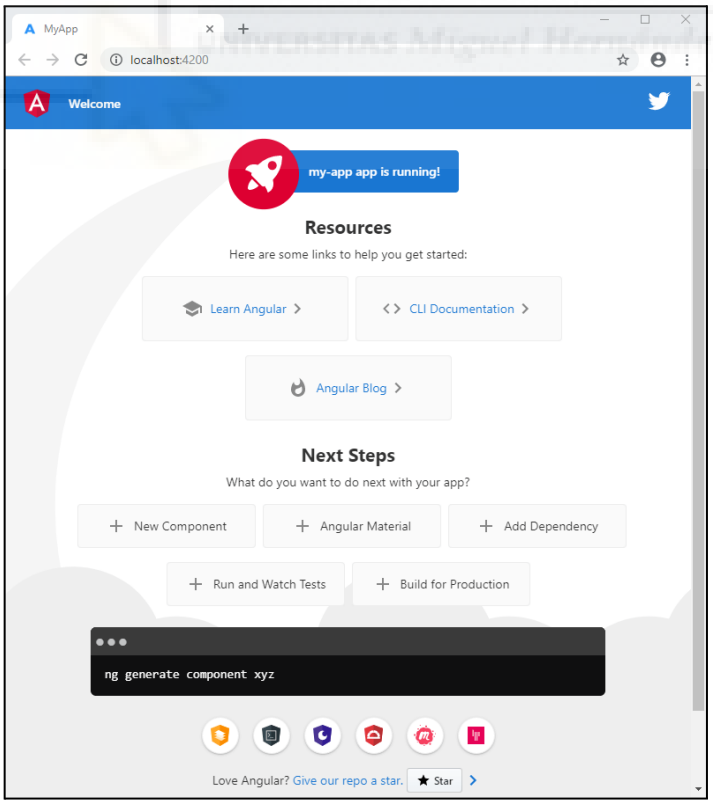

Figura 2.16.- Aplicación desarrollada con Angular

2.2.4.- Servidores

2.2.4.1.- Node.js

Node.js no es un servidor propiamente dicho, es un entorno de ejecución de JavaScript orientado a eventos asíncronos, pero dispone de la funcionalidad para trabajar como servidor, por lo que también se emplea como tal en muchos proyectos. Es gratuito y de código abierto. Node.js se ejecuta en un sólo hilo y, para evitar bloqueos causados por las peticiones de entrada/salida, cede el control de las operaciones al kernel y, dado que la mayoría tienen múltiples hilos, pueden ejecutarse varias funciones simultáneamente. Cuando el kernel completa una acción, Node.js añade la función "callback" al bucle de eventos donde se ejecutará llegado su turno [37] [38].

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Figura 2.17.- Código JavaScript para configurar un servidor en Node.js

2.2.4.2.- Apache

Apache es un servidor HTTP de código abierto diseñado para sistemas UNIX y Windows distribuido bajo la licencia *Apache License 2.0*. Dispone de módulos que extienden las funcionalidades básicas del servidor. Por ejemplo, sistemas de autentificación, soporte para lenguajes como Perl, Python y PHP, y la capacidad de crear varios hosts virtuales en un mismo ordenador [39].

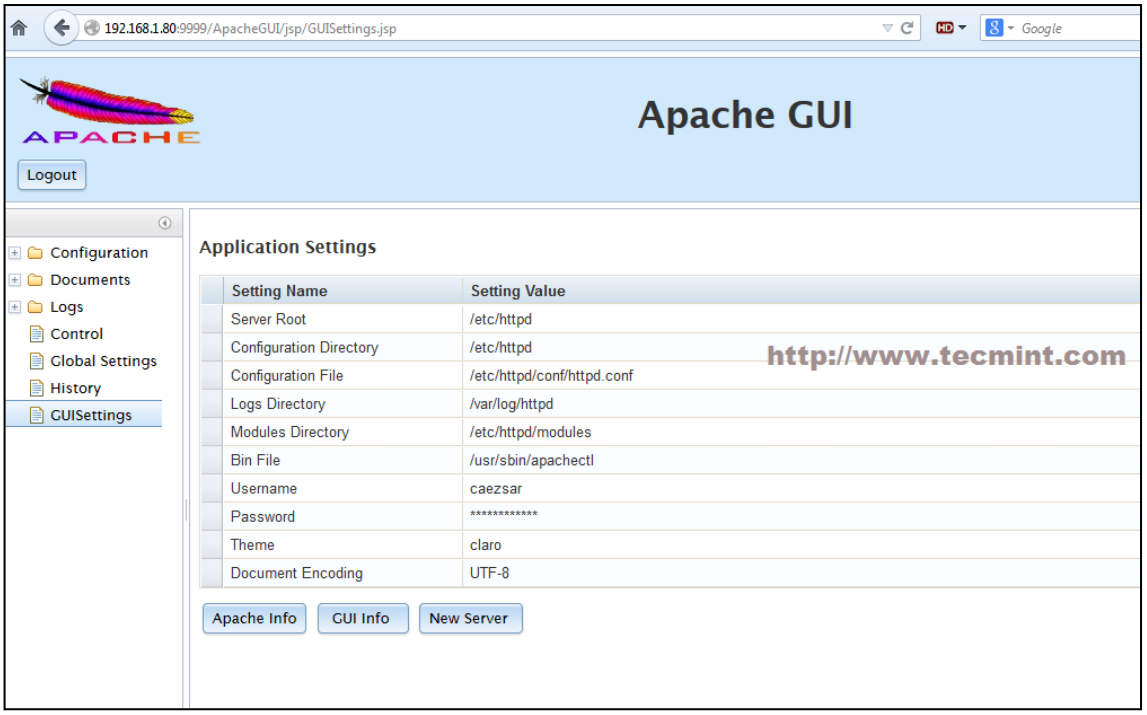

Figura 2.18.- Ventana de configuración del servidor Apache

2.2.5.- Bases de datos

2.2.5.1.- MongoDB

MongoDB es un programa para la gestión de bases de datos NoSQL (no relacional). Es gratuito y, dependiendo de la versión, está publicado bajo la AGPL o la Licencia Pública del Lado del Servidor (SSPL). Almacena los datos en documentos flexibles con estructura similar a JSON. Dispone de una versión que almacena los datos en sus servidores (diferentes versiones, una gratuita y las demás de pago) y de otra administrada localmente.

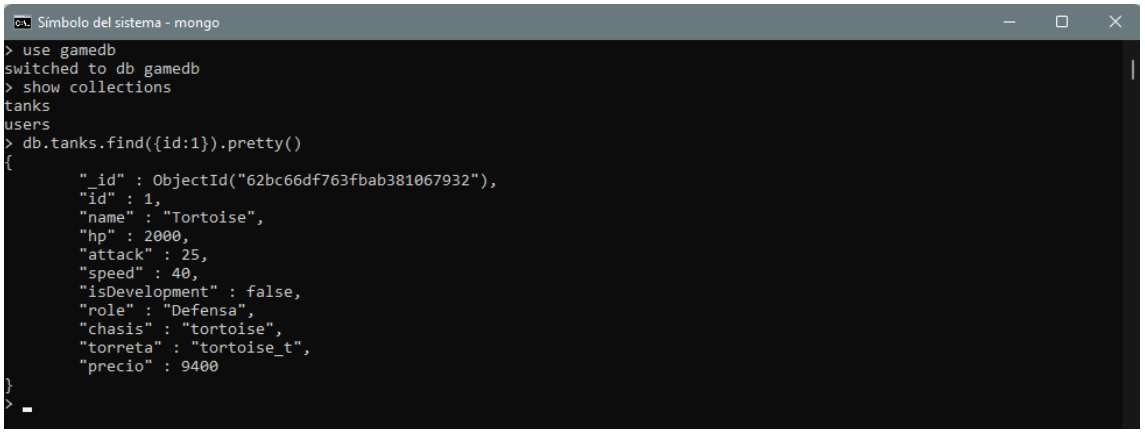

Figura 2.19.- Búsqueda de un registro con MongoDB local

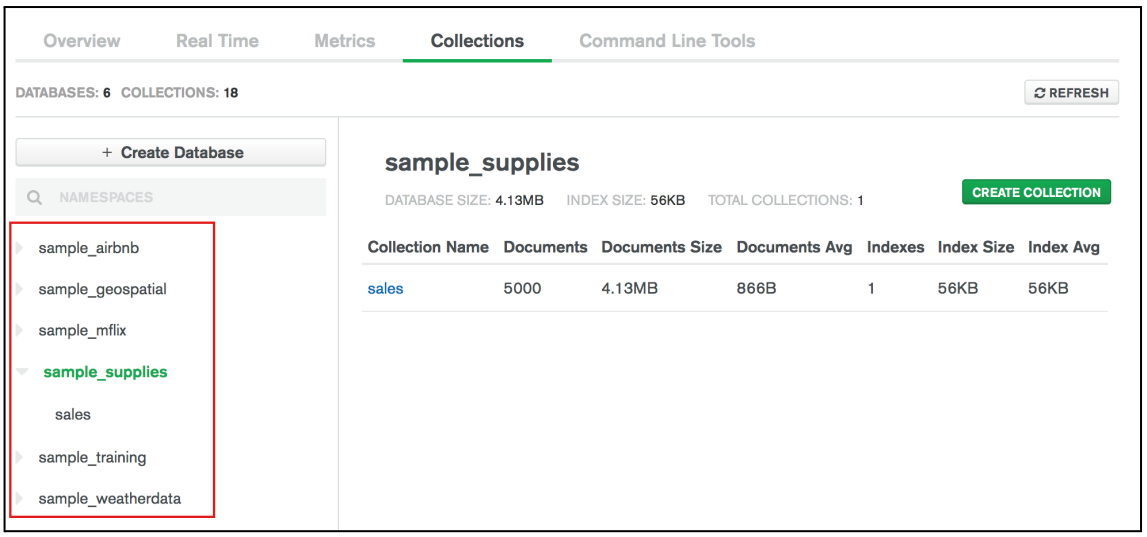

Figura 2.20.- MongoDB Atlas

2.2.5.2.- MySQL Server

MySQL es un gestor de bases de datos relacionales de código abierto, está desarrollado en C y C++. Tras la compra de Oracle, además de la licencia GNU GPL v2, hay una versión con licencia propietaria. Sigue el modelo cliente-servidor, donde el cliente realiza solicitudes mediante instrucciones SQL (Structured Query Language), el servidor realiza la acción pertinente y devuelve el resultado [40].

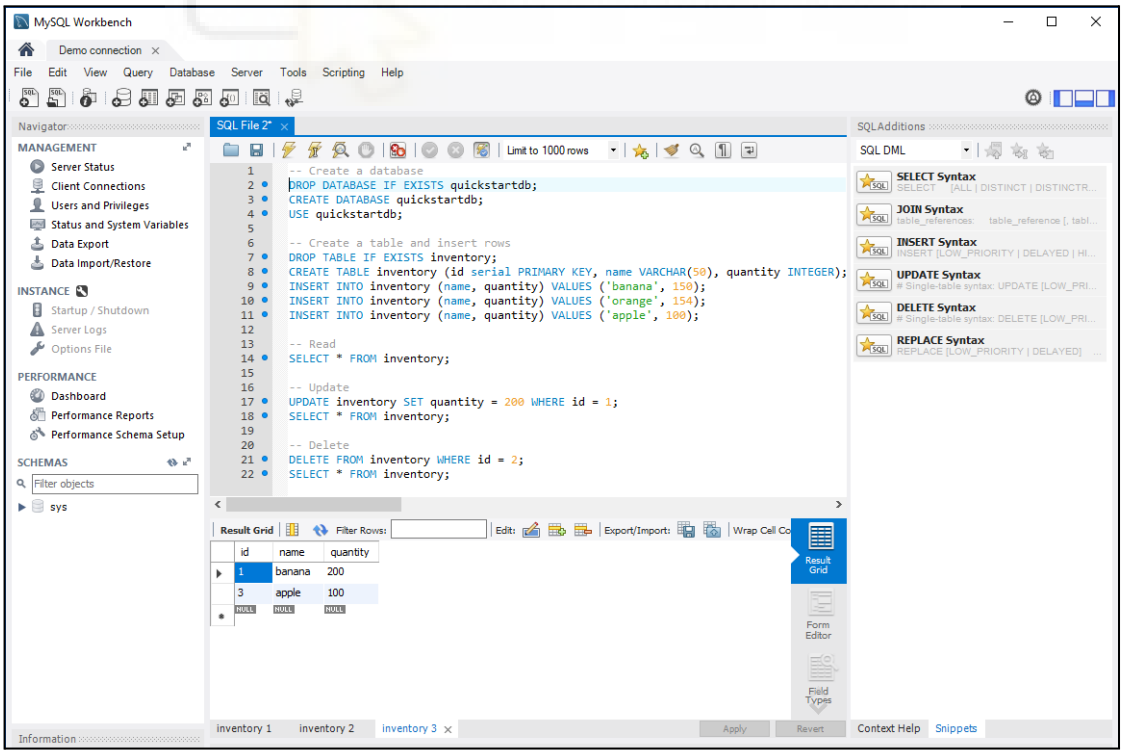

Figura 2.21.- MySQL

2.2.5.3.- MariaDB

MariaDB es un sistema gestor de bases de datos relacionales. Creado por los desarrolladores originales de MySQL (antes de que lo comprara Oracle), es de código abierto y está publicado bajo la licencia GPL v2. El enfoque de los desarrolladores es el rendimiento, la estabilidad y los principios del software libre [41].

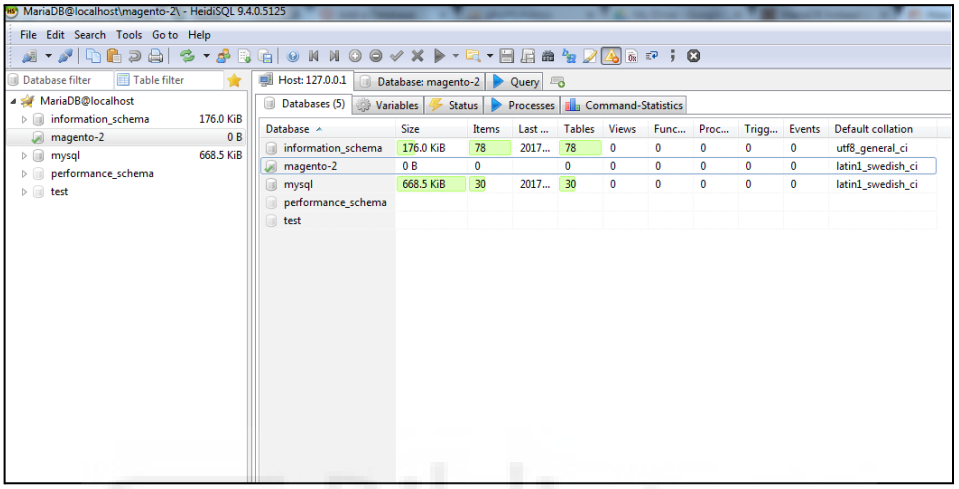

Figura 2.22.- MariaDB

2.2.6.- Herramientas de diseño gráfico

2.2.6.1.- Adobe Photoshop

Photoshop es uno de los editores gráficos más conocidos y utilizados. Adobe, la compañía que lo desarrolla, ofrece una prueba gratuita por tiempo limitado (7 días), pero para usarlo a largo plazo es necesario suscribirse a un plan de Creative Cloud. Photoshop tiene herramientas para diseñar, dibujar y modificar imágenes y animaciones. [42].

