UNIVERSIDAD DE OVIEDO

MÁSTER UNIVERSITARIO EN INGENIERÍA WEB

TRABAJO FIN DE MÁSTER

"Age of Code: Desarrollo de un Videojuego para aprender a Programar"

DIRECTOR: Begoña Cristina Pelayo García Bustelo

CODIRECTOR: Cristian González García

AUTOR: Gabriel Manteca Muñoz

Agradecimientos

Quiero aprovechar esta sección para dar las gracias a mis directores por su apoyo, así como a mis compañeros que han estado conmigo a lo largo del desarrollo de este proyecto.

Muchas gracias a todos, ha sido una gran experiencia.

Resumen

Se ha desarrollado un juego con el propósito de facilitar el aprendizaje de un lenguaje de programación moderno. Se pretende que el juego ayude a desarrollar un pensamiento más programático e introducir esquemas típicos de programación que puedan ayudar a un usuario con su desarrollo como programador.

El tipo de juego propuesto es de estrategia en tiempo real, es decir, en el videojuego no se dispondrá de tiempos límite ni turnos para ejecutar acciones sino que estas se ejecutarán conforme se reciban. Debido a esto, el tiempo avanzará continuamente para los enemigos, que podrán atacar a los personajes del usuario en cualquier momento.

Para la realización de este proyecto se ha desarrollado una aplicación web que de forma gráfica permita la interacción del usuario con el juego. Para la implementación se han empleado tecnologías actuales como *AngularJS* y *WebGL*, que permite la representación de gráficos 2D y 3D en un navegador gráfico moderno.

Dado que el concepto del juego propuesto se puede extender de forma muy amplia, se ha propuesto la creación de una base que pueda ser extendida en un futuro. El principal objetivo será crear una serie de esqueletos que muestren una funcionalidad mínima y que ayuden a la futura ampliación del juego.

Para la realización de este proyecto, será necesario definir y crear una gramática que soporte el lenguaje implementado en el juego. Además, se hace necesaria la implementación de una serie de comprobaciones que ayuden al usuario a detectar sus errores y que le especifiquen en que se equivocó, así como por parte del juego detectar si es correcta o no la programación realizada por el usuario.

En el juego se permite el uso de estructuras de control programáticas y diferentes acciones sobre los objetos del juego que permiten, por ejemplo, interactuar con el mundo mediante movimientos, construcción, recolección de recursos y ataque a enemigos, entre otras acciones.

Palabras Clave

Videojuego, Videojuego de estrategia en tiempo real, Python, TypeScript, Programación, Web, WebGL, AngularJS.

Abstract

The objective of the creation of this game is to make easier the learning of a modern programming language. The game should help to develop a more programmatic thinking and introduce some typical schemes used during development so it can be of use with the user development as a programmer.

The type of game proposed is Real-Time Strategy, it will not have a turn system or time limit to execute actions; instead, these are executed as they are received. In the same way, the enemies can attack the user's characters in real time, and can even kill them if the user does not take the opportune actions.

With these objectives in mind, a web application that allowed the interaction with the user graphically was created. For the implementation the latest technologies were used, some of them being *AngularJS* and *WebGL*. The first one allows the automatic refresh of the site and some modern directives presented in one-page applications and the another one allows the representation of 2D and 3D graphics in a modern web browser.

Given the broad concept of this game, it was proposed to create an extensible base, to be expanded in the future. With this, the objective was to create skeletons with a minimal functionality but extensible without the necessity of rewriting any actual code.

For the realization of this project, it will be necessary to define and create a grammar supporting the implemented language. Also, it will be necessary the creation of checks that help the user to detect its errors and help him to fix them, and detect if the code is correct as part of the game.

The game permits the use of programming control structures and different actions on the game objects. These actions supports, for example, interactions with the world using movements, construction of new objects, mining of resources, attack enemies…

Keywords

Videogame, Real-time strategy, Python, TypeScript, Programming, Web, WebGL, AngularJS.

Índice General

Índice de Figuras

Índice de Código Fuente

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Este proyecto se ha desarrollado principalmente con el objetivo de cubrir la necesidad actual de enseñar a programar a todos los estratos de la población, mediante juegos y métodos más amigables que los tradicionales libros y manuales, pues puede observarse el resultado directo del código introducido y facilita el acercamiento de más sectores de la sociedad.

Si bien el objetivo del proyecto no parece excesivamente ambicioso, resulta serlo bastante pues se ha de asegurar todo lo posible para evitar posibles vulnerabilidades y agujeros de seguridad.

Para este desarrollo se crearán distintas partes, principalmente *backend* y *frontend* como todo servidor web y se tratará de hacer lo más seguro y fiable posible, así como intentar que su rendimiento sea notable.

Debido al requisito de seguridad, todo el código introducido por el usuario jamás se ejecutará directamente ni en *backend* ni en *frontend*, sino que se analizará en *backend* y se generará un código cerrado y seguro ya definido anteriormente.

Igualmente debido al requisito de aprendizaje, se tratará de detectar la mayor cantidad de errores en tiempo de análisis en la parte *backend* de la aplicación y otorgar la mayor cantidad de información sobre los mismos al usuario para facilitar su corrección.

Otro requisito es que, debido a que el código finalmente se ejecuta en *frontend*, será necesario cerrar todo lo posible su ejecución para que no pueda ser fuente de ataques o *exploits*, por este motivo, se crearán una serie de plantillas que ejecuten el código de forma limitada evitando los *exploits*, en la medida de lo posible.

Cabe destacar que debido a la jugabilidad que presenta el juego, se ha creado una arquitectura fácilmente ampliable mediante la herencia de determinadas clases, facilitando futuras expansiones en determinados campos como nuevos personajes, edificios o enemigos.

No obstante, el juego actualmente presenta una jugabilidad mínima con unos personajes, enemigos, objetos, edificios, etc., permitiendo que los usuarios puedan interactuar con el juego sin necesidad de ampliaciones en su arquitectura.

Finalmente y para facilitar que el usuario se familiarice con el entorno, se desarrollará un sistema de tutoriales que permitan que el usuario se habitúe a los elementos de programación del lenguaje y se le propondrá algún reto de desarrollo básico.

1.2 Resumen de Todos los Aspectos

1.2.1 Introducción

La cantidad de usuarios que consideran que aprender a programar es algo básico que debería impartirse en los niveles de educación obligatorios va en aumento, igualmente los juegos que pretenden ayudar a dicha formación también está creciendo como se muestra en las alternativas analizadas en esta memoria.

Igualmente, la mayor parte de estos juegos se encuentran disponibles de forma gratuita a través de Internet, ya que solo requieren un navegador moderno y visitar una página web con un usuario registrado para poder mantener su progreso.

Siendo así, se decidió crear este proyecto con la ambición de ir un paso más allá y permitir su expansión futura con un mundo abierto que permitiese al usuario explorar más allá que los pasos de un tutorial guiado, los tipos de juego existentes o un mero reto intelectual.

Finalmente, se ha optado por realizar una jugabilidad basada en los videojuegos de estrategia en tiempo real siendo, hasta donde se ha podido averiguar, el primero en mezclar este tipo de juego con los videojuegos para aprender a programar.

1.2.2 Análisis y diseño

Como se especificará más adelante el principal objetivo del sistema es el desarrollo de un videojuego que facilite el aprendizaje de un usuario en cuanto a programación.

Para ello se pretende cumplir los siguientes objetivos:

1. Registro de nuevos usuarios.
2. *Login* seguro de nuevos usuarios.
3. Modificaciones de los datos de los usuarios.
4. Borrado de los datos de los usuarios.
5. Dibujado del estado de la partida.
6. Reconocimiento de expresiones del lenguaje.
7. Ejecución de las expresiones del lenguaje.
8. Persistir correctamente los datos del juego.
9. Mundo con el que interactuar.
10. Recursos disponibles en el mundo.
11. Tutorial de juego explicando sus funcionalidades.
12. Aplicación web disponible 24 horas al día, 7 días a la semana.
13. Tiempo de respuesta del servidor *backend* aceptable.
14. Juego accesible desde cualquier navegador moderno.

1.2.3 Implementación

El sistema desarrollado empleará tecnologías web *backend* y *frontend* y tratará de ser lo más estándar, seguro y eficiente posible.

Por ello se ha seleccionado Python como tecnología de desarrollo en *backend* junto a la base de datos MySQL y el servidor Apache como *proxy* y HTML, CSS y TypeScript como tecnología de desarrollo en el *frontend*.

Durante el desarrollo se utilizó el IDE PyCharm para codificar la parte *backend* y el editor Atom para la parte *frontend*.

1.2.4 Pruebas

Las pruebas se ejecutarán siempre tras finalizar cada módulo usando una metodología ágil, arreglando todos aquellos desperfectos que estas detecten antes de cerrar todos y cada uno de ellos.

Las pruebas de accesibilidad se realizaron al finalizar la mayor parte de los módulos, debido a que unos dependían de otros por lo que era complejo realizar una prueba fiable, no obstante se realizó el diseño intentando que fuera accesible desde el primer momento.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

La creación de este proyecto viene justificada por el auge actual de la programación sobre todo entre los jóvenes con eventos como Hour of Code **[CO16].**

Basándonos en premisas como la anterior, una de las mejores formas de facilitar el aprendizaje de la programación es a través de juegos. En este aspecto también se trata de enfocar a jugadores más adultos mediante una jugabilidad de tipo estrategia en tiempo real similar al videojuego Age of Empires.

La elección de este juego como modelo a seguir es por el éxito que tuvo en su día y el rotundo éxito que sigue teniendo a día de hoy con records de ventas y de jugadores en plataformas como Steam **[VC16]**.

Para hacer que el juego sea accesible a la mayor cantidad de gente posible se ha optado por realizar el proyecto orientado completamente a la web. Otro de los motivos para esta decisión frente a la actual hegemonía de los teléfonos móviles es la universalidad de la web y la dificultad añadida que puede suponer desarrollar código en un dispositivo móvil.

El juego soportará, entre otras cosas, el reconocimiento de sentencias de un lenguaje moderno y ejecución de las mismas, el soporte a registro, modificación y eliminación de usuarios, etc.

2.2 Objetivos del Proyecto

El objetivo del proyecto es desarrollar la aplicación web descrita en la justificación anterior junto al análisis, diseño y las pruebas de la misma, así como esta documentación y el oportuno mantenimiento. Los objetivos desglosados son los que siguen:

1. Registro de nuevos usuarios.
2. *Login* seguro de nuevos usuarios.
3. Modificaciones de los datos de los usuarios.
4. Borrado de los datos de los usuarios.
5. Dibujado del estado de la partida.
6. Reconocimiento de expresiones del lenguaje.
7. Ejecución de las expresiones del lenguaje.
8. Persistir correctamente los datos del juego.
9. Mundo con el que interactuar.
10. Recursos disponibles en el mundo.
11. Tutorial de juego explicando sus funcionalidades.
12. Aplicación web disponible 24 horas al día, 7 días a la semana.
13. Tiempo de respuesta del servidor *backend* aceptable.
14. Juego accesible desde cualquier navegador moderno.

2.3 Estudio de la Situación Actual

Actualmente existe un número considerable de videojuegos cuyo principal objetivo es enseñar a programar al usuario bien mediante el uso de un lenguaje de programación escribiendo sentencias o bien mediante controles gráficos, que intente enseñar la mentalidad programática que todo programador ha de poseer.

En cuanto a los primeros, CodeCombat es una aplicación web que se basa en este principio, dispone de un lenguaje de programación similar a JavaScript y su objetivo es completar una serie de tutoriales y puzles **[CC16].**

En cuanto a los segundos existen distintos lenguajes gráficos como puede ser LightBot o RoboZZle donde pueden aplicarse distintos acercamientos típicos de programación para resolver problemas **[Ostrovsky09] [NoAuthor16]**.

2.3.1 Evaluación de Alternativas

2.3.1.1 CodeCombat

CodeCombat dispone de una sencilla interfaz web cuyo principal cometido es resolver un puzle programáticamente. Una de las ventajas más atractivas de esta aplicación es que dispone de una visualización gráfica en 2D de la situación del programa y del avance del mismo así como el uso de un lenguaje de programación real.

En los primeros niveles se intenta centrar en la sintaxis básica del lenguaje así como pequeños tutoriales que pretenden ayudar al usuario a familiarizarse con el uso de su personaje y los distintos paradigmas disponibles como bucles, condiciones, etc.

Las principales ventajas a parte de una gran cantidad de niveles es el sistema de misiones y bonificaciones por usar un código optimizado. También cabe mencionar el hecho de lo cuidado de su estética y lo fácil que resulta engancharse a resolver misiones en el juego.

La principal desventaja es lo cerrado del mundo, pues más allá de pequeñas fases de puzle, el juego no abarca la posibilidad de combatir contra otros usuarios, formar alianzas o controlar un grupo de héroes.

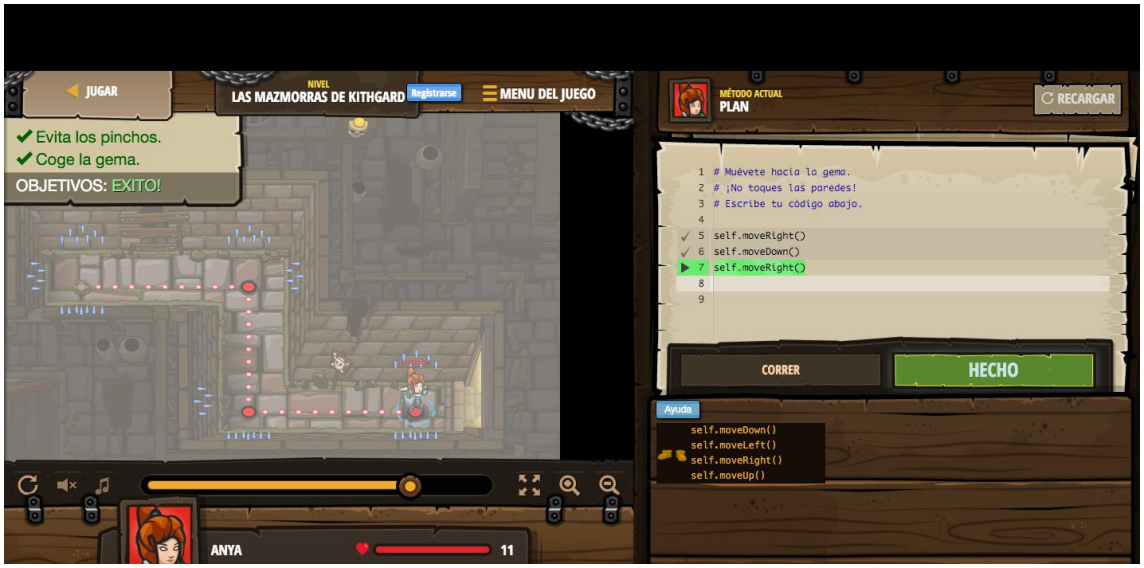


Figura 2.1 CodeCombat

2.3.1.2 LightBot

LightBot se trata de un pequeño juego disponible para distintas plataformas con versión web basada en flash cuyo objetivo es iluminar ciertas baldosas mediante la programación de un pequeño robot mediante acciones simples. Detrás de esta sencillísima premisa se encuentra un juego lleno de posibilidades que enseña a planificar antes de realizar acciones así como tener visión suficiente para realizar las mismas de forma eficiente ya que el espacio de las mismas es limitado.

La principal ventaja del videojuego es la aparente sencillez del mismo y lo poco que cuesta hacerse con el control del juego. Igualmente la sencillez aparente se convierte en complejidad una vez los puzles han de ser resueltos de forma óptima.

La mayor desventaja es que este tipo de juego a pesar de transmitir una mentalidad programática no hace que un usuario se enfrente al código, al igual que se dispone de una jugabilidad excesivamente limitada.

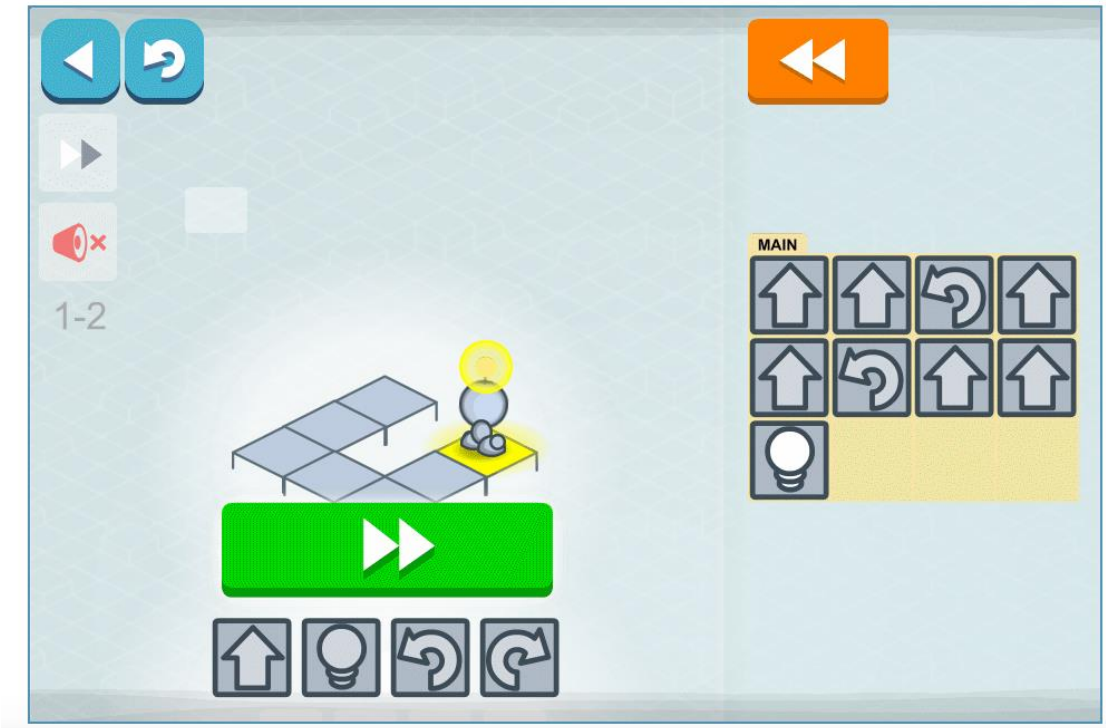


Figura 2.2 LightBot

2.3.1.3 RoboZZle

Al igual que en el caso anterior RoboZZle traza los movimientos de un objeto sobre el tablero de juego cuyo objetivo es recoger todos los elementos del mismo sin dejar ninguno. A diferencia de LightBot, RoboZZle persigue una mentalidad más programática, enfocando al usuario en funciones repetibles, bucles, recursividad, etc.

La principal ventaja es su aparente sencillez que se complica empleando bucles y recursividad.

Al igual que en el caso de LightBot, RoboZZle no es capaz de hacer que el usuario se enfrente a código real, si bien se encuentra considerablemente más cerca de la mentalidad programática de un desarrollador.

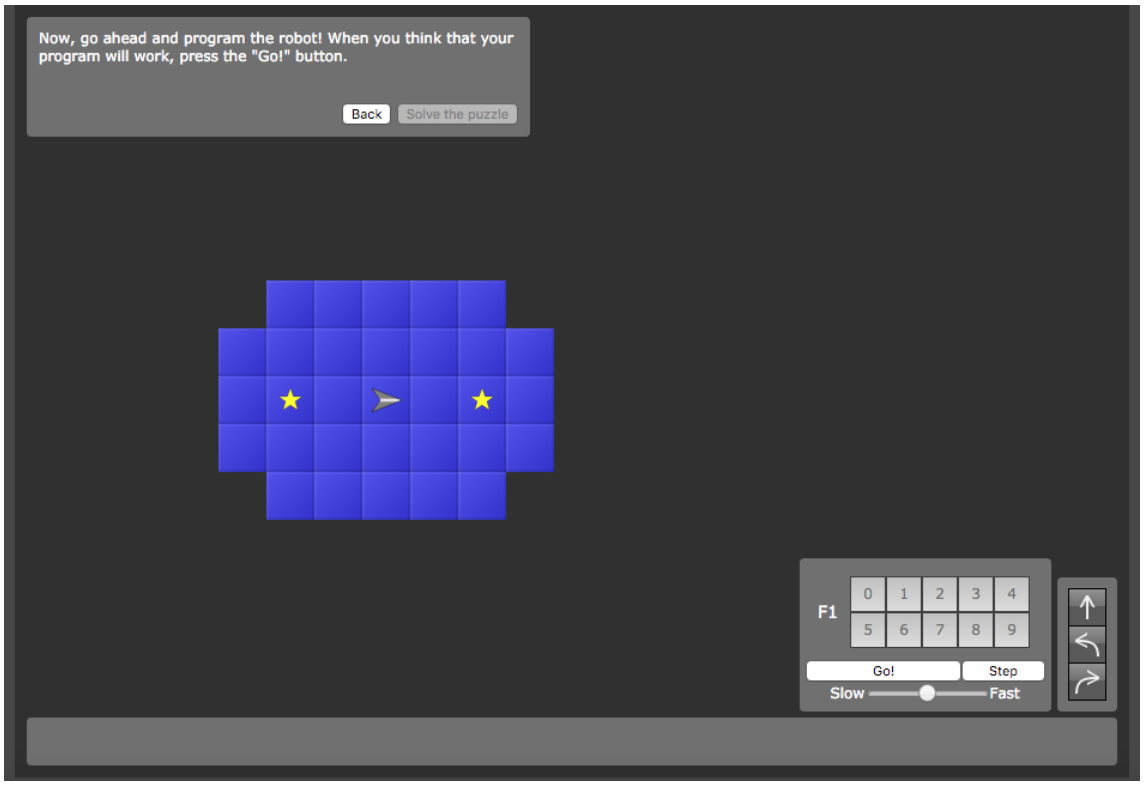


Figura 2.3 RoboZZle

2.3.1.4 Logo

Logo se menciona a nivel histórico ya que fue de los primeros lenguajes para ayudar a que el usuario obtenga una mentalidad de programador.

Logo es el famoso “lenguaje de la tortuga”, el lenguaje se basa en hacer que una tortuga se mueva por la pantalla y dibuje una línea allá por donde pase. Este lenguaje se basa en instrucciones sencillas que todo lo que hacen es mover la tortuga por la pantalla.

Este lenguaje ha inspirado otros lenguajes modernos entre los cuales quizás cabe destacar Scratch, desarrollado por el MIT y está influenciado a su vez por LISP.

En cuanto a sus ventajas cabe destacar la simplicidad detrás de su idea y la terrible complejidad que puede llegar a alcanzar una vez dominado.

En cuanto a sus desventajas, la más obvia es que no es un juego en sí y que no sirve para mucho más que dibujar gráficos.

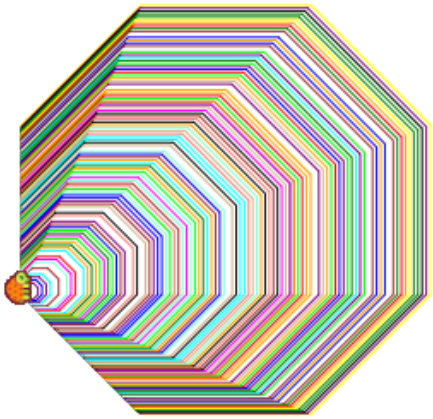


Figura 2.4 Logo

Capítulo 3. Aspectos Teóricos

3.1 Servidor Backend

El servidor *backend* es un programa ejecutado en el servidor web al que el usuario únicamente puede lanzar consultas desde *frontend*. La principal diferencia con el *frontend* de la aplicación es que el usuario no puede ver directamente el *backend* y tampoco puede modificarlo.

La principal función de este servidor *backend* es servir las paginas *frontend* que verá posteriormente el usuario en su navegador, así como acceder a la base de datos del sistema y procesar la lógica de la aplicación. Entre esta lógica se encuentra la gestión de los usuarios o la jugabilidad del sistema desarrollado.

3.2 Frontend

El *frontend* es la parte visual de la aplicación que el usuario puede ver e interactuar con ella. La principal función de esta parte de la aplicación es mostrar al usuario las funciones para interactuar con la aplicación así como las visualizaciones gráficas y la interfaz general de la aplicación.

El *frontend* soporta desde gráficos hasta funciones asíncronas así como un muy leve procesamiento no obstante se suele delegar todo lo que supone carga computacional a la parte *backend* de la aplicación.

Tradicionalmente el *frontend* se construye mediante HTML, CSS y JavaScript, usando este último para la escasa computación que se realiza en el mismo y para mejorar la usabilidad de la página.

3.3 Videojuego de estrategia en tiempo real

Un videojuego en tiempo real es aquel en el que el tiempo transcurre igual y de forma continua para todos los usuarios, es decir, no existen turnos ni pausas para tomar decisiones sino que todas han de ser tomadas en el momento.

Estos videojuegos están diseñados para ser jugados de forma rápida, obligando a que la toma de decisiones sea en el momento no permitiendo pausar el juego para pensar la estrategia.

Esto facilita bastante el sistema de implementación multijugador que ya se encuentra en muchos juegos, pues permite que las acciones puedan realizarse en el momento que el usuario las solicita.

Algunos ejemplos de este tipo de juegos son los Age of Empires o la saga Total War.

3.3.1 Age of Empires I y II

Age of Empires fue desarrollado por la empresa Ensemble Studios y distribuido por Microsoft Studios en 1997 en su primera versión para PC. Esta primera versión contó con una expansión y dos versiones posteriores, ambas con sus expansiones correspondientes.

Si hay algo que debe ser destacado de la franquicia Age of Empires fue su tremendo y rotundo éxito en la comunidad de videojuegos mundial. Consiguiendo ventas en todas y cada una de sus versiones, un total de 3, e incluso consiguiendo una remasterización en HD hace pocos años, la cual incluye dos nuevas expansiones.

En cuanto al juego en sí, su principal objetivo era derrotar a las unidades enemigas utilizando unidades amistosas y militares junto a distintas estrategias de guerra.

El juego también cuenta con un sistema de recursos similar al implementado en este juego y distintas unidades militares. No obstante este se encontraba bastante más avanzado que el juego desarrollado en este TFM, disponiendo de unidades navales, un sistema de niveles y un sistema de edades que permitían modernizar la civilización.

Figura 3.1 Age of Empires

3.4 Base de datos

Una base de datos es una aplicación normalmente ejecutada en la parte *backend* de la aplicación cuyo principal objetivo es el almacenamiento de información de forma estructurada usando abstracciones que faciliten su uso al desarrollador.

Las bases de datos suelen ser o bien relacionales o bien no relacionales, siendo las primeras las más utilizadas de forma tradicional y siendo las segundas las que últimamente están consiguiendo una cuota de mercado bastante elevada.

Siendo las primeras las más empleadas a día de hoy es relevante comentar su funcionamiento en mayor profundidad.

Las bases de datos relacionales se componen mediante estructuras denominadas tablas en las que se almacenan las diferentes estructuras de datos, cada estructura se introduce como una nueva fila de la tabla mientras que cada dato se encasilla dentro de una columna determinada.

Cada fila se identifica de forma única mediante una clave primaria, normalmente indicada en una columna o, posiblemente, varias. La restricción evidente de la clave primaria es que debe ser única en la tabla pues si se encontrase repetida no sería posible identificar la tabla.

La práctica totalidad de bases de datos relacionales usan el estándar de lenguaje SQL para realizar consultas sobre los datos, introducir datos nuevos, modificarlos o modificar la base de datos en general añadiendo nuevas tablas, usuarios o modificando permisos.

3.5 WebSockets

WebSocket es una tecnología web de comunicación cliente-servidor que proporciona un canal bidireccional full dúplex usando un socket TCP sobre la web. Esta tecnología de comunicación se encuentra implementada en la totalidad de los navegadores web modernos y la principal ventaja que dispone con respecto a otras tecnologías de comunicación es que permite que el servidor envíe datos al cliente en cualquier momento y sin que el cliente tenga que solicitarlos.

El protocolo WebSocket también soporta una versión segura que añade una capa de TLS a la comunicación, cifrándola y mejorando notablemente la seguridad.

Un detalle importante de esta tecnología es que el cliente debe iniciar la comunicación con el servidor siempre, es decir, el servidor en ningún caso puede iniciar la petición de creación de un WebSocket con un cliente. Siendo así, es imposible comunicar directamente a dos clientes mediante WebSockets.

3.6 WebGL

Figura 3.2 WebGL

WebGL es la aproximación que se ha realizado desde los estándares web a la inclusión de gráficos 3D dentro de un navegador y a través de un sitio web. El objetivo es la creación de una API basada en la de OpenGL 2 que permita la visualización de gráficos 2D y 3D de forma sencilla y eficiente en un navegador web sin necesidad de usar un *canvas* con la API nativa de JavaScript.

Debido a la complejidad que sigue suponiendo el uso de WebGL en la actualidad, la comunidad de desarrollo ha creado numerosas bibliotecas que simplifican su uso en las tareas más comunes y repetitivas como en la animación de *sprites* o bien simplemente en el dibujado en el navegador.