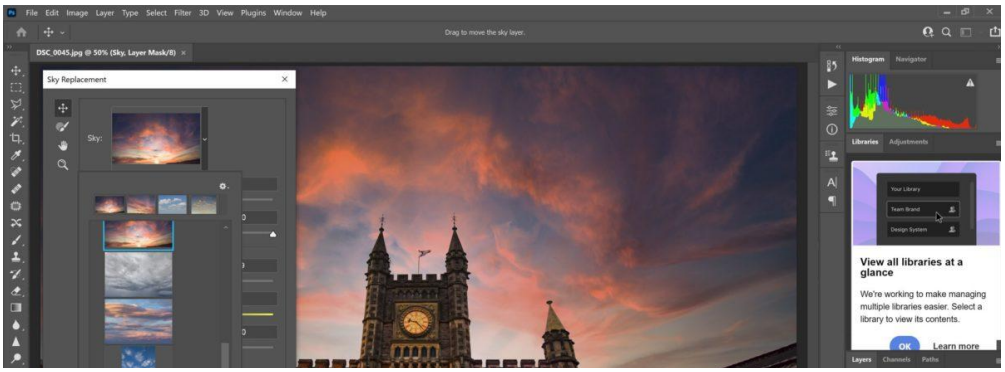

Figura 2.23.- Adobe Photoshop

2.2.6.2.- GIMP

Es un editor de imágenes gratuito y de código abierto. El nombre es un acrónimo de GNU Image Manipulation Program. Está disponible para Windows, macOS y Linux. Como Photoshop, GIMP tiene herramientas para modificar imágenes, para diseñar elementos gráficos y crear dibujos. Existen scripts y plugins (programados con Python, Perl, Scheme o C, entre otros) creados por la comunidad que añaden funcionalidades al programa base [43].

Figura 2.24.- GIMP

2.2.6.3.- Inkscape

Inkscape es un editor de gráficos vectoriales gratuito y de código abierto para Linux, Windows y macOS X. Está orientado a la creación de ilustraciones técnicas y artísticas como dibujos, logos, tipografías y diagramas. Aunque soporta múltiples extensiones para exportar e importar, utiliza el formato SVG como predeterminado. A diferencia de los editores de gráficos rasterizados, Inkscape permite renderizar imágenes sin límite de resolución. Cualquier usuario puede modificar o añadir funcionalidades al programa y compartirlas con el resto de la comunidad a través de add-ons [44].

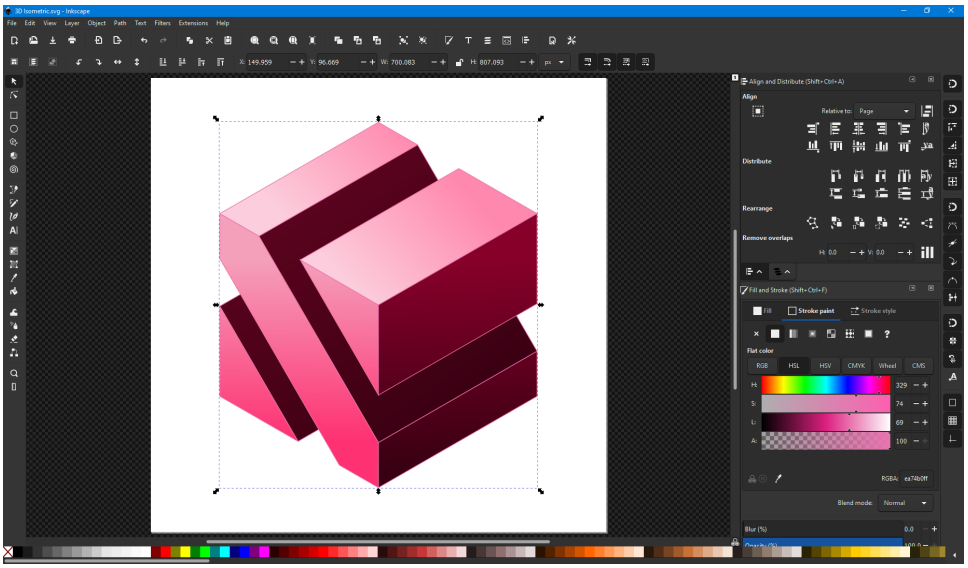

Figura 2.25.- Inkscape

2.2.7.- Programas de edición de audio

2.2.7.1.- Audacity

Audacity es un editor y grabador de audio gratuito y de código abierto. Está disponible para Windows, macOS y Linux. Tiene soporte para sonido de 16, 24 y 32 bits y para exportar e importar múltiples formatos de audio, con y sin pérdidas. Además, permite cortar, pegar, aplicar efectos a las pistas y hacer análisis y selección de frecuencias [45].

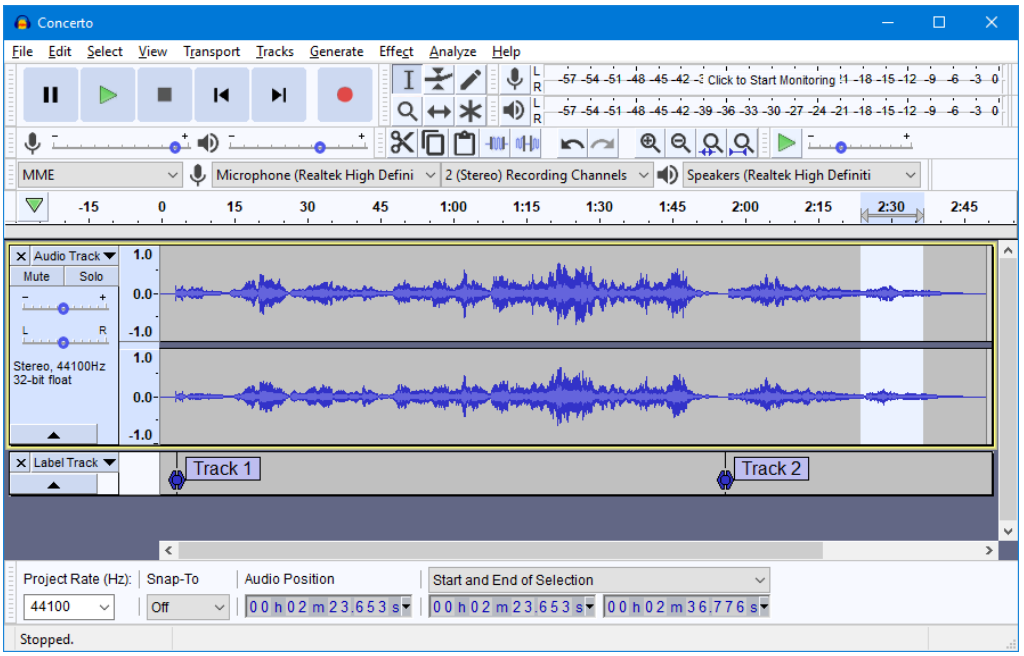

Figura 2.26.- Audacity

2.2.7.2.- Adobe Audition

Es un programa de la suite de Adobe enfocado a la edición y grabación de audio. Tiene herramientas para limpiar, restaurar y reparar audio, añadir efectos y para organizar las pistas. Audition está disponible sólo a través de la suscripción mensual al paquete Creative Cloud de Adobe, aunque se puede probar durante 7 días [46].

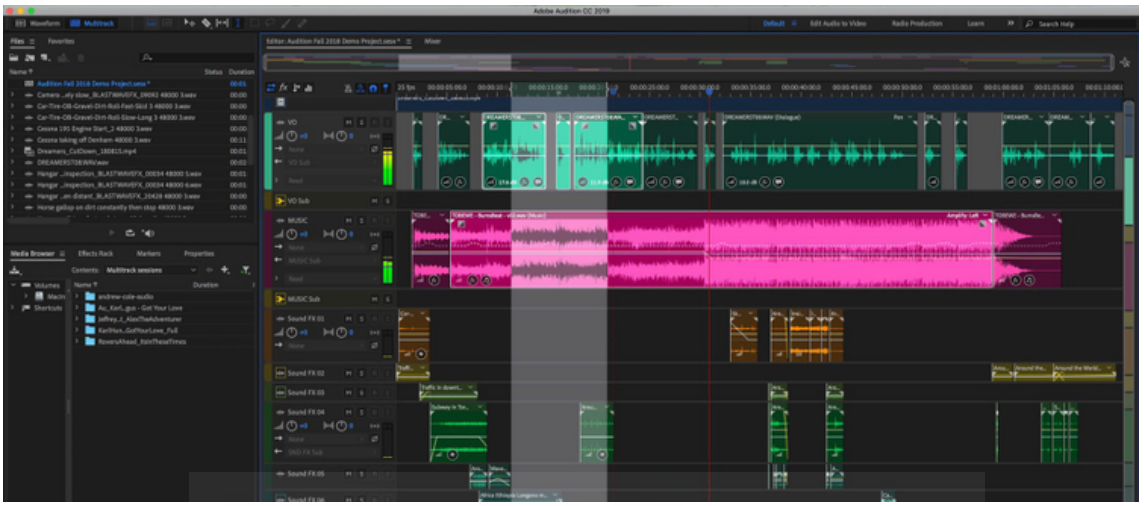

Figura 2.27.- Adobe Audition

2.2.8.- Resumen

En las siguientes tablas se muestra de forma resumida las principales ventajas y desventajas que se aprecian en las distintas herramientas y tecnologías que se han expuesto en los apartados anteriores.

Tabla 2.1.- Editores de código

Editor	Ventajas	Desventajas
Visual Studio Code	- Gratuito - Tienda de plugins - Versiones para Windows, macOS y Linux - Muchas funcionalidades - Desarrollado por Microsoft	- Envía información sobre el uso a Microsoft - Más lento que otros editores
Notepad++	- Gratuito - Software libre - Código abierto - Portable - Rápido - Ligero	- Sólo funciona en Windows
Sublime Text 4	- Funciona en Windows, macOS, Linux y dispositivos con procesadores ARM64	- Es de pago - Ni es libre ni de código abierto
Brackets	- Gratuito - Código abierto - Visualizador de los cambios en tiempo real	- Limitado a programación web (HTML, CSS y JavaScript)

Tabla 2.2.- Herramientas para el desarrollo de videojuegos

Software	Ventajas	Desventajas
Phaser 3	- Gratis - Libre y de código abierto - Tiene motor gráfico y de físicas en 2D - Comunidad grande y activa - Dispone de muchos tutoriales - Sencillo de utilizar - Se pueden crear juegos para navegador y para dispositivos móviles	- El soporte para juegos 3D es mínimo y a través de plugins - No sirve para crear aplicaciones de escritorio - Sólo soporta JavaScript - Limitado para proyectos muy grandes
Unity	- Herramientas para crear entornos 2D y 3D - Fácil de utilizar - Soporta C# y JavaScript - Se pueden crear juegos de escritorio, navegador, dispositivos móviles y videoconsolas - Versión gratuita	- Utiliza muchos recursos, por lo que se necesita un ordenador potente
Unreal Engine	- Multiplataforma: PC, móvil, videoconsolas y navegador - Utiliza C++, un lenguaje muy conocido - Herramientas de calidad para 3D	- Curva de aprendizaje bastante alta - Más limitado en 2D que otras opciones

Biblioteca UNIVERSITAS Miguel Hernández

Tabla 2.3.- Frameworks para la creación de interfaces

Framework	Ventajas	Desventajas
React.js	- Gratis - Fácil de usar - Permite crear interfaces para páginas web y dispositivos móviles - Muy utilizado y, por tanto, muchos tutoriales y recursos	- Al ser tan popular y evolucionar tan rápido, la documentación oficial es limitada o está desactualizada.
Vue	- Gratis - Flexible y eficiente	- Menos popular - Documentación limitada
Angular	- Gratis - Documentación detallada	- Utiliza TypeScript, por lo que requiere un aprendizaje adicional

Tabla 2.4.- Servidores

Servidor	Ventajas	Desventajas
Node.js	- Gratis - Código abierto - Permite usar JavaScript en cliente y servidor - Funciona a base de eventos	- No soporta PHP
Apache	- Gratis - Código abierto - Parches de seguridad - Configuración sencilla - Soporta Wordpress	- Problemas de rendimiento en webs con mucho tráfico.

Tabla 2.5.- Bases de datos

Base de Datos	Ventajas	Desventajas
MongoDB	- Gratis - Altamente compatible con JavaScript - NoSQL - Muy flexible y escalable	- No es adecuada para proyectos en los que se necesite una base de datos relacional
MySQL	- Gratis - Buena seguridad - Fácil de instalar y configurar - Muy utilizado - Muy estable	- Documentación limitada
MariaDB	- Gratis - Eficiente	- Menos popular que MySQL

Tabla 2.6.- Herramientas de diseño gráfico

Software	Ventajas	Desventajas
Adobe Photoshop	- Es el editor más popular - Mucha documentación y tutoriales - Muchas funcionalidades - Soporta 3D - Windows y macOS	- Licencia propietaria - Alto coste económico - Requiere mucha práctica para aprovecharlo al máximo - No está disponible en Linux
GIMP	- Libre y de código abierto - Sencillo de utilizar - Soporta gráficos vectoriales - Funciona en Windows, macOS y Linux.	- Menos funcionalidades que Photoshop
Inkscape	- Libre y de código abierto - Sencillo de utilizar - Soporta gráficos vectoriales - Funciona en Windows, macOS y Linux.	- Herramientas limitadas para la edición de fotografías

Tabla 2.7.- Programas de edición de audio

Software	Ventajas	Desventajas
Audacity	- Gratuito - Fácil de usar - Muy útil para proyectos amateur - Funciona en Windows, macOS y Linux.	- Menos herramientas que otros editores de pago
Adobe Audition	- Buen soporte - Mucha documentación - Gran cantidad de funcionalidades - Versión para Windows y macOS	- Licencia propietaria - Alto coste económico - Difícil de masterizar - No está disponible en Linux

2.3.- VALORACIÓN

Los distintos tipos de herramientas que se han mostrado en este capítulo son las necesarias para la realización del proyecto. Dado que es un TFG y no un videojuego comercial, se ha priorizado el uso de programas libres. Es por eso que, a excepción de dos, todas las aplicaciones disponen de una versión gratuita..

Empezando por lo más básico, para la creación de un videojuego es necesario contar con un editor de código. De los cuatro propuestos, Visual Studio Code y Notepad++ cuentan con mayor popularidad y, por tanto, gran soporte y funcionalidad. Sublime Text 4, aunque

también es bastante versátil, requiere pagar una licencia para su uso prolongado y Brackets, pese a que es gratuito, es muy limitado.

Dado que el proyecto trata del desarrollo de un videojuego ejecutado en navegador, de los tres programas propuestos, Phaser 3 es el más apropiado. Unity y Unreal Engine son muy potentes y excelentes para la creación de juegos para PC y videoconsolas, pero están limitados para los juegos de navegador. Por el contrario, Phaser 3 está diseñado para ejecutarse en una web, pero carece de herramientas para los juegos de escritorio.

En cuanto a los frameworks orientados a la creación de interfaces, React, Vue y Angular son muy buenas opciones. Los tres son gratuitos, de código abierto y funcionan de forma similar. Pese a que React es el más usado, la elección sobre cuál usar es principalmente una cuestión de preferencias. En el proyecto he utilizado React pues me resulta más cómoda e intuitiva la forma en la que se se dividen las diferentes partes de la aplicación (interfaz, manejo de eventos, gestión del estado…).

Apache y Node.js son dos tipos de servidores muy utilizados y efectivos. Ambos son gratuitos y de código abierto. La principal diferencia es que Apache tiene soporte para lenguajes como PHP, Perl o Python y Node.js está diseñado para funcionar únicamente con JavaScript. Por tanto, la elección del servidor a usar dependerá de los lenguajes que se quieran utilizar. Dado que en este proyecto se utilizan frameworks que requieren JavaScript del lado del servidor, es necesario usar Node.js.

Los sistemas gestores de bases de datos propuestos son todos gratuitos. Igual que en el caso anterior, el factor decisivo es el tipo de proyecto que se vaya a desarrollar. Para aplicaciones que requieran una base de datos relacional se podría utilizar MySQL o MariaDB. Por otro lado, MongoDB es de tipo NoSQL y organiza la base de datos en colecciones y documentos. La sintaxis es muy similar a JSON, por lo que la conversión entre MongoDB y JavaScript es muy sencilla.

Al tratarse de un videojuego, se necesitan editores de imágenes para la creación de los sprites. Photoshop es el mejor programa para profesionales, pero tiene un coste elevado. En el caso de este proyecto, GIMP o Inkscape son perfectamente utilizables. Ambos son gratuitos y tienen herramientas suficientes para la creación de sprites.

Por último, en un videojuego es esencial el sonido. Para la grabación y edición de audio se han incluido dos opciones, Adobe Audition y Audacity. El primero forma parte de la suite creativa de Adobe, es muy potente, pero tiene un coste elevado. Por otro lado, Audacity, aunque es más limitado, es gratuito y tiene herramientas suficientes para la creación de los sonidos necesarios en un juego retro.

Biblioteca
UNIVERSITAS Miguel Hernández

Capítulo 3 Hipótesis de trabajo

El proyecto se puede dividir en dos partes diferentes: la aplicación web con funcionalidades de red social y el videojuego multijugador. Aunque comparten elementos como el entorno de desarrollo o los lenguajes, se necesitan herramientas específicas en cada parte. Dado que este es un Trabajo de Fin de Grado sin financiación, todas las tecnologías utilizadas son gratuitas y, en muchos casos, de código abierto.

3.1.- LENGUAJES DE PROGRAMACIÓN Y EDITOR DE CÓDIGO

Al tratarse de una aplicación web, los lenguajes utilizados son HTML, CSS y JavaScript. HTML (*Hypertext Markup Language*), es un lenguaje de marcado para estructurar el contenido de una página web. Su versión más actual, la 5, estandariza su funcionamiento y uso con otros lenguajes complementarios. HTML, por tanto, se centra en indicar la

estructura, jerarquía y tipos de elementos que componen una página web haciendo uso de etiquetas ("<p>" para párrafos de texto, "<img>" imágenes, etc.). Además, cada etiqueta dispone de ciertos atributos que permiten añadir identificadores o especificar el comportamiento del elemento [47].

CSS, del inglés *Cascading Style Sheets*, es un lenguaje de estilo utilizado para la presentación de documentos HTML. Mediante el tipo de etiqueta o las propiedades de HTML '*class*' o '*id*', se puede cambiar el estilo de un conjunto, de un elemento en particular o de ciertas etiquetas. Entre otros atributos, CSS permite modificar el color, la altura, anchura, la posición en el documento, la fuente o los márgenes [48].

El último lenguaje es JavaScript, está basado en prototipos, se ejecuta en un sólo hilo, y soporta varios paradigmas: orientado a objetos, imperativo y declarativo [49]. JavaScript permite crear elementos interactivos en los documentos HTML como galerías de imágenes, diseños dinámicos, juegos y animaciones 2D y 3D. Además, existen APIs (*Application Programming Interface*) que aumentan la funcionalidad básica del lenguaje y facilitan el manejo y la creación de datos, gráficos y audio [50].

De los cuatro editores presentados en el capítulo 2, se ha elegido Visual Studio Code. Pese a que tiene una versión de código abierto, se ha utilizado la distribución proporcionada por Microsoft. Además de soportar una gran cantidad de lenguajes de programación, dispone de plugins que aumentan sus funcionalidades [24]. Visual Studio Code también dispone de una herramienta llamada IntelliSense que ayuda a autocompletar código de manera inteligente, lo que agiliza la programación. Otra funcionalidad similar es la de los Snippets, una estructura básica de componentes o funciones, ya sean de JavaScript (en este TFG, pero hay para muchos lenguajes) o para los frameworks más populares. Por ejemplo, escribiendo "for", el editor escribiría un bucle con el que recorrer un array sencillo [51]. También se ha elegido este editor porque dispone de un menú lateral en el que navegar por la carpeta del proyecto. Además, permite ejecutar múltiples consolas de comando que, para el caso del presente trabajo, se usan para iniciar los servicios del servidor localhost.

3.2.- APLICACIÓN WEB

3.2.1.- Modelo-Vista-Controlador, SPA y RESTful API

La aplicación web está fundamentada en tres conceptos: modelo-vista-controlador (MVC), API de tipo REST (*Representational State Transfer*) y aplicaciones de una sola página (SPA). El primer concepto, MVC, es una arquitectura software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El

modelo contiene una representación de los datos, la lógica de negocio y los mecanismos de persistencia. Es el responsable de la capa de almacenamiento de datos y de notificar cualquier cambio en dichos datos. La vista contiene la interfaz de usuario y la información enviada al cliente, además, informa al controlador de las acciones del cliente. Por último, el controlador es un intermediario entre las otras dos capas. Dispone de unas reglas con las que gestionar los eventos de entrada (un clic, cambios en un campo de texto…) y transforma los datos para que ambas partes los entiendan [52].

Una API de tipo REST es una interfaz entre sistemas que usa HTTP para obtener datos y operar con ellos. Las características que definen el enfoque REST son [53][54]:

- Protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, sin necesidad de recordar ningún estado previo. Opcionalmente, se pueden almacenar las respuestas en caché para que, si se repite una petición en el futuro, los datos estén disponibles de forma inmediata.

- Operaciones fundamentales: las operaciones elementales de la especificación HTTP que debe proporcionar un sistema REST son POST, GET, PUT y DELETE.

- Interfaz uniforme: para estandarizar la transferencia de información es necesario que cada recurso utilice una URI única como identificador. Además, las respuestas deben contener la información necesaria para describir cómo procesarlas y manipular los datos recibidos.

- Sistema jerárquico en capas: los servidores que participan en la recuperación de la información solicitada están jerarquizados en capas invisibles para el cliente.

El último concepto es SPA (aplicaciones de una sola página). Este tipo de aplicaciones cargan todo su contenido al acceder a la web y cambia la información mostrada de manera dinámica, sin necesidad de recargar la página. Respecto a las URLs, pueden cambiar dependiendo de la vista que se esté mostrando. La principal ventaja de este tipo de diseño es que agilizan la navegación, haciéndola más fluida. Dado que todo el contenido se carga al principio, la transición entre secciones es prácticamente instantánea. Su principal inconveniente es la dificultad para hacer SEO (Search Engine Optimization), en comparación con las webs tradicionales [55].

3.2.2.- MERN Stack

Cada vez es más habitual trabajar utilizando frameworks, ya que acelera el flujo de trabajo y permite aplicar las mejores prácticas de desarrollo de manera sencilla. En la actualidad, uno de los conjuntos o *stacks* más utilizados es el conocido como MERN Stack. Se utiliza

para el desarrollo de aplicaciones web y sus siglas corresponden a las iniciales de sus componentes: MongoDB, Express.js, React.js y Node.js [56].

MongoDB es un sistema gestor de bases de datos NoSQL orientado a documentos. Es decir, en vez de tablas, tiene colecciones en las que guarda documentos, no registros. Los datos se guardan mediante pares clave-valor, creando una estructura muy similar a los objetos JSON. Su principal diferencia respecto a las bases de datos relacionales es que en MongoDB no es necesario seguir ningún esquema, los documentos de una colección pueden tener estructuras diferentes. Otra característica es que no se pueden realizar *JOINS*. Dado que las tablas o colecciones no están relacionadas entre sí, para obtener datos de varias tablas, es necesario realizar varias consultas [57].

React.js es un framework orientado al diseño de interfaces. Está basado en componentes, partes de la interfaz independientes, cada una con su propio estado y lógica. Gracias a estos elementos, se pueden construir aplicaciones modulares de forma sencilla. Además, se pueden jerarquizar los componentes y pasar datos de un módulo a otro inferior. React.js, dentro de la arquitectura MVC previamente comentada, funciona como vista, ya que renderiza la interfaz de usuario, y también como controlador. Dado que funciona con JavaScript, el framework dispone de herramientas para gestionar los eventos y la comunicación con las otras capas [34].

Express.js es un framework para Node.js. Su función es simplificar la configuración de un servidor HTTP y la conexión con el *middleware* que se necesite. Es decir, aporta herramientas que permiten, por ejemplo, almacenar cookies, guardar la sesión o crear rutas de manera sencilla, en vez de tener que programar cada función a mano [58].

Por último, Node.js es un entorno de ejecución que permite utilizar JavaScript del lado del servidor. Sirve como base donde trabajar con otros módulos o frameworks que faciliten el desarrollo o aumenten la funcionalidad de un proyecto como por ejemplo, Express.js, React.js u otras mencionadas más adelante. Además, como con Node.js se puede utilizar JavaScript en cliente y servidor, la comunicación entre estos se puede realizar de manera sencilla con, por ejemplo, objetos JSON [59].

3.2.3.- Frameworks y herramientas adicionales

La aplicación web está principalmente desarrollada con la pila de tecnologías MERN vistas en el apartado anterior, pero, además, se han usado otras herramientas que aportan funcionalidad extra, simplifican algunas tareas o hacen más cómodo el desarrollo de la aplicación web. La primera viene por defecto con Node.js ya que se trata de su gestor de paquetes. NPM o *Node Package Manager* permite instalar las librerías o módulos que se vayan a usar [60]. Por ejemplo, a través de la consola de comandos. Con esta herramienta

se han instalado el resto de frameworks utilizados en el proyecto, que se van a comentar en los siguientes párrafos.

Se ha usado Nodemon, una librería para Node.js que permite automatizar el reinicio del servidor cuando se detecta un cambio en alguno de los archivos. Funciona simplemente ejecutando el servidor con el comando "*nodemon*" (en vez de el estándar de "*npm*") [61].

En el funcionamiento de una API de tipo REST es necesario garantizar la comunicación entre la base de datos, la API y el cliente. Para cumplir estas funciones se han utilizado dos librerías: Mongoose y Axios. La primera permite escribir consultas a una base de datos de MongoDB y definir modelos, que especifiquen los campos, requisitos y valores por defecto de los documentos (el equivalente a los registros de una base de datos relacional) [62]. Mongoose se ha utilizado para conectar la API y la base de datos.

En segundo lugar, Axios es un cliente HTTP para Node.js. Es asíncrono, basado en el uso de "promesas". Una promesa es un objeto de JavaScript que se queda a la espera de un recurso cuando se hace una llamada asíncrona [63][64]. Axios permite hacer consultas desde el cliente a la API, indicando una operación HTTP (POST, GET, PUT, DELETE…), una URI y los datos a enviar en formato JSON.

Biblioteca
UNIVERSITAS Miguel Hernández

3.3.- JUEGO

3.3.1.- Phaser 3

Además de la aplicación web, el proyecto consta de un videojuego multijugador en línea. Para el desarrollo de esta parte, se ha utilizado Phaser 3, un framework Javascript para HTML5 para desarrollo de videojuegos para navegador o dispositivo móvil. Utiliza WebGL o canvas para el renderizado de gráficos, gestiona la lógica del juego y los eventos, y dispone de un motor propio de físicas para la interacción entre los distintos elementos del videojuego [28].

Un proyecto Phaser está dividido en 3 partes principales. La primera es la configuración del juego. Aquí se ajustan parámetros como el ancho y alto de la ventana de juego o el tipo de físicas que se va a usar. Por otro lado están las escenas, que funcionan como ventanas o zonas independientes del juego, aunque se puede pasar información de una a otra. Por ejemplo, una escena sería la pantalla inicial, el menú principal o un nivel del juego. El tercer elemento son las clases. Phaser sigue el paradigma de la programación orientada a objetos y permite crear clases personalizadas y crear objetos de esos tipos en las escenas.

3.3.2.- Socket.io

Socket.io es un framework que funciona en el cliente y en el lado del servidor (con Node.js) y permite establecer una conexión por WebSockets. Este tipo de comunicación es en tiempo real, con baja latencia, bidireccional y está basada en eventos. En caso de fallar, Socket.io vuelve a la conexión HTTP estándar [65].

En el proyecto se utiliza este framework para enviar desde el cliente la información del jugador y, desde el servidor, compartir esos datos con el resto de usuarios. Dado que es un juego de combate, también se tiene que transmitir la posición y tipo de proyectiles. Los datos se envían en objetos JSON. Como las batallas del juego son de 3 contra 3, para evitar que sólo puedan jugar 6 jugadores en un momento dado, Socket.io permite crear salas independientes y realizar un combate en cada una.

3.3.3.- Sprites

Al ser un juego en 2D, el apartado gráfico está formado por sprites. De los 3 editores de imágenes propuestos en el capítulo 2, aunque Adobe Photoshop es el más potente, se descarta por su alto coste. En su lugar, se ha elegido GIMP, que es gratuito y dispone de herramientas más que suficientes para diseñar simples sprites de estilo Pixel Art. Esta técnica artística se centra en resaltar los píxeles del dibujo y no suavizar las imágenes.

3.3.4.- Sonido

El juego está inspirado en los arcade e igual que el apartado gráfico, el sonido es bastante básico. El programa utilizado para crear y editar los sonidos es Audacity. Es gratuito, de código abierto y tiene herramientas para grabar, editar y aplicar efectos a pistas de audio mono y estéreo.

Biblioteca
UNIVERSITAS Miguel Hernández

Capítulo 4
Metodología y resultados

4.1.- PLANIFICACIÓN DEL PROYECTO

El proyecto consta de dos partes diferenciadas: la aplicación web con funcionalidades de red social y el videojuego multijugador en línea. Pese a que ambos elementos están diseñados para funcionar en conjunto, gran parte del desarrollo se ha hecho por separado. Además, cada parte tiene unos roles de usuario, casos de uso y requerimientos propios, aunque estén estrechamente relacionados.

4.1.1.- Ciclo de vida

Sucede lo contrario con los ciclos de vida. Las páginas web y los videojuegos, especialmente los de tipo multijugador, parten de una versión base y van evolucionando con el tiempo, conforme la tecnología avanza y los usuarios demandan variedad y mejoras.

Por lo tanto, el ciclo de vida de ambas partes es de tipo iterativo. Es decir, las fases del desarrollo se repiten en iteraciones en las que se va añadiendo funcionalidad al producto. Además, este ciclo de vida permite mejorar la aplicación según la retroalimentación que aporta el cliente [66].

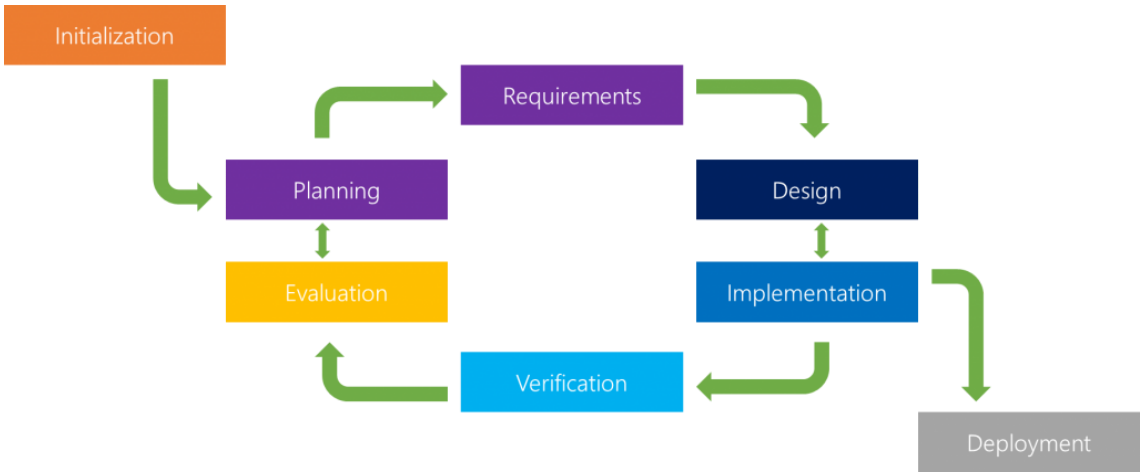

Figura 4.1.- Ciclo de vida iterativo

4.1.2.- Diagrama de Gantt

La planificación, igual que el proyecto, se puede dividir en varias partes: la aplicación web, el videojuego, la base de datos, la documentación y, dado que las principales tecnologías utilizadas en este Trabajo de Fin de Grado no se ven durante la carrera, el trabajo de investigación previo al desarrollo también se ha tenido en cuenta.