Cabe destacar que el rendimiento de una aplicación en WebGL sigue siendo cuanto menos cuestionable dado que el navegador requiere una gran cantidad de capacidad de procesamiento para *renderizar* en tiempo real algo creado con esta API.

3.7 Metodologías

3.7.1 Métrica 3

Métrica 3 es una metodología de Planificación, Desarrollo y Mantenimiento de sistemas de la información que puede ser utilizada libremente con la única restricción de citar la fuente de su propiedad intelectual, es decir, el Ministerio de Hacienda y Administraciones Públicas.

Los objetivos de esta metodología de trabajo son:

- Tratar de proporcionar o definir sistemas de información que ayuden a conseguir los fines de la organización.
- Dotar a la Organización de productos software.
- Mejorar la productividad.
- Facilitar la comunicación y entendimiento.
- Facilitar la operación, mantenimiento y uso de los productos software obtenidos.

Este documento está desarrollado usando la metodología Métrica 3, pero de forma adaptada, contemplando solo aquellos apartados que se han considerado más adecuados para un trabajo de fin de máster.

3.7.2 SCRUM

SCRUM es una de las metodologías agiles modernas más usadas a día de hoy, su principal característica es el desarrollo en equipos de pequeño tamaño y el trabajo en periodos de tiempo cortos, normalmente de una o dos semanas, en los que al final del mismo se realiza una entrega parcial y se valida o rechaza el trabajo realizado. Igualmente se realizan reuniones de forma diaria donde se valora el trabajo que se está realizando y se toman decisiones al respecto del mismo.

Usando esta metodología se pretenden evitar imprevistos que puedan retrasar el proyecto, mejorar la productividad, realizar desarrollos rápidos y trabajar con el cliente para que el proyecto se adapte realmente a lo que él necesite.

La metodología funciona en iteraciones fijas de una semana a un mes, en el fin de cada iteración se ha de entregar un incremento del producto final que pueda ser evaluado por el equipo y el cliente.

Durante la iteración se producen reuniones diarias en las que se trata de sincronizar al equipo y solventar aquellos imprevistos que hayan surgido durante el transcurso de la misma.

3.8 Git

Figura 3.3 Git

Git es un sistema de control de versiones creado por Linus Torvalds (Linux) pensado para aplicaciones con un gran número de ficheros de código fuente.

Git se encuentra centrado en el trabajo a gran escala con una gran cantidad de desarrolladores trabajando en paralelo por lo que se trabaja con una gran cantidad de “ramas” en cada repositorio de código.

El trabajo en Git se hace de forma distribuida por lo que cada desarrollador tiene una copia del trabajo completo que posteriormente contribuye al proyecto global cuando se encuentre completada.

Como muchos de los sistemas de control de versiones se soportan las opciones de deshacer y se incluye soporte a la solución de conflictos.

3.9 Gramática de libre contexto

Una gramática de libre contexto se define como una 4-tupla de la siguiente forma:

$$G = (V_t, V_n, P, S)$$

Figura 3.4 Definición formal de gramática

Dónde:

- V_t es un conjunto finito de símbolos terminales.
- V_n es un conjunto finito de símbolos no terminales.
- P es un conjunto finito de reglas de producción.
- S se denomina símbolo inicial y forma parte de los símbolos no terminales.

Las reglas de P se crean de la siguiente forma:

$$V_n \longrightarrow (V_t \cup V_n)^*$$

Figura 3.5 Definición de regla de producción

Es decir, los símbolos no terminales pueden construirse mediante la unión de cero o más símbolos terminales con símbolos no terminales o bien cero o más símbolos terminales o símbolos no terminales. **[Cueva01]**

La notación más empleada para definir gramáticas es la Forma de Backus-Naur o más típicamente *Backus Naur Form* (BNF). Esta notación será la empleada para describir formalmente la gramática desarrollada.

Debido a como se definen las gramáticas, típicamente existen dos formas de comprobar si una cadena de entrada pertenece al lenguaje definido por la gramática.

La primera de ella es partiendo desde el símbolo inicial y aplicando reglas de producción, sustituyendo el símbolo no terminal de más a la izquierda siempre primero, recorriendo el árbol formado por la gramática de arriba a abajo, este tipo de derivación se denomina “derivación por la izquierda” o LL.

El segundo caso, partiendo desde el símbolo inicial y aplicando reglas de producción sustituyendo el símbolo no terminal de más a la derecha siempre primero, recorriendo el árbol de la gramática desde abajo hacia arriba, se denomina “derivación por la derecha” o LR.

Esta diferencia es crítica en un lenguaje de programación ya que determina el orden de ejecución de las sentencias introducidas.

En el caso de este proyecto se ha empleado una biblioteca que abstrae gran parte de este proceso empleando un *parseador* de tipo LR.

Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales

4.1 Backlog

A continuación se indican los módulos que se eligieron para realizar su desarrollo. Se indica como sombreado y subrayado cada módulo principal, mientras que se indica en cursiva los submódulos correspondientes al módulo. Los módulos contienen la suma total en horas de todos sus submódulos, en caso de tenerlos.

Módulo	**Prioridad**	**Duración**	**Detalle**
Investigar aplicaciones similares	10	16h	Para tomar ideas, pros y contras, mejoras, etc.
Investigar *frameworks frontend* y *backend*	10	16h	Encontrar un *framework frontend* para los gráficos 2D y otro *backend* para el servidor
Selección de la base de datos a emplear	10	4h	Seleccionar un Sistema Gestor de Bases de Datos para el proyecto
Construcción del servidor *backend*	9	32h	Definir el punto de entrada del WebSocket y los puntos de entrada REST
Instalación del entorno de ejecución	9	3h	Instalación del lenguaje y módulos que emplee el servidor *backend*
Instalación de la base de datos	9	2h	Instalación de la base de datos en el servidor *backend*
Configuración del entorno de ejecución y base de datos	9	3h	Configuración de los paquetes instalados anteriormente
Definición de los puntos de entrada de WebSocket y REST	9	8h	Definición de los puntos de entrada y construcción de la estructura básica que compondrá el servidor *backend*
Creación de la base de datos de la aplicación, usuarios y acceso a la misma	9	8h	Creación de la base de datos de la aplicación, los usuarios con acceso correspondiente y el acceso remoto oportuno en caso de ser necesario

Conexión del servidor backend con la base de datos mediante un conector	9	8h	Conectar el servidor *backend* con el servicio de base de datos mediante un conector disponible para el mismo o mediante los medios que se estimen oportunos
Construcción del *frontend* básico	9	24h	Conectar el *frontend* con el WebSocket y los puntos de entrada REST
Instalación del servidor que servirá el frontend	9	3h	Instalación y aseguración del servidor que servirá las paginas *frontend*
Configuración del servidor frontend	9	3h	Configuración del servidor *frontend* y aseguración del mismo
Codificación de la estructura HTML básica	9	8h	Codificación de los formularios y marcado HTML de las paginas *game*, usuario, *registro* y *login*
Creación de una hoja de estilo básica	9	10h	Creación de una hoja de estilo para las páginas ampliable
Definir el analizador léxico y sintáctico en *backend*	9	64h	Crear la gramática que reconozca el lenguaje en *backend*
Definición de las expresiones regulares que reconozcan los tokens del lenguaje	9	6h	Creación del analizador léxico en *backend* que reconozca los *tokens* y los separe
Definición de la gramática que reconozca el lenguaje	9	12h	Creación del analizador sintáctico del lenguaje que reconozca expresiones del mismo
Construcción del AST del lenguaje	9	12h	Creación del árbol sintáctico del lenguaje que almacene expresiones del mismo
Almacenamiento de las expresiones detectadas en el AST	9	8h	Almacenamiento de las expresiones detectadas por la gramática en el AST definido
Construcción del analizador semántico sobre el AST construido	9	22h	Reconocimiento semántico del lenguaje comprobando que los *lvalue* son correctos así como las ultimas comprobaciones
Pruebas de los analizadores con ejemplos del lenguaje	9	4h	Pruebas de reconocimiento usando muestras del lenguaje para comprobar su correcto funcionamiento
Definir el protocolo de comunicación del websocket	9	40h	Definir los distintos mensajes que soportará el protocolo

Definición de los tipos de mensajes a tratar en el protocolo	9	8h	Mensajes de código enviados desde el cliente, Mensajes de estado desde el servidor, mensajes de plantillas a ejecutar desde el servidor…
Definición de los campos a tratar dentro de cada tipo de mensaje	9	8h	Definir los campos que son necesarios en cada tipo de mensajes así como su formato
Codificación de la creación de mensajes de parte del servidor y recepción y tratado de mensajes del cliente	9	12h	Codificación en el servidor del tratado de mensajes que envía y recibe, incluyendo parámetros y ecos así como si tratamiento si se producen
Codificación de la creación de mensajes en el cliente	9	12h	Codificación en el cliente de los tipos de mensaje que envía y espera recibir así como su tratamiento si se producen
Registro + *login* + modificación de datos (CRUD)	9	40h	Creación del CRUD de Usuario
Creación de la tabla usuarios	9	6h	Creación de la tabla usuarios en la base de datos del sistema con todos los campos relevantes a los datos de registro del usuario
Codificación de la recuperación	9	6h	Codificación del método GET del punto de entrada, devolverá los datos de un usuario concreto
Codificación de la creación de usuarios	9	6h	Codificación del método POST del punto de entrada, creará un nuevo registro en la base de datos en caso de que los identificadores no estén repetidos
Codificación de la actualización de usuarios	9	6h	Codificación de la actualización de datos de un usuario siempre que sea sobre el usuario con el que el usuario se encuentra conectado
Codificación de la eliminación de usuarios	9	6h	Codificación de la eliminación de usuarios de la base de datos siempre que se disponga de los permisos para realizarlo o sea el propio usuario
Codificación del login	9	10h	Codificación del *login* y la estructura de cookies para mantenerlo mediante la creación de un *token* en la base de datos que comprobara posteriormente el WebSocket
Definir y codificar el objeto *World*	8	80h	Definir y codificar el objeto global del sistema *World* representando el mundo y las acciones que pueden efectuarse sobre el mismo.
Definición del objeto World del juego	8	8h	Definición de las funciones que permitan al usuario interactuar con el mundo así como los atributos del mismo

Definir cómo se va a representar el mundo	8	8h	Definir como se almacenarán los obstáculos, recursos y demás objetos del mundo
Crear la tabla mundo	8	8h	Creación de la tabla Mundo en la base de datos almacenando todos los campos relevantes del mundo para cada usuario
Codificar el objeto World en backend	8	24h	Codificar las funciones de las que dispondrá el objeto *World* como puede ser exploración, búsqueda de usuarios, lista de usuarios, búsqueda de recursos…
Documentar el objeto World	8	12h	Crear la documentación del objeto *World* que ayude a un usuario a interactuar con el mismo
Definir y codificar el objeto *Me*	8	80h	Definir y codificar el objeto global *Me*, representando al usuario y todas las acciones que puede efectuar sobre sí mismo, sus unidades, edificios y recursos.
Definición del objeto Me del juego	8	8h	Definición de las funciones y atributos del usuario y la interacción con todo lo que posee
Crear la tabla Me en la base de datos	8	12h	Crear la tabla *Me* que contenga toda la información de cada usuario en el mundo del juego
Codificar la lista de edificios	8	8h	Codificación del acceso a la lista de edificios del usuario
Codificar la lista de personajes	8	20h	Codificación de la lista que contendrá todos los personajes del jugador en cuestión
Codificar los recursos	8	12h	Codificar los distintos recursos de los que dispone el usuario
Codificar la creación de edificios	8	8h	Codificar las funciones y objetos que permitan la creación de un edificio concreto
Documentar el objeto Me	8	12h	Crear la documentación del objeto Me que ayude a un usuario a interactuar con el mismo
Codificar un personaje genérico	8	40h	Codificar los atributos y funciones de un personaje genérico del que heredaran los demás
Definir el personaje genérico	8	8h	Definir los atributos y funciones que componen a todo personaje
Codificar las acciones del personaje genérico	8	16h	Codificar las acciones y atributos correspondientes al personaje genérico, atributos de personaje (salud, ataque, coste, defensa…), funciones de movimiento, etc.
Crear la tabla personaje	8	8h	Crear la tabla Personaje que contenga los datos y funciones de cada personaje creado en el juego
Documentar la creación de personaje	8	8h	Crear el texto de manual de usuario que explique la creación de un personaje y los atributos de los que dispone
Codificar un objeto edificio genérico	8	40h	Codificar los atributos y funciones de un edificio genérico del que heredaran los demás
Definir el edificio genérico	8	8h	Definir los atributos y funciones que componen a todo personaje

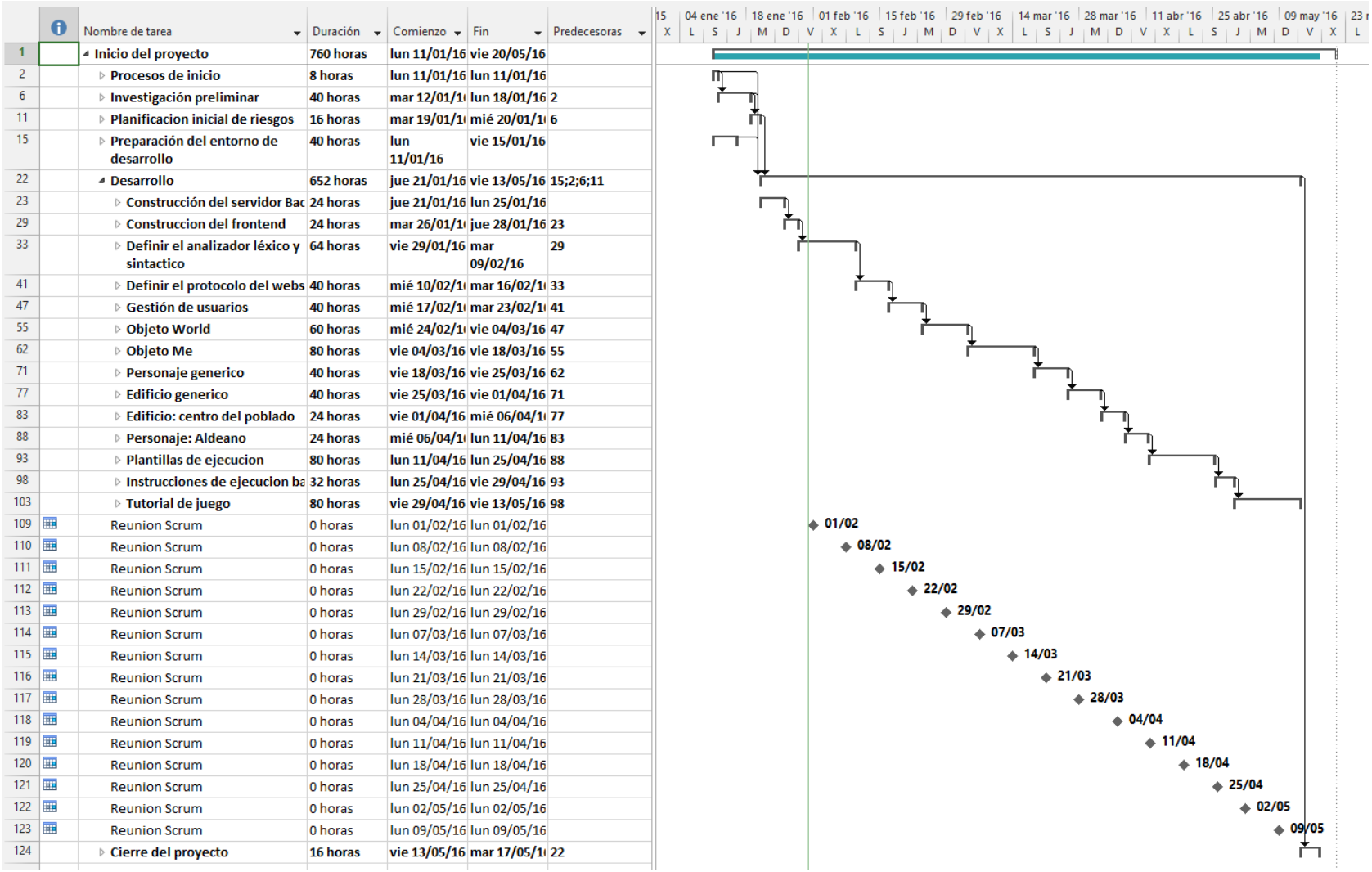
Codificar las acciones del edificio genérico	8	16h	Codificar las acciones y atributos correspondientes al edificio genérico, atributos de edificio (salud, ataque, defensa, coste…), funciones de reclutamiento, etc.
Crear la tabla edificio	8	8h	Crear la tabla edificio que contenga los datos y funciones de cada edificio creado en el juego
Documentar la creación de edificios	8	8h	Crear el texto de manual de usuario que explique la creación de un edificio y los atributos de los que dispone
Codificar el edificio del centro del poblado	7	24h	Codificar los atributos y funciones del edificio central del usuario
Definir el objeto centro del poblado	7	8h	Definir los atributos y funciones concretas del edificio centro del poblado
Codificar las funciones y atributos del centro del poblado	7	8h	Codificar las funciones de reclutamiento de aldeanos y soldados básicos así como el resto de funciones que se estimen oportunas, al igual que los atributos de defensa, ataque, salud, etc.
Documentar el edificio	7	8h	Crear el texto del manual que permita interactuar con este edificio
Codificar el personaje Aldeano	7	24h	Codificar los atributos y funciones del personaje aldeano, que permita al usuario construir nuevos edificios y recolectar recursos.
Definir el objeto aldeano	7	8h	Definir los atributos y funciones concretas de un aldeano
Codificar las funciones y atributos del personaje aldeano	7	8h	Codificar las funciones de recolección de recursos y construcción de edificios, así como los atributos de personaje correspondientes: coste, ataque, defensa, salud…
Documentar el personaje	7	8h	Crear el texto del manual que permita interactuar con este personaje
Creación de plantillas de ejecución de código	7	80h	Creación de las plantillas de código a ejecutar en *frontend* según las ordenes introducidas por el usuario
Definición de las plantillas de ejecución	7	12h	Definir e identificar las plantillas necesarias para ejecutar el código introducido por el usuario de forma segura en el *frontend*
Codificación de las plantillas frontend	7	52h	construcción y codificación de las plantillas en JavaScript *frontend* que permitan la ejecución del código introducido por el usuario de forma segura, correcta y controlada
Pruebas sobre las plantillas creadas	7	16h	Pruebas de ejecución de código sobre las plantillas creadas para comprobar su correcto funcionamiento

Envío de instrucciones de ejecución tras análisis de código desde *backend*	7	32h	Tras el análisis del código en *backend* se generan las instrucciones a ejecutar en forma de plantillas y se envían a *frontend*
Definición de mensajes necesarios para ejecución de instrucciones	7	8h	Definición de los mensajes necesarios para la ejecución de instrucciones en *frontend*, concretando los atributos en cada uno de ellos
Codificación del generador de mensajes	7	12h	Fase similar a la de generación de código en la estructura de un compilador, se generará el conjunto de mensajes necesarios para ejecutar una instrucción en *frontend* a partir del código introducido por el usuario
Interpretación de mensajes en frontend	7	12h	Se interpretarán los mensajes recibidos de *backend* y se ejecutarán mediante las plantillas definidas anteriormente en *frontend*
Tutorial	7	80h	El último mes de desarrollo se dedicará al tutorial, en caso de hacerlo rápido y sobrar tiempo se ampliara el proyecto
Definición de los pasos del tutorial	7	8h	Definición de los pasos e instrucciones necesarios para el tutorial
Codificación de las distintas fases del tutorial	7	32h	Codificación de las distintas fases, mensajes y documentación para que el usuario pueda completar el tutorial y aprender a controlar el juego
Persistir las fases del tutorial	7	16h	Almacenar el tutorial en el sistema de base de datos para que pueda ser accesible en cualquier momento. También se almacena que usuarios han completado el tutorial y en caso de no haberlo hecho en qué punto se quedaron.
Creación de pruebas unitarias	7	24h	Creación de las pruebas unitarias que comprobarán el éxito de las acciones realizadas por el usuario en cada fase del tutorial

4.2 Planificación Inicial

A continuación se muestra el diagrama de Gantt correspondiente a las tareas del *backlog* indicado anteriormente.

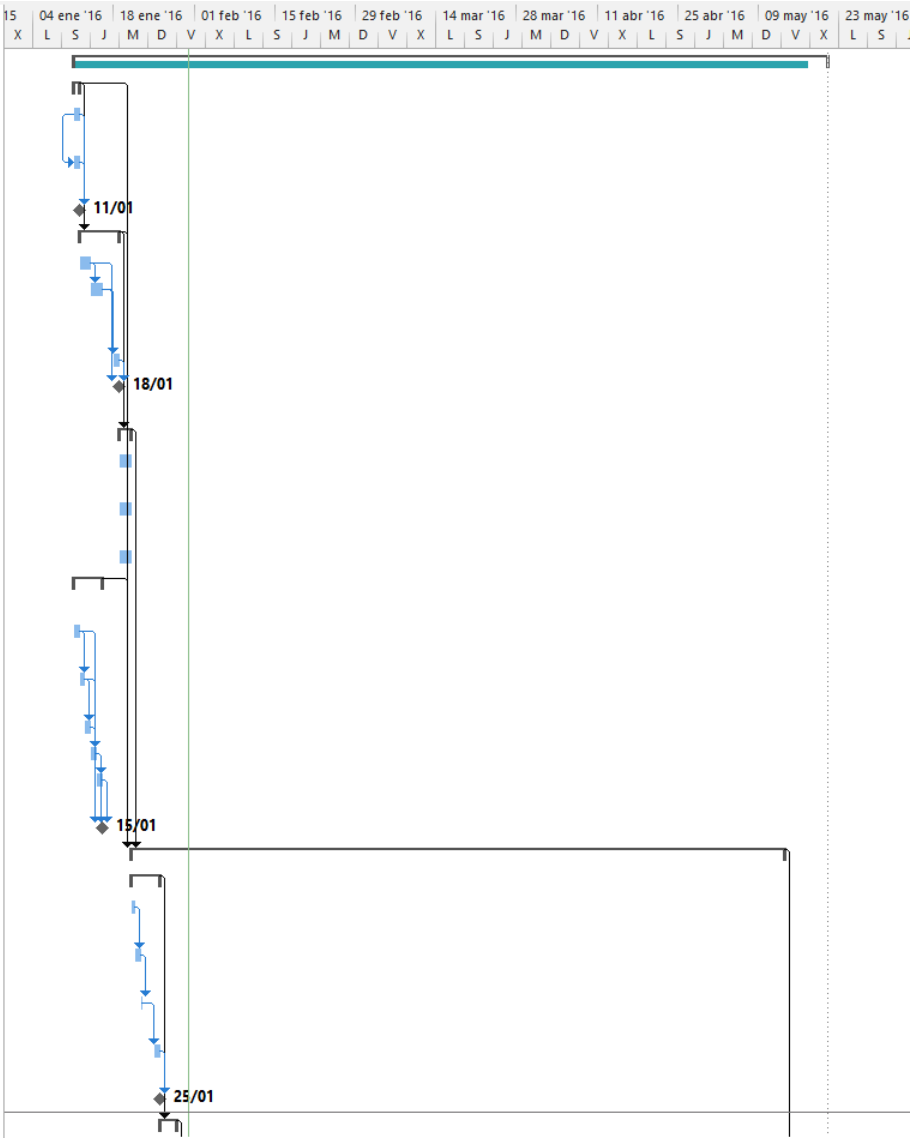
Este diagrama se ha realizado para visualizar la estimación de cuánto tiempo llevaría implementar los módulos y calcular los tiempos de los mismos de forma correcta.



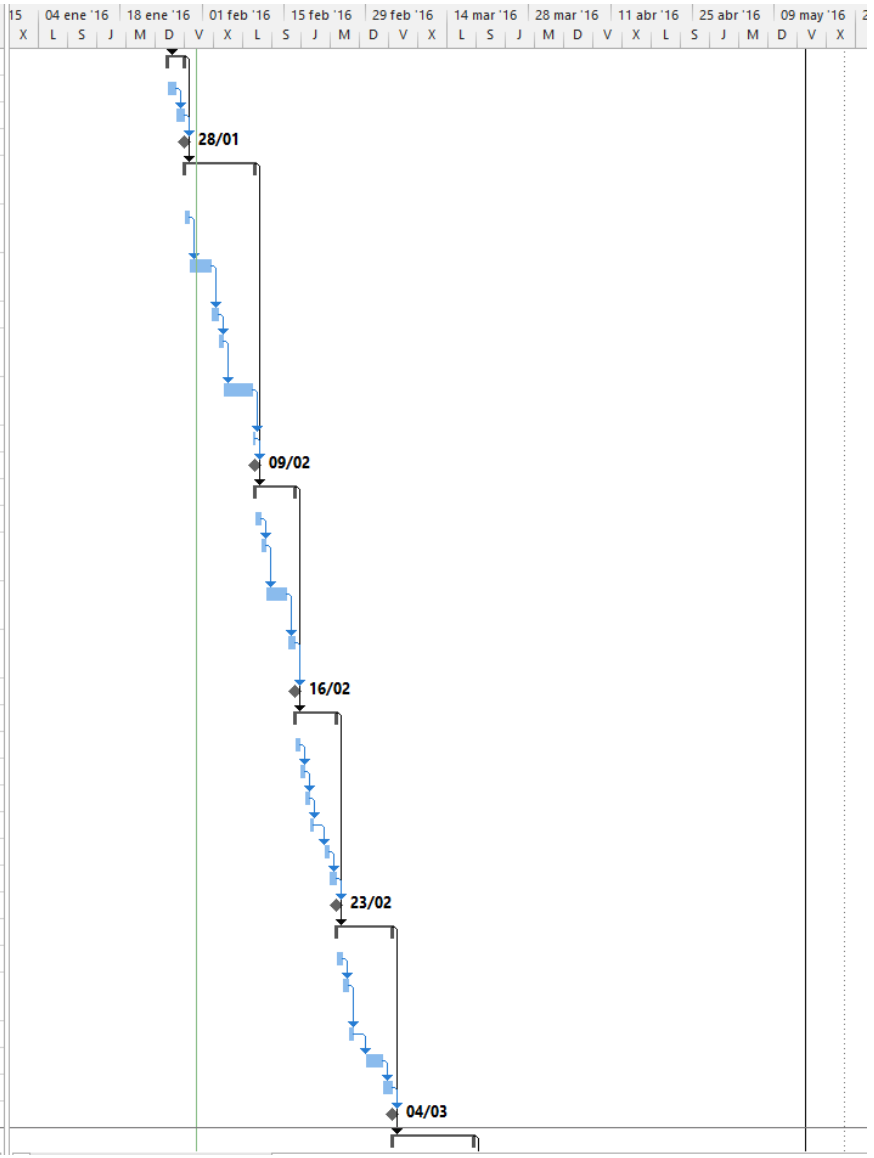
	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio del proyecto	760 horas	lun 11/01/16	vie 20/05/16	
2	Procesos de inicio	8 horas	lun 11/01/16	lun 11/01/16	
6	Investigación preliminar	40 horas	mar 12/01/1	lun 18/01/16	2
11	Planificacion inicial de riesgos	16 horas	mar 19/01/1	mié 20/01/1	6
15	Preparación del entorno de desarrollo	40 horas	lun 11/01/16	vie 15/01/16	
22	Desarrollo	652 horas	jue 21/01/16	vie 13/05/16	15;2;6;11
23	Construcción del servidor Bac	24 horas	jue 21/01/16	lun 25/01/16	
29	Construccion del frontend	24 horas	mar 26/01/1	jue 28/01/16	23
33	Definir el analizador léxico y sintactico	64 horas	vie 29/01/16	mar 09/02/16	29
41	Definir el protocolo del webs	40 horas	mié 10/02/1	mar 16/02/1	33
47	Gestión de usuarios	40 horas	mié 17/02/1	mar 23/02/1	41
55	Objeto World	60 horas	mié 24/02/1	vie 04/03/16	47
62	Objeto Me	80 horas	vie 04/03/16	vie 18/03/16	55
71	Personaje generico	40 horas	vie 18/03/16	vie 25/03/16	62
77	Edificio generico	40 horas	vie 25/03/16	vie 01/04/16	71
83	Edificio: centro del poblado	24 horas	vie 01/04/16	mié 06/04/1	77
88	Personaje: Aldeano	24 horas	mié 06/04/1	lun 11/04/16	83
93	Plantillas de ejecucion	80 horas	lun 11/04/16	lun 25/04/16	88
98	Instrucciones de ejecucion ba	32 horas	lun 25/04/16	vie 29/04/16	93
103	Tutorial de juego	80 horas	vie 29/04/16	vie 13/05/16	98
109	Reunion Scrum	0 horas	lun 01/02/16	lun 01/02/16	
110	Reunion Scrum	0 horas	lun 08/02/16	lun 08/02/16	
111	Reunion Scrum	0 horas	lun 15/02/16	lun 15/02/16	
112	Reunion Scrum	0 horas	lun 22/02/16	lun 22/02/16	
113	Reunion Scrum	0 horas	lun 29/02/16	lun 29/02/16	
114	Reunion Scrum	0 horas	lun 07/03/16	lun 07/03/16	
115	Reunion Scrum	0 horas	lun 14/03/16	lun 14/03/16	
116	Reunion Scrum	0 horas	lun 21/03/16	lun 21/03/16	
117	Reunion Scrum	0 horas	lun 28/03/16	lun 28/03/16	
118	Reunion Scrum	0 horas	lun 04/04/16	lun 04/04/16	
119	Reunion Scrum	0 horas	lun 11/04/16	lun 11/04/16	
120	Reunion Scrum	0 horas	lun 18/04/16	lun 18/04/16	
121	Reunion Scrum	0 horas	lun 25/04/16	lun 25/04/16	
122	Reunion Scrum	0 horas	lun 02/05/16	lun 02/05/16	
123	Reunion Scrum	0 horas	lun 09/05/16	lun 09/05/16	
124	Cierre del proyecto	16 horas	vie 13/05/16	mar 17/05/1	22