La implementación del videojuego se puede subdividir en tres secciones: la codificación de la lógica del juego, la conexión online en tiempo real y el diseño de los sprites. Como se ve en el diagrama de Gantt que aparece más adelante (Tabla 4.1 y 4.2), la lógica en cliente es la parte a la que más tiempo se le dedica, aunque siempre acompañada de uno de los otros dos elementos. En segundo lugar, el desarrollo de la aplicación web consta de tres secciones: la implementación de la API de tipo REST, el diseño de la interfaz de usuario y la codificación de las funcionalidades (inicio de sesión, búsqueda y gestión de usuarios…). El último apartado es la base de datos. Al ser de tipo no relacional, su diseño es más flexible y ha ido mutando según las necesidades de los otros elementos del proyecto.

Tabla 4.1.- Diagrama de Gantt (1)

Semana	S-1	S-2	S-3	S-4	S-5	S-6	S-7	S-8	S-9	S-10
Investigación										
Lógica del juego										
Online del juego										
Sprites										
Interfaz web										
Funciones web										
RESTful API										
Base de datos										
Documentación										

Biblioteca
UNIVERSITAS Miguel Hernández

Tabla 4.2.- Diagrama de Gantt (2)

Semana	S-11	S-12	S-13	S-14	S-15	S-16	S-17	S-18	S-19	S-20
Investigación										
Lógica del juego										
Online del juego										
Sprites										
Interfaz web										
Funciones web										
RESTful API										
Base de datos										
Documentación										

4.2.- CAPTURA DE REQUISITOS

4.2.1.- Requisitos funcionales y no funcionales

Tabla 4.3.- Requisito Funcional. Cuenta de usuario

RF-1	**Cuenta de usuario**
Descripción	Dado que es necesario guardar el progreso del juego y las conversaciones entre jugadores, cada usuario debe tener una cuenta. Además, para utilizar la plataforma, se deberá iniciar sesión o, en caso de no tener cuenta, registrarse.

Tabla 4.4.- Requisito Funcional. Seguir/Dejar de seguir un usuario

RF-2	**Seguir/Dejar de seguir un usuario**
Descripción	Los usuarios podrán seguir o dejar de seguir a otros jugadores.

Tabla 4.5.- Requisito Funcional. Mensajes

RF-3	**Mensajes**
Descripción	Los usuarios, una vez iniciada la sesión, tendrán un apartado donde enviar mensajes a los usuarios que siguen y ver las respuestas.

Tabla 4.6.- Requisito Funcional. Partida online

RF-4	**Partida online**
Descripción	Los usuarios podrán acceder al juego y que se les asigne aleatoriamente una sala en la que jugar con otros usuarios.

Tabla 4.7.- Requisito Funcional. Múltiples tanques

RF-5	**Múltiples tanques**
Descripción	Los usuarios tendrán acceso a diferentes tanques con los que jugar.

Tabla 4.8.- Requisito Funcional. Economía del juego

RF-6	**Economía del juego**
Descripción	Los usuarios ganarán recursos al jugar y podrán desbloquear vehículos de combate con ellos.

Tabla 4.9.- Requisito Funcional. Estadísticas

RF-7	**Estadísticas**
Descripción	Los jugadores podrán ver su rendimiento en el juego mediante una serie de estadísticas (batallas totales, porcentaje de victorias, daño medio…).

Tabla 4.10.- Requisito Funcional. Gestión de usuarios

RF-8	**Gestión de usuarios**
Descripción	Los administradores podrán eliminar a otros usuarios o cambiar su rol.

Tabla 4.11.- Requisito Funcional. Modificación cuenta de usuario

RF-9	**Modificación cuenta de usuario**
Descripción	Los usuarios podrán modificar su nombre de usuario, correo electrónico y contraseña.

Tabla 4.12.- Requisito Funcional. Tanques en desarrollo

RF-10	**Tanques en desarrollo**
Descripción	Los *testers* y administradores tendrán acceso a tanques en desarrollo.

Tabla 4.13.- Requisito No Funcional. Seguridad

RNF-1	**Seguridad**
Descripción	Como el juego es online, habrá que implementar métodos para que la comunicación con el servidor y otros usuarios sea segura.

4.2.2.- Roles de usuarios

La aplicación dispone de 3 roles o tipos de usuario: jugador, *tester* y administrador. El primero es el rol predeterminado, pudiendo explorar la aplicación web y jugar de forma normal (aunque es el único que puede cambiar su nombre de usuario y registrarse). El *tester*, es un jugador que, además, tiene acceso a contenido en desarrollo. Por último, el administrador puede eliminar usuarios o cambiar su rol. En cuanto a la herencia, se ha creado un rol ficticio llamado "funciones básicas" que agrupa los casos de uso comunes al resto de tipos de usuario. De esta manera, jugador y *tester* heredan de "funciones básicas" y administrador quedaría como una especialización de *tester.*

Tabla 4.14.- Rol de usuario. Administrador

Rol de usuario	Administrador
Descripción	Tiene control sobre el acceso y privilegios de otros jugadores (Excepto de otros administradores). Es decir, puede eliminar cuentas y cambiar el rol de otros usuarios. Por otro lado, dentro del juego, dispone de recursos ilimitados y puede acceder a elementos en desarrollo.
Casos de uso	C.U.2, C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22, C.U.23, C.U.24, C.U.25, C.U.26.

Tabla 4.15.- Rol de usuario. *Tester*

Rol de usuario	*Tester*
Descripción	Tiene acceso a todo el contenido del juego, incluso al que está en desarrollo y dispone de recursos ilimitados. Su función es probar modificaciones o elementos nuevos para su adecuado equilibrio jugable.
Casos de uso	C.U.2, C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22.

Tabla 4.16.- Rol de usuario. Jugador

Rol de usuario	Jugador
Descripción	Puede jugar, ver otros perfiles de usuario, seguirles y conversar con ellos. Dentro del juego, tiene recursos limitados. A diferencia de los otros roles, puede cambiar su nombre de usuario.
Casos de uso	C.U.1, C.U.2, C.U.3, C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22.

Tabla 4.17.- Rol de usuario. Funciones básicas

Rol de usuario	Funciones básicas
Descripción	Conjunto de casos de uso que son comunes al resto de tipos de usuarios.
Casos de uso	C.U.2, C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22.

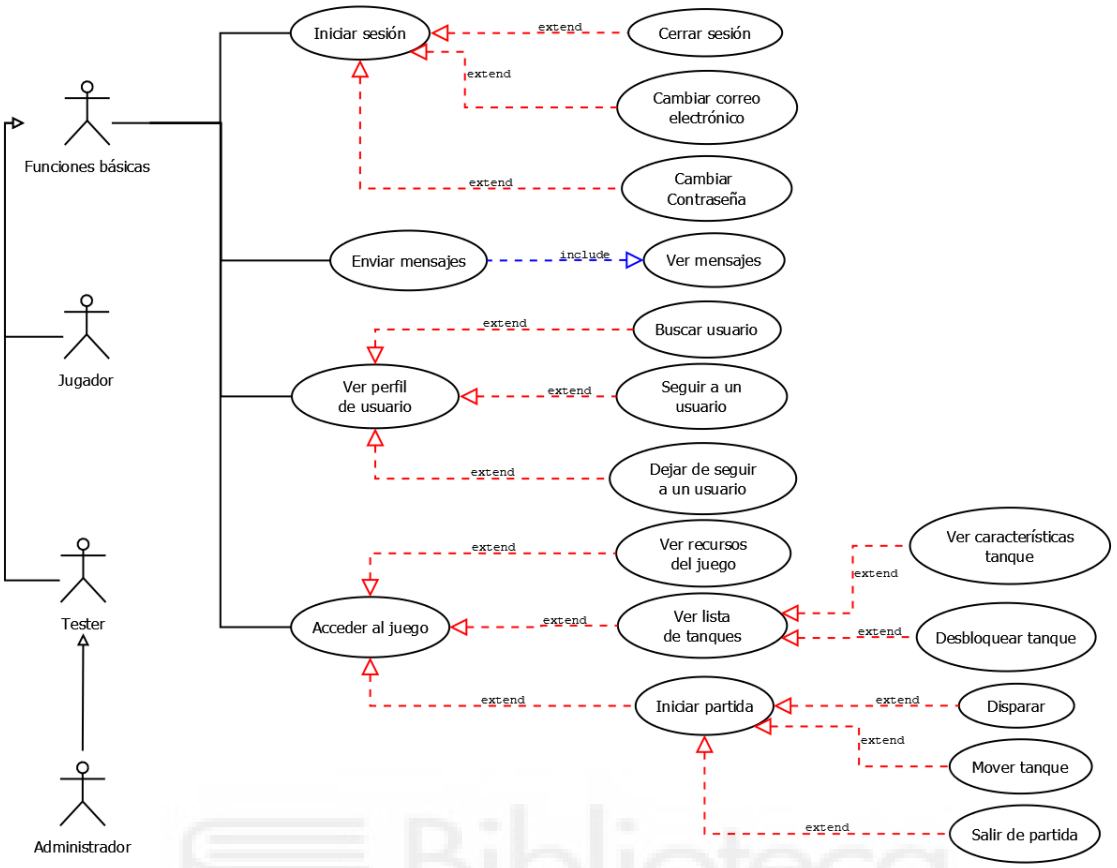

Figura 4.2.- Diagrama de casos de uso. Funciones básicas

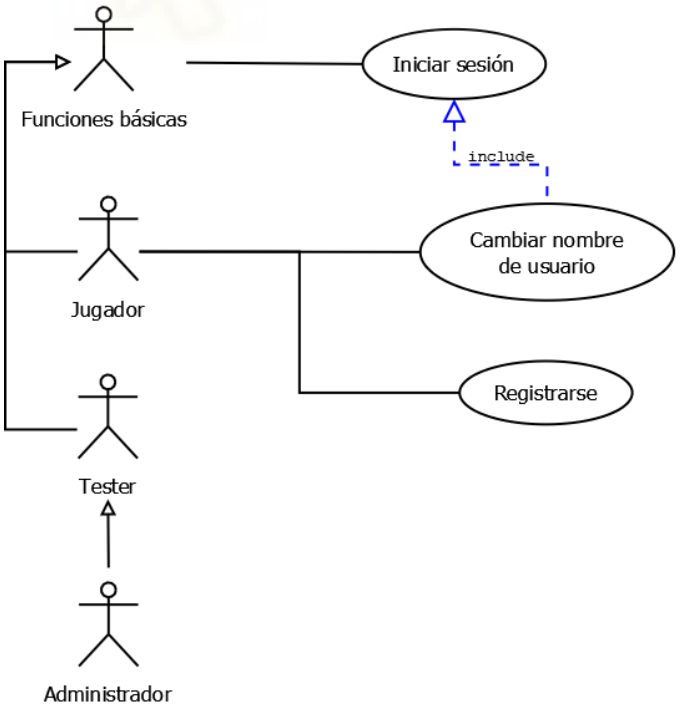

Figura 4.3.- Diagrama de casos de uso. Jugador.

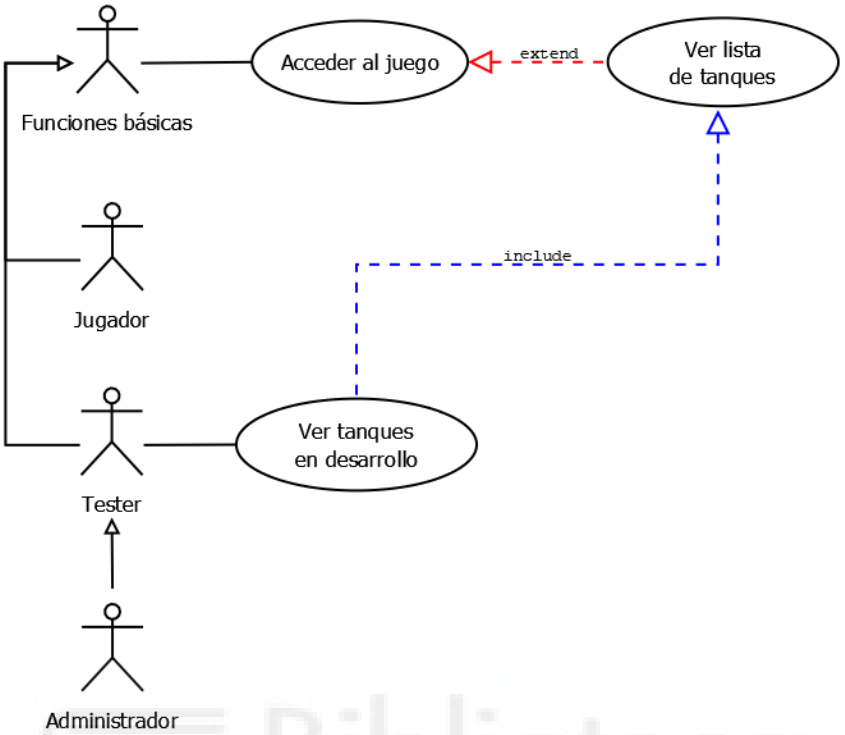

Figura 4.4.- Diagrama de casos de uso. *Tester*.

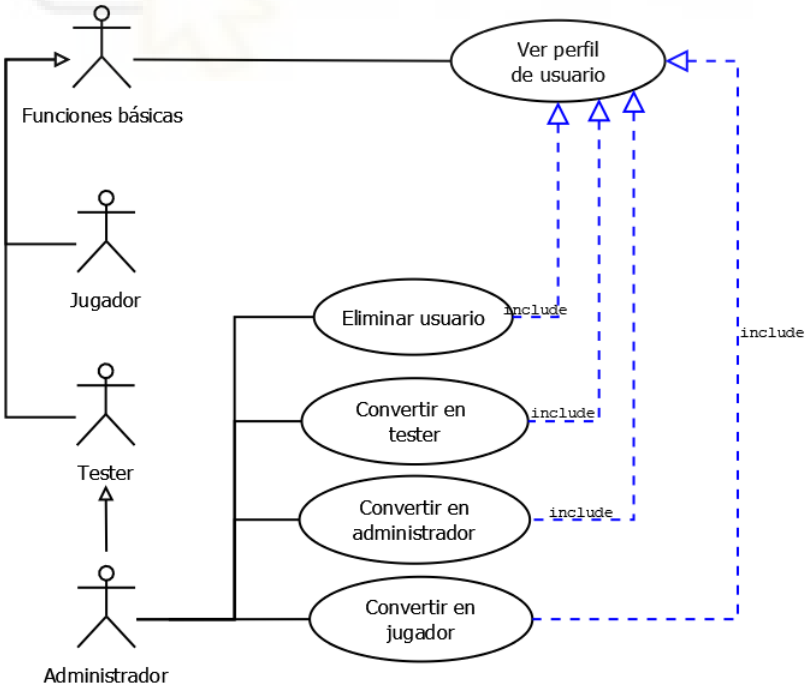

Figura 4.5.- Diagrama de casos de uso. Administrador.

4.2.3.- Casos de uso

Tabla 4.18.- Caso de uso. Registrar cuenta

C.U.1	Registrar cuenta
Actores	Jugador
Descripción	Creación de una cuenta mediante un formulario
Precondición	
Secuencia normal	P1 - Introducir nombre de usuario P2 - Introducir email P3 - Introducir contraseña P4 - Repetir contraseña P5 - Enviar formulario
Postcondición	El usuario queda registrado y su sesión se inicia automáticamente. Se le redirige a la página de inicio.
Excepciones	- Si el usuario o email ya existen en la base de datos, se muestra un mensaje por pantalla y se resetea el formulario. - Si el campo "contraseña" y "repetir contraseña" no son iguales, se le indica al usuario y el formulario se resetea. - Si la contraseña no cumple con los requisitos mínimos, aparece un mensaje informando al usuario.
Frecuencia	Alta
Importancia	Alta

Tabla 4.19.- Caso de uso. Iniciar sesión

C.U.2	Iniciar Sesión
Actores	Administrador, *tester*, jugador y "funciones básicas"
Descripción	Identificación del usuario mediante un formulario
Precondición	C.U.1
Secuencia normal	P1 - Introducir email P2 - Introducir contraseña P3 - Enviar formulario
Postcondición	El usuario queda identificado, pudiendo acceder a toda la funcionalidad de la aplicación. Se le redirige al inicio.
Excepciones	- Si la contraseña no corresponde con la del usuario, se le indica al usuario. - Si el email indicado no existe, se muestra mensaje y se resetea el formulario.
Frecuencia	Alta
Importancia	Alta

Tabla 4.20.- Caso de uso. Cambiar nombre de usuario

C.U.3	Cambiar nombre de usuario
Actores	Jugador
Descripción	El usuario puede cambiar su nombre de usuario a través de un formulario
Precondición	C.U.2
Secuencia normal	P1 - Introducir nuevo nombre de usuario P2 - Introducir contraseña P3 - Enviar formulario
Postcondición	A partir de ese momento, el usuario se identificará con el nuevo nombre
Excepciones	- Si el nuevo nombre ya pertenece a otro usuario, se muestra un mensaje de error. El cambio no tiene efecto. - Si la contraseña no es correcta, se indica por pantalla. No se realiza el cambio.
Frecuencia	Baja
Importancia	Baja

Tabla 4.21.- Caso de uso. Cambiar contraseña

C.U.4	Cambiar contraseña
Actores	Administrador, *tester*, jugador y “funciones básicas”.
Descripción	El usuario puede cambiar su contraseña
Precondición	C.U.2
Secuencia normal	P1 – Introducir contraseña actual P2 – Introducir nueva contraseña P3 – Repetir nueva contraseña P4 – Enviar formulario
Postcondición	La contraseña del usuario cambia por la nueva
Excepciones	- Si la nueva contraseña no cumple los requisitos de seguridad, se muestra un mensaje y el cambio no se efectúa. - Si la nueva contraseña no se repite exactamente, se informa al usuario. No se cambia la contraseña. - Si la contraseña actual no es correcta, se indica por pantalla. La contraseña no cambia.
Frecuencia	Baja
Importancia	Alta

Tabla 4.22.- Caso de uso. Cambio de correo electrónico

C.U.5	Cambio de correo electrónico
Actores	Administrador, *tester*, jugador y "funciones básicas"
Descripción	El usuario puede cambiar su correo electrónico
Precondición	C.U.2
Secuencia normal	P1 - Introducir correo electrónico actual P2 - Introducir el nuevo correo electrónico P3 - Repetir el correo electrónico nuevo P4 - Introducir contraseña P5 - Enviar formulario
Postcondición	El correo electrónico del usuario cambia
Excepciones	- Si el correo electrónico actual es erróneo, se muestra un mensaje y no se hace el cambio. - Si el nuevo correo electrónico ya pertenece a otro usuario, se indica por pantalla. El cambio no se efectúa. - Si el nuevo correo electrónico no se repite exactamente, se informa al usuario. No se realiza el cambio. - Si la contraseña no es correcta, se indica por pantalla. No cambia el correo electrónico.
Frecuencia	Baja
Importancia	Alta

Tabla 4.23.- Caso de uso. Cerrar sesión

C.U.6	Cerrar Sesión
Actores	Administrador, *tester*, jugador y "funciones básicas".
Descripción	El usuario puede cerrar su sesión en cualquier momento
Precondición	C.U.2
Secuencia normal	P1 - Hacer clic en el botón de cerrar sesión
Postcondición	Se cierra la sesión del usuario y se redirige a la página de inicio
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.24.- Caso de uso. Buscar un usuario

C.U.7	Buscar un usuario
Actores	Administrador, *tester*, jugador y "funciones básicas".
Descripción	Buscar un usuario mediante su nombre
Precondición	C.U.2
Secuencia normal	P1 – Introducir un nombre de usuario en la barra de búsqueda P2 – Hacer clic en el botón de buscar
Postcondición	Se redirige al perfil del usuario buscado
Excepciones	- Si el usuario no existe, aparece un mensaje de error.
Frecuencia	Alta
Importancia	Alta

Tabla 4.25.- Caso de uso. Ver perfil de usuario

C.U.8	Ver perfil de usuario
Actores	Administrador, *tester*, jugador y "funciones básicas".
Descripción	Ver la información y rendimiento en batalla de un usuario
Precondición	C.U.2, C.U.6
Secuencia normal	P1 – Buscar un usuario mediante la barra de búsqueda
Postcondición	
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.26.- Caso de uso. Seguir a un usuario

C.U.9	Seguir a un usuario
Actores	Administrador, *tester*, jugador y "funciones básicas".
Descripción	Añadir a un usuario a la lista de amigos
Precondición	C.U.2, C.U.8
Secuencia normal	P1 – Hacer clic en el botón de seguir usuario
Postcondición	El botón cambia el texto a "seguido" y se guarda a ese usuario en la lista de amigos
Excepciones	- Si el usuario ya lo seguía, entonces lo dejará de seguir. - Un usuario no se puede seguir a si mismo. No aparece botón en el perfil.
Frecuencia	Alta
Importancia	Alta

Tabla 4.27.- Caso de uso. Dejar de seguir a un usuario

C.U.10	Dejar de seguir a un usuario
Actores	Administrador, *tester*, jugador y "funciones básicas".
Descripción	Dejar de ser amigo de otro usuario
Precondición	C.U.2, C.U.8, C.U.9
Secuencia normal	P1 – Hacer clic en el botón que pone "seguido"
Postcondición	Se elimina usuario de lista de amigos y el botón cambia a "seguir"
Excepciones	- Si no se sigue al usuario, el botón es diferente y, al clicar, se le incluirá en amigos. - En el perfil propio no aparece el botón.
Frecuencia	Alta
Importancia	Alta

Tabla 4.28.- Caso de uso. Ver mensajes

C.U.11	Ver mensajes
Actores	Administrador, *tester*, jugador y "funciones básicas".
Descripción	Se pueden enviar mensajes a los jugadores que se sigue
Precondición	C.U.2
Secuencia normal	P1 – Ir a la pestaña de mensajes
Postcondición	Aparece una lista con los mensajes recibidos y enviados
Excepciones	- Si no hay mensajes, se muestra mensaje indicativo
Frecuencia	Alta
Importancia	Alta

Tabla 4.29.- Caso de uso. Enviar mensajes

C.U.12	Enviar mensajes
Actores	Administrador, *tester*, jugador y "funciones básicas".
Descripción	Se pueden enviar mensajes a los jugadores que se sigue
Precondición	C.U.2, C.U.9, C.U.11
Secuencia normal	P1 - Hacer clic en el usuario al que se quiera enviar el mensaje P2 - Escribir mensaje en la casilla P3 - Enviar mensaje
Postcondición	El mensaje enviado se guardará en la base de datos y se le entregará al destinatario cuando acceda a sus mensajes
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.30.- Caso de uso. Acceder al juego

C.U.13	Acceder al juego
Actores	Administrador, *tester*, jugador y “funciones básicas”.
Descripción	El usuario puede acceder a la pestaña del juego
Precondición	C.U.2
Secuencia normal	P1 - Hacer clic en la pestaña del juego
Postcondición	Se abre la página del juego y aparece la página de inicio del juego
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.31.- Caso de uso. Ver recursos del juego

C.U.14	Ver recursos del juego
Actores	Administrador, *tester*, jugador y “funciones básicas”.
Descripción	El usuario puede ver los recursos de que dispone
Precondición	C.U.2, C.U.13
Secuencia normal	P1 - Pulsar el botón “comenzar”
Postcondición	Se le muestra al jugador los recursos de los que dispone
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.32.- Caso de uso. Ver lista de tanques

C.U.15	Ver lista de tanques
Actores	Administrador, *tester*, jugador y “funciones básicas”.
Descripción	El usuario puede ver una lista de los tanques disponibles
Precondición	C.U.2, C.U.13
Secuencia normal	P1 - Pulsar el botón “comenzar”
Postcondición	Se muestran los tanques existentes
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.33.- Caso de uso. Ver tanques en desarrollo

C.U.16	Ver tanques en desarrollo
Actores	Administrador y *tester*
Descripción	El usuario puede ver una lista de los tanques disponibles
Precondición	C.U.2, C.U.13
Secuencia normal	P1 - Pulsar el botón “comenzar”
Postcondición	Se muestran los tanques existentes, normales y en desarrollo.
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.34.- Caso de uso. Ver características tanque

C.U.17	Ver características tanque
Actores	Administrador, *tester*, jugador y “funciones básicas”.
Descripción	Visualizar las características detalladas de cada vehículo de combate del juego
Precondición	C.U.2, C.U.15
Secuencia normal	P1 - Seleccionar el tanque que se quiera consultar
Postcondición	Se muestra la información del tanque
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.35.- Caso de uso. Desbloquear tanque

C.U.18	Desbloquear tanque
Actores	Administrador, tester, jugador y “funciones básicas”.
Descripción	El jugador puede desbloquear tanques y usarlos en batalla
Precondición	C.U.2, C.U.17
Secuencia normal	P1 – Hacer clic en el botón de desbloquear
Postcondición	El vehículo estará disponible para utilizarlo en batalla
Excepciones	- Si el jugador no tiene recursos suficientes para desbloquearlo, aparece un mensaje de error.
Frecuencia	Media
Importancia	Alta

Tabla 4.36.- Caso de uso. Iniciar partida

C.U.19	Iniciar partida
Actores	Administrador, tester, jugador y “funciones básicas”.
Descripción	El jugador puede entrar en partida y jugar con el tanque seleccionado
Precondición	C.U.2, C.U.17
Secuencia normal	P1 - Darle al botón “batalla”
Postcondición	El jugador es incluido en una sala en la que jugar con otros jugadores
Excepciones	- Si no hubiera jugadores disponibles, se creará una sala nueva y el jugador esperará a otros jugadores.
Frecuencia	Alta
Importancia	Alta

Tabla 4.37.- Caso de uso. Mover tanque

C.U.20	Mover tanque
Actores	Administrador, tester, jugador y “funciones básicas”.
Descripción	El jugador puede mover su vehículo en el campo de batalla
Precondición	C.U.2, C.U.19
Secuencia normal	P1 - Hacer clic a una de las flechas del teclado
Postcondición	El tanque se mueve en la dirección de la flecha pulsada
Excepciones	- Si el tanque está en contacto con un elemento inamovible, no se moverá en esa dirección.
Frecuencia	Alta
Importancia	Alta

Tabla 4.38.- Caso de uso. Disparar

C.U.21	Disparar
Actores	Administrador, tester, jugador y “funciones básicas”.
Descripción	El jugador, dentro de la partida, puede disparar a otros tanques
Precondición	C.U.2, C.U.19
Secuencia normal	P1 - Mover el ratón a la posición a la que se quiere disparar P2 - Hacer clic en el botón izquierdo del ratón
Postcondición	Un proyectil es disparado, teniendo como origen el tanque, y como destino la posición del ratón.
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.39.- Caso de uso. Salir de partida

C.U.22	Salir de partida
Actores	Administrador, tester, jugador y “funciones básicas”.
Descripción	Salir de la partida y volver a la página del jugador
Precondición	C.U.2, C.U.19
Secuencia normal	P1 – Pulsar la tecla “Esc” P2 - Hacer clic en “salir”
Postcondición	La partida termina y el jugador vuelve a la página donde ver sus recursos
Excepciones	
Frecuencia	Alta
Importancia	Alta

Tabla 4.40.- Caso de uso. Eliminar usuario

C.U.23	Eliminar usuario
Actores	Administrador
Descripción	El administrador puede eliminar a un usuario de la base de datos y, por tanto, de la aplicación
Precondición	C.U.2, C.U.8
Secuencia normal	P1 – Hacer clic en eliminar usuario P2 – Introducir contraseña del administrador P3 – Hacer clic en eliminar
Postcondición	El usuario eliminado pierde el acceso a la cuenta para siempre.
Excepciones	- Si la contraseña es incorrecta, aparece un mensaje de error y el usuario no se borra.
Frecuencia	Baja
Importancia	Alta

Tabla 4.41.- Caso de uso. Convertir usuario en administrador

C.U.24	Convertir usuario en administrador
Actores	Administrador
Descripción	Un administrador puede convertir a cualquier usuario en administrador.
Precondición	C.U.2, C.U.8
Secuencia normal	P1 – Hacer clic en cambiar rol P2 – Seleccionar el rol de administrador P3 – Introducir contraseña del administrador P4 – Hacer clic en cambiar
Postcondición	El usuario se convertirá en administrador y ganará los privilegios de dicho rol.
Excepciones	- Si la contraseña es incorrecta, aparece un mensaje de error y no se cambia el rol del usuario. - La opción no está disponible si el usuario ya pertenece al rol administrador.
Frecuencia	Baja
Importancia	Alta

Tabla 4.42.- Caso de uso. Convertir usuario en *tester*

C.U.25	Convertir usuario en *tester*
Actores	Administrador
Descripción	Un administrador puede convertir a cualquier jugador en *tester*
Precondición	C.U.2, C.U.8
Secuencia normal	P1 – Hacer clic en cambiar rol P2 – Seleccionar el rol de *tester* P3 – Introducir contraseña del administrador P4 – Hacer clic en cambiar
Postcondición	El usuario seleccionado se convertirá en *tester* y obtendrá los privilegios de dicho rol.
Excepciones	- Si la contraseña es incorrecta, aparece un mensaje de error y no se cambia el rol del usuario. - La opción no está disponible si el usuario ya pertenece al rol jugador. - Si el usuario es administrador, está opción no aparece.
Frecuencia	Baja
Importancia	Alta

Tabla 4.43.- Caso de uso. Convertir usuario en jugador

C.U.26	Convertir usuario en jugador
Actores	Administrador
Descripción	Un administrador puede cambiar el rol de *tester* a jugador
Precondición	C.U.2, C.U.8
Secuencia normal	P1 – Hacer clic en cambiar rol P2 – Seleccionar el rol de jugador P3 – Introducir contraseña del administrador P4 – Hacer clic en cambiar
Postcondición	El usuario seleccionado se convertirá en jugador y obtendrá los privilegios de dicho rol.
Excepciones	- Si la contraseña es incorrecta, aparece un mensaje de error y no se cambia el rol del usuario. - La opción no está disponible si el usuario ya pertenece al rol jugador. - Si el usuario es administrador, está opción no aparece.
Frecuencia	Baja
Importancia	Alta

4.3.- DISEÑO

4.3.1.- Diagrama de secuencia

La aplicación, tal y como se explica en el apartado 3.2.1, sigue la arquitectura Modelo-Vista-Controlador. Por tanto, para la realización de muchos de los casos de uso disponibles, el paso de mensajes entre las distintas partes del sistema es el mismo. Se pueden diferenciar cuatro diagramas principales, dependiendo del elemento con el que se interactúe.

El primer componente con el que se puede interactuar es el formulario. Ya sea, para registrarse, iniciar sesión, cambiar el nombre de usuario o, incluso, el envío de mensajes en tiempo real, el proceso que sigue el sistema para la ejecución de las tareas es el mismo. El usuario envía el formulario y la aplicación hace una solicitud HTTP (POST, GET, PUT…) a la API de tipo REST. Después, la API realiza la query sobre la base de datos. Tras recibir el resultado de la query, los datos se envían al controlador junto con el código de respuesta HTTP y se le muestra al usuario la vista o el mensaje (en caso de error) pertinente.