Figura 4.1 Planificación inicial

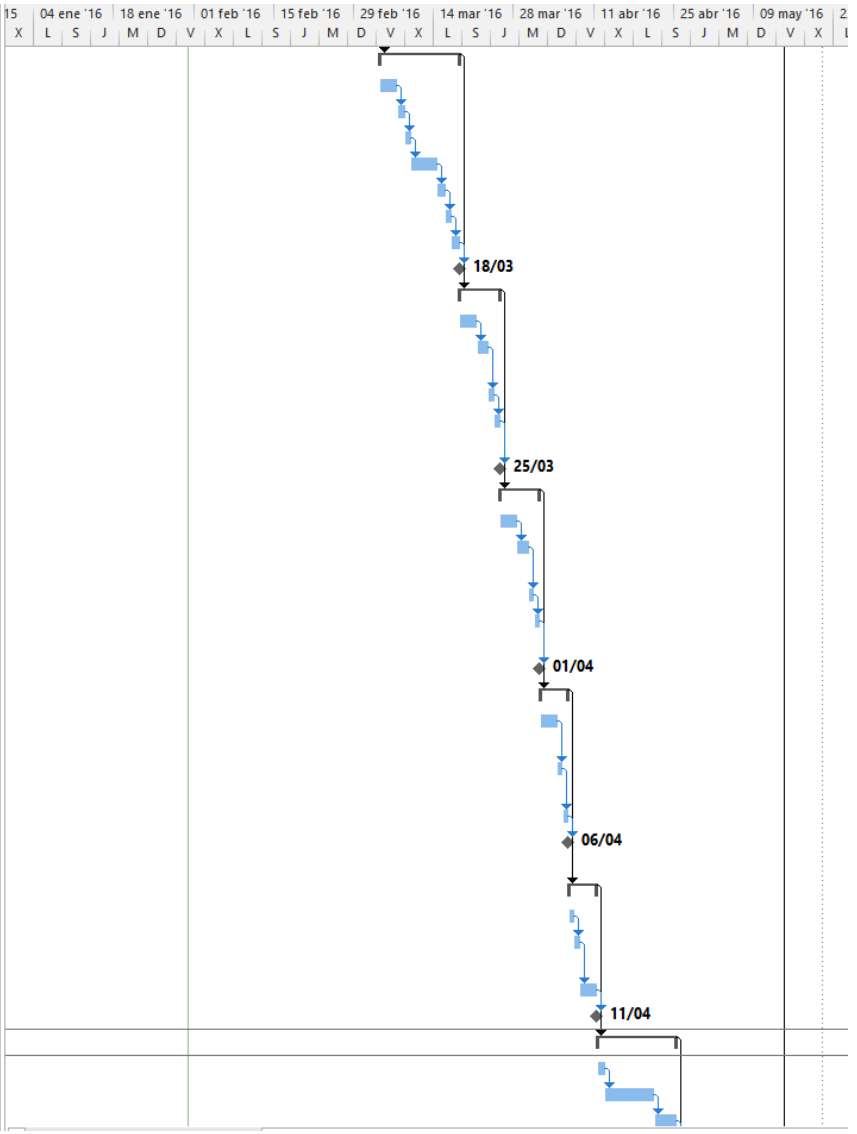
	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Inicio del proyecto**	**760 horas**	**lun 11/01/16**	**vie 20/05/16**	
2	**Procesos de inicio**	**8 horas**	**lun 11/01/16**	**lun 11/01/16**	
3	Establecer las comunicaciones entre los	8 horas	lun 11/01/16	lun 11/01/16	
4	Establecer los requisitos del proyecto	8 horas	lun 11/01/16	lun 11/01/16	3CC
5	Validar los requisitos del proy	0 horas	lun 11/01/16	lun 11/01/16	4;3
6	**Investigación preliminar**	**40 horas**	**mar 12/01/1**	**lun 18/01/16**	**2**
7	Busqueda de aplicaciones sim	16 horas	mar 12/01/1	mié 13/01/1	
8	Búsqueda y selección de un framework backend y un framework frontend	16 horas	jue 14/01/16	vie 15/01/16	7
9	Búsqueda y selección de un S	8 horas	lun 18/01/16	lun 18/01/16	8
10	Documentación: Aplicaciones relacionadas	0 horas	lun 18/01/16	lun 18/01/16	7;8;9
11	**Planificacion inicial de riesgos**	**16 horas**	**mar 19/01/1**	**mié 20/01/1**	**6**
12	Tormenta de ideas para la identificación de riesgos	16 horas	mar 19/01/16	mié 20/01/16	
13	Cuantificación y priorización de riesgos	16 horas	mar 19/01/16	mié 20/01/16	
14	Toma de decisiones de riesg	16 horas	mar 19/01/1	mié 20/01/1	
15	**Preparación del entorno de desarrollo**	**40 horas**	**lun 11/01/16**	**vie 15/01/16**	
16	Instalación del entorno de ejecucion en el servidor	8 horas	lun 11/01/16	lun 11/01/16	
17	Instalación de IDEs y compiladores en el equipo	8 horas	mar 12/01/16	mar 12/01/16	16
18	Securización del servidor	8 horas	mié 13/01/1	mié 13/01/1	17
19	Preparacion del entorno de p	8 horas	jue 14/01/16	jue 14/01/16	18
20	Preparacion del control de versiones	8 horas	vie 15/01/16	vie 15/01/16	19
21	Entorno de desarrollo	0 horas	vie 15/01/16	vie 15/01/16	16;17;18;19;20
22	**Desarrollo**	**652 horas**	**jue 21/01/16**	**vie 13/05/16**	**15;2;6;11**
23	**Construcción del servidor Bac**	**24 horas**	**jue 21/01/16**	**lun 25/01/16**	
24	Definición de los puntos de entrada REST	6 horas	jue 21/01/16	jue 21/01/16	
25	Definicion de los puntos de entrada de WebSocket	8 horas	jue 21/01/16	vie 22/01/16	24
26	Creación de la base de datos de la aplicación	2 horas	vie 22/01/16	vie 22/01/16	25
27	Conexión del backend con la base de datos	8 horas	lun 25/01/16	lun 25/01/16	26
28	Servidor Backend	0 horas	lun 25/01/16	lun 25/01/16	27
29	**Construccion del frontend**	**24 horas**	**mar 26/01/1**	**jue 28/01/16**	**23**



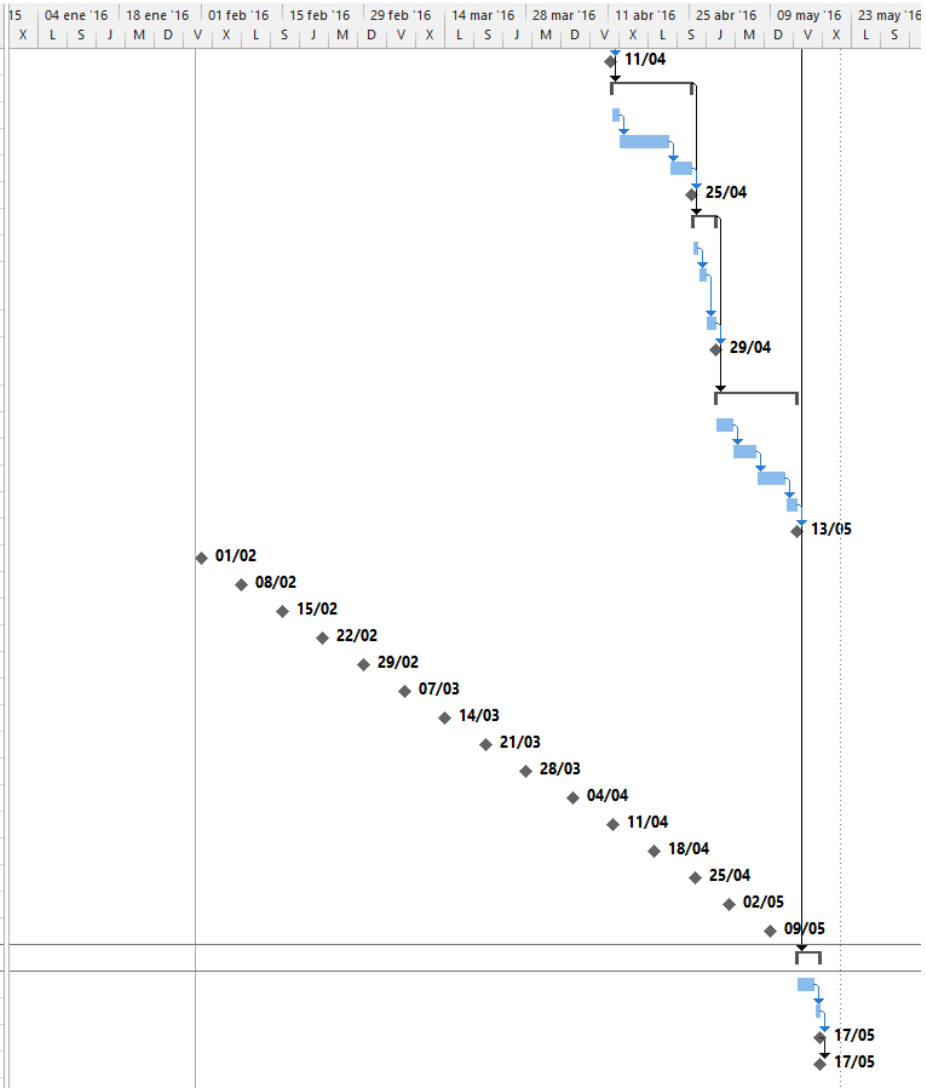
	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
29	**Construccion del frontend**	**24 horas**	**mar 26/01/1**	**jue 28/01/16**	**23**
30	Codificación del HTML	10 horas	mar 26/01/1	mié 27/01/1	
31	Creación de las hojas de e:	14 horas	mié 27/01/1	jue 28/01/16	30
32	Entrega del Frontend	0 horas	jue 28/01/16	jue 28/01/16	31
33	**Definir el analizador léxico y sintactico**	**64 horas**	**vie 29/01/16**	**mar 09/02/16**	**29**
34	Definicion de las expresiones regulares del	6 horas	vie 29/01/16	vie 29/01/16	
35	Definicion de la gramatica del lenguaje	12 horas	vie 29/01/16	mar 02/02/16	34
36	Construcción del AST del l	12 horas	mar 02/02/1	mié 03/02/1	35
37	Almacenar las expresiones en el AST	8 horas	mié 03/02/16	jue 04/02/16	36
38	Construccion del analizador semantico	22 horas	jue 04/02/16	mar 09/02/16	37
39	Prueba del analizador	4 horas	mar 09/02/1	mar 09/02/1	38
40	Entrega del analizador	0 horas	mar 09/02/1	mar 09/02/1	39
41	**Definir el protocolo del webs**	**40 horas**	**mié 10/02/1**	**mar 16/02/1**	**33**
42	Definir los mensajes a trat	8 horas	mié 10/02/1	mié 10/02/1	
43	Definir los campos necesarios en cada	8 horas	jue 11/02/16	jue 11/02/16	42
44	Codificar la creación de mensajes en Servidor	12 horas	vie 12/02/16	lun 15/02/16	43
45	Codificar la creación de mensajes en cliente	12 horas	lun 15/02/16	mar 16/02/16	44
46	Protocolo del websocket	0 horas	mar 16/02/1	mar 16/02/1	45
47	**Gestión de usuarios**	**40 horas**	**mié 17/02/1**	**mar 23/02/1**	**41**
48	Creación de la tabla de usu	6 horas	mié 17/02/1	mié 17/02/1	
49	Recuperación de datos	6 horas	mié 17/02/1	jue 18/02/16	48
50	Creación de usuarios	6 horas	jue 18/02/16	vie 19/02/16	49
51	Actualización de usuarios	6 horas	vie 19/02/16	vie 19/02/16	50
52	Eliminación de usuarios	6 horas	lun 22/02/16	lun 22/02/16	51
53	Codificacion del login	10 horas	lun 22/02/16	mar 23/02/1	52
54	Gestión de usuarios	0 horas	mar 23/02/1	mar 23/02/1	53
55	**Objeto World**	**60 horas**	**mié 24/02/1**	**vie 04/03/16**	**47**
56	Definición del objeto Wor	8 horas	mié 24/02/1	mié 24/02/1	
57	Definición de la representación del mundo	8 horas	jue 25/02/16	jue 25/02/16	56
58	Crear la tabla Mundo en el	8 horas	vie 26/02/16	vie 26/02/16	57
59	Codificar el objeto en Back	24 horas	lun 29/02/16	mié 02/03/1	58
60	Documentar el objeto	12 horas	jue 03/03/16	vie 04/03/16	59
61	Entrega: objeto World	0 horas	vie 04/03/16	vie 04/03/16	60
62	**Objeto Me**	**80 horas**	**vie 04/03/16**	**vie 18/03/16**	**55**



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
62	**Objeto Me**	**80 horas**	**vie 04/03/16**	**vie 18/03/16**	**55**
63	Definición del objeto Me	8 horas	vie 04/03/16	lun 07/03/16	
64	Creación de la tabla en el S	12 horas	lun 07/03/16	mar 08/03/1(	63
65	Codificar la lista de edificí	8 horas	mié 09/03/1(	mié 09/03/1(	64
66	Codificar la lista de persor	20 horas	jue 10/03/16	lun 14/03/16	65
67	Codificar los recursos	12 horas	lun 14/03/16	mar 15/03/1(	66
68	Codificar la creación de ed	8 horas	mié 16/03/1(	mié 16/03/1(	67
69	Documentar el objeto	12 horas	jue 17/03/16	vie 18/03/16	68
70	Entrega: objeto Me	0 horas	vie 18/03/16	vie 18/03/16	69
71	**Personaje generico**	**40 horas**	**vie 18/03/16**	**vie 25/03/16**	**62**
72	Definir el personaje gener	8 horas	vie 18/03/16	lun 21/03/16	
73	Codificar las acciones del personaje generico	16 horas	lun 21/03/16	mié 23/03/16	72
74	Crear la tabla personaje	8 horas	mié 23/03/1(	jue 24/03/16	73
75	Documentar la creación de personajes	8 horas	jue 24/03/16	vie 25/03/16	74
76	Entrega: Personaje Generi	0 horas	vie 25/03/16	vie 25/03/16	75
77	**Edificio generico**	**40 horas**	**vie 25/03/16**	**vie 01/04/16**	**71**
78	Definir el edificio generic	8 horas	vie 25/03/16	lun 28/03/16	
79	Codificar las acciones del edificio generico	16 horas	lun 28/03/16	mié 30/03/16	78
80	Crear la tabla edificio	8 horas	mié 30/03/1(	jue 31/03/16	79
81	Documentar la creación de edificios	8 horas	jue 31/03/16	vie 01/04/16	80
82	Entrega: Edificio Generico	0 horas	vie 01/04/16	vie 01/04/16	81
83	**Edificio: centro del poblado**	**24 horas**	**vie 01/04/16**	**mié 06/04/1(**	**77**
84	Definir el objeto centro del poblado	8 horas	vie 01/04/16	lun 04/04/16	
85	Codificar las funciones y atributos del centro del	8 horas	lun 04/04/16	mar 05/04/16	84
86	Documentar el edificio	8 horas	mar 05/04/1(	mié 06/04/1(	85
87	Entrega: Edificio Centro del poblado	0 horas	mié 06/04/16	mié 06/04/16	86
88	**Personaje: Aldeano**	**24 horas**	**mié 06/04/1(**	**lun 11/04/16**	**83**
89	Definir el objeto aldeano	8 horas	mié 06/04/1(	jue 07/04/16	
90	Codificar funciones y atributos del personaje	8 horas	jue 07/04/16	vie 08/04/16	89
91	Documentar el personaje	8 horas	vie 08/04/16	lun 11/04/16	90
92	Entrega: personaje Aldean	0 horas	lun 11/04/16	lun 11/04/16	91
93	**Plantillas de ejecucion**	**80 horas**	**lun 11/04/16**	**lun 25/04/16**	**88**
94	Definicion de las plantillas	12 horas	lun 11/04/16	mar 12/04/1(	
95	Codificacion de las plantill	52 horas	mié 13/04/1(	jue 21/04/16	94
96	Pruebas sobre las plantilla	16 horas	jue 21/04/16	lun 25/04/16	95



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
92	Entrega: personaje Aldean	0 horas	lun 11/04/16	lun 11/04/16	91
93	**Plantillas de ejecucion**	**80 horas**	**lun 11/04/16**	**lun 25/04/16**	**88**
94	Definicion de las plantillas	12 horas	lun 11/04/16	mar 12/04/1	
95	Codificacion de las plantill	52 horas	mié 13/04/1	jue 21/04/16	94
96	Pruebas sobre las plantilla	16 horas	jue 21/04/16	lun 25/04/16	95
97	Entrega: Plantillas de ejec	0 horas	lun 25/04/16	lun 25/04/16	96
98	**Instrucciones de ejecucion ba**	**32 horas**	**lun 25/04/16**	**vie 29/04/16**	**93**
99	Definicion de mensajes	8 horas	lun 25/04/16	mar 26/04/1	
100	Codificacion del generador de mensajes	12 horas	mar 26/04/16	mié 27/04/16	99
101	Interpretacion de mensaje	12 horas	jue 28/04/16	vie 29/04/16	100
102	Entrega: Instrucciones de ejecucion	0 horas	vie 29/04/16	vie 29/04/16	101
103	**Tutorial de juego**	**80 horas**	**vie 29/04/16**	**vie 13/05/16**	**98**
104	Definicion de los pasos	8 horas	vie 29/04/16	lun 02/05/16	
105	Codificacion de las fases	32 horas	lun 02/05/16	vie 06/05/16	104
106	Creación de pruebas unita	24 horas	vie 06/05/16	mié 11/05/1	105
107	Persistir el tutorial en el S	16 horas	mié 11/05/1	vie 13/05/16	106
108	Entrega: Tutorial de juego	0 horas	vie 13/05/16	vie 13/05/16	107
109	Reunion Scrum	0 horas	lun 01/02/16	lun 01/02/16	
110	Reunion Scrum	0 horas	lun 08/02/16	lun 08/02/16	
111	Reunion Scrum	0 horas	lun 15/02/16	lun 15/02/16	
112	Reunion Scrum	0 horas	lun 22/02/16	lun 22/02/16	
113	Reunion Scrum	0 horas	lun 29/02/16	lun 29/02/16	
114	Reunion Scrum	0 horas	lun 07/03/16	lun 07/03/16	
115	Reunion Scrum	0 horas	lun 14/03/16	lun 14/03/16	
116	Reunion Scrum	0 horas	lun 21/03/16	lun 21/03/16	
117	Reunion Scrum	0 horas	lun 28/03/16	lun 28/03/16	
118	Reunion Scrum	0 horas	lun 04/04/16	lun 04/04/16	
119	Reunion Scrum	0 horas	lun 11/04/16	lun 11/04/16	
120	Reunion Scrum	0 horas	lun 18/04/16	lun 18/04/16	
121	Reunion Scrum	0 horas	lun 25/04/16	lun 25/04/16	
122	Reunion Scrum	0 horas	lun 02/05/16	lun 02/05/16	
123	Reunion Scrum	0 horas	lun 09/05/16	lun 09/05/16	
124	**Cierre del proyecto**	**16 horas**	**vie 13/05/16**	**mar 17/05/1**	**22**
125	Despliegue a produccion	8 horas	vie 13/05/16	lun 16/05/16	
126	Pruebas en produccion	8 horas	lun 16/05/16	mar 17/05/1	125
127	Validación final	0 horas	mar 17/05/1	mar 17/05/1	126
128	Cierre del proyecto	0 horas	mar 17/05/1	mar 17/05/1	127



4.3 Presupuesto Inicial

A continuación se presentan los presupuestos calculados inicialmente, el primero de ellos se encuentra detallado e incluye toda la información en cuanto a beneficios y costes para trabajar de forma interna en la empresa que presupuesta mientras que el segundo es un presupuesto reducido que se entrega al cliente sin entrar en excesivos detalles.

4.3.1 Desarrollo de Presupuesto Detallado (Empresa)

El presupuesto a continuación es el presupuesto interno de la empresa que ofrece el proyecto, este se encuentra lo más detallado posible, incluyendo los precios unitarios de cada unidad y las amortizaciones correspondientes. Igualmente se calcula el margen de beneficios y todos aquellos recursos que serán necesarios de forma detallada.

Ítem	Concepto	Cantidad	Amortización	Precio Unitario (€)	Total (€)
1	*Investigación Inicial*				
1.1	Procesos de inicio	8h	100%	10	80
1.2	Investigación preliminar	40h	100%	10	400
1.3	Planificación de riesgos	16h	100%	10	160
1.4	Preparación del entorno de desarrollo	40h	100%	10	400
2	*Recursos Software*				
2.1	Microsoft Windows 10	2	10%	135	27
2.2	Microsoft Office Professional 2013	2	7%	539	75,46
2.3	Ubuntu Server 14.04 LTS	1	10%	0	0
2.4	Licencia PyCharm	1	40%	199	79,60
2.5	Vim	1	5%	0	0
2.6	Apache Server	1	100%	0	0
2.7	Python 3	3	100%	0	0
2.8	Servidor Tornado	1	100%	0	0
2.9	Framework *frontend*	1	100%	0	0

2.10	Analizador léxico y sintáctico Python	1	100%	0	0
2.11	SGBD MySQL	1	100%	0	0
3	*Recursos Hardware*				
3.1	Servidor Dedicado	1	100%	84,62	84,62
3.2	Equipo Sobremesa	1	10%	1400	140
3.3	Equipo portátil	1	10%	1200	120
4	*Módulos del proyecto*				
4.1	Construcción del servidor *backend*	24h	100%	10	240
4.2	Construcción del *frontend*	24h	100%	10	240
4.3	Definir el analizador léxico y sintáctico	64h	100%	10	640
4.4	Definir el protocolo del *websocket*	40h	100%	10	400
4.5	Gestión de usuarios	40h	100%	10	400
4.6	Objeto *World*	60h	100%	10	600
4.7	Objeto Me	80h	100%	10	800
4.8	Personaje genérico	40h	100%	10	400
4.9	Edificio genérico	40h	100%	10	400
4.10	Edificio: Centro del poblado	24h	100%	10	240
4.11	Personaje: Aldeano	24h	100%	10	240
4.12	Plantillas de ejecución	80h	100%	10	800
4.13	Instrucciones de ejecución *backend*	32h	100%	10	320
4.14	Tutorial de juego	80h	100%	10	800
4.15	Cierre y despliegue	16h	100%	10	160
5	*Otros Gastos*				
5.1	Desplazamientos	-	100%	130	130
5.2	Electricidad	-	100%	120	120
5.3	Gestión	-	100%	100	100
5.4	Gastos de oficina	-	100%	125	125
5.5	Contingencias	-	100%	600	600
5.6	Alojamiento	-	100%	800	800

Subtotal	10.121,68
Beneficio (10%)	1.012,17
IVA (21%)	2.338,11
TOTAL	**13.471,96**

Se ha calculado el precio unitario de la hora de programación como un sueldo medio de desarrollador, siendo su coste de 10€/h y un total de 1600 €/mes y teniendo en cuenta un 25% de retenciones para seguridad social, hacienda y otras retribuciones, siendo el sueldo real percibido por el desarrollador de 1200€.

Dentro de los gastos de oficina se incluye todo el material fungible como tinta de impresora y papel así como el agua y otros gastos posibles derivados del uso diario.

Se incluye un presupuesto para posibles contingencias de 600€.

El coste de alojamiento es de 200€/mes, duración estimada del proyecto 4 meses.

En cuanto a las amortizaciones se han calculado de la forma que sigue:

- Microsoft Windows: Renovación de licencia cada 4 años, duración del proyecto 4 meses.
- Microsoft Office: Renovación de licencia cada 5 años, duración del proyecto 4 meses.
- Licencia PyCharm: Renovación anual, duración del proyecto 4 meses, extra por posible consulta con servicio técnico.
- Renovación de equipos cada 4 años, duración del proyecto 4 meses, extra por reparación de piezas e imprevistos.

4.3.2 Desarrollo de Presupuesto Simplificado (Cliente)

El siguiente presupuesto es el simplificado para el cliente, obviando detalles que el cliente no debe conocer como el margen de beneficios, amortizaciones o precios unitarios de los elementos.

Se indican los submódulos implementados en cursiva.

Ítem	Concepto	Total (€)
1	Investigación inicial	1144
2	Recursos Software	200,27
3	Recursos Hardware	379,08
4	Módulos del proyecto	9410,5
4.1	*Construcción del servidor backend*	401,5
4.2	*Construcción del frontend*	401,5
4.3	*Definir el analizador léxico y sintáctico*	841,5
4.4	*Definir el protocolo del websocket*	577,5
4.5	*Gestión de usuarios*	577,5
4.6	*Objeto World*	797,5
4.7	*Objeto Me*	1017,5
4.8	*Personaje genérico*	577,5
4.9	*Edificio genérico*	577,5
4.10	*Edificio: Centro del poblado*	401,5
4.11	*Personaje: Aldeano*	401,5
4.12	*Plantillas de ejecución*	1017,5
4.13	*Instrucciones de ejecución backend*	489,5
4.14	*Tutorial de juego*	1017,5

4.15	*Cierre y despliegue*	313,5
Subtotal		11.133,85
IVA (21%)		2.338,11
TOTAL		**13.471,96**

Capítulo 5. Análisis

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

Este proyecto trata de facilitar al usuario el aprendizaje de la programación en un lenguaje de alto nivel mediante la integración en un videojuego web de tipo estrategia en tiempo real. Se plantea que en un futuro el videojuego sea ampliable, mediante otra gramática que simule un lenguaje de programación distinto así como la ampliación del videojuego del sistema con nuevas opciones.

Este videojuego estará sostenido mediante un servidor web que se encargará de servir las páginas para que sea accesible mediante un navegador web moderno. Estas páginas lanzarán peticiones contra otra aplicación que se encargue de proporcionar la lógica del sistema. Dado que este sistema se encontrará accesible a todo el mundo mediante internet se valorará de forma muy importante y crítica la seguridad del sistema.

El sistema será accesible desde los principales navegadores web de escritorio del momento, usando aquellas tecnologías de las que se disponga en los mismos. No obstante cabe destacar que no se proporcionará acceso dedicado a navegadores móviles dado que no se estima que el acceso al videojuego desde este tipo de dispositivo sea relevante.

La aplicación únicamente ofrecerá acceso al juego a usuarios registrados en el sistema por lo que se considera necesario como primer requisito de la misma soportar a distintos usuarios, su registro, modificación de datos, conexión y desconexión de la aplicación y por supuesto, el borrado de sus datos. Una vez se soportan usuarios es necesario soportar distintos tipos, pues puede ser necesario efectuar acciones con más privilegios que los de un usuario corriente como por ejemplo impedir el acceso a un usuario concreto.

Basado en la descripción anterior se muestran a continuación los distintos roles de usuario y las acciones que pueden realizar:

- Usuario Anónimo:
  - Registro de usuario.
  - Conexión al sistema.
  - Visualización del sitio web.
- Usuario Registrado:
  - Mismas acciones que el usuario anónimo.
  - Desconexión del sistema.
  - Interacción con el videojuego.
  - Compilación de código.
  - Ejecución de código.
  - Comprobación de errores.

    - Modificación de sus datos.
    - Interacción con su propio objeto mundo.
    - Interacción con su propio objeto usuario.
    - Persistencia de sus propios datos del juego.
- Usuario administrador:
    - Mismas acciones que el usuario registrado.
    - Modificación de datos de otros usuarios.
    - Restricción del acceso a usuarios.