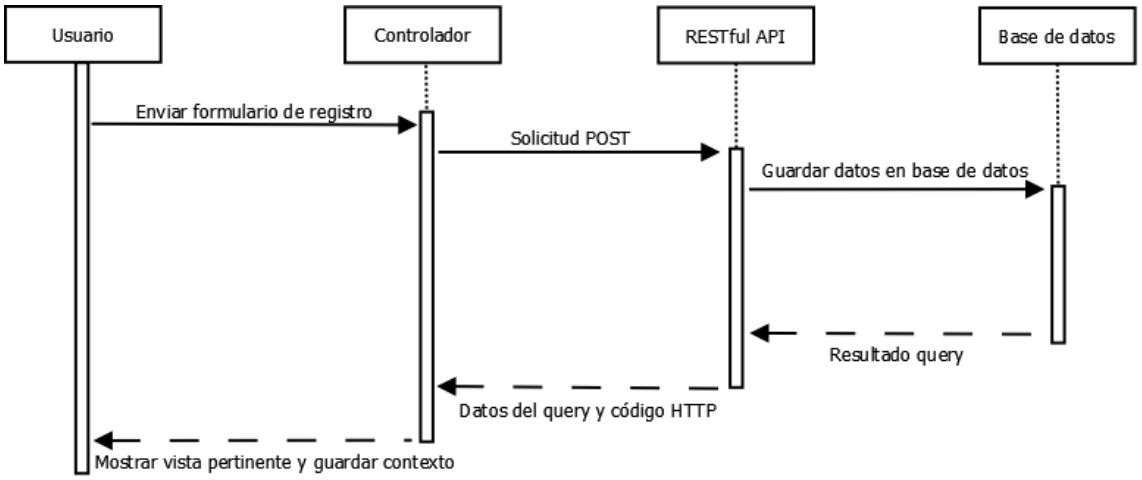

Figura 4.6.- Diagrama de secuencia. Interacción con formularios.

En segundo lugar, el usuario puede interactuar con los enlaces y botones de la aplicación. En este caso, la secuencia es mucho más sencilla, ya que no es necesario solicitar información a la API o a la base de datos. El controlador comprueba, mediante el contexto que nos facilita React.js, que el usuario ha iniciado sesión y tiene los permisos suficientes para acceder al recurso solicitado.

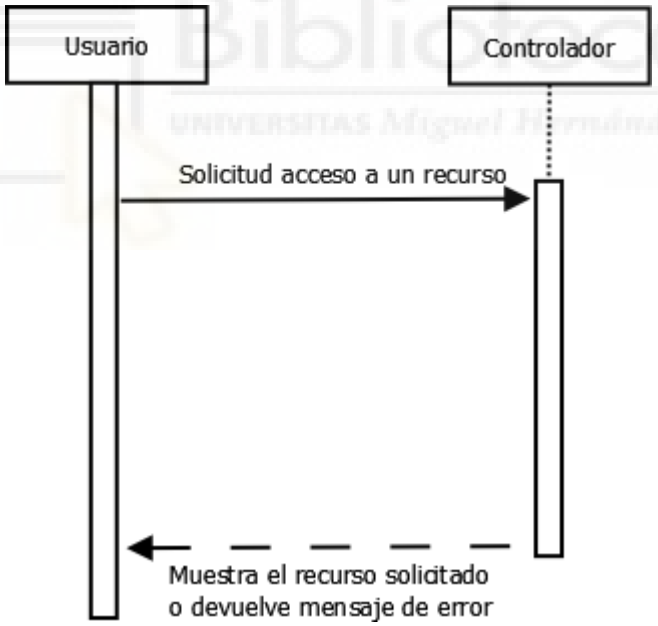

Figura 4.7.- Diagrama de secuencia. Navegación por la aplicación web.

Por último, el tercer y cuarto diagrama de secuencia representan las tareas disponibles dentro del juego, fuera y dentro de partida. Para asegurar que los jugadores no hagan trampa, cualquier acción que realicen se tiene que comparar con los datos del servidor o de la base de datos. Por ejemplo, si el usuario quiere desbloquear un tanque, habrá que comprobar, solicitando la información a la base de datos, que dispone de los recursos

suficientes para hacerlo. Por otro lado, dentro de batalla, hay que enviar la posición del jugador al servidor y, de ahí, al resto de jugadores.

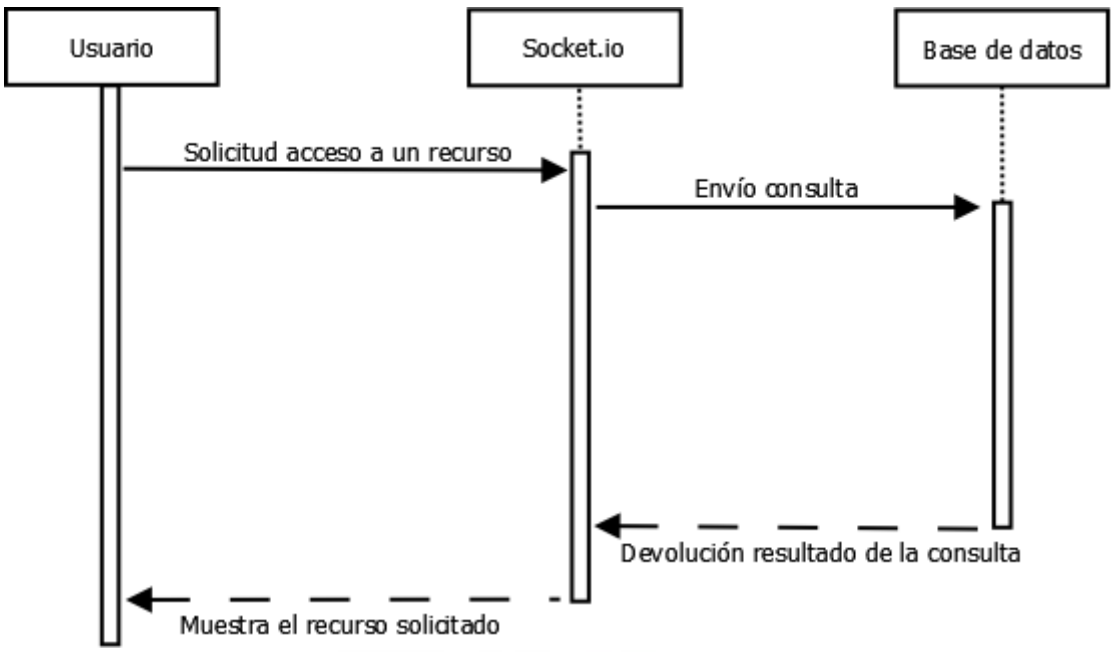

Figura 4.8.- Diagrama de secuencia. Solicitud recurso del juego, fuera de batalla.

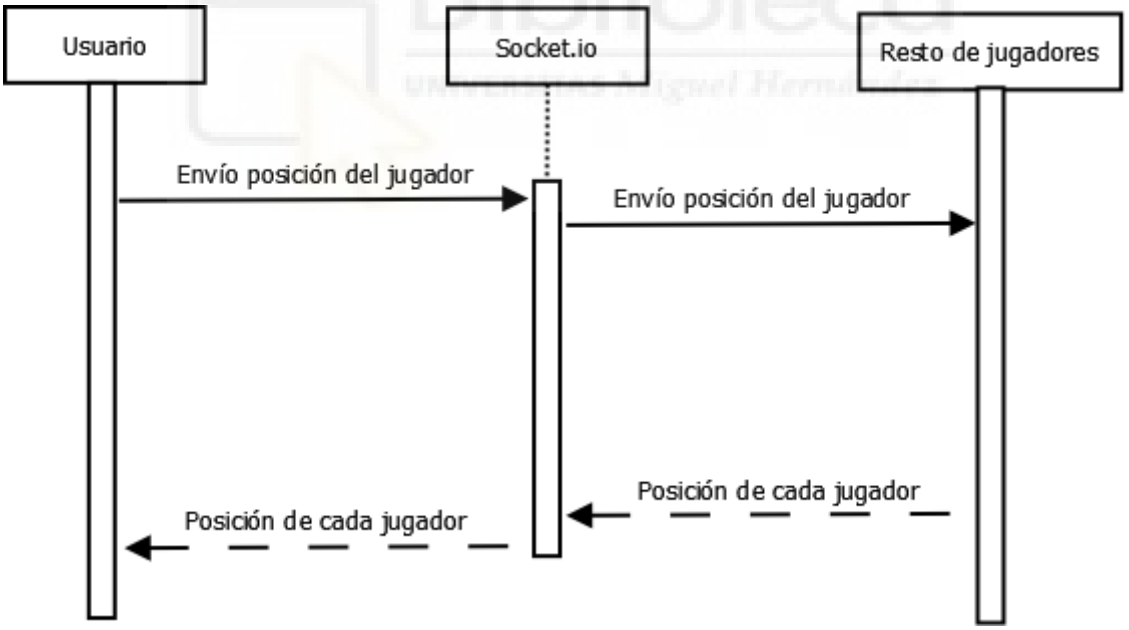

Figura 4.9.- Diagrama de secuencia. Posición de los jugadores en batalla.

4.3.2.- Base de datos

Para el proyecto se ha utilizado MongoDB, un gestor de bases de datos de tipo NoSQL y, por tanto, no se puede hacer un diagrama entidad/relación. Los datos que almacena la

aplicación están distribuidos en cuatro colecciones (equivalentes a tablas en el modelo relacional): usuarios, tanques, conversaciones entre jugadores y mensajes de cada conversación. Pese a que las colecciones son flexibles, se ha creado una estructura predeterminada mediante el framework Mongoose.

En la colección de usuarios se almacena una clave identificativa, el nombre de usuario, el correo electrónico, la contraseña debidamente encriptada, los usuarios que sigue y que le siguen, los tanques que tiene desbloqueados y las estadísticas de combate (número de batallas, de victorias, experiencia media, daño máximo…).

```
const UserSchema = new mongoose.Schema(
    {
        username: {
            type: String,
            require: true,
            min:3,
            max:20,
            unique: true
        },
        email: {
            type: String,
            require: true,
            max:50,
            unique: true
        },
```

Figura 4.10.- Parte de la estructura de la colección de usuarios.

Por otro lado, de los tanques se guarda un número de identificación, el nombre, el rol que desempeña en batalla, la vida, la potencia de fuego, la velocidad, el chasis y la torreta que utiliza, los recursos que necesita para ser desbloqueado y un booleano para comprobar si es un vehículo que está en desarrollo o no.

La colección de conversaciones, además de tener un identificador propio, almacena la clave de identificación de cada usuario que participa en la conversación. Por otro lado, la colección de mensajes guarda el identificador de la conversación, el del emisor y el texto del mensaje.

4.3.3.- Diseño de una escena en Phaser 3

El juego está compuesto por escenas, conjuntos de elementos relacionados entre sí. Además, dispone de métodos que permiten establecer un estado inicial que se puede

modificar durante la ejecución mediante la interacción del jugador o datos recibidos del servidor. Estas escenas pueden ser la pantalla de carga, el menú principal o cada campo de batalla.

Aunque cada una se programe para un propósito diferente, todas siguen la misma estructura de tres componentes y uno opcional. El primer método es "preload". Sirve para cargar todos los assets (sprites y sonidos) que se vayan a utilizar en la escena o, si se quisiera, todo el juego.

```
preload(){
    this.load.path = '/assets/imgs/';
    this.load.image('cargando', 'text/cargando.png');
    this.load.image('fondo', 'fondo.png');
    this.load.image('logo', 'text/mainTitle.png');
    this.load.image('comenzar', 'text/comenzar.png');
}
```

Figura 4.11.- Estructura de una escena. Método preload.

El segundo elemento se llama "create". En él se declaran e inicializan las variables que va a utilizar la escena. También se pueden crear grupos de objetos y asignarles un comportamiento específico, como las colisiones con otro grupo o si les afecta la gravedad. Los eventos también se establecen dentro de este método.

```
create(){

    this.otherPlayers = this.physics.add.group();

    this.bricks = this.physics.add.staticGroup();

    this.balasAmigas = this.physics.add.group({classType: Bala, runChildUpdate: true});
    this.balasEnemigas = this.physics.add.group({classType: BalaEnemiga, runChildUpdate: true});

    this.cursors = this.input.keyboard.createCursorKeys();

    crearObstaculos(this.game, this.bricks);
```

Figura 4.12.- Estructura de una escena. Método create.

En tercer lugar está la función "update". Este método es un bucle en el que se puede comprobar y/o actualizar variables, objetos y condiciones. En el proyecto, se utiliza este método para comprobar si el jugador se ha movido respecto a la última vez que se recorrió el bucle y, en caso verdadero, enviar la posición al servidor. De esta manera se evita sobrecargar la conexión con el servidor.

```
update(time,delta){

    if(this.hull){

        if(this.cursors.left.isDown){
            this.hull.angle = 270;
            this.hull.setVelocity(-80,0);
            this.turret.setVelocity(-80,0);
        }else if(this.cursors.right.isDown){
            this.hull.angle = 90;
            this.hull.setVelocity(80,0);
            this.turret.setVelocity(80,0);
        }else if(this.cursors.down.isDown){
            this.hull.angle = 180;
            this.hull.setVelocity(0,80);
            this.turret.setVelocity(0,80);
```

Figura 4.13.- Estructura de una escena. Método update.

Dentro de una escena se puede saltar a otra y, opcionalmente, se pueden enviar datos de la escena original a la nueva mediante objetos JSON. La segunda escena, para guardar en su instancia los datos recibidos, utiliza el método "init". Esta función recibe un sólo argumento con la información transmitida por la escena anterior.

```
init(data){
    this.jugador = data.jugador;
    this.tanques = data.tanques;
}
```

Figura 4.14.- Estructura de una escena. Método init.

4.3.4.- Diseño de sprites

El juego está inspirado en juegos arcade y en la estética pixel art. Este tipo de diseño busca resaltar los píxeles que forman las imágenes y evita el suavizado. Los objetos más importantes del juego, los tanques, están formados por dos sprites distintos: el chasis y la torreta. El primer elemento es de 50x50 píxeles y el segundo de 22x60 píxeles. Dentro del juego los tanques tomarán el color del equipo en el que estén (rojo o azul). Para que el color aplicado por el juego se vea adecuadamente, torreta y chasis se han tenido que diseñar en distintos tonos de gris.

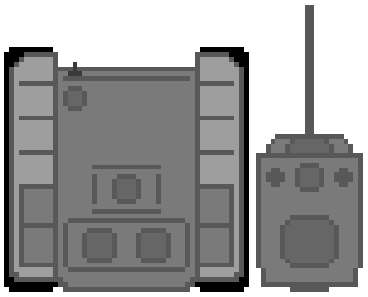

Figura 4.15.- Chasis y torreta de un tanque

La pantalla de inicio, la de carga y el menú tienen un fondo que consiste en un color sólido y un borde formado por bloques de 24x24 píxeles. Los distintos campos de batalla están diseñados de forma similar. Tienen un fondo básico y se diferencian por las estructuras y barreras formadas por los bloques.

Figura 4.16.- Fondo básico de los menús

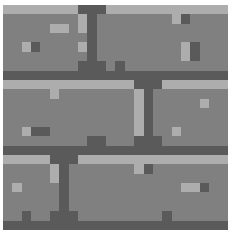

Figura 4.17.- Pared del juego (bloque de 24x24 píxeles)

Por último, los proyectiles disparados por los tanques son sprites de 16x16 píxeles y son de color rojo o azul, dependiendo del equipo. Otro elemento de pequeño tamaño es la retícula. En el juego, este elemento se controla con el ratón e indica la dirección hacia dónde se va a disparar. Cada usuario ve la suya y no tiene acceso a la del resto de jugadores. Es roja y de tamaño 13x13 píxeles.

Figura 4.18.- Proyectiles y retícula

4.4.- IMPLEMENTACIÓN

4.4.1.- Comunicación por socket

El juego está pensado para que múltiples usuarios jueguen uno contra otro en tiempo real. Para garantizar que la experiencia sea la mejor, es necesario utilizar un sistema de comunicación rápido y eficiente, que minimice la latencia o retardo. El framework Socket.io suple dichas necesidades y, además, lo hace mediante eventos, lo que simplifica la programación.