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

5.2.1.1 Explicación textual

Se va a desarrollar una aplicación web que facilite el aprendizaje de un usuario en un lenguaje de alto nivel moderno. Para ello se desarrollara un pequeño juego en web de tipo estrategia en tiempo real con el que el usuario interactuará empleando dicho lenguaje.

Para que el usuario pueda interactuar con el videojuego será necesario que se encuentre registrado y autenticado en la aplicación. Por supuesto, esta aplicación soportara la edición de los datos y el borrado de los mismos siempre que el usuario así lo solicite.

Para que el juego sea visualmente más atractivo se creará una interfaz web que muestre el estado del juego mediante una visualización gráfica. Esta aplicación solo interactuará con una aplicación mediante peticiones que ejecutara toda la lógica del sistema.

La aplicación empleará tecnologías modernas soportadas por los navegadores de escritorio actuales, se valora como muy importante que la aplicación mantenga unos niveles altos de seguridad, dado que se encontrará disponible de cara al público a través de internet.

Igualmente se valora positivamente que la aplicación sea lo más usable posible y que emplee tecnologías eficientes y modernas.

En cuanto a la jugabilidad del sistema, se ha de crear un sistema que reconozca el código introducido por el usuario y ejecute las acciones correspondientes. Igualmente el usuario ha de poder interactuar con sus objetos y con su mundo de juego. Estos datos deben estar persistidos para que puedan ser accedidos cada vez que el usuario se conecte al sistema.

Finalmente, los lenguajes, tecnologías y herramientas a emplear serán elegidos por el desarrollador siempre que se puedan cumplir estos requisitos descritos.

5.2.1.2 Requisitos funcionales

Código	Nombre Requisito	Descripción del Requisito
R1	Usuario Anónimo	
R1.1	Registrar usuario	Se debe añadir un nuevo usuario al sistema comprobando que su dirección de correo electrónico no se encuentre introducida ya en la base de datos. Igualmente ha de comprobarse que el resto de datos introducidos sean correctos.
R1.2	Acceder al sistema	Debe permitir el acceso al sistema a un usuario correctamente autentificado mediante la comprobación de la contraseña y el nombre de usuario asignado.
R1.3	Visualización del sitio web	Debe permitirse que se visualice el sitio web público a excepción del juego que solo será accesible a usuarios registrados. Igualmente todos aquellos apartados de administración o modificación de datos no accesibles sin autenticación han de estar restringidos.
R2	Usuario Registrado	
R2.1	Desconectar del sistema	Se debe permitir la desconexión del sistema del usuario sin pérdida de datos y asegurando que esta desconexión es efectiva hasta la siguiente conexión.
R2.2	Interacción con el videojuego	Se debe permitir la interacción con la página del videojuego, así como las funciones derivadas de esta interacción.
R2.3	Compilación de código	Se ha de permitir la compilación del código introducido a tal efecto en la aplicación por parte del usuario.
R2.4	Comprobación de errores	Se debe comprobar que el código introducido para compilación por parte del usuario se encuentra libre de errores y es compatible con la especificación del sistema. En caso de haber errores estos han de ser notificados al usuario aportando la mayor cantidad de información posible.
R2.5	Ejecución de código	Se debe ejecutar el código introducido por el usuario y obtener los resultados esperados del mismo siempre que este no tenga errores.
R2.6	Modificación de datos	Se debe permitir la modificación de los datos del usuario siempre que este lo solicite.
R2.7	Interacción con objeto mundo	Se debe permitir que el usuario interactúe con su mundo en el juego, permitiéndole realizar acciones sobre el mismo.
R2.8	Interacción con objeto usuario	Se debe permitir que el usuario interactúe con su objeto usuario del juego y realizar las acciones que considere oportunas.
R2.9	Persistencia de datos de usuario	Se deben persistir los datos del usuario de forma que cada vez que se conecte este disponga de las

		modificaciones que haya realizado sobre aquellos objetos modificables.
R3	Usuario Administrador	
R3.1	Modificación de datos de otros usuarios	Se debe permitir que los datos de otros usuarios sean modificados mediante el usuario administrador.
R3.2	Restricción de acceso al sistema	Se debe permitir que el usuario administrador pueda restringir el acceso a determinados usuarios.

5.2.1.3 Requisitos no funcionales

Código	Nombre Requisito	Descripción del Requisito
R4	Requisitos de Usuario	
R4.1	Usuarios con edad suficiente	Se debe contemplar que los usuarios del sistema tengan edad suficiente para aceptar los términos del mismo según la legislación del país.
R5	Requisitos tecnológicos	
R5.1	Estandarización	El sistema debe funcionar según los estándares modernos y estar soportado en los navegadores de escritorio actuales. Para ello han de emplearse tecnologías estándar y soportadas.
R5.2	Persistencia de datos	Los datos de los usuarios han de estar persistidos y no pueden perderse por fallos de la aplicación.
R5.3	Disponibilidad	El sistema ha de encontrarse disponible 24/7 en la medida de lo posible.
R5.4	Eficiencia	Dado que puede darse un número elevado de jugadores simultáneos y la carga de un compilador es elevada, ha de crearse una aplicación eficiente que no derive en retrasos a los jugadores.
R6	Requisitos de usabilidad	
R6.1	Usabilidad	El sistema ha de ser accesible a los usuarios, facilitando la interacción con el mismo.
R7	Requisitos de Seguridad	
R7.1	Conexiones seguras	El sistema deberá emplear conexiones seguras siempre que sea posible.
R7.2	Control de la entrada de datos	El sistema deberá controlar la entrada de datos al sistema por parte del usuario para evitar vulnerabilidades y fallos típicos de seguridad.
R7.3	Entorno de ejecución seguro	El sistema deberá encontrarse asegurado contra ataques en la medida de lo posible.
R8	Requisitos de plataforma	
R8.1	Implementación de requisitos de la LOPD	El sistema deberá implementar la seguridad y requisitos exigidos por la Ley Orgánica de Protección de Datos Española.

5.2.2 Identificación de Actores del Sistema

En este proyecto se definen 3 actores principales:

- Usuario Anónimo: Este usuario es todo aquel que accede al sitio web sin estar debidamente autenticado, bien porque no dispone de un usuario o porque aún no se ha conectado con el mismo. Este usuario únicamente puede navegar por el sitio web o realizar aquellas acciones que le hagan estar registrado.
- Usuario Registrado: Se reconoce como este usuario a todo aquel que dispone de un usuario registrado y se encuentra identificado en el sistema como el mismo. Puede efectuar todas las acciones del usuario anónimo además de la práctica totalidad del resto de funciones que se ofrecen.
- Usuario Administrador: Este usuario ha sido modificado con un rol específico para permitir aquellas funciones que afecten a otros usuarios como modificar datos y restringir acceso.

5.2.3 Especificación de Casos de Uso

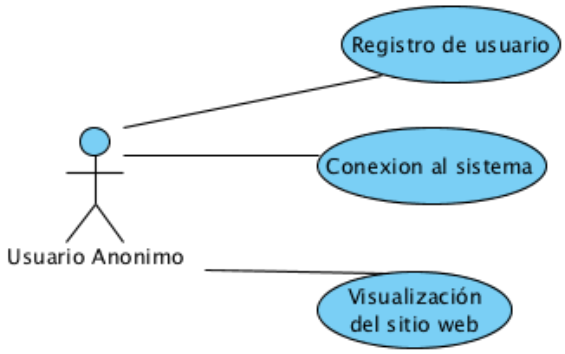


Figura 5.1 Casos de uso de Usuario Anónimo

Nombre del Caso de Uso
Registro de usuario
Descripción
El sistema mostrará un formulario con los campos que el usuario deberá cumplimentar para pasar a estar registrado en el sistema. Posteriormente si esta acción se ha realizado con éxito será redirigido a la página de autenticación.

Nombre del Caso de Uso
Conexión al sistema
Descripción
El sistema mostrará un formulario solicitando los datos de nombre de usuario y contraseña al usuario. Si esta información se provee de forma correcta el usuario pasara a estar autenticado en el sistema y podrá cambiar su rol de usuario anónimo a usuario registrado. En caso de que la información no sea correcta o haya errores se informará al usuario mediante un mensaje de error genérico.

Nombre del Caso de Uso
Visualización del sitio web
Descripción
El sistema deberá servir las páginas del sitio web a todos los usuarios, si bien deberá restringir el acceso a partes del mismo como el juego o la modificación de datos.

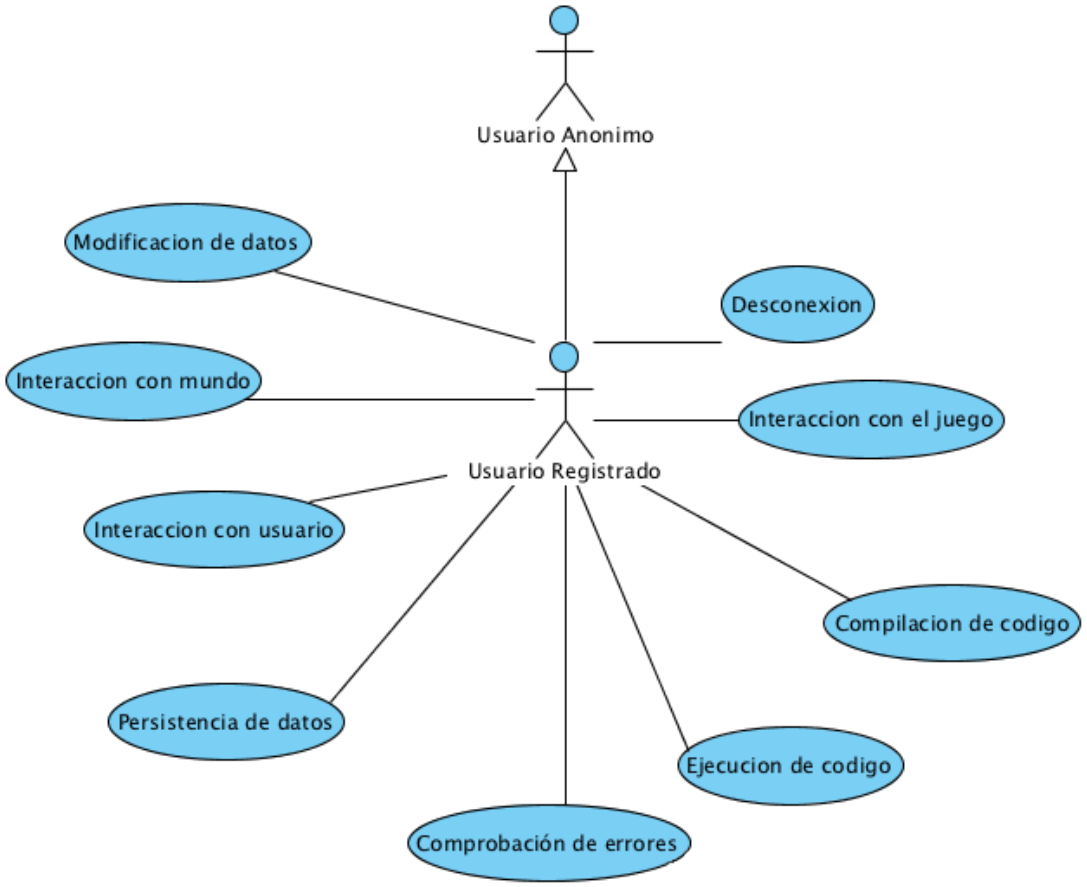


Figura 5.2 Casos de uso de Usuario Registrado

Nombre del Caso de Uso
Desconexión
Descripción
El sistema deberá proveer un acceso que solicite la desconexión del sistema al usuario, tras dicha solicitud el usuario deberá desconectarse de la aplicación y a recibir los accesos de usuario anónimo únicamente.

Nombre del Caso de Uso
Interacción con el juego
Descripción
El sistema deberá permitir que el usuario interactúe con el videojuego permitiéndole introducir código y visualizar el resultado de dicha introducción. Si el usuario se desconecta se le deberá dejar de atender y redirigir a una página desde la que pueda interactuar con sus permisos.

Nombre del Caso de Uso
Compilación de código
Descripción
El sistema deberá atender a todo usuario registrado que desee enviar código al servidor para comprobar si este está libre de errores y posteriormente ejecutarlo. Igualmente deberá informarse de los errores que hayan sucedido.

Nombre del Caso de Uso
Ejecución de Código
Descripción
El sistema deberá ejecutar el código introducido por el usuario si este se encuentra libre de errores y es correcto. Esta ejecución deberá ser controlada desde la aplicación para evitar posibles problemas.

Nombre del Caso de Uso
Comprobación de errores
Descripción
El sistema deberá mostrar la mayor cantidad de información posible al usuario en cuanto a los errores encontrados en su código para facilitar la depuración de los mismos.

Nombre del Caso de Uso
Persistencia de datos
Descripción
Todos los datos del sistema han de estar persistidos en una base de datos en la parte *backend* de la aplicación y han de ser guardados de forma fiable y segura, manteniendo su integridad.

Nombre del Caso de Uso
Interacción con usuario
Descripción
El sistema ha de permitir al usuario interaccionar con su propio usuario en el juego, esto es, proveer acceso a sus edificios, recursos y demás atributos relevantes del mismo en la medida que no ocasione problemas de integridad con el juego, es decir, que no se permita la modificación de parámetros como los recursos de forma directa.

Nombre del Caso de Uso
Interacción con mundo
Descripción
El sistema deberá permitir al usuario interaccionar con su propio objeto mundo, es decir, ver los recursos disponibles en el mismo así como todo aquello con lo que pueda interactuar. Se deberá, no obstante, restringir el acceso directo a la modificación del mismo siempre que no se espere que el usuario lo haga.

Nombre del Caso de Uso
Modificación de datos
Descripción
El sistema deberá proveer los mecanismos y formularios pertinentes para permitir que el usuario modifique los datos que haya introducido en el sistema con las pertinentes validaciones.

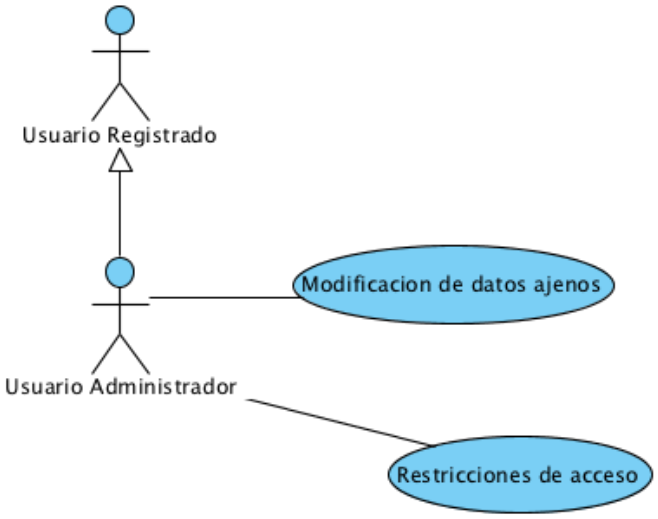


Figura 5.3 Casos de uso de Usuario Administrador

Nombre del Caso de Uso
Modificación de datos ajenos
Descripción
El sistema deberá proveer los mecanismos y formularios pertinentes para permitir que el usuario administrador modifique los datos de otros usuarios. Este caso siempre ha de mostrar un aviso al administrador y solicita que lo confirme antes de proceder a la modificación.

Nombre del Caso de Uso
Restricciones de acceso
Descripción
El sistema deberá proveer los mecanismos y formularios pertinentes para que el usuario administrador pueda restringir el acceso a la aplicación a un determinado usuario. Este caso siempre ha de mostrar un aviso al administrador y solicitar que confirme las acciones a realizar.

5.3 Identificación de los Subsistemas en la Fase de Análisis

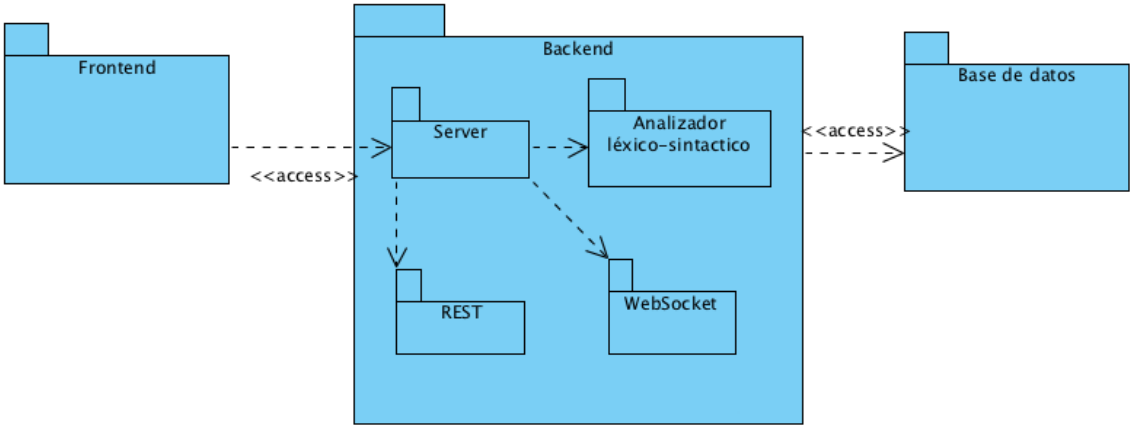


Figura 5.4 Subsistemas

5.3.1 Descripción de los Subsistemas

***Frontend*:**

- Mostrar páginas al usuario: *index*, *login*, registro y juego.
- Enviar peticiones a servidor *backend*.

Base de datos:

- Persistencia de datos.
- Responder consultas.

***Backend*:**

- Atender peticiones de *frontend*.
- Acceder a la base de datos.

Server:

- Atender peticiones entrantes.
- Responder a las peticiones.
- Atender peticiones conforme a usuarios.
- Atender peticiones conforme a otros mensajes.

Analizador léxico-sintáctico:

- Analizar texto y construir código.
- Reportar errores.
- Controlar incorrecciones en el lenguaje.

REST:

- Gestionar alta de usuarios.
- Gestionar modificaciones de usuarios.
- Gestionar borrado de usuarios.
- Gestionar autenticación de usuarios.
- Notificar posibles errores.

WebSocket:

- Solicitar análisis léxico y sintáctico de texto.
- Atender peticiones de clientes.
- Enviar mensajes a clientes.
- Comprobar autenticación de clientes.

5.3.2 Descripción de los Interfaces entre Subsistemas

La comunicación entre los distintos subsistemas se realiza de la forma que sigue:

- Entre los subsistemas *frontend* y *backend* en lo concerniente al subsistema REST se realiza mediante peticiones HTTPS REST, usando llamadas AJAX desde el subsistema *frontend* a las que responde el subsistema *backend* de forma estándar.
- Entre los subsistemas *frontend* y *backend* en lo concerniente al subsistema WebSocket se realiza empleando un WebSocket estándar usando comunicación cifrada.
- Entre los subsistemas *backend* y base de datos se usa una conexión estándar mediante un conector. Aunque ambos se encuentran en el mismo equipo, la conexión se realiza a través de red usando la interfaz *loopback* del sistema.
- Finalmente cabe destacar la conexión entre los subsistemas que componen *backend*, todos ellos son partes de código Python que se comunican entre ellos mediante llamadas a métodos estándar del lenguaje.
- Cada uno de los subsistemas de *backend* puede necesitar acceder a la base de datos en algún momento por lo que se ha creado un objeto que facilite dichas consultas.

5.4 Análisis de Casos de Uso y Escenarios

5.4.1 Caso de uso 1.1: Registro de usuario

Escenario 1.1: Registro de usuario	
Precondiciones	El usuario no debe estar identificado
Poscondiciones	El usuario quedará registrado
Actores	Usuario Anónimo
Descripción	1. El usuario entra en el sitio web y pulsa sobre el apartado de registro. 2. El usuario rellena el formulario. 3. El usuario pulsa el botón de registro. 4. El sistema comprueba que todos los datos estén incluidos. 5. El sistema inicia la inclusión de los datos siempre que estos no estén repetidos. 6. El sistema concluye el registro y devuelve al usuario a la página principal.
Variaciones (escenarios secundarios)	• Si el usuario no ha incluido todos los campos, el sistema notificará los campos que no ha incluido al usuario y solicitará su inclusión. • Si los datos ya se encuentran ingresados se mostrará una notificación al usuario de la situación y se le solicitará que rectifique.
Excepciones	-
Notas	-

5.4.2 Caso de uso 1.2: Conexión al sistema

Escenario 2.1: Conexión al sistema	
Precondiciones	El usuario no debe estar identificado
Poscondiciones	El usuario quedará identificado
Actores	Usuario Anónimo
Descripción	1. El usuario no identificado entra en el sitio web y pulsa sobre el apartado de *login*. 2. El usuario rellena el formulario con su nombre de usuario y contraseña. 3. El usuario pulsa sobre el botón de inicio de sesión. 4. El sistema realiza un hash de la contraseña antes de enviarla. 5. El sistema comprueba que se han introducido ambos datos. 6. El sistema realiza un hash nuevamente de la contraseña. 7. El sistema compara la contraseña del usuario introducida con la almacenada en la base de datos. 8. El sistema crea una cookie en la que introduce el id, el nombre y un *token* generado aleatoriamente además de otras comprobaciones de seguridad. 9. El sistema responde al usuario que el *login* ha sido correcto.
Variaciones (escenarios secundarios)	• Si el usuario no ha incluido todos los campos, el sistema notificará los campos que no ha incluido al usuario y solicitará su inclusión. • Si los datos introducidos no son correctos, el sistema informará del error y permitirá rectificar al usuario.
Excepciones	-
Notas	-

5.4.3 Caso de uso 1.3: Visualización del sitio web

Escenario 3.1: Visualización del sitio web	
Precondiciones	El usuario no debe estar identificado
Poscondiciones	-
Actores	Usuario Anónimo
Descripción	1. El usuario accede al sitio web. 2. El sistema responde al usuario con las páginas con las que éste puede interactuar.
Variaciones (escenarios secundarios)	-
Excepciones	Si el servidor no puede responder se mostrará una página de error.
Notas	-

5.4.4 Caso de uso 1.4: Desconexión

Escenario 4.1: Desconexión	
Precondiciones	El usuario debe estar identificado
Poscondiciones	El usuario pasara a ser un usuario anónimo
Actores	Usuario Registrado
Descripción	1. El usuario entra en el sitio web. 2. El usuario pulsa sobre el apartado de desconexión. 3. El sistema borra la *cookie* del usuario. 4. El sistema borra el *token* del usuario de la base de datos.
Variaciones (escenarios secundarios)	
Excepciones	-
Notas	-

5.4.5 Caso de uso 1.5: Interacción con el juego

Escenario 5.1: Interacción con el juego	
Precondiciones	El usuario debe estar identificado
Poscondiciones	-
Actores	Usuario Registrado
Descripción	1. El usuario accede al sitio web. 2. El usuario pulsa sobre el apartado de juego. 3. El sistema carga la pantalla de juego de forma dinámica. 4. El sistema conecta mediante un WebSocket con la parte *backend*. 5. El sistema comprueba que el usuario se encuentra autenticado, su IP es la misma que cuando hizo *login* y que el *token* es correcto. 6. El sistema cachea los datos del usuario. 7. El sistema envía al usuario su mundo. 8. El sistema envía al usuario su objeto usuario. 9. El sistema queda a la espera de peticiones de compilación por parte del usuario.
Variaciones (escenarios secundarios)	• Si el usuario no se encuentra debidamente autenticado, el sistema notificará el error al usuario mediante el pertinente mensaje y cerrará la conexión.
Excepciones	-
Notas	El proceso de compilación y ejecución se detallan en los correspondientes casos de uso.

5.4.6 Caso de uso 1.6: Compilación de código

Escenario 6.1: Compilación de código	
Precondiciones	El usuario debe estar identificado y haber completado los pasos del caso de uso Interacción con el juego
Poscondiciones	-
Actores	Usuario Registrado
Descripción	1. El usuario introduce código en el editor preparado a tal efecto. 2. El sistema envía dicho código al servidor en un mensaje de tipo Compile. 3. El sistema envía el código del usuario al analizador sintáctico. 4. El analizador divide el código en *tokens* del lenguaje y se lo envía al analizador semántico. 5. El analizador semántico construye un árbol del lenguaje con los *tokens* recibidos y se lo devuelve al sistema. 6. El sistema recorre el árbol identificando las variables utilizadas y uniéndolas con sus definiciones. 7. El sistema rastrea el árbol en busca de posibles errores semánticos. 8. El sistema rastrea las llamadas a función comprobando los parámetros enviados. 9. El sistema genera el código equivalente a dicho código. 10. El sistema notifica al usuario la correcta compilación con un mensaje “compileOK”. 11. El sistema solicita la ejecución del código equivalente con un mensaje “execute”.
Variaciones (escenarios secundarios)	• Si el analizador sintáctico encuentra algún problema durante el análisis estos serán notificados al usuario con el correspondiente mensaje “compileFail”. • Si durante el resto de comprobaciones se produce algún error estos serán reportados al usuario mediante el correspondiente mensaje “compileFail”. • Si en cualquier caso se produce un error, se detendrán las comprobaciones en el momento y se solicitará al usuario que rectifique, no permitiéndole ejecutar código.
Excepciones	-
Notas	El proceso de ejecución se detalla en el correspondiente caso de uso.

5.4.7 Caso de uso 1.7: Ejecución de código

Escenario 7.1: Ejecución de código	
Precondiciones	El usuario debe estar identificado y haber completado los pasos del caso de uso compilación de código
Poscondiciones	-
Actores	Usuario Registrado
Descripción	1. Tras la compilación exitosa el sistema envía un mensaje de tipo *execution* con el código generado en la parte *backend* de la aplicación. 2. El sistema *frontend* recibe el código, el cual ejecuta mostrando el resultado esperado
Variaciones (escenarios secundarios)	• Si durante la ejecución del código se produce algún error, este será capturado y enviado al servidor junto con el código generado para facilitar las tareas de depuración. Igualmente el usuario será notificado.
Excepciones	-
Notas	-

5.4.8 Caso de uso 1.8: Comprobación de errores

Escenario 8.1: Comprobación de errores	
Precondiciones	El usuario debe estar identificado y haber completado los pasos del caso de uso Compilación de código
Poscondiciones	-
Actores	Usuario Registrado
Descripción	1. En caso de que se hayan producido errores estos habrán sido enviados al usuario mediante un mensaje de tipo “compileFail”. 2. Dichos mensajes deberán mostrarse al usuario en la consola creada a tal efecto en la página. 3. El usuario deberá corregir los fallos detectados y repetir la operación de compilación.
Variaciones (escenarios secundarios)	-
Excepciones	-
Notas	-

5.4.9 Caso de uso 1.9: Persistencia de datos

Escenario 9.1: Persistencia de datos	
Precondiciones	El usuario debe estar identificado
Poscondiciones	La base de datos será actualizada
Actores	Usuario Registrado
Descripción	1. El usuario pulsa desconectar o sale del juego. 2. El sistema es notificado mediante un evento de dicha desconexión. 3. El sistema procede a recopilar los datos de la partida cambiados. 4. El sistema actualiza la base de datos con los datos modificados. 5. El sistema cierra la conexión y desconecta al usuario.
Variaciones (escenarios secundarios)	-
Excepciones	-
Notas	El proceso de ejecución se detalla en el correspondiente caso de uso.