En cliente y servidor la comunicación se realiza mediante dos tipos de funciones, una para enviar datos y otra para recibirlos ("emit" y "on"). El cliente tan sólo puede comunicarse con el servidor, mientras que este último, puede conectar con uno, con varios o con todos los clientes al mismo tiempo. Para coordinar las dos partes, los métodos de envío y recibo tienen un string que los identifica. Por ejemplo, el cliente enviará el evento "disparo" y, el servidor, ejecutará la función que tenga ese identificador.

```
var x = this.hull.x;
var y = this.hull.y;
var a = this.hull.angle;
var r = this.turret.rotation = Phaser.Math.Angle.Between(this.turret.x, this.turret.y, this.reticula.x, this.reticula.y) + Math.PI/2;

if (this.hull.oldPosition && (x !== this.hull.oldPosition.x || y !== this.hull.oldPosition.y || a !== this.hull.oldPosition.angle) ||
    this.socket.emit('playerMovement', { x: this.hull.x, y: this.hull.y, angle: this.hull.angle,  rotation: this.turret.rotation});
}

this.hull.oldPosition = {
    x: this.hull.x,
    y: this.hull.y,
    angle: this.hull.angle
};

this.turret.oldRotation = this.turret.rotation;
```

Figura 4.19.- Función que envía la posición del jugador al servidor

```
socket.on('playerMovement', function(movementData){
    players[socket.id].x = movementData.x;
    players[socket.id].y = movementData.y;
    players[socket.id].angle = movementData.angle;
    players[socket.id].rotation = movementData.rotation;
    // Se envía la posición a todos los jugadores conectados
    socket.broadcast.emit('playerMoved', players[socket.id]);
});
```

Figura 4.20.- Servidor recibe la posición del cliente y la envía al resto de jugadores

En el servidor, para iniciar el servicio, gracias a Node.js tan sólo es necesario asignarle a una constante el servicio que se va a usar (mediante la palabra clave "require" y el nombre del framework), el puerto en el que va a actuar y el puerto con el que debe comunicarse. En el cliente, tan sólo hay que importar una función de Socket.io y guardar el resultado en una variable que se utilizará para llamar a los otros métodos ("emit" y "on").

```
const io = require("socket.io")(8800, {
    cors: {
      origin: "http://localhost:3000",
    },
});
```

Figura 4.21.- Inicio del servicio de socket.io en el servidor

4.4.2.- Diseño de la aplicación web

A continuación se van a mostrar capturas de pantalla de la aplicación web terminada. Las páginas con las que cuenta son las siguientes: página de inicio, página del juego, formulario de registro y de inicio de sesión, perfil y ajustes de usuario y ventana con las conversaciones. Además, todas y cada una de las páginas cuentan con el mismo *header* y *footer*.

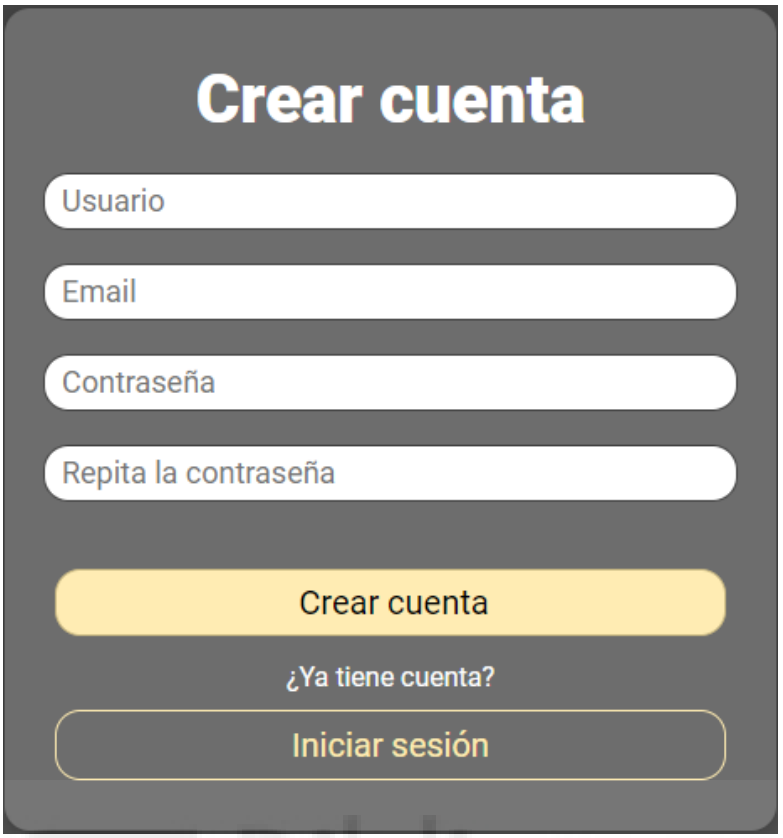

Figura 4.22.- Formulario de registro

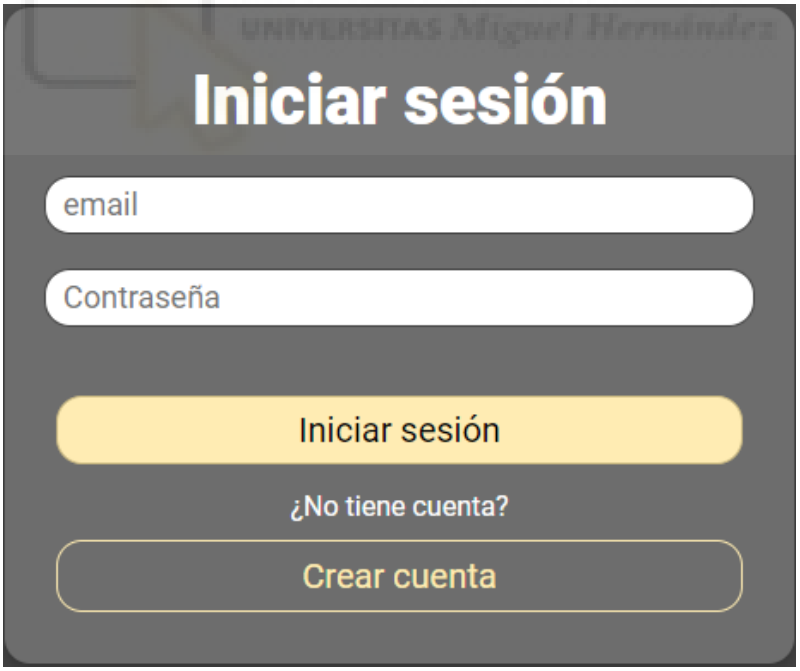

Figura 4.23.- Formulario de inicio de sesión

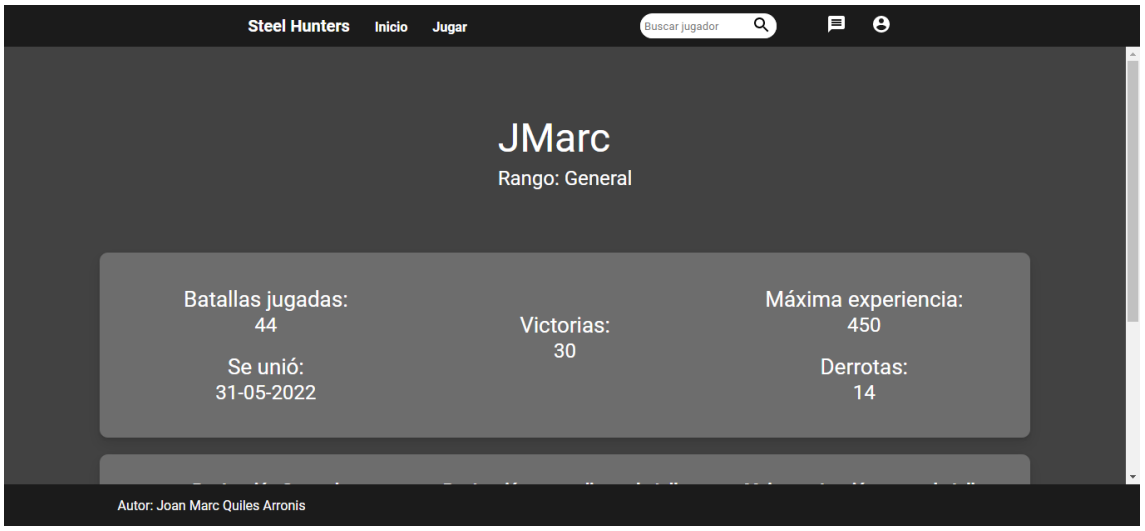

Figura 4.24.- Página de perfil de usuario

Figura 4.25.- Página de ajustes de cuenta. Datos generales.

Figura 4.26.- Página de ajustes de cuenta. Cambio de nombre.

Figura 4.27.- Página de ajustes de cuenta. Cambio de contraseña

Figura 4.28.- Página de ajustes de cuenta. Cambio de email.

Figura 4.29.- Página de ajustes de cuenta. Eliminar cuenta.

Figura 4.30.- Ventana del juego. Pantalla de inicio.

Figura 4.31.- Ventana del juego. Pantalla de carga.

Biblioteca
UNIVERSITAS Miguel Hernández

Figura 4.32.- Ventana del juego. Menú de selección y tanque desbloqueado.

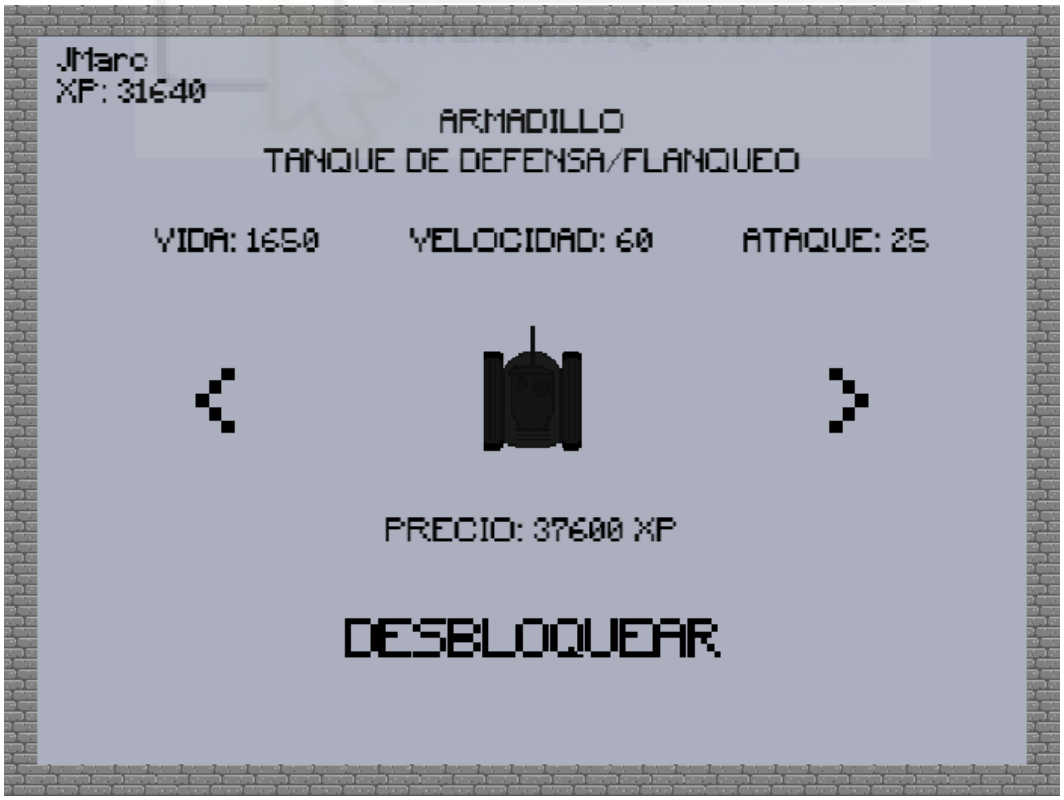

Figura 4.33.- Ventana del juego. Menú de selección y tanque bloqueado.

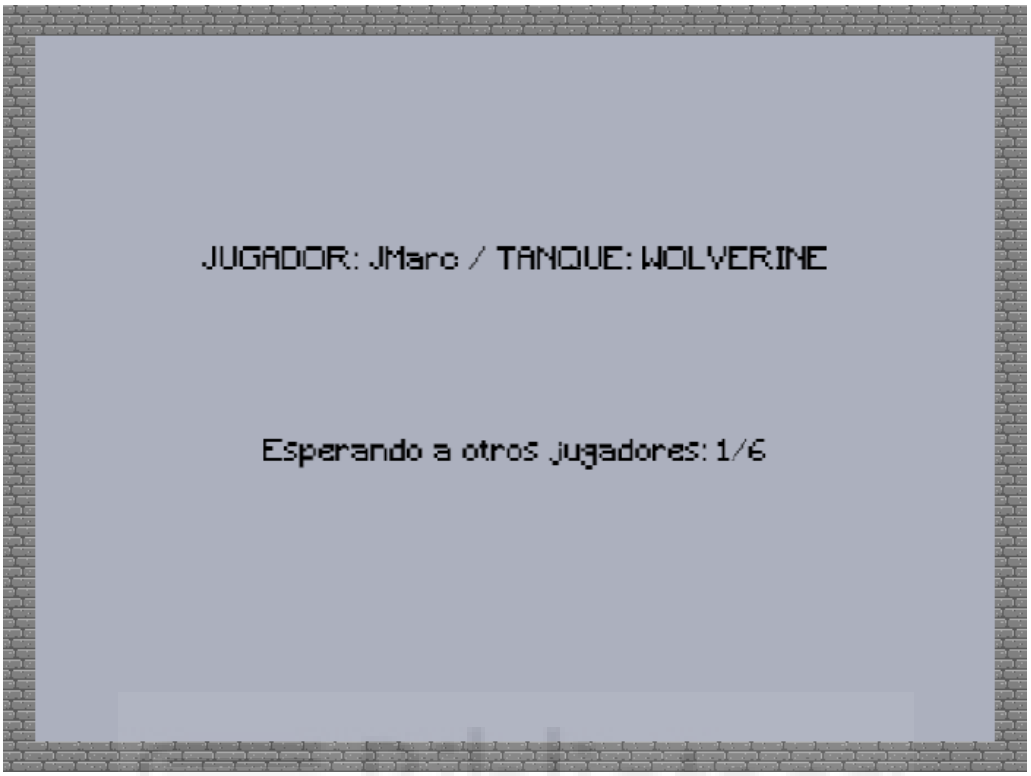

Figura 4.34.- Ventana del juego.Matchmaking.

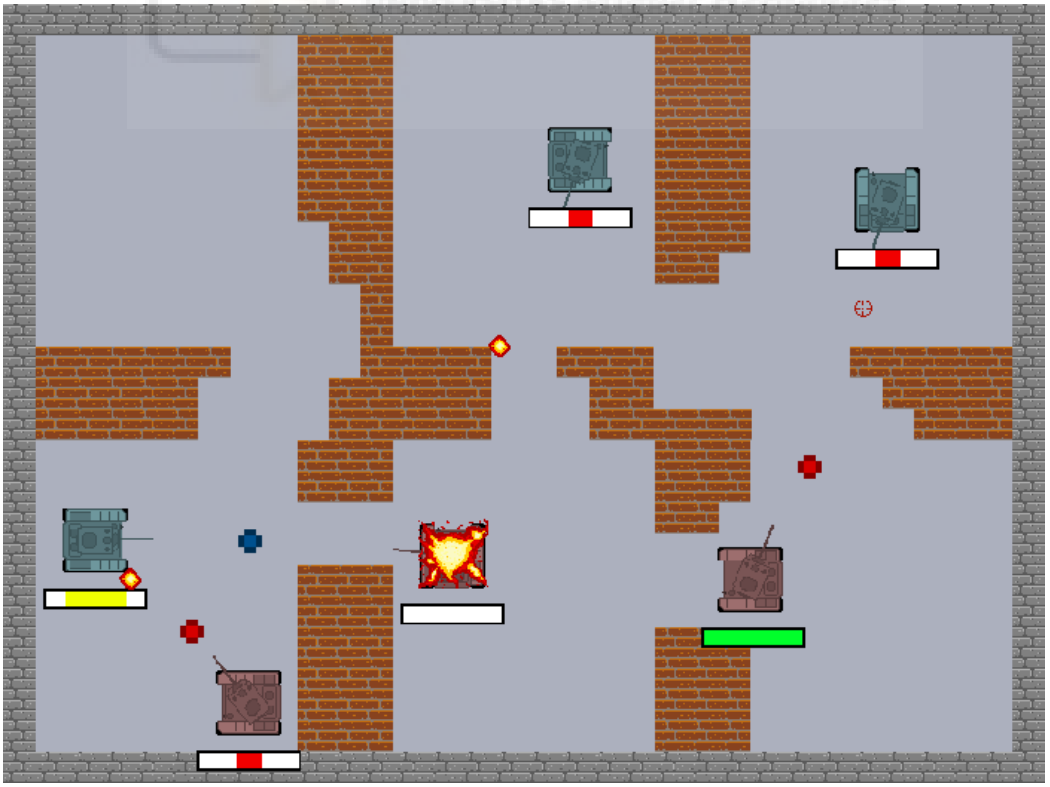

Figura 4.35.- Ventana del juego. Batalla.

Figura 4.36.- Ventana del juego. Fin de partida.

Figura 4.37.- Página del chat.

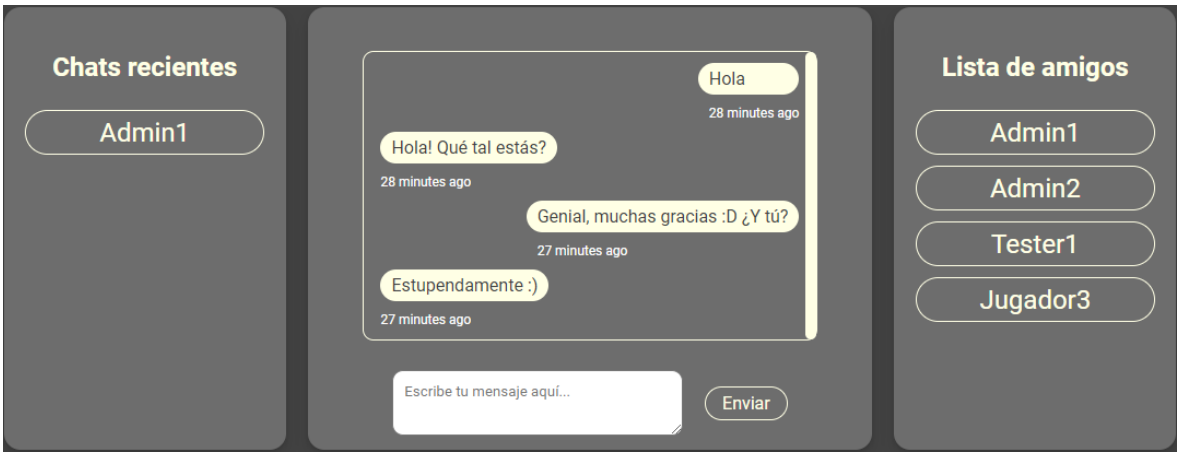

Figura 4.38.- Página del chat con una conversación abierta.

4.5.- IMPLANTACIÓN

Dado que el proyecto consiste en una aplicación web y un juego ejecutado en navegador, tan sólo se necesita un servidor que ejecute el programa. Para el desarrollo y las pruebas, se ha usado un servidor en localhost, que permite acceder a la web a los usuarios conectados en la misma red de área local. En caso de que se comercializara, tan sólo haría falta contratar un servicio de *hosting*, subir la aplicación al servidor y un dominio.

Biblioteca
UNIVERSITAS Miguel Hernández

Capítulo 5
Conclusiones y trabajo futuro

5.1.- CONCLUSIONES

Este proyecto es fruto del interés personal por los videojuegos. De acuerdo con esta motivación, el objetivo principal, conforme se indica en la introducción, era el aprendizaje de las tecnologías necesarias para desarrollar e integrar un videojuego multijugador en línea dentro de una aplicación web.

El resultado de tales aspiraciones y del conocimiento adquirido durante el desarrollo de este proyecto es ***Steel Hunters***, una aplicación web que es, al mismo tiempo, una red social y un videojuego multijugador en línea. Las funcionalidades implementadas para el apartado social de la aplicación son los perfiles de usuario donde ver la información de cada jugador, la posibilidad de seguir (es decir, hacerse amigo) de otro usuario, las conversaciones en tiempo real y la gestión de usuarios por parte de los administradores.

El juego, aunque gráficamente sencillo y sin ningún tipo de narrativa, se centra en la jugabilidad y la función multijugador. Además del menú donde cada jugador puede ver los recursos que tiene y los tanques disponibles, el apartado más importante es el combate. La dinámica implementada para las batallas es la siguiente: los usuarios, divididos en 2 equipos de 3 jugadores, pueden mover el tanque por el mapa y disparar a los enemigos. Por cada proyectil que haya impactado, excepto los disparos de los aliados, se le restan puntos de vida al tanque hasta su destrucción. La partida termina cuando un equipo acaba todos los vehículos enemigos.

El proyecto es completamente funcional, pero de contenido limitado. Como se ha visto a lo largo de la memoria, para el desarrollo de un videojuego entran muchas áreas de trabajo (diseño gráfico, programación, sonido, interfaces…), imposibles de abarcar por una sóla persona. Por tanto, para comercializar el producto, sería esencial contratar a un equipo de profesionales, cada uno especializado en una disciplina distinta.

En el supuesto de tener un equipo de desarrollo, para ayudar a la evolución el juego y aportar *feedback*, se ha creado un rol llamado *tester* que se encarga de probar el contenido nuevo o cambios al existente, para asegurar que, al sacar la actualización al público general, no existan *bugs* (errores o funcionamiento no previsto del juego) y que el equilibrio entre los tanques sea adecuado, es decir, que ningún vehículo sea excesivamente mejor o peor que los demás.

Los videojuegos, aparte de ofrecer entretenimiento, facilitan las relaciones entre personas. Este aspecto social está cada vez más presente en la industria y se puede ver en las distintas plataformas de streaming o en las tiendas digitales más populares [4] [5] [6] [7] [8]. Según el artículo *Gaming: The Next Super Platform*, el 84% de las personas veía los videojuegos como una manera de conectar con la gente [3]. ***Steel Hunters*** al fusionar la red social con los videojuegos, está especialmente enfocado a este sector de la industria.

5.2.- POSIBLES DESARROLLOS FUTUROS

5.2.1.- Red Social

El proyecto tiene dos ramas por las que puede evolucionar. La primera es la red social. Una posible mejora sería la customización de los perfiles. Actualmente la página de cada usuario sólo muestra su nombre y las estadísticas del juego, pero se podrían añadir imágenes y descripciones personalizadas.

Otra mejora que fomenta la relación entre usuarios y la competitividad en el juego, es la posibilidad de grabar las batallas, que los jugadores puedan publicarlas en sus perfiles y que los seguidores puedan reaccionar. Así, en caso de que un usuario rinda especialmente bien en una batalla, lo puede compartir y presumir del resultado.

5.2.2.- Juego

En cuanto al juego, el margen para mejorar es muy elevado. El primer paso sería añadir más vehículos y escenarios, que aporten variedad al juego. No sólo en el apartado visual, sino en el jugable. Se podrían crear vehículos con mecánicas únicas como dos cañones, armas orientadas al combate cercano como un lanzallamas (rango limitado, pero mucho daño). En cuanto a los campos de batalla, además del cambio en la posición de los obstáculos, se podrían diseñar zonas móviles o que vayan cambiando a lo largo del combate. Esto añadiría dinamismo y un componente estratégico al juego.

Las tecnologías utilizadas para el desarrollo del proyecto están orientadas a la creación de aplicaciones multiplataforma. Dado que uno de los sectores que más jugadores tiene y más ingresos genera es el de los dispositivos móviles, se podría modificar el juego para hacerlo compatible con Android e iOS. Aunque los controles táctiles habría que crearlos de cero, gran parte del código se podría reutilizar.

Por último, una modificación que sí conllevaría el desarrollo de mucho código y la utilización de tecnologías distintas a las empleadas en el proyecto, sería el paso de gráficos 2D a 3D. No sólo sería necesario rehacer las físicas, las texturas (el equivalente a los sprites, pero en 3D) y los escenarios, sino que habría que cambiar los controles y la cámara.

Biblioteca
UNIVERSITAS Miguel Hernández

Bibliografía

[1] EOM - La evolución del mercado de los videojuegos
https://elordenmundial.com/mapas-y-graficos/evolucion-mercado-videojuegos/
14/07/2022

[2] Mordor Intelligence - Gaming Marke
https://www.mordorintelligence.com/industry-reports/global-gaming-market
14/07/2022

[3] Accenture - Gaming: The next super platform
https://www.accenture.com/us-en/insights/software-platforms/gaming-the-next-super-platform
15/07/2022

[4] Steam - Juegos
https://store.steampowered.com/games/?l=latam#p=0&tab=ConcurrentUsers
14/07/2022

[5] Epic - Más populares
https://store.epicgames.com/es-ES/collection/most-popular
14/07/2022

[6] Play Store
https://play.google.com/store/games?device=phone
14/07/2022

[7] Sensor Tower - Top Grossing Mobile Games Worldwide for June 2022
https://sensortower.com/blog/top-mobile-games-by-worldwide-revenue-june-2022
14/07/2022

[8] Twitch - Explorar categorías (nº espectadores orden descendiente)
https://www.twitch.tv/directory
14/07/2022

[9] Definición.de - Definición de videojuego
https://definicion.de/videojuego/
18/07/2022

[10] Computer Hope - Browser-Based game
https://www.computerhope.com/jargon/b/browserbased-game.htm
18/07/2022

[11] Techopedia - Arcade Game
https://www.techopedia.com/definition/1903/arcade-game
18/07/2022

[12] Game Informer - How Flash games changed video game history
https://www.gameinformer.com/2018/12/22/how-flash-games-changed-video-game-history
18/07/2022

[13] Wikipedia - Adobe Flash Player
https://es.wikipedia.org/wiki/Adobe_Flash_Player
18/07/2022

Biblioteca
Universitas Miguel Hernández

[14] Polygon - The rise and fall of Flash gaming, explained
https://www.polygon.com/2017/7/8/15942194/flash-video-games-where-are-they-now
18/07/2022

[15] MacRumors - Adobe is officially dead after 25 years
https://www.macrumors.com/2021/01/12/adobe-flash-era-is-officially-over/
18/07/2022

[16] Future - The future of games is instant
https://future.com/instant-games/
18/07/2022

[17] GeoGuessr
https://www.geoguessr.com/
18/07/2022

[18] Jomesa - Historia de las máquinas recreativas
https://jomesa.es/historia-de-las-maquinas-recreativas/
19/07/2022

[19] Pixfans.com - Máquinas Arcade: Historia y Evolución
https://pixfans.com/maquinas-arcade-historia-y-evolucion/
19/07/2022

[20] PowerUps - Máquinas recreativas: Auge y decadencia de un sector casi extinto
https://powerups.es/maquinas-recreativas-auge-y-decadencia-de-un-sector-casi-extinto/
19/07/2022

[21] MAME - About
https://www.mamedev.org/about.html
19/07/2022

[22] RetroPie
https://retropie.org.uk/
19/07/2022

[23] Github - Microsoft/VSCode
https://github.com/Microsoft/vscode/
20/07/2022

Biblioteca
Universitaria Miguel Hernández

[24] Visual Studio Code
https://code.visualstudio.com/docs
20/07/2022

[25] Notepad++
https://notepad-plus-plus.org/downloads/v8.4.2/
20/07/2022

[26] Wikipedia - Sublime Text
https://es.wikipedia.org/wiki/Sublime_Text
20/07/2022

[27] Github - Brackets
https://github.com/brackets-cont/brackets
20/07/2022

[28] Github - Phaser
https://github.com/photonstorm/phaser
20/07/2022

[29] Open Webinars - Qué es Unity
https://openwebinars.net/blog/que-es-unity/
20/07/2022

[30] Unity - Plan Unity Student
https://unity.com/es/products/unity-student
20/07/2022

[31] Unreal Engine - FAQs
https://www.unrealengine.com/en-US/faq
20/07/2022

[32] Unreal Engine - Tools for General Platform Support
https://docs.unrealengine.com/5.0/en-US/sharing-and-releasing-projects-for-unreal-engine-5/
20/07/2022

[33] Unreal Engine - Pixel Streaming
https://docs.unrealengine.com/5.0/en-US/pixel-streaming-in-unreal-engine/
20/07/2022

Biblioteca
Universidad Miguel Hernández

[34] Github - React
https://github.com/facebook/react
20/07/2022

[35] Vue - Introduction
https://vuejs.org/guide/introduction.html#what-is-vue
21/07/2022

[36] Angular
https://angular.io/
21/07/2022

[37] Node.js - About Node.js
https://nodejs.org/en/about/
21/07/2022

[38] Node.js - The Node.js Event Loop
https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/
21/07/2022

[39] Wikipedia - Apache HTTP Server
https://en.wikipedia.org/wiki/Apache_HTTP_Server#Performance
21/07/2022

[40] Hostinger - ¿Qué es MySQL? Explicación detallada para principiantes
https://www.hostinger.es/tutoriales/que-es-mysql
21/07/2022

[41] MariaDB
https://mariadb.org/
21/07/2022

[42] Adobe - Photoshop
https://www.adobe.com/es/products/photoshop/free-trial-download.html
21/07/2022

[43] GIMP
https://www.gimp.org/
21/07/2022

[44] Inkscape - About
https://inkscape.org/about/
21/07/2022

[45] Audacity
https://www.audacityteam.org/
21/07/2022

[46] Adobe - Audition
https://www.adobe.com/es/products/audition.html
21/07/2022

[47] Desarrolloweb.com - HTML
https://desarrolloweb.com/home/html#track3
25/07/2022

[48] MDN Web Docs - CSS
https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/CSS_basics
25/07/2022

[49] MDN Web Docs - JavaScript
https://developer.mozilla.org/en-US/docs/Web/JavaScript
25/07/2022

[50] MDN Web Docs - Fundamentos de JavaScript
https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics
25/07/2022

[51] Visual Studio Code - JavaScript in Visual Studio Code
https://code.visualstudio.com/docs/languages/javascript
25/07/2022

[52] Universidad de Alicante - Modelo Vista Controlador (MVC)
https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html
26/07/2022

Biblioteca
Universitas Miguel Hernández

[53] BBVA - API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos
https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/
26/07/2022

[54] Red Hat - ¿Qué es una API de REST?
https://www.redhat.com/es/topics/api/what-is-a-rest-api
26/07/2022

[55] Digital55 - ¿Qué son las Single-Page-Application? El desarrollo elegido por Gmail y LinkedIn
https://digital55.com/que-son-single-page-application-spa-desarrollo-elegido-por-gmail-linkedin/
26/07/2022

[56] OpenWebinars - MERN Stack: Qué es y qué ventajas ofrece
https://openwebinars.net/blog/mern-stack-que-es-y-que-ventajas-ofrece/
26/07/2022

[57] Genbeta - MongoDB: qué es, cómo funciona y cuándo podemos usarlo
https://www.genbeta.com/desarrollo/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no
26/07/2022

[58] Stack Overflow - What is Express.js?
https://stackoverflow.com/questions/12616153/what-is-express-js
26/07/2022

[59] Node.js - Introduction to Node.js
https://nodejs.dev/learn
26/07/2022

[60] Open Webinars - Qué es NPM y para qué sirve
https://openwebinars.net/blog/que-es-node-package-manager/
26/07/2022

[61] NPM - Nodemon
https://www.npmjs.com/package/nodemon
26/07/2022

[62] códigofacilito - Qué es Mongoose
https://codigofacilito.com/articulos/que-es-mongoose
26/07/2022

[63] Axios
https://axios-http.com/docs/intro
26/07/2022

[64] JavaScript.info
https://javascript.info/promise-basics
26/07/2022

[65] Socket.io - Introduction
https://socket.io/docs/v4/
27/07/2022

[66] ITM Platform - Ciclos de vida clásico, iterativo y ágil
https://www.itmplatform.com/es/blog/ciclos-de-vida-clasico-iterativo-y-agil
29/07/2022

Biblioteca
UNIVERSITAS Miguel Hernández