5.4.10 Caso de uso 1.10: Interacción con usuario

Escenario 10.1: Interacción con usuario	
Precondiciones	El usuario debe estar identificado y haber completado los pasos del caso de uso Interacción con el juego
Poscondiciones	-
Actores	Usuario Registrado
Descripción	1. El usuario puede introducir código que interactúe con el objeto usuario mediante la clase “me”. 2. El código introducido será enviado al analizador donde seguirá el proceso de compilación. 3. Finalmente se ejecutará en *frontend*.
Variaciones (escenarios secundarios)	• Si el usuario trata de modificar variables que no deben ser modificadas de forma directa como por ejemplo los recursos, se tomara como un error de compilación.
Excepciones	-
Notas	-

5.4.11 Caso de uso 1.11: Interacción con mundo

Escenario 11.1: Interacción con mundo	
Precondiciones	El usuario debe estar identificado y haber completado los pasos del caso de uso Interacción con el juego
Poscondiciones	-
Actores	Usuario Registrado
Descripción	1. El usuario puede introducir código que interactúe con el objeto mundo mediante la clase *World*. 2. El código introducido será enviado al analizador donde seguirá el proceso de compilación. 3. Finalmente se ejecutará en *frontend*.
Variaciones (escenarios secundarios)	• Si el usuario trata de modificar variables que no deben ser modificadas de forma directa como por ejemplo los objetos del mundo, se tomara como un error de compilación.
Excepciones	-
Notas	-

5.4.12 Caso de uso 1.12: Modificación de datos

Escenario 12.1: Modificación de datos	
Precondiciones	El usuario debe estar identificado
Poscondiciones	Los datos del usuario se modificarán
Actores	Usuario Registrado
Descripción	1. El usuario entra al sitio web. 2. El usuario pulsa sobre su perfil. 3. El usuario pulsa sobre el acceso a editar. 4. El usuario introduce los nuevos datos y pulsa confirmar. 5. El sistema envía una petición al servidor con los nuevos datos. 6. El sistema comprueba las credenciales del usuario y modifica los datos.
Variaciones (escenarios secundarios)	• Si el usuario no introduce todos los datos estos no serán modificados y se le avisará del error.
Excepciones	-
Notas	-

5.4.13 Caso de uso 1.13: Modificación de datos ajenos

Escenario 13.1: Modificación de datos ajenos	
Precondiciones	El usuario debe estar identificado
Poscondiciones	Los datos del usuario se modificarán
Actores	Usuario Administrador
Descripción	1. El usuario administrador accede a la página de administración. 2. El usuario administrador se autentica con sus credenciales. 3. El sistema comprueba las credenciales del usuario y comprueba que el rol del usuario sea efectivamente de administrador. 4. El administrador introduce el usuario a modificar. 5. El sistema recupera los datos del usuario a modificar por parte del administrador. 6. El administrador modifica los datos en cuestión y pulsa sobre el botón de confirmación. 7. El sistema comprueba la autoridad del administrador para realizar el cambio. 8. El sistema realiza el cambio.
Variaciones (escenarios secundarios)	• Si el administrador no dispone de la autoridad suficiente para realizar el cambio, el sistema le notificará de la situación. • Si el usuario no existe se mostrará al administrador una notificación sobre la situación.
Excepciones	-
Notas	-

5.4.14 Caso de uso 1.14: Restricciones de acceso

Escenario 14.1: Restricciones de acceso	
Precondiciones	El usuario debe estar identificado
Poscondiciones	El acceso del usuario objetivo será restringido
Actores	Usuario Administrador
Descripción	1. El usuario administrador accede a la página de administración. 2. El usuario administrador se autentica con sus credenciales. 3. El sistema comprueba las credenciales del usuario y comprueba que el rol del usuario sea efectivamente de administrador. 4. El administrador introduce el nombre del usuario a restringir. 5. El sistema recupera los datos del usuario. 6. El administrador pulsa sobre el botón de bloquear. 7. El sistema comprueba las credenciales del administrador y realiza el bloqueo.
Variaciones (escenarios secundarios)	• Si el usuario no existe se mostrará al administrador una notificación sobre la situación.
Excepciones	-
Notas	-

5.5 Análisis de Interfaces de Usuario

5.5.1 Descripción de la Interfaz

5.5.1.1 Interfaz 1: Página principal

Figura 5.5 Mockup Página Inicial

La página principal mostrará únicamente un botón que redirigirá al juego, igualmente se incluirán los enlaces al *login* que cambiará a *logout* si se encuentra conectado, el perfil siempre que esté conectado y la página de registro si está desconectado.

El fondo de la página podrá ser un ejemplo del mundo del juego de forma de imagen estática.

5.5.1.2 Interfaz 2: Login

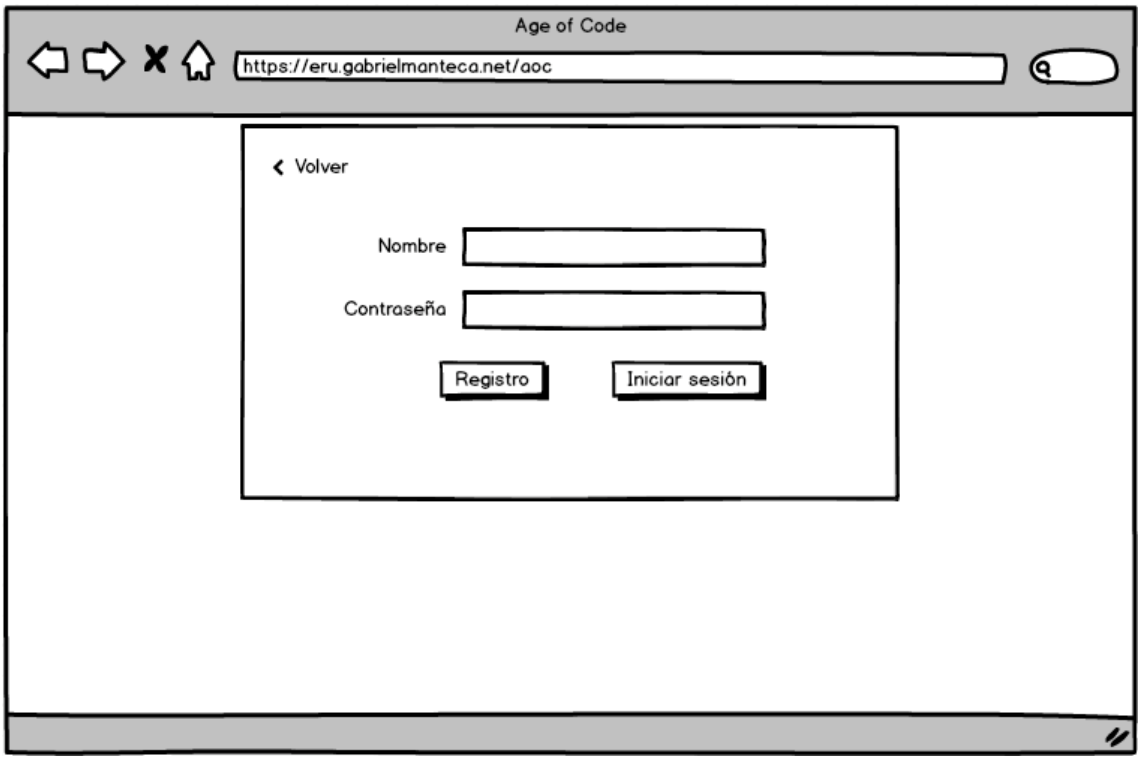


Figura 5.6 Mockup Login

La página de *login* sencillamente muestra un formulario que solicita el nombre de usuario y la contraseña del mismo. A su vez provee enlaces para volver a la página principal y un enlace a la página de registro por si el usuario desea registrarse.

5.5.1.3 Interfaz 3: Registro de usuario

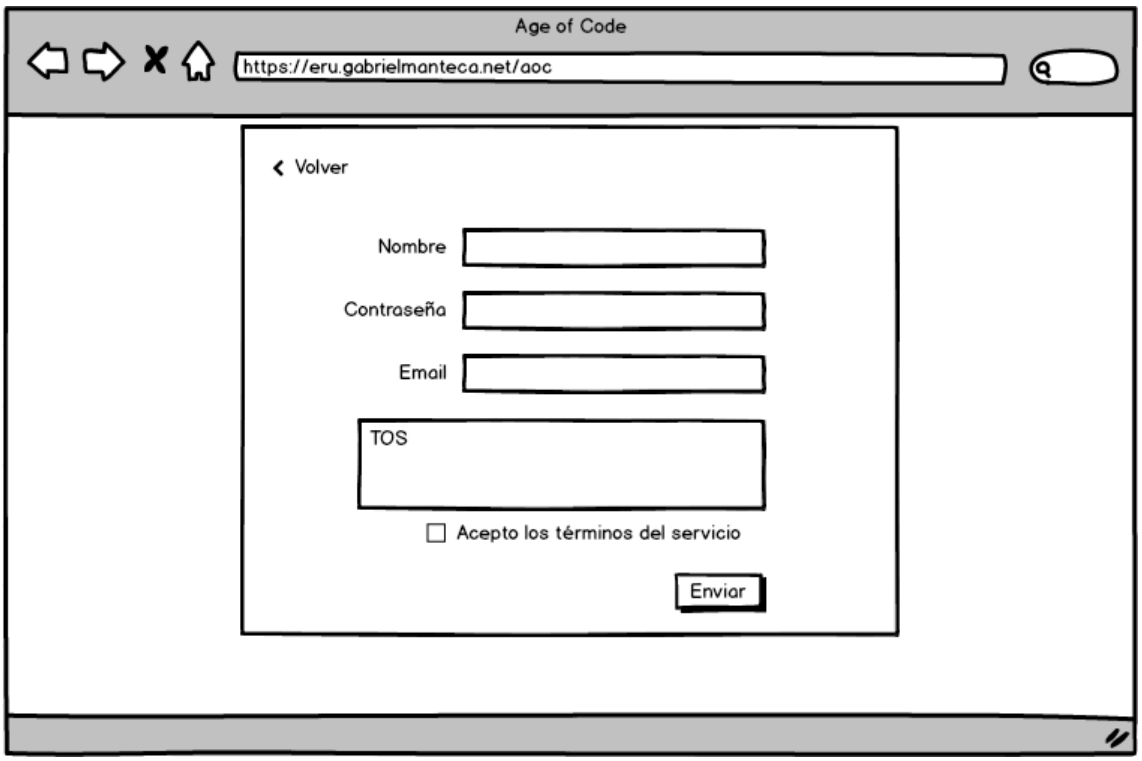


Figura 5.7 Mockup Registro

La interfaz de registro muestra los campos mínimos a rellenar por el usuario así como los términos del sistema informando de las condiciones y las leyes como la LOPD que rigen el sitio. También se dispone de un botón para volver a la página principal.

5.5.1.4 Interfaz 4: Página de juego

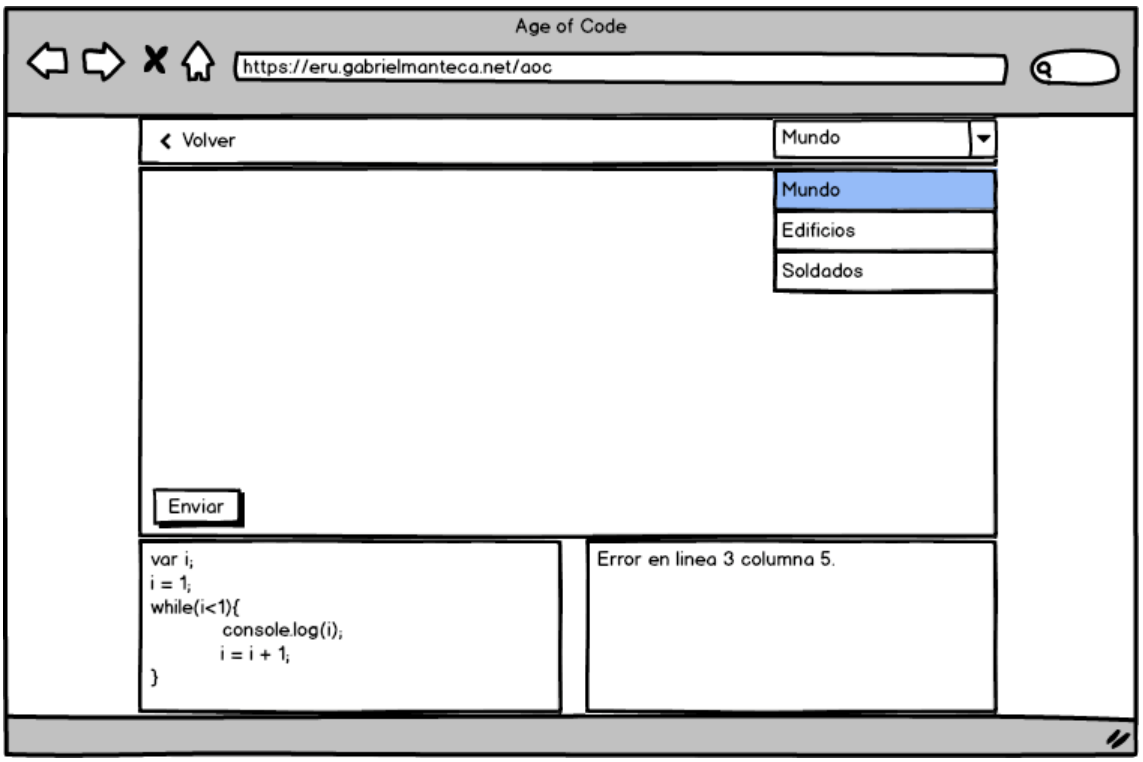


Figura 5.8 Mockup Juego

La página de juego presenta la funcionalidad más compleja. En la parte inferior izquierda de la misma se presenta el editor de texto en el cual el usuario puede introducir el código para enviarlo al servidor a ser analizado. En la parte derecha en cambio es donde se muestran los mensajes de error y de consola que el usuario ejecute.

En la parte central de la página se mostrará el estado del juego en tiempo real mediante una representación 2D del mismo y en forma de rejilla. Finalmente se podrá ver el estado del mundo en formato JSON desplegando el menú de la parte superior derecha.

5.5.2 Descripción del Comportamiento de la Interfaz

Los mensajes de error producidos durante la ejecución del programa se mostrarán al usuario en el apartado dedicado a tal efecto, una línea en rojo para los formularios o una línea de error en la consola del juego en el caso de ser la pantalla de juego.

Todas las páginas se interconectarán a través del *index* debido a la escasa cantidad de páginas web de las que dispone el sitio.

5.5.3 Diagrama de Navegabilidad

A continuación se muestra, mediante un diagrama de tipo máquina de estados, la interacción que existe entre las diferentes páginas del sitio descritas anteriormente.

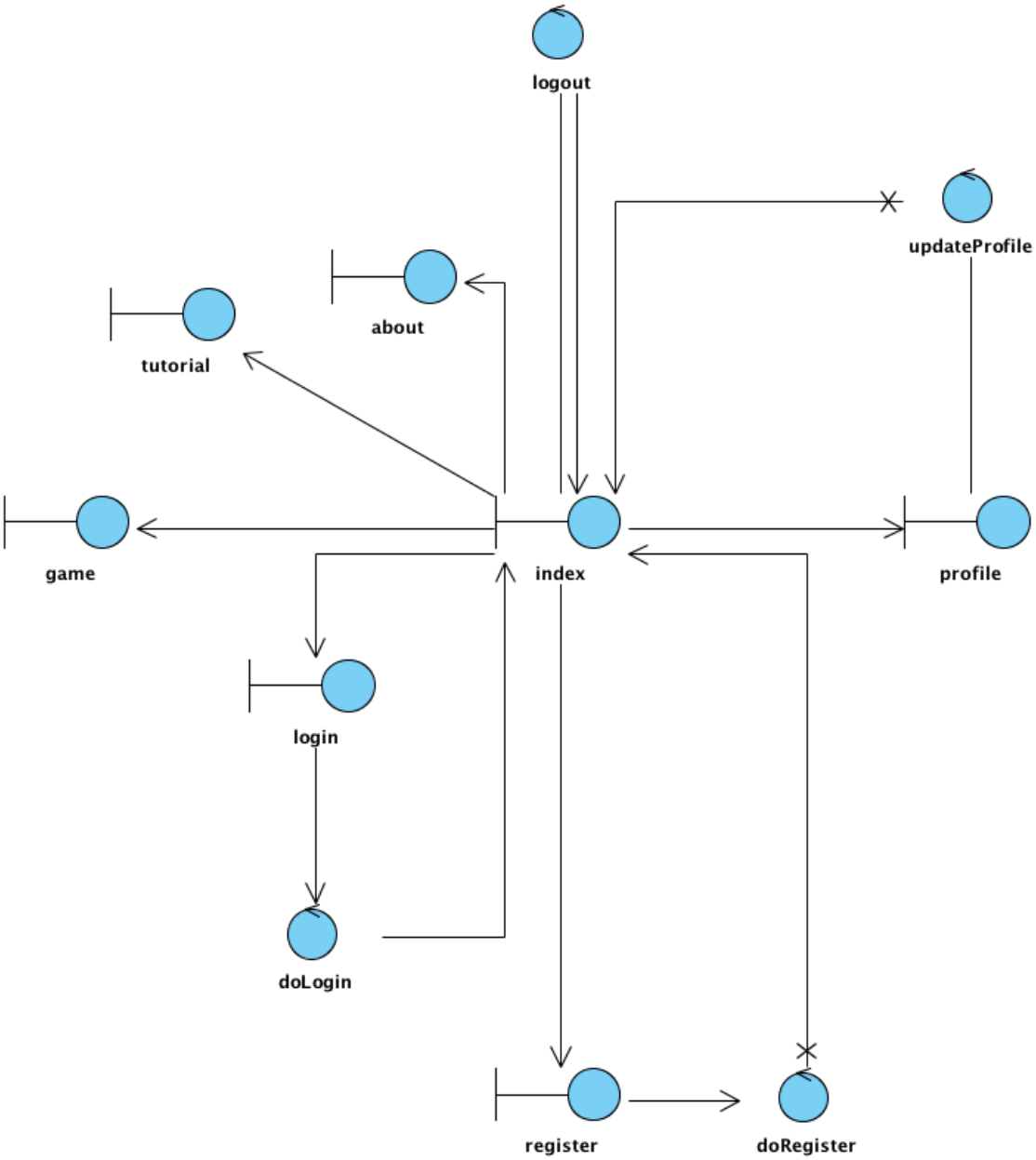


Figura 5.9 Diagrama de Navegabilidad

El diagrama anterior se muestra simplificado con el único objeto de mostrar los enlaces entre las páginas principales.

5.6 Especificación del Plan de Pruebas

Al emplearse una metodología ágil, se ejecutaran las pruebas correspondientes a cada módulo al acabarlo para así poder confirmar su correcta programación y en caso de no ser así, realizar los cambios oportunos.

Se realizarán distintos tipos de pruebas, indicados a continuación

5.6.1 Pruebas unitarias

Se emplearán *frameworks* para la realización de pruebas unitarias principalmente en la gramática del *backend*, dado que es el punto en el que más fallos pueden detectar este tipo de pruebas.

Las pruebas tratarán de incluir muestras de todo el tipo de código soportado en el proyecto, para tratar de detectar posibles fallos introducidos durante las distintas iteraciones del desarrollo.

Para la creación de dichas pruebas se empleará el módulo *unittest* de Python y el paquete *nosetests* del mismo.

5.6.2 Pruebas de integración

Se realizarán principalmente en la interconexión entre *frontend* y *backend*, igualmente se estudiará el correcto funcionamiento del código introducido por el usuario.

Estas pruebas se realizarán siempre en su totalidad al acabar cada módulo, comprobando que las pruebas realizadas anteriormente se siguen realizando de forma correcta y en caso de no ser así proceder a las correspondientes correcciones.

A continuación se describen las pruebas mínimas a realizar en el desarrollo del sistema. Este plan puede ampliarse durante las distintas iteraciones si se considera necesario.

Caso de Uso 1: Registro de usuario	
Prueba	**Resultado Esperado**
Registrar un usuario no existente	Se crea un nuevo usuario y podrá proceder a iniciar sesión.
Prueba	**Resultado Esperado**
Registrar un usuario existente	El sistema notificará el error al usuario mediante un mensaje y le permitirá cambiar los parámetros de registro.

Caso de Uso 2: Conexión al sistema	
Prueba	**Resultado Esperado**
Introducir un usuario y contraseña correctos	El sistema autentica al usuario y le redirige a la página de inicio.
Prueba	**Resultado Esperado**
Introducir un usuario o contraseña incorrectos	El sistema informa al usuario de que existe un error con los datos provistos y permite que el usuario rectifique o cree un nuevo usuario.
Prueba	**Resultado Esperado**
Introducir un usuario y contraseña inexistentes	El sistema informa al usuario de que existe un error con los datos provistos y permite que el usuario rectifique o cree un nuevo usuario.

Caso de Uso 3: Visualización del sitio web	
Prueba	**Resultado Esperado**
Acceder al sitio web	El sitio web será visualizado.

Caso de Uso 4: Desconexión	
Prueba	**Resultado Esperado**
Desconectar a un usuario	El sistema desconecta al usuario.

Caso de Uso 5: Interacción con el juego	
Prueba	**Resultado Esperado**
Iniciar el juego	El sistema envía el mundo y el usuario e inicializa la conexión con *backend*.
Prueba	**Resultado Esperado**
Interactuar con el juego	El sistema reconoce el código insertado y lo convierte en plantillas de ejecución correctas para su posterior uso en *frontend*.

Caso de Uso 6: Compilación de código	
Prueba	**Resultado Esperado**
Introducción de código sin errores	El código se compila sin errores y se ejecuta obteniendo el resultado esperado
Prueba	**Resultado Esperado**
Introducción de código con errores	El código no se compila ni ejecuta y el sistema reporta los errores léxicos, sintácticos o semánticos detectados.

Caso de Uso 9: Persistencia de datos	
Prueba	**Resultado Esperado**
Modificación de datos de la partida del usuario y desconexión del mismo	El sistema recopila los cambios y actualiza la partida. Cuando el usuario solicite jugar otra vez se le mostrará la partida modificada.

Caso de Uso 10: Interacción con usuario	
Prueba	**Resultado Esperado**
Consultar datos del usuario	El sistema retornará el valor correcto solicitado sobre el usuario siempre que se disponga acceso al mismo, en caso contrario mostrará un error.
Prueba	**Resultado Esperado**
Editar datos del usuario	El usuario será correctamente modificado si dichos datos son modificables, en caso contrario mostrará un error de compilación al usuario.

Caso de Uso 11: Interacción con mundo	
Prueba	**Resultado Esperado**
Consultar datos del mundo	El sistema retornará el valor correcto solicitado sobre el mundo siempre que se disponga acceso al mismo, en caso contrario mostrará un error.
Prueba	**Resultado Esperado**
Editar datos del mundo	El mundo será correctamente modificado si dichos datos son modificables, en caso contrario mostrará un error de compilación al usuario.

Caso de Uso 12: Modificación de datos	
Prueba	**Resultado Esperado**
Un usuario registrado y autenticado realiza un cambio de sus datos con datos válidos	El sistema los valida y realiza el cambio, permitiendo que el usuario se mantenga *logueado* y se pueda conectar la próxima vez con los nuevos datos.
Prueba	**Resultado Esperado**
Un usuario registrado y autenticado realiza un cambio de sus datos con datos inválidos	El sistema trata de validarlos e informa al usuario de un error en los mismos permitiéndole corregirlos.

Caso de Uso 13: Modificación de datos ajenos	
Prueba	**Resultado Esperado**
Un usuario administrador realiza un cambio de los datos de un usuario con datos válidos	El sistema los valida y realiza el cambio, permitiendo que el usuario se pueda conectar la próxima vez con los nuevos datos.
Prueba	**Resultado Esperado**
Un usuario administrador realiza un cambio de los datos de un usuario con datos inválidos	El sistema trata de validarlos e informa al administrador de un error en los mismos permitiéndole corregirlos.

Caso de Uso 14: Restricciones de acceso	
Prueba	**Resultado Esperado**
Un usuario administrador solicita la restricción de acceso de un usuario existente	El sistema comprueba la existencia del usuario, realiza la restricción al usuario e informa al administrador.
Prueba	**Resultado Esperado**
Un usuario administrador solicita la restricción de acceso de un usuario inexistente	El sistema comprueba la inexistencia del usuario e informa al administrador de la inexistencia del usuario.

Capítulo 6. Módulos del sistema

6.1 Módulo: Construcción del servidor Backend

6.1.1 Diagrama de diseño del módulo

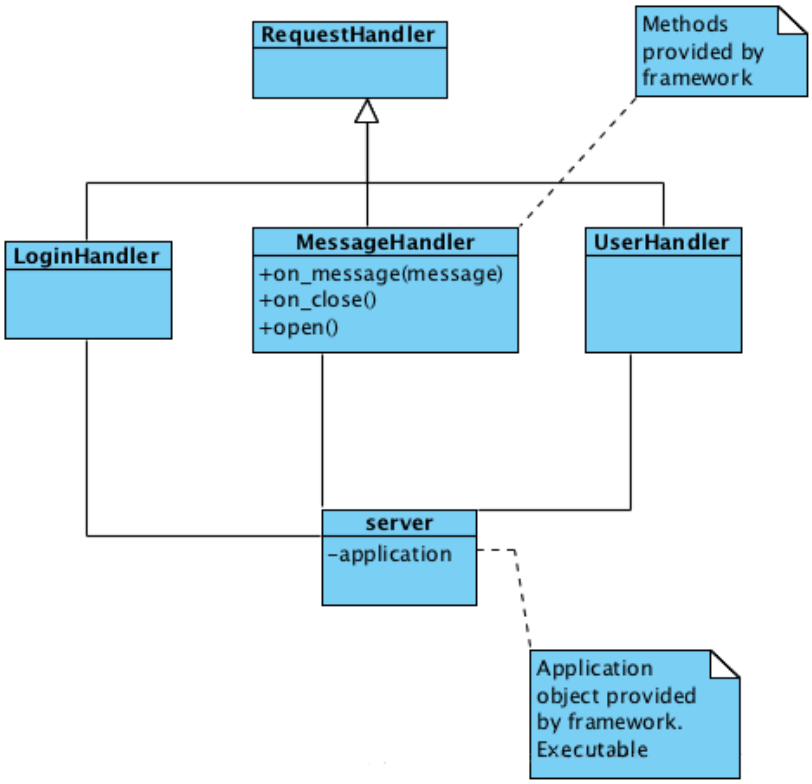


Figura 6.1 Diagrama del Backend

El servidor *backend* dispondrá de un punto central en la clase server. Esta clase dispone del atributo *application* en el que se especifican los puntos de acceso a la API, que serán atendidos por los distintos *handlers* definidos en el sistema.

Nombre de la Clase
server
Descripción
Punto inicial de la aplicación *backend*, es el archivo que se ejecutará como *main* e incluye la lógica de los puntos de acceso a la API.
Responsabilidades
Servir de punto de entrada a la aplicación
Atributos Propuestos
application: Objeto que provee la lógica del *framework* del servidor. Provisto por el *framework* elegido.
Métodos Propuestos

Nombre de la Clase
UserHandler
Descripción
Encargada de controlar aquellas peticiones contra el punto de entrada *user*. Gestiona toda la lógica relacionada con usuarios
Responsabilidades
Gestionar toda la lógica relacionada con usuarios.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
MessageHandler
Descripción
Encargada de controlar aquellas peticiones contra el punto de entrada *wss*. Gestionará toda la lógica relacionada con el WebSocket de la aplicación.
Responsabilidades
Atender, controlar y responder aquellas peticiones recibidas mediante el WebSocket de forma correcta.
Atributos Propuestos
Métodos Propuestos
on_message: Atiende un nuevo mensaje del WebSocket **on_close**: Notifica el cierre de una conexión en el WebSocket **open**: Procesa la apertura de una nueva conexión en el WebSocket

Nombre de la Clase
LoginHandler
Descripción
Encargada de recibir peticiones de *login* al sistema.
Responsabilidades
Controlar el acceso y autenticación de los usuarios al sistema.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
RequestHandler
Descripción
Provista por el *framework*.
Responsabilidades
Proveer la lógica de todos los manejadores de peticiones http.
Atributos Propuestos
Métodos Propuestos

6.2 Módulo: Construcción del *frontend*

6.2.1 Diagrama de diseño del modulo

El diagrama siguiente se encuentra simplificado únicamente para facilitar su lectura y comprensión.

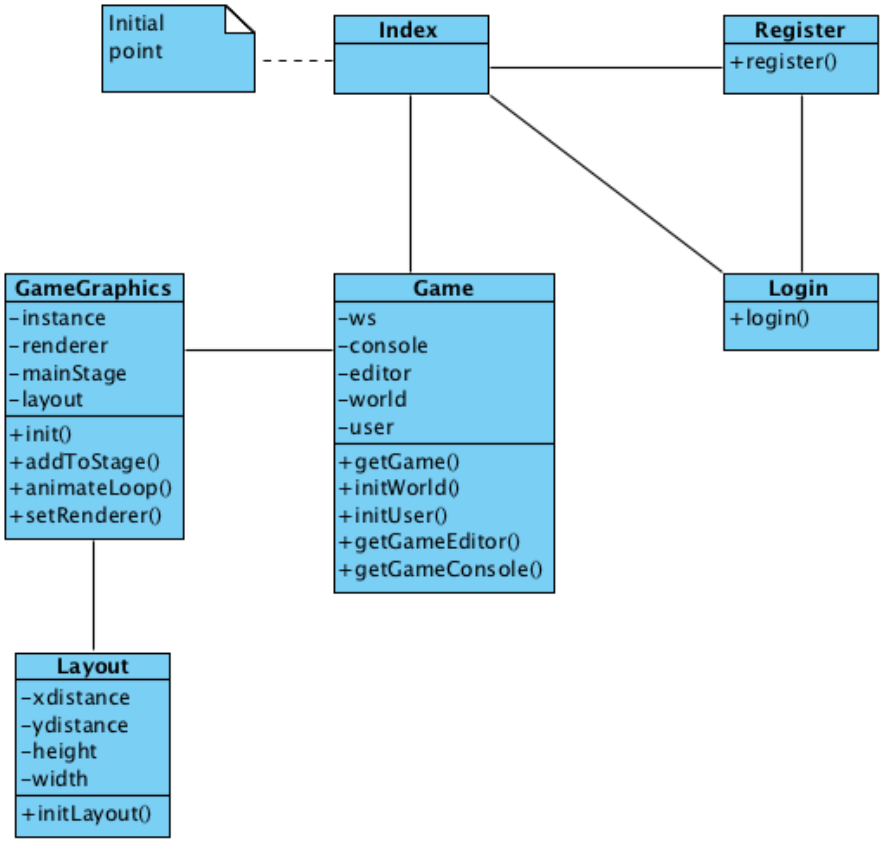


Figura 6.2 Diagrama del frontend

Nombre de la Clase
Index
Descripción
Página inicial de acceso al sistema
Responsabilidades
Proveer información sobre el sitio y acceso a los demás espacios del mismo.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
Register
Descripción
Página de registro del sistema, se encargará de proveer toda la información relevante del mismo y los medios para realizarlo.
Responsabilidades
Proveer la lógica y la información necesarias para realizar el registro de un nuevo usuario.
Atributos Propuestos
Métodos Propuestos
register: Enviará al servidor la información de registro del nuevo usuario y controlará los posibles errores que puedan producirse.

Nombre de la Clase
Login
Descripción
Página de autenticación del sistema, se encargara de proveer la lógica de autenticación del mismo
Responsabilidades
Proveer la lógica para realizar la autenticación del sistema.
Atributos Propuestos
Métodos Propuestos
login: Enviará al servidor la información de autenticación del nuevo usuario y controlará los errores que se produzcan en el *login*.

Nombre de la Clase
Game
Descripción
Vista encargada de la representación del juego y de la interacción con el mismo.
Responsabilidades
Proveer la lógica de visualización del videojuego y de interacción con el mismo.
Atributos Propuestos
ws: WebSocket empleado para la conexión con el servidor. **console**: Consola del usuario en la que aparecerán todos aquellos mensajes que se le deban transmitir. **editor**: Editor con el que interactuará el usuario. **world**: Mundo del usuario con el que podrá interactuar. **user**: Objeto usuario del usuario conectado actualmente.
Métodos Propuestos
getGame: Recupera la instancia de este objeto. **initWorld**: Inicializa el mundo con los datos recibidos. **initUser**: Inicializa el usuario con los datos recibidos. **getGameEditor**: Recupera el objeto editor con el que interactuará el usuario. **getGameConsole**: Recupera el objeto consola que enviará mensajes al usuario.

Nombre de la Clase
GameGraphics
Descripción
Clase utilidad encargada de la representación gráfica del juego y de todo aquello relacionado con los gráficos 2D del mismo.
Responsabilidades
Facilitar la interacción con los gráficos 2D.
Atributos Propuestos
instance: Instancia de esta clase única. **renderer**: *Renderer* de la escena. **mainStage**: Escena principal del juego. **layout**: Rejilla e interfaz del juego
Métodos Propuestos
init: Inicializa los gráficos del juego. **addToStage**: Añade el objeto a la escena. **animateLoop**: Bucle de animación de la escena. **setRenderer**: Establece el *renderer* de la escena y las propiedades del mismo.

Nombre de la Clase
Layout
Descripción
Encargada de dibujar la rejilla de juego y de inicializar la interfaz de los gráficos del juego.
Responsabilidades
Dibujar la rejilla del juego e inicializar la interfaz gráfica.
Atributos Propuestos
xdistance: Distancia en x de las casillas del juego. **ydistance**: Distancia en y de las casillas del juego. **width**: Ancho total del *renderer*. **height**: Altura total del *renderer*.
Métodos Propuestos
initLayout: Inicializa la pantalla de juego y su interfaz.

6.3 Módulo: Definir el analizador léxico y sintáctico

6.3.1 Diagramas de diseño del módulo

Este módulo se ha dividido en distintos diagramas para facilitar su lectura y comprensión. Estos sub módulos son el analizador sintáctico y semántico y el AST del lenguaje.

6.3.1.1 Estructura del analizador general

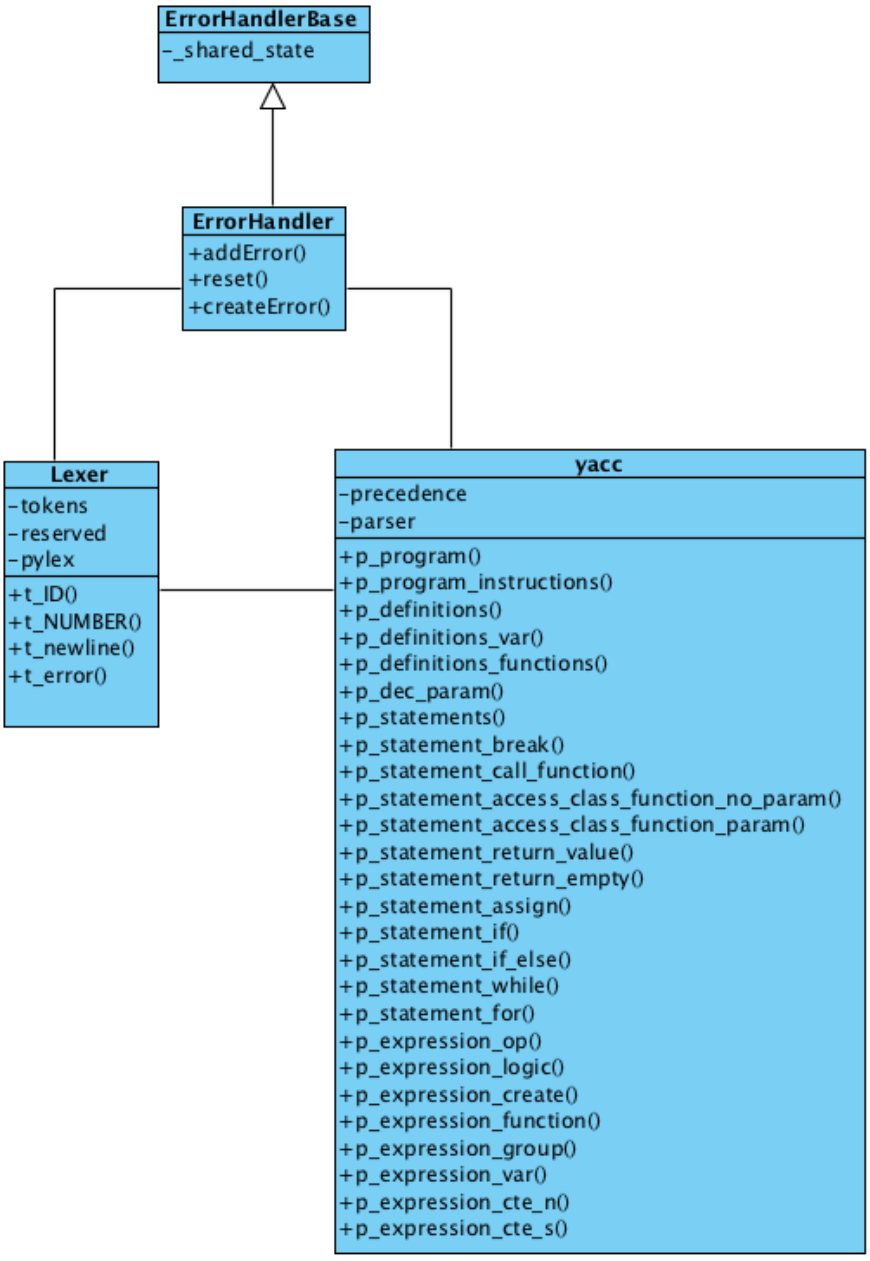


Figura 6.3 Analizador léxico y sintáctico

Nombre de la Clase
Lexer
Descripción
Se encargará de trabajar como la herramienta *flex*, dividiendo el texto introducido en *tokens* del lenguaje.
Responsabilidades
Dividir el texto introducido en *tokens* del lenguaje y reportar aquellos *tokens* no conocidos al manejador de errores.
Atributos Propuestos
tokens: Lista de *tokens* general que se esperan encontrar en el lenguaje. **reserved**: Lista de palabras reservadas del lenguaje. **pylex**: Objeto que contiene todo el analizador léxico. Provisto por el *framework*
Métodos Propuestos
Los distintos métodos que figuran en el diagrama se encargan de realizar distintas comprobaciones sobre los determinados *tokens* introducidos. Todos ellos vienen forzados por la biblioteca empleada.

Nombre de la Clase
yacc
Descripción
Se encargará de reconocer sentencias del lenguaje a partir de los *tokens* detectados en la clase *lexer*, construir el AST y reportar los errores encontrados.
Responsabilidades
Reconocimiento de expresiones del lenguaje, construcción del AST y reporte de errores.
Atributos Propuestos
precedence: Lista de precedencias en el caso de conflictos a la hora de realizar el reconocimiento. **parser:** Objeto general encargado del reconocimiento del lenguaje, ofrecido por la biblioteca.
Métodos Propuestos
Todos los métodos se encuentran forzados por la biblioteca y se basan en la documentación de Python para crear la gramática. No se considera relevante de explicar, para más detalle por favor consulten los anexos técnicos.

Nombre de la Clase
ErrorHandler
Descripción
Encargada de registrar todos los errores producidos durante el análisis de lenguaje y reportarlos aportando la mayor cantidad posible de información.
Responsabilidades
Registro de errores producidos en el análisis.
Atributos Propuestos
Métodos Propuestos
addError: Añade un nuevo error al manejador. **reset:** Vacía la lista de errores del manejador. **createError:** Crea un nuevo error y lo introduce en la lista de errores del manejador.

Nombre de la Clase
ErrorHandlerBase
Descripción
Nexo de unión entre todos los manejadores de error que comparten estado.
Responsabilidades
Compartir datos entre todos los manejadores de error.
Atributos Propuestos
_shared_state: diccionario que contiene el estado de cada uno de los manejadores.
Métodos Propuestos

6.3.1.2 Definiciones en el AST

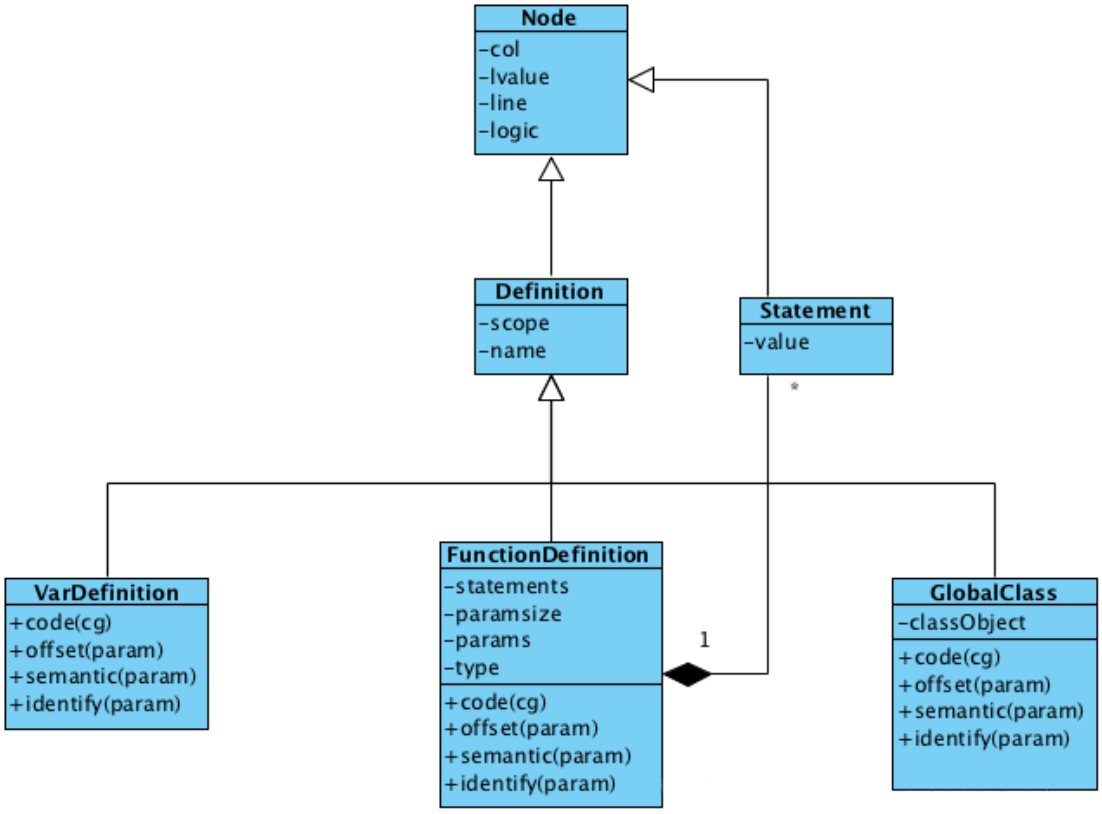


Figura 6.4 Definiciones en el AST

Nombre de la Clase
Node
Descripción
Nodo base del AST, incluye información genérica común a todos los nodos del árbol.
Responsabilidades
Servir de base para los nodos del AST.
Atributos Propuestos
col: Indica el número de columna en el que se ha encontrado el nodo. **lvalue:** Indica si el nodo puede encontrarse a la izquierda de una asignación. **line:** Indica el número de línea en el que se ha encontrado el nodo. **logic:** Indica si el nodo tiene un valor booleano o no.
Métodos Propuestos

Nombre de la Clase
Definition
Descripción
Nodo base de todas las definiciones encontradas en el código.
Responsabilidades
Servir de base para todas las definiciones.
Atributos Propuestos
scope: Indica el ámbito en el que se encuentra la definición del elemento. **name:** Indica el nombre del elemento definido.
Métodos Propuestos

Nombre de la Clase
VarDefinition
Descripción
Nodo que representa la definición de una variable
Responsabilidades
Representar la definición de una variable
Atributos Propuestos
Métodos Propuestos
code: Encargado de la creación del código de este nodo. **offset**: Este método no realiza acción en este nodo. **semantic**: Este método no realiza acción en este nodo. **identify**: Incluye la definición de la variable como un identificador nuevo.

Nombre de la Clase
FunctionDefinition
Descripción
Nodo que representa la definición de una función.
Responsabilidades
Representar la definición de una función.
Atributos Propuestos
statements: Conjunto de sentencias dentro de la función definida. **paramsize:** Numero de parámetros que recibe la función. **params**: Lista de parámetros que recibe la función. **type**: Tipo devuelto por la función.
Métodos Propuestos
code: Encargado de la creación del código de este nodo. **offset**: Calcula el número de parámetros que debe recibir la función. **semantic**: Determina el tipo de retorno de la función. **identify**: Incluye la definición de la función como un identificador nuevo, al igual que sus parámetros.

Nombre de la Clase
GlobalClass
Descripción
Nodo que representa una clase global del sistema.
Responsabilidades
Representar una clase global del sistema
Atributos Propuestos
classObject: Instancia de la clase global a la que representa.
Métodos Propuestos
code: Lanza un aviso al desarrollador que indica que se ha alcanzado este nodo en este método que ayude a depurar. **offset**: Lanza un aviso al desarrollador que indica que se ha alcanzado este nodo en este método que ayude a depurar. **semantic**: Lanza un aviso al desarrollador que indica que se ha alcanzado este nodo en este método que ayude a depurar. **identify**: Lanza un aviso al desarrollador que indica que se ha alcanzado este nodo en este método que ayude a depurar.

6.3.1.3 Expresiones en el AST

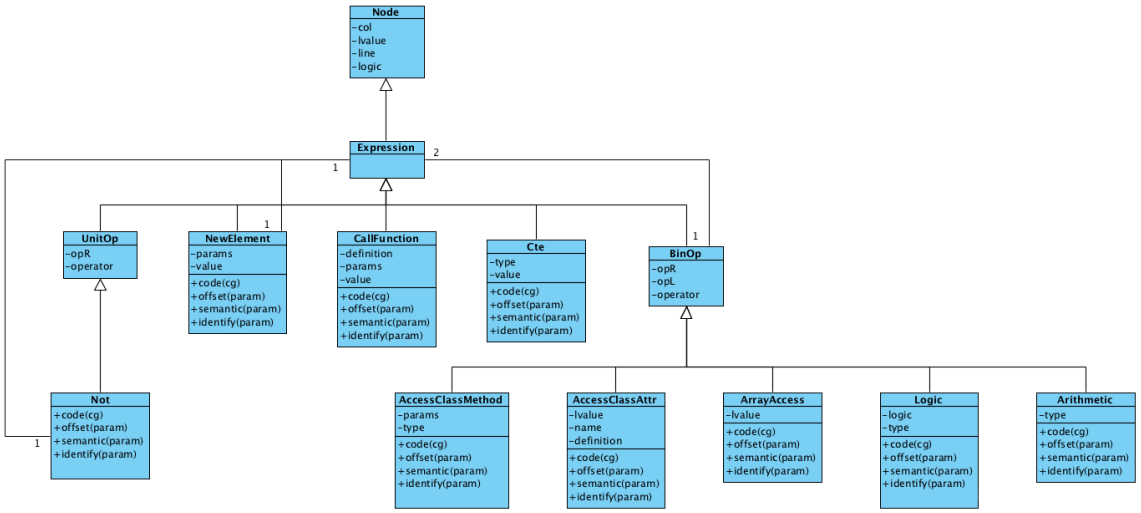


Figura 6.5 Expresiones en el AST

Nombre de la Clase
Expression
Descripción
Nodo que representa una expresión del lenguaje.
Responsabilidades
Actúa como clase base de todas las expresiones del lenguaje.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
NewElement
Descripción
Nodo que representa la instanciación de una clase.
Responsabilidades
Representar una nueva instanciación.
Atributos Propuestos
params: Parámetros de la nueva instancia. **value**: Clase a instanciar.
Métodos Propuestos
code: Encargado de la creación del código de este nodo. **offset**: Calcula el número de parámetros que debe recibir la instancia. **semantic**: Establece el tipo del identificador. **identify**: Comprueba que la clase sea conocida.

Nombre de la Clase
CallFunction
Descripción
Nodo que representa una llamada a una función que retorna valor.
Responsabilidades
Representar una llamada a una función.
Atributos Propuestos
params: Parámetros de la llamada. **value**: Función a llamar. **definition**: Definición de la función a ejecutar.
Métodos Propuestos
code: Encargado de la creación del código de este nodo. **offset**: Calcula el número de parámetros que debe recibir la invocación. **semantic**: No realiza acción. **identify**: Comprueba que la función sea conocida.

Nombre de la Clase
Cte
Descripción
Nodo que representa una constante, numérica o cadena
Responsabilidades
Representar una constante
Atributos Propuestos
type: Tipo de la constante, numérica o cadena. **value**: Valor de la constante.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: No realiza acción. **identify**: No realiza acción.

Nombre de la Clase
UnitOp
Descripción
Nodo que representa una operación unaria
Responsabilidades
Representar una operación unaria
Atributos Propuestos
opR: Operando de la parte derecha de la expresión. **operator**: Operador de la operación.
Métodos Propuestos

Nombre de la Clase
Not
Descripción
Nodo que representa la negación de una expresión
Responsabilidades
Representar la negación de una expresión
Atributos Propuestos
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Comprueba que el operando derecho sea un valor lógico para poder ser negado. **identify**: No realiza acción.

Nombre de la Clase
BinOp
Descripción
Nodo que representa una operación entre dos operandos.
Responsabilidades
Representar una operación entre dos operandos
Atributos Propuestos
opR: Operando de la parte derecha de la expresión. **opL**: Operando de la parte izquierda de la expresión. **operator**: Operador de la expresión.
Métodos Propuestos

Nombre de la Clase
AccessClassMethod
Descripción
Nodo que representa un acceso a un método de una clase.
Responsabilidades
Representar un acceso a un método de una clase.
Atributos Propuestos
type: Tipo de dato devuelto por el método de la clase. **params**: Parámetros recibidos por el método.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: Comprueba el número de parámetros pasados al método. **semantic**: Recupera el tipo de dato devuelto por el método. **identify**: Comprueba que no se esté accediendo a una clase o método desconocidos.

Nombre de la Clase
AccessClassAttr
Descripción
Nodo que representa un acceso a un atributo de una clase.
Responsabilidades
Representar un acceso a un atributo de una clase.
Atributos Propuestos
lvalue: Indica si este atributo es editable. **name**: Nombre del acceso a atributo. **definition**: Definición de la clase a la que se accede.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Comprueba que el atributo puede ser asignable. **identify**: Comprueba que existe la clase y que existe el atributo.

Nombre de la Clase
Logic
Descripción
Nodo que representa una operación lógica con resultado booleano.
Responsabilidades
Representar una operación lógica.
Atributos Propuestos
logic: Indica si la operación es lógica. **type**: Especifica el tipo de la operación.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Comprueba que los operandos se pueden operar de forma lógica. **identify**: No realiza acción.

Nombre de la Clase
Arithmetic
Descripción
Nodo que representa una operación aritmética.
Responsabilidades
Representar una operación aritmética.
Atributos Propuestos
type: Indica el tipo de la operación aritmética.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Comprueba que el resultado de los operandos sea de tipo *number*. **identify**: No realiza acción.

6.3.1.4 Sentencias en el AST

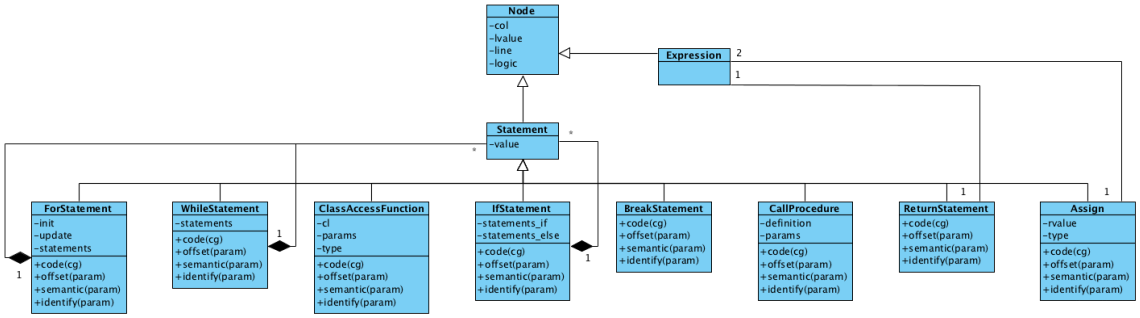


Figura 6.6 Sentencias en el AST

Nombre de la Clase
Statement
Descripción
Nodo que representa todas las sentencias del lenguaje.
Responsabilidades
Representar la base de las sentencias del lenguaje.
Atributos Propuestos
value: Representa las condiciones, métodos de acceso u otros atributos de las sentencias.
Métodos Propuestos

Nombre de la Clase
ForStatement
Descripción
Nodo que representa un bucle *for* del lenguaje.
Responsabilidades
Representar un bucle *for*.
Atributos Propuestos
init: Indica el primer bloque de inicialización del bucle *for*. **update**: Indica el valor de actualización del bucle *for*. **statements**: Lista de sentencias a ejecutar en el bucle.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Comprueba que el atributo *value* sea de tipo lógico. **identify**: No realiza acción.

Nombre de la Clase
WhileStatement
Descripción
Nodo que representa un bucle *while*.
Responsabilidades
Representar un bucle *while*.
Atributos Propuestos
statements: Lista de sentencias a ejecutar en el bucle.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Comprueba que el atributo *value* sea lógico. **identify**: No realiza acción.

Nombre de la Clase
ClassAccessFunction
Descripción
Nodo que representa un acceso a un método de una clase.
Responsabilidades
Representar un acceso a un método de una clase.
Atributos Propuestos
type: Indica el tipo devuelto por el método de la clase. **params**: Lista de parámetros a recibir por el método. **cl**: Instancia de la clase a la que representa.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: Comprueba el número de parámetros pasados al método. **semantic**: establece el tipo de retorno de la sentencia. **identify**: Comprueba que la clase y el método sean conocidos.

Nombre de la Clase
IfStatement
Descripción
Nodo que representa una sentencia *If*.
Responsabilidades
Representar una sentencia *If*.
Atributos Propuestos
statements_if: Sentencias a ejecutar si la condición se cumple. **statements_else**: Sentencias a ejecutar si la condición no se cumple.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Comprueba que la condición del *if* sea de tipo lógico. **identify**: No realiza acción.

Nombre de la Clase
BreakStatement
Descripción
Nodo que representa una sentencia break.
Responsabilidades
Representar una sentencia break.
Atributos Propuestos
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: No realiza acción. **identify**: No realiza acción.

Nombre de la Clase
CallProcedure
Descripción
Nodo que representa una llamada a procedimiento que no retorna valores.
Responsabilidades
Representar una llamada a procedimiento.
Atributos Propuestos
definition: Definición de la función a llamar. **params**: Lista de parámetros a recibir por la función.
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: Comprueba que el número de parámetros pasado sea correcto. **semantic**: No realiza acción. **identify**: Comprueba que la función a llamar se encuentre definida.

Nombre de la Clase
ReturnStatement
Descripción
Nodo que representa una operación de retorno.
Responsabilidades
Representar una operación de retorno.
Atributos Propuestos
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Retorna el tipo de retorno de esta función para establecerlo en la definición. **identify**: No realiza acción.

Nombre de la Clase
Assign
Descripción
Nodo que representa una sentencia de asignación.
Responsabilidades
Representar una sentencia de asignación.
Atributos Propuestos
type: Indica el tipo resultante en la asignación. **rvalue**: Expresión a la derecha de la asignación
Métodos Propuestos
code: Encargado de la creación de código de este nodo. **offset**: No realiza acción. **semantic**: Comprueba que el elemento *value* sea *lvalue* y establece el tipo de la parte izquierda de la asignación y de la asignación en general. **identify**: No realiza acción.

6.3.1.5 Otras clases del AST

Con el único objeto de simplificar los diagramas anteriores se incluyen a continuación las clases utilidad que se emplean en el AST.

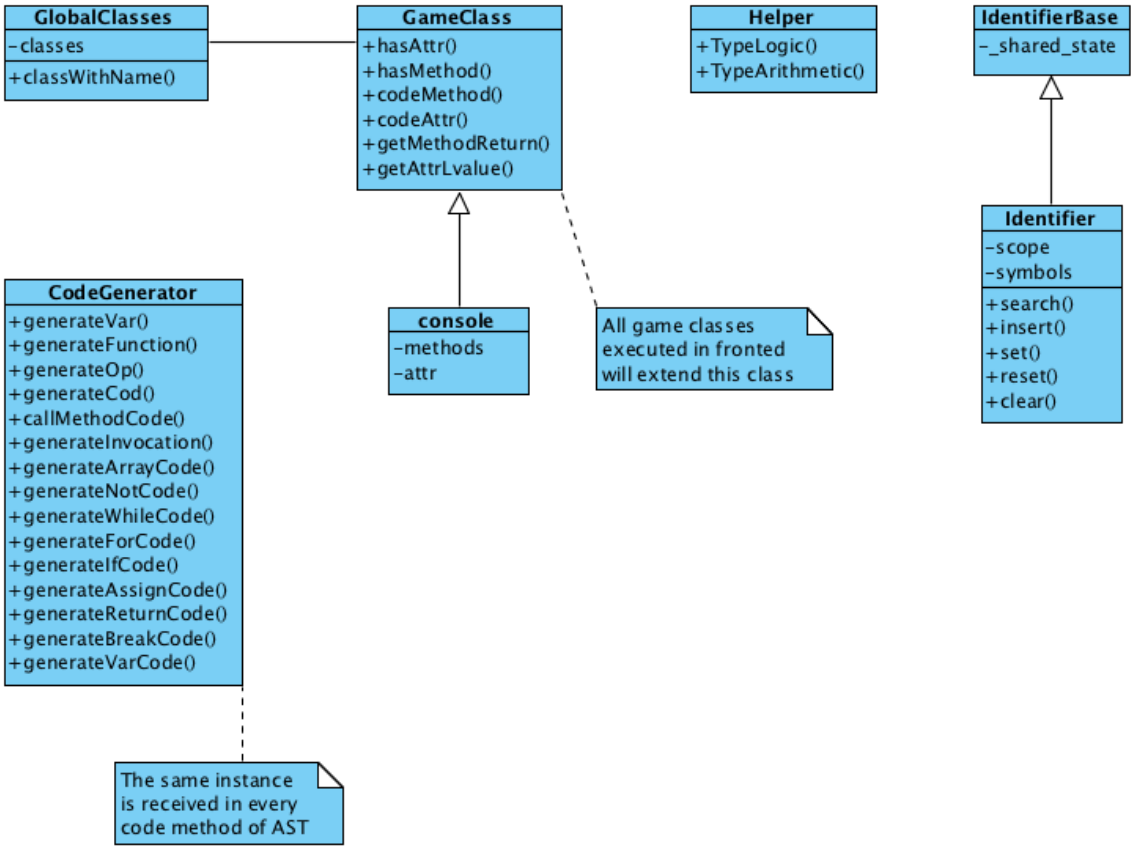


Figura 6.7 Otras clases del AST

Nombre de la Clase
GlobalClasses
Descripción
Contiene instancias de todas las clases globales del juego como consola, mundo o usuario.
Responsabilidades
Devolver instancias de clases globales del juego y comprobar la existencia de las mismas.
Atributos Propuestos
classes: Lista de clases de las que se dispone en el juego
Métodos Propuestos
classWithName: Comprueba la existencia de una clase concreta, en caso de existir devuelve su instancia.

Nombre de la Clase
GameClass
Descripción
Clase utilidad que actúa como padre de toda la jerarquía de clases del juego.
Responsabilidades
Actuar como padre de la jerarquía de clases del juego.
Atributos Propuestos
Métodos Propuestos
hasAttr: Comprueba si la clase tiene el atributo por el que se pregunta. **hasMethod**: Comprueba si la clase tiene el método por el que se pregunta **codeMethod**: Devuelve el código a ejecutar en *frontend* correspondiente al método solicitado para una determinada clase. **codeAttr**: Devuelve el código a ejecutar en *frontend* correspondiente al acceso a un atributo solicitado para una determinada clase. **getMethodReturn**: Devuelve el tipo de dato retornado por el método preguntado. **getAttrLvalue**: Comprueba si el atributo por el que se pregunta puede ser *lvalue*.

Nombre de la Clase
console
Descripción
Clase global que representa la consola del navegador.
Responsabilidades
Representar la consola del navegador.
Atributos Propuestos
methods: Lista de métodos soportados por la aplicación dentro de la clase consola. **attr**: Lista de atributos y su posibilidad de ser *lvalue* dentro de la clase consola.
Métodos Propuestos

Nombre de la Clase
CodeGenerator
Descripción
Generador de código para cada una de las clases del AST. Una instancia de esta clase es pasada por el árbol en el método *code* de cada nodo.
Responsabilidades
Generar el código equivalente de un AST concreto.
Atributos Propuestos
Métodos Propuestos
generateVar: Genera el código de una definición de variable. **generateFunction**: Genera el código de una definición de función. **generateOp**: Genera el código de una operación binaria. **generateCode**: Inserta el código recibido dentro del generador. **callMethodCode**: Genera el código de una llamada a método. **generateInvocation**: Genera el código de una invocación a función. **generateNotCode**: Genera el código de una negación unaria. **generateWhileCode**: Genera el código correspondiente a una sentencia *while*. **generateForCode**: Genera el código correspondiente a un bucle *for*. **generateIfCode**: Genera el código correspondiente a una sentencia *if-else*. **generateAssignCode**: Genera el código correspondiente a una sentencia de asignación. **generateReturnCode**: Genera el código correspondiente a una sentencia *return*. **generateBreakCode**: Genera el código correspondiente a una sentencia *break*. **generateVarCode**: Genera el código correspondiente al valor de una variable.

Nombre de la Clase
Helper
Descripción
Clase utilidad que determina los tipos de retorno y si los valores operados son lógicos.
Responsabilidades
Facilitar las utilidades de tipos de datos y lógica de operaciones.
Atributos Propuestos
Métodos Propuestos
TypeLogic: Determina si el resultado de la operación será lógico o no. **TypeArithmetic**: Determina el tipo resultado de la operación aritmética.

Nombre de la Clase
Identifier
Descripción
Clase utilidad que registra las definiciones de variables detectadas hasta el momento y posteriormente comprueba que las variables usadas se encuentran definidas.
Responsabilidades
Registrar todas las definiciones y comprobar que las variables usadas se encuentran registradas.
Atributos Propuestos
scope: Ámbito actual en el que se encuentra registrando o buscando variables. **symbols**: Definiciones de variables actualmente encontradas.
Métodos Propuestos
search: Busca en la tabla de símbolos actual la variable solicitada, si no la encuentra busca en una tabla de símbolos superior hasta encontrarla o quedarse en el ámbito inicial. **insert**: Inserta una definición de variable en la tabla de símbolos del ámbito actual. **set**: Incrementa el ámbito actual en una unidad. **reset**: Decrementa el ámbito actual en una unidad. **clear**: Vacía todas las tablas de símbolos y reinicia todo el identificador.

Nombre de la Clase
IdentifierBase
Descripción
Clase base de todos los identificadores que se encarga de compartir datos entre ellos.
Responsabilidades
Compartir los datos entre los identificadores
Atributos Propuestos
_shared_state: Diccionario que contiene el estado compartido de los identificadores.
Métodos Propuestos

6.3.2 Diagramas de flujo del modulo

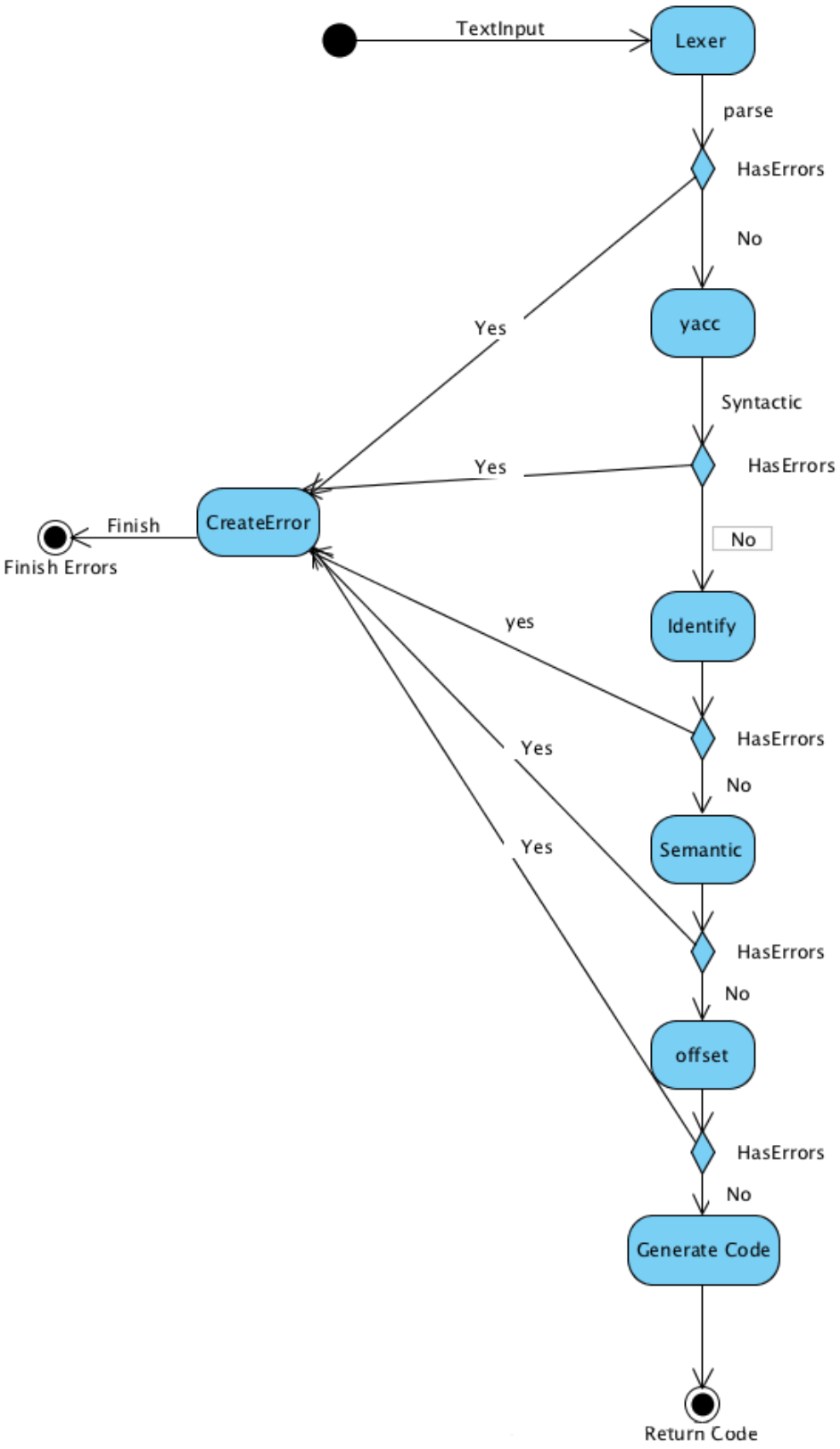


Figura 6.8 Análisis y generación de código

6.4 Módulo: Definir el protocolo del websocket

6.4.1 Diagrama de diseño del módulo

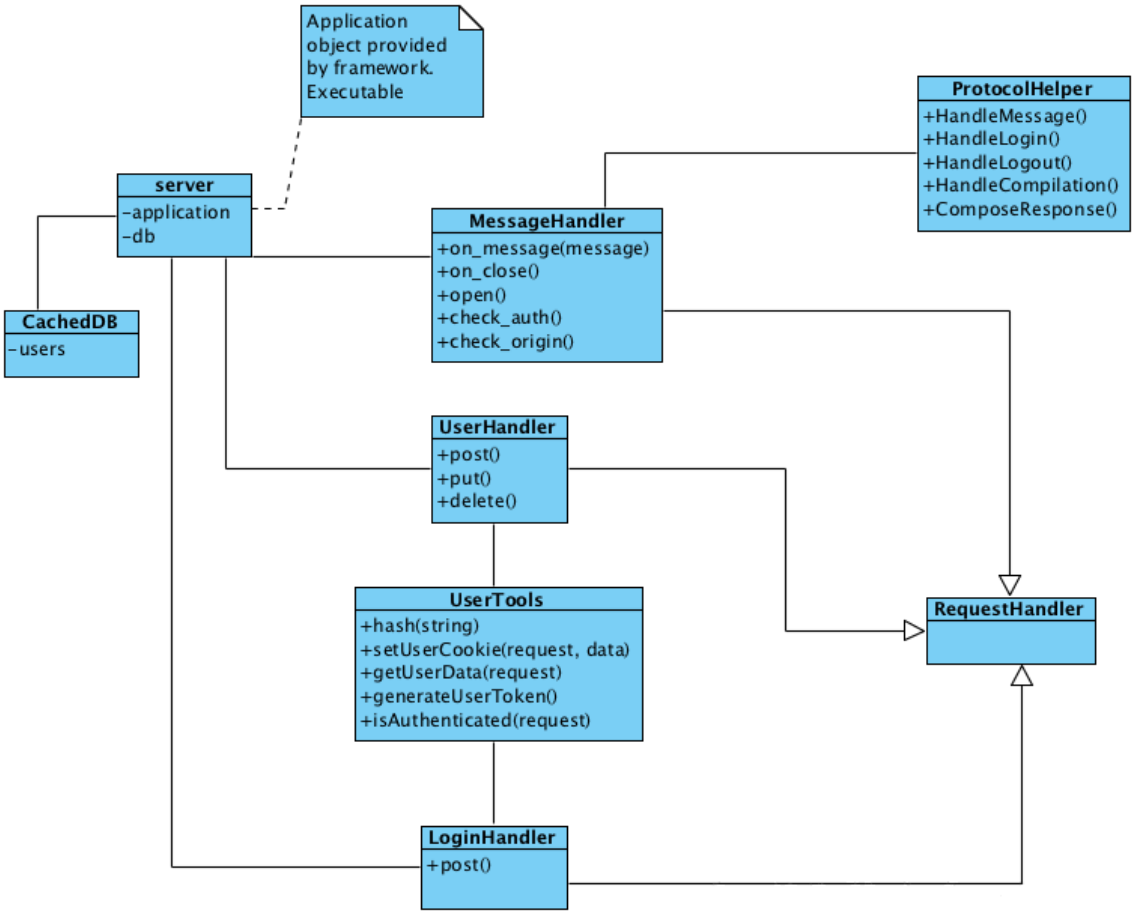


Figura 6.9 Backend: Protocolo del websocket

Nombre de la Clase
server
Descripción
Punto inicial de la aplicación *backend*, es el archivo que se ejecutará como *main* e incluye la lógica de los puntos de acceso a la API.
Responsabilidades
Servir de punto de entrada a la aplicación
Atributos Propuestos
application: Objeto que provee la lógica del *framework* del servidor. Provisto por el *framework* elegido. **db**: Objeto que provee una caché de la base de datos accesible en forma de diccionario.
Métodos Propuestos

Nombre de la Clase
MessageHandler
Descripción
Encargada de controlar aquellas peticiones contra el punto de entrada wss. Gestiona toda la lógica relacionada con el WebSocket de la aplicación.
Responsabilidades
Atender, controlar y responder aquellas peticiones recibidas mediante el WebSocket de forma correcta.
Atributos Propuestos
Métodos Propuestos
on_message: Atiende un nuevo mensaje del WebSocket **on_close**: Notifica el cierre de una conexión en el WebSocket **open**: Procesa la apertura de una nueva conexión en el WebSocket y comprueba que el usuario está debidamente autenticado llamando a check_auth. **check_auth**: Comprueba que el usuario se encuentra debidamente autenticado en el sistema y si lo está cachea sus datos. **check_origin**: Comprueba que el origen de la conexión es el sitio web esperado, en caso contrario deniega la conexión.

Nombre de la Clase
ProtocolHelper
Descripción
Encargada de procesar todos los mensajes recibidos en el WebSocket de acuerdo al protocolo establecido, dependiendo de su tipo y de los parámetros incluidos en el mensaje.
Responsabilidades
Clasificar y responder los mensajes del protocolo de forma correcta y ordenada.
Atributos Propuestos
Métodos Propuestos
HandleMessage: Atiende un nuevo mensaje del protocolo clasificándolo según su tipo. **HandleLogin**: Atiende un mensaje de *login* del WebSocket. **HandleLogout**: Atiende un mensaje de *logout* del WebSocket, actualizando la base de datos con el nuevo mundo del usuario y con su nuevo objeto de usuario. **HandleCompilation**: Invoca al analizador léxico-sintáctico y *parsea* el código del usuario, notifica si la compilación ha sido correcta y en caso contrario notifica los errores. **ComposeResponse**: Compone una nueva respuesta del protocolo.

6.4.2 Diagramas de flujo del módulo

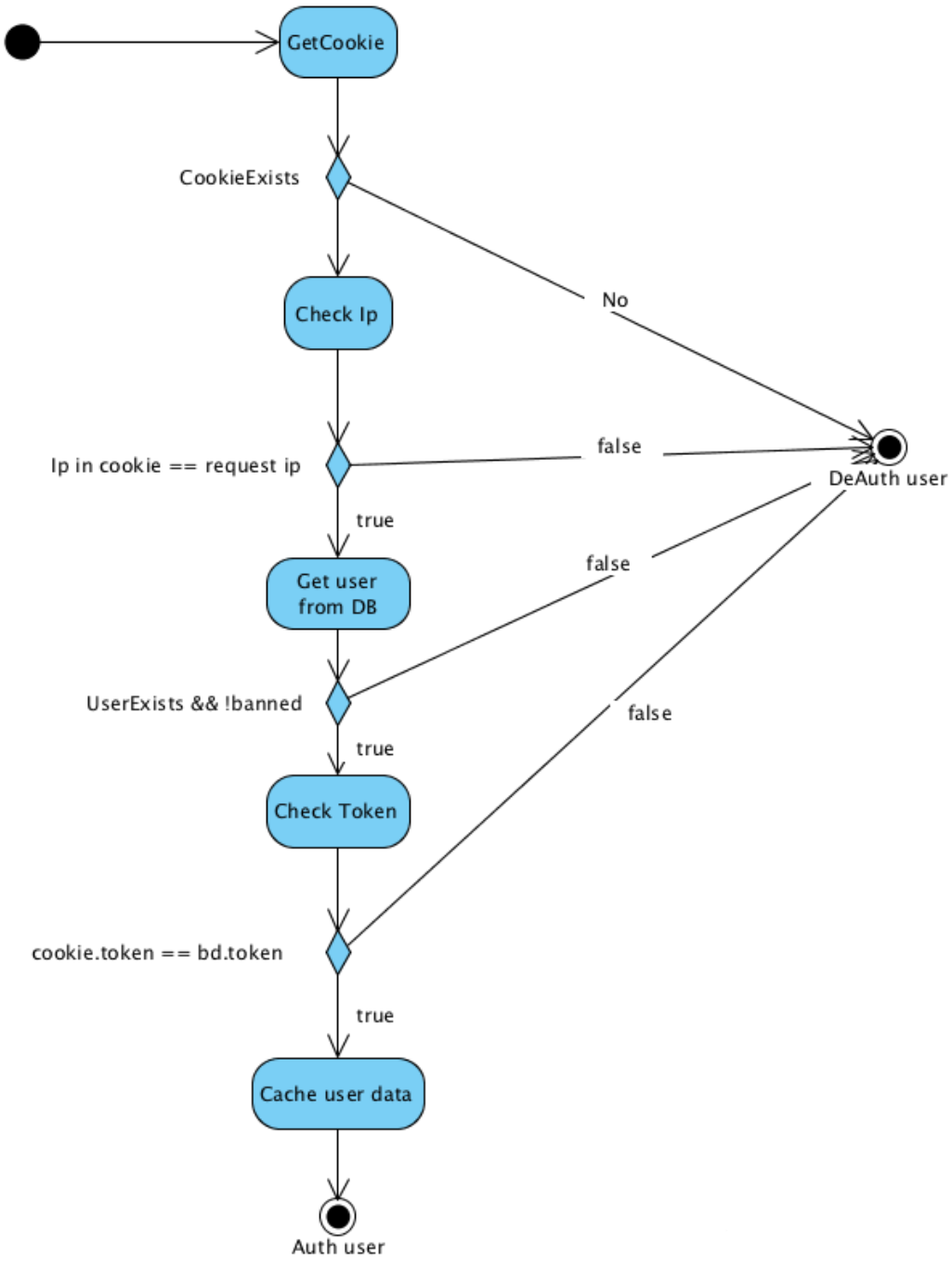


Figura 6.10 Autenticación de usuarios

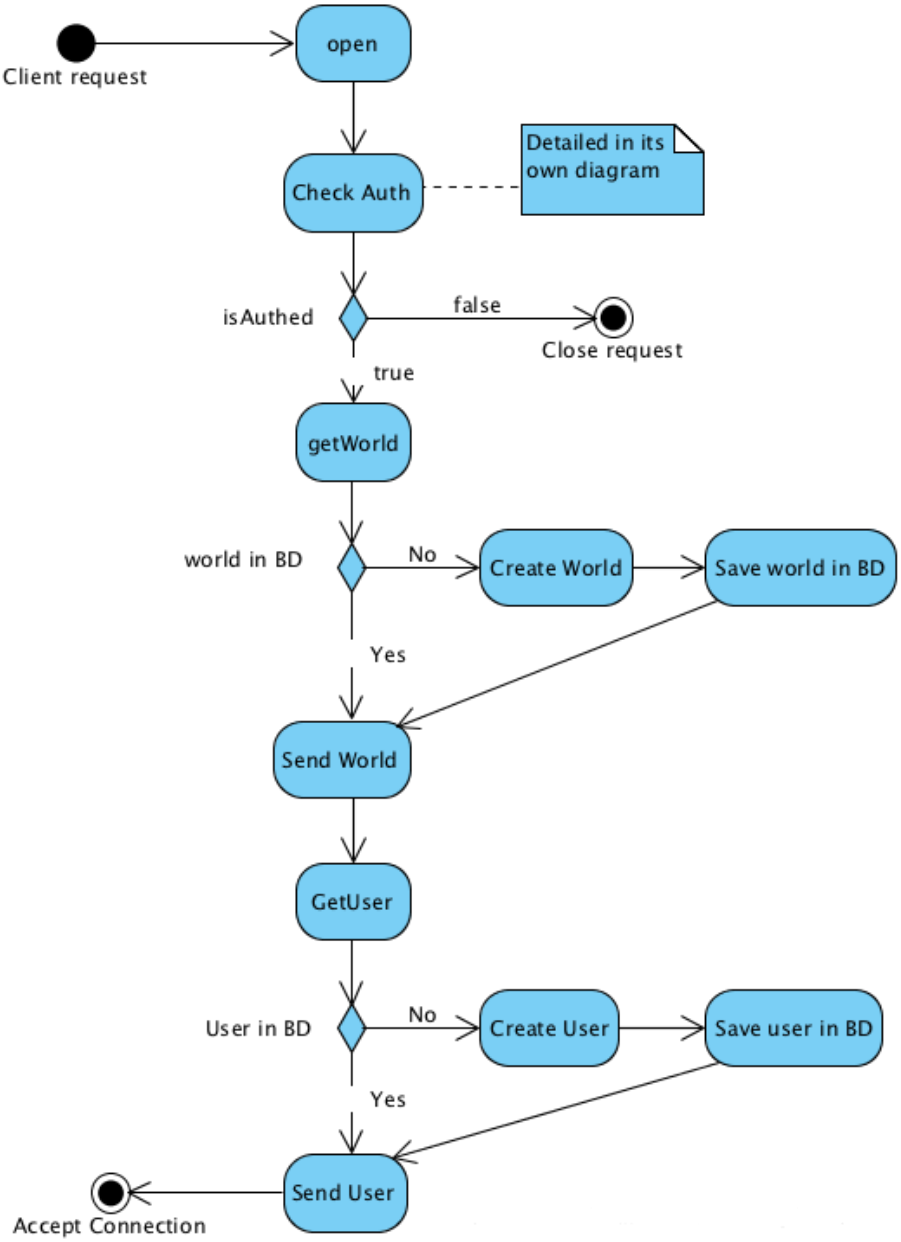


Figura 6.11 Conexión de usuario

6.5 Módulo: Gestión de usuarios

6.5.1 Diagrama de diseño del módulo

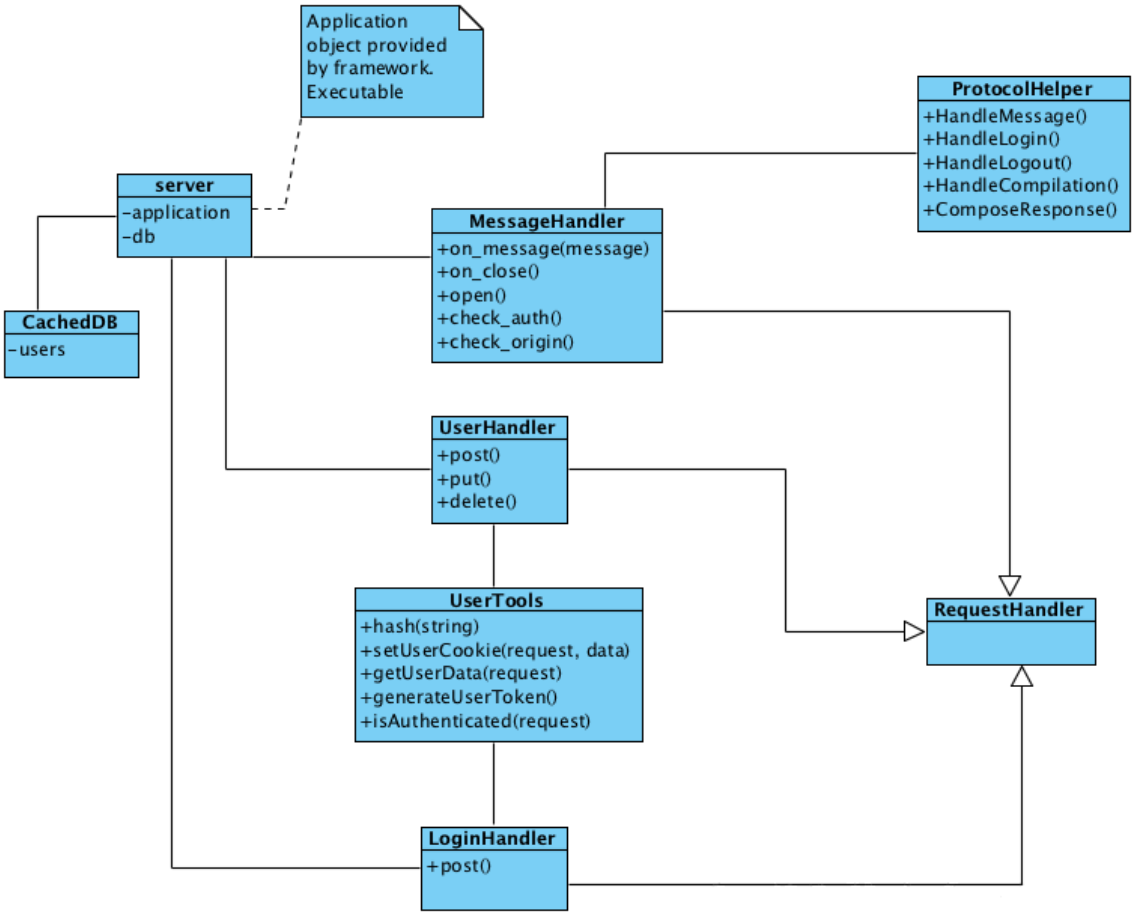


Figura 6.12 Backend: Gestión de usuarios

Nombre de la Clase
UserHandler
Descripción
Encargada de controlar aquellas peticiones contra el punto de entrada *user*. Gestiona toda la lógica relacionada con usuarios
Responsabilidades
Gestionar toda la lógica relacionada con usuarios: creación, actualización y borrado de los mismos.
Atributos Propuestos
Métodos Propuestos
post: Gestiona la creación de un nuevo usuario **put**: Gestiona la actualización de un usuario **delete:** Gestiona la eliminación de un usuario

Nombre de la Clase
LoginHandler
Descripción
Encargada de recibir peticiones de *login* al sistema.
Responsabilidades
Controlar el acceso y autenticación de los usuarios al sistema.
Atributos Propuestos
Métodos Propuestos
post: Recibirá los datos de *login* de un usuario y comprobará que las credenciales sean correctas, en caso de no serlo enviará un mensaje genérico de error. En caso de serlo establecerá las cookies pertinentes.

Nombre de la Clase
UserTools
Descripción
Conjunto de utilidades para controlar el acceso y autenticación de los usuarios.
Responsabilidades
Unificar en un objeto todas aquellas utilidades relacionadas con los usuarios como las cookies o los *tokens*.
Atributos Propuestos
Métodos Propuestos
hash: Retorna un hash del texto recibido como parámetro. Usado para generar el hash de las contraseñas. **setUserCookie**: Establece las cookies del usuario de forma cifrada. En la cookie se cifra un diccionario con datos del usuario. **getUserData**: Recupera el diccionario introducido en las cookies y lo retorna de forma legible. **generateUserToken**: Genera un *token* seguro nuevo para el usuario. **isAuthenticated**: Comprueba la existencia de la cookie del usuario y que la *IP* desde la que realiza las peticiones es la misma que la *IP* desde la que realizó el *login*.

6.5.2 Diagrama de la base de datos

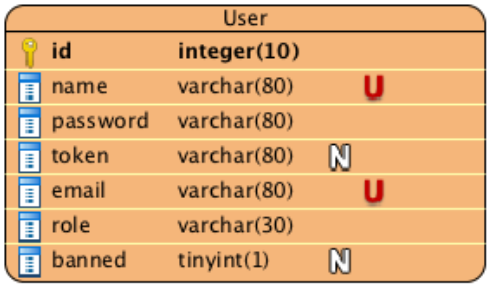


Figura 6.13 Base de datos: Usuario

En esta primera iteración únicamente se ha incluido en la base de datos la tabla de usuarios con los datos indicados en la parte superior.

Nombre de la Clase
User
Descripción
Tabla de la base de datos que contendrá los datos concretos del usuario.
Responsabilidades
Contener los datos concretos de cada usuario.
Atributos Propuestos
id: Identificador único del usuario. **name**: Nombre único del usuario. **password**: Hash de la contraseña del usuario. **token**: *Token* aleatorio generado para asegurar la autenticación del usuario. **email**: Email único del usuario. **role**: Rol del usuario en el sistema. Por defecto será *USER*. **banned**: Indicador de si el acceso del usuario se encuentra restringido.
Métodos Propuestos

6.5.3 Diagramas de flujo del módulo

A continuación se detallan los diagramas de *login* de un usuario y registro del mismo.

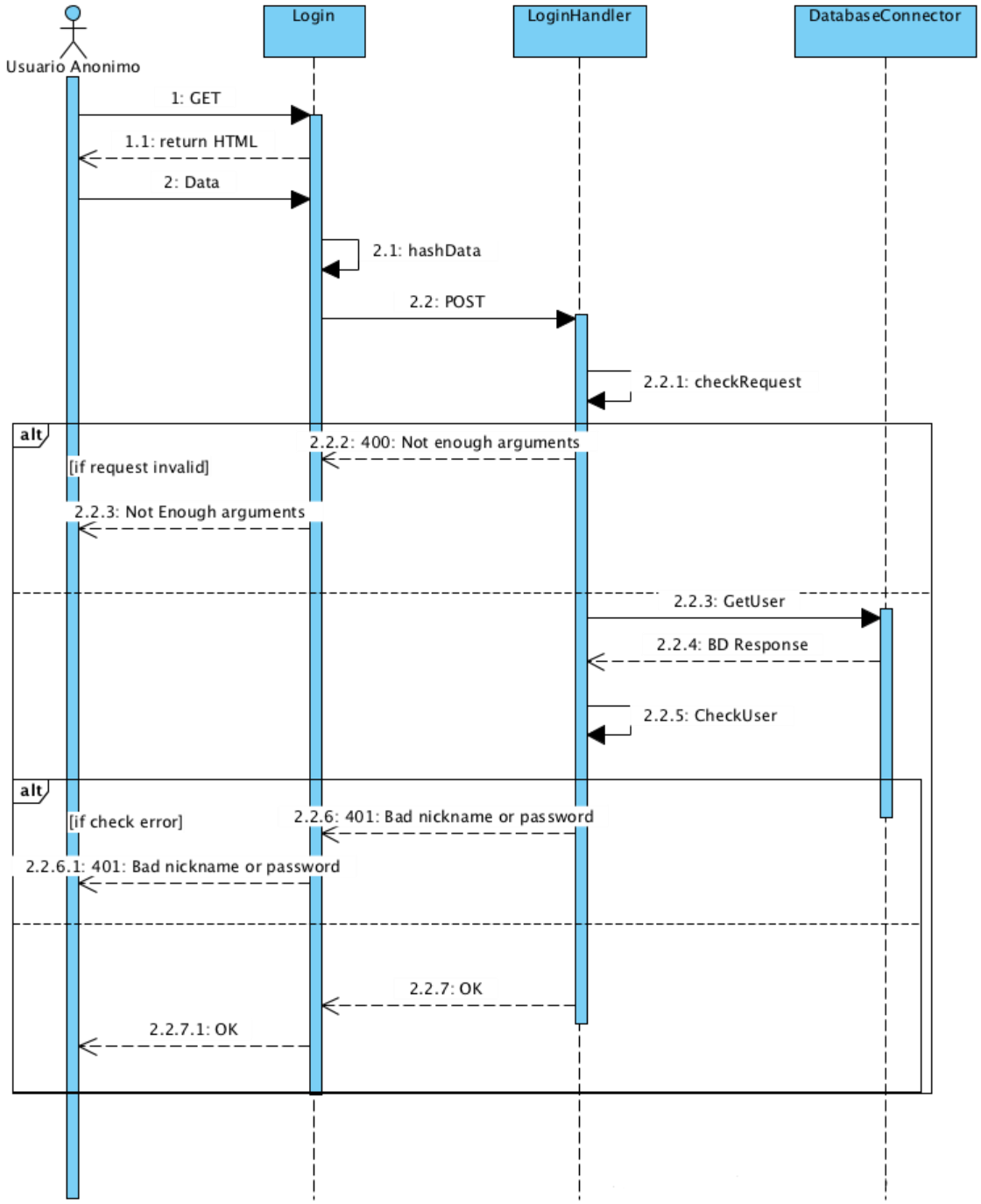


Figura 6.14 Login de Usuario

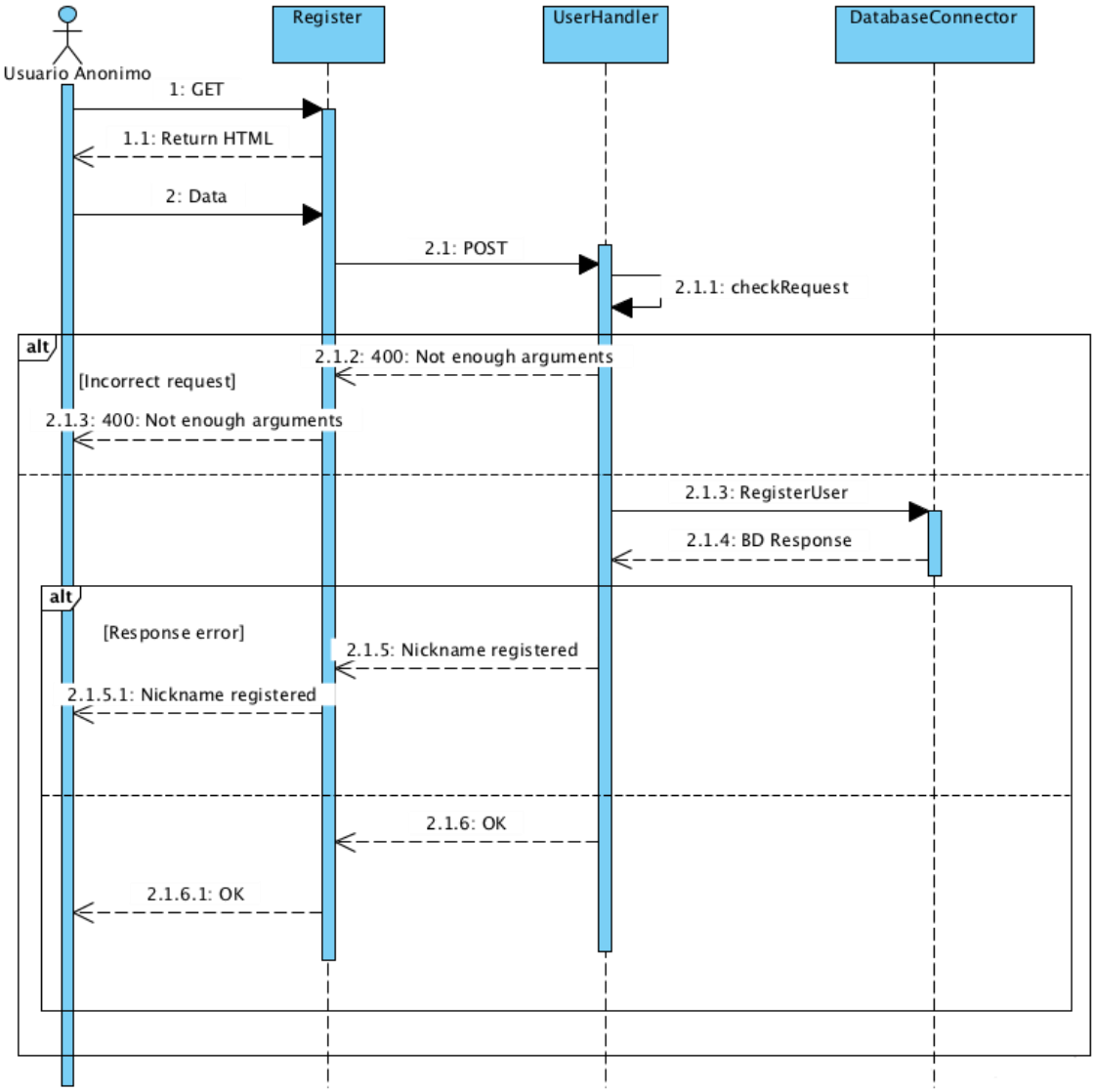


Figura 6.15 Registro de usuario

6.6 Módulo: Objeto World

6.6.1 Diagrama de la base de datos del módulo

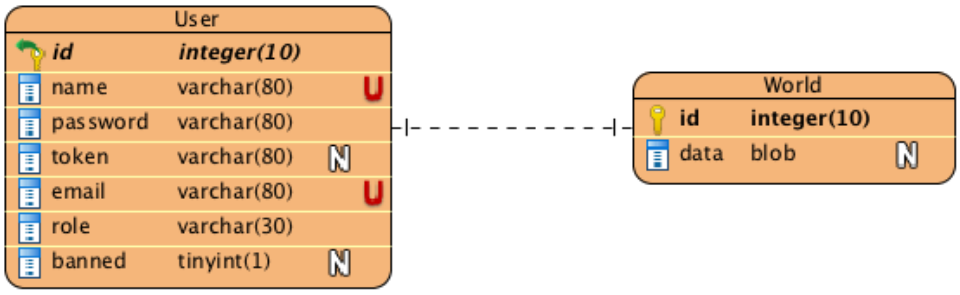


Figura 6.16 Base de datos: Mundo

Se ha añadido a la parte *backend* una nueva tabla para almacenar el mundo de cada usuario.

Nombre de la Clase
World
Descripción
Tabla de la base de datos que contendrá los datos del mundo del usuario.
Responsabilidades
Contener los datos concretos del mundo de cada usuario.
Atributos Propuestos
id: Identificador único del usuario. **data**: Mundo del usuario.
Métodos Propuestos

6.6.2 Diagrama de diseño del módulo

6.6.2.1 Diagramas de diseño del backend

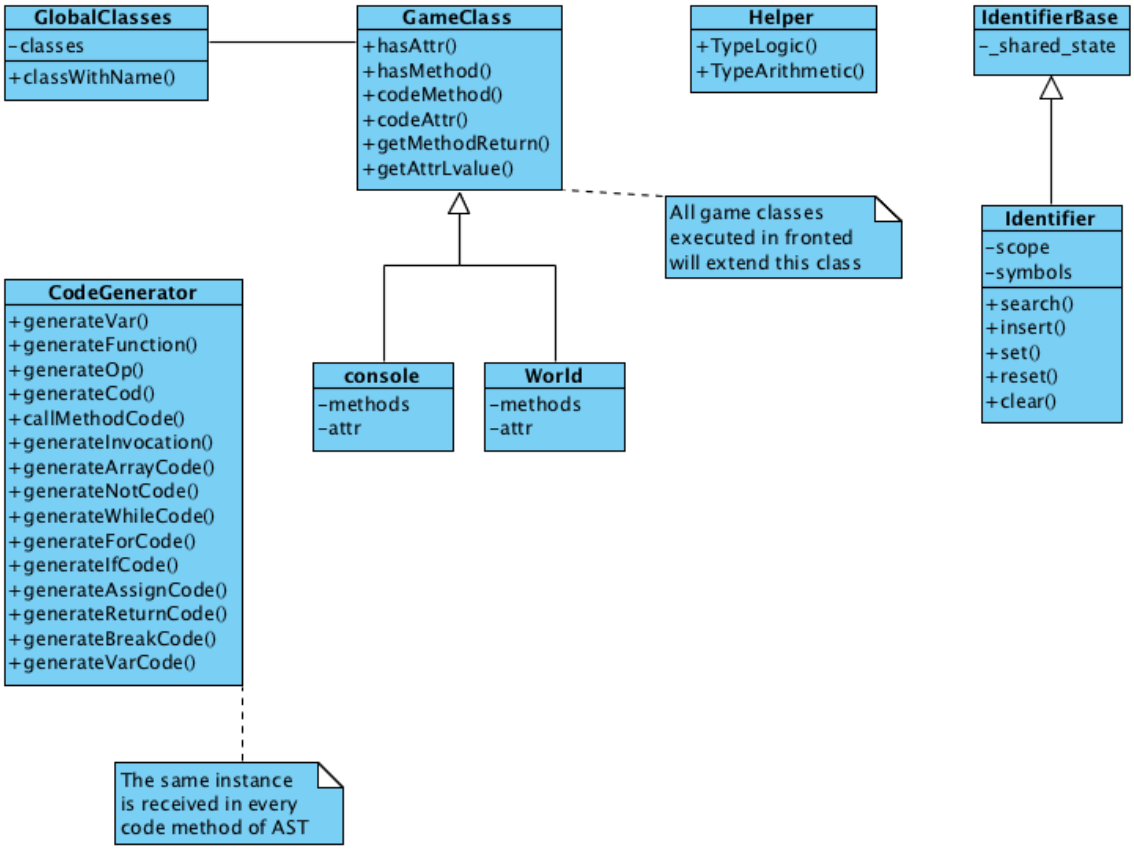


Figura 6.17 Utilidades del AST: Mundo

Se ha añadido una nueva clase a las utilidades del AST denominada *World* que contiene toda la información sobre esta nueva clase.

Nombre de la Clase
World
Descripción
Clase global que representa el mundo del usuario en el juego.
Responsabilidades
Representar el mundo del usuario.
Atributos Propuestos
methods: Lista de métodos soportados por la aplicación dentro del mundo. **attr**: Lista de atributos y su posibilidad de ser *lvalue* dentro del mundo.
Métodos Propuestos

Además ahora al conectar mediante WebSocket si el usuario se encuentra autenticado, el sistema enviará mediante un mensaje *World* el mundo del usuario.

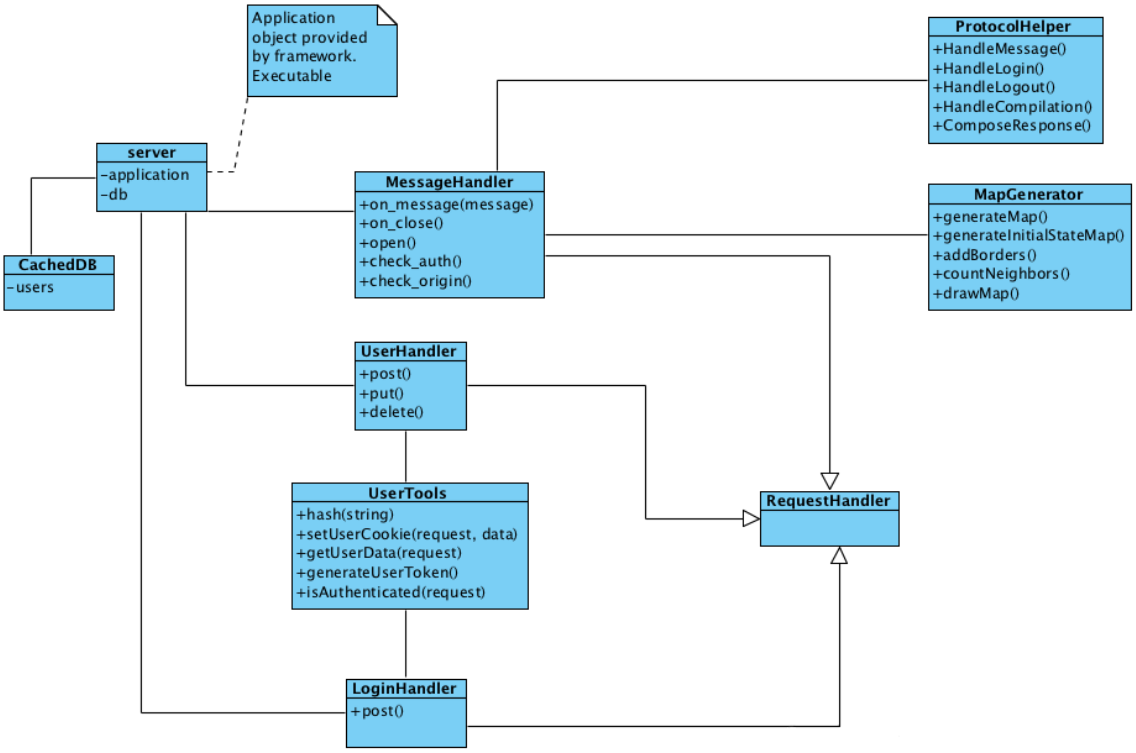


Figura 6.18 Backend: Mundo

Nombre de la Clase
MapGenerator
Descripción
Utilidad empleada para generar mapas de juego aleatorios usando un algoritmo de tipo *Cellular autómata*.
Responsabilidades
Generar mapas aleatorios para el usuario.
Atributos Propuestos
Métodos Propuestos
generateMap: Genera un nuevo mapa de usuario del tamaño y con la probabilidad especificada. **generateInitialStateMap**: Genera un nuevo mapa completamente aleatorio sin realizar el algoritmo del autómata celular. **addBorders**: Fuerza los bordes del mapa para que sea un mapa cerrado. **countNeighbors**: Cuenta los vecinos ocupados a un nodo dado en el mapa. **drawMap**: Realiza una representación mediante *ASCII* en la consola del mapa creado.

6.6.2.2 Diagrama de diseño del frontend

El diagrama siguiente se encuentra simplificado únicamente para facilitar su lectura y comprensión.

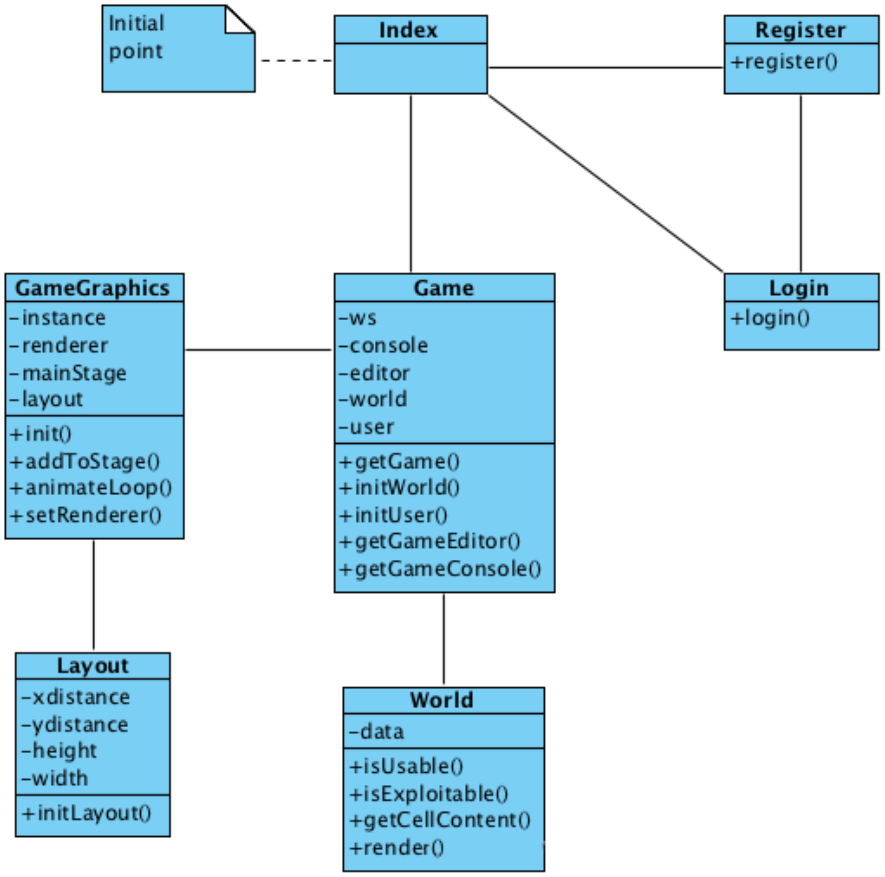


Figura 6.19 Frontend: Mundo

Nombre de la Clase
World
Descripción
Clase que almacena el mundo del usuario en *frontend* junto a sus acciones.
Responsabilidades
Almacenar el mundo del usuario junto a sus acciones y ejecutarlas.
Atributos Propuestos
data: Datos del mundo del usuario
Métodos Propuestos
isUsable: Determina si la casilla especificada puede usarse para construir o caminar. **isExploitable**: Determina si la casilla especificada dispone de un recurso que se pueda explotar. **getCellContent**: Recupera el contenido de la casilla especificada. **render**: Dibuja el mundo de forma gráfica.

6.7 Módulo: Objeto Me

6.7.1 Diagrama de la base de datos del módulo

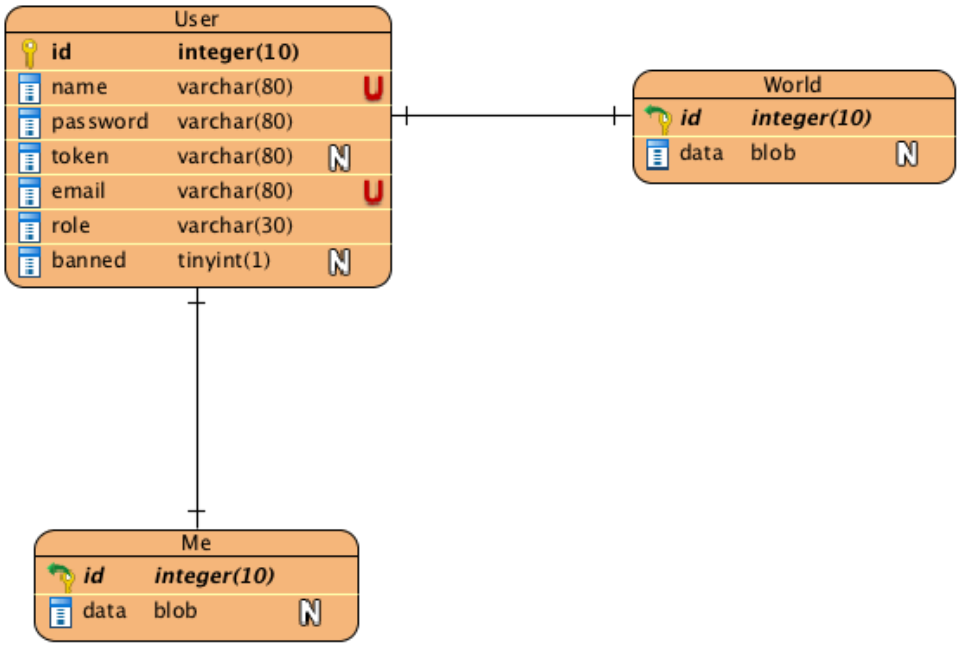


Figura 6.20 Base de datos: Me

Se añade una nueva clase *Me* que almacena los objetos usuario en formato *json*, identificados por el id del usuario.

Nombre de la Clase
Me
Descripción
Tabla de la base de datos que contendrá los datos del objeto *Me* del usuario.
Responsabilidades
Contener los datos concretos del objeto usuario de cada usuario.
Atributos Propuestos
id: Identificador único del usuario. **data**: Objeto *Me* del usuario almacenado en formato *JSON*.
Métodos Propuestos

6.7.2 Diagramas de diseño del modulo

6.7.2.1 Diagrama de diseño del frontend

El diagrama siguiente se encuentra simplificado únicamente para facilitar su lectura y comprensión.

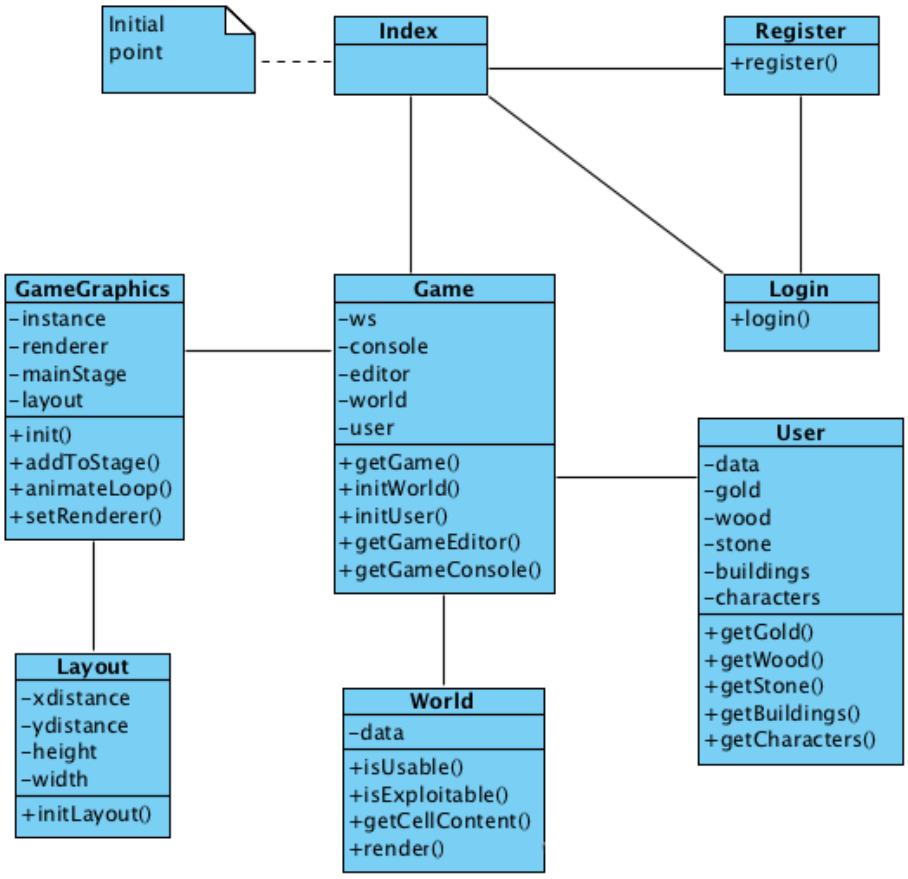


Figura 6.21 Frontend: Me

Nombre de la Clase
User
Descripción
Clase que contendrá los datos del usuario en *frontend* para que sean consultados.
Responsabilidades
Contener los datos concretos del objeto usuario del usuario y facilitar su interacción.
Atributos Propuestos
data: Contiene los datos brutos del objeto usuario. **gold**: Cantidad de oro de la que dispone el usuario. **wood**: Cantidad de madera de la que dispone el usuario. **stone**: Cantidad de piedra de la que dispone el usuario. **buildings**: Lista de edificios del usuario. **characters**: Lista de personajes del usuario.
Métodos Propuestos
getGold: Recupera el oro del usuario. **getWood**: Recupera la madera del usuario. **getStone**: Recupera la piedra del usuario. **getBuildings**: Recupera la lista de edificios del usuario. **getCharacters**: Recupera la lista de personajes del usuario.

6.7.2.2 Diagrama de diseño del backend

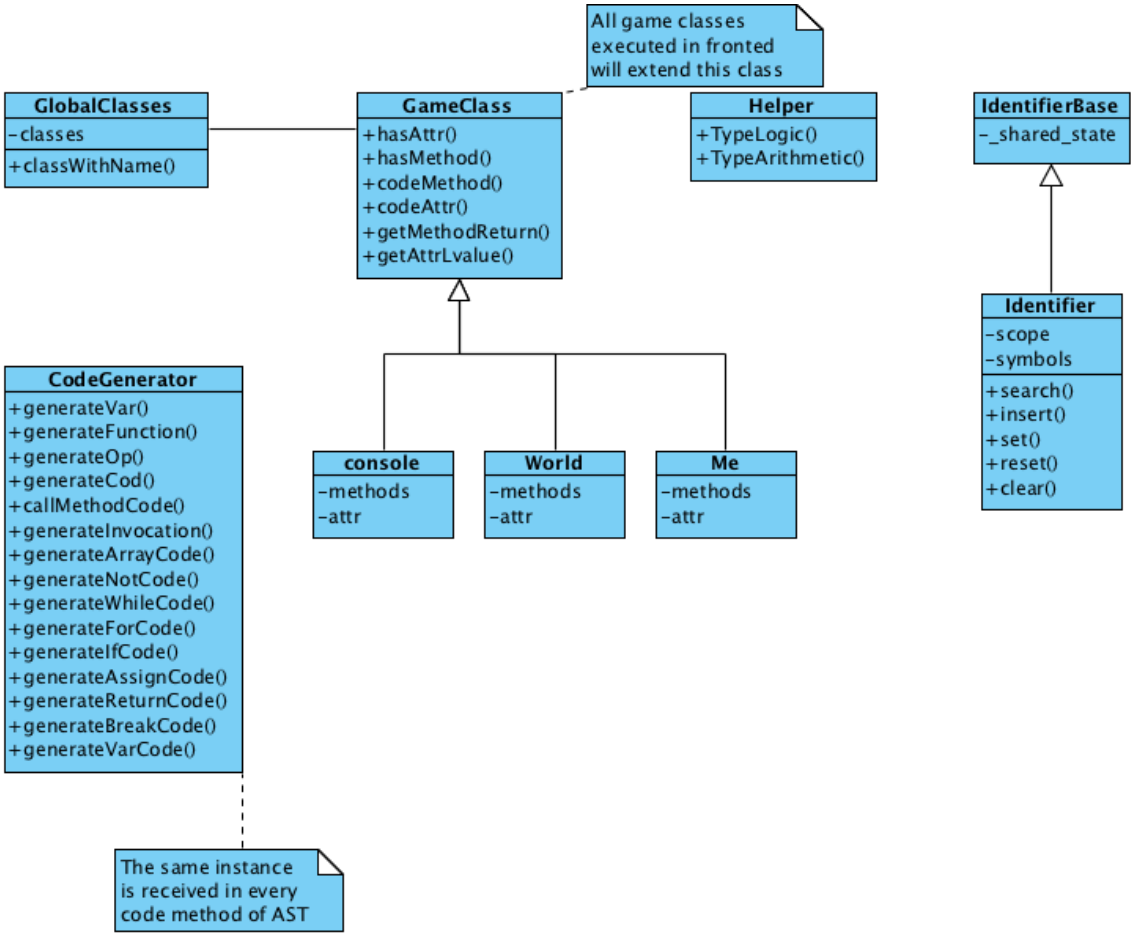


Figura 6.22 Utilidades del AST: Me

Nombre de la Clase
World
Descripción
Clase global que representa el objeto del usuario en el juego.
Responsabilidades
Representar el objeto del usuario en el juego.
Atributos Propuestos
methods: Lista de métodos soportados por la aplicación en cuanto al usuario. **attr**: Lista de atributos y su posibilidad de ser *lvalue* dentro del usuario así como su tipo.
Métodos Propuestos

6.8 Módulo: Edificio Genérico

6.8.1 Diagrama de diseño del backend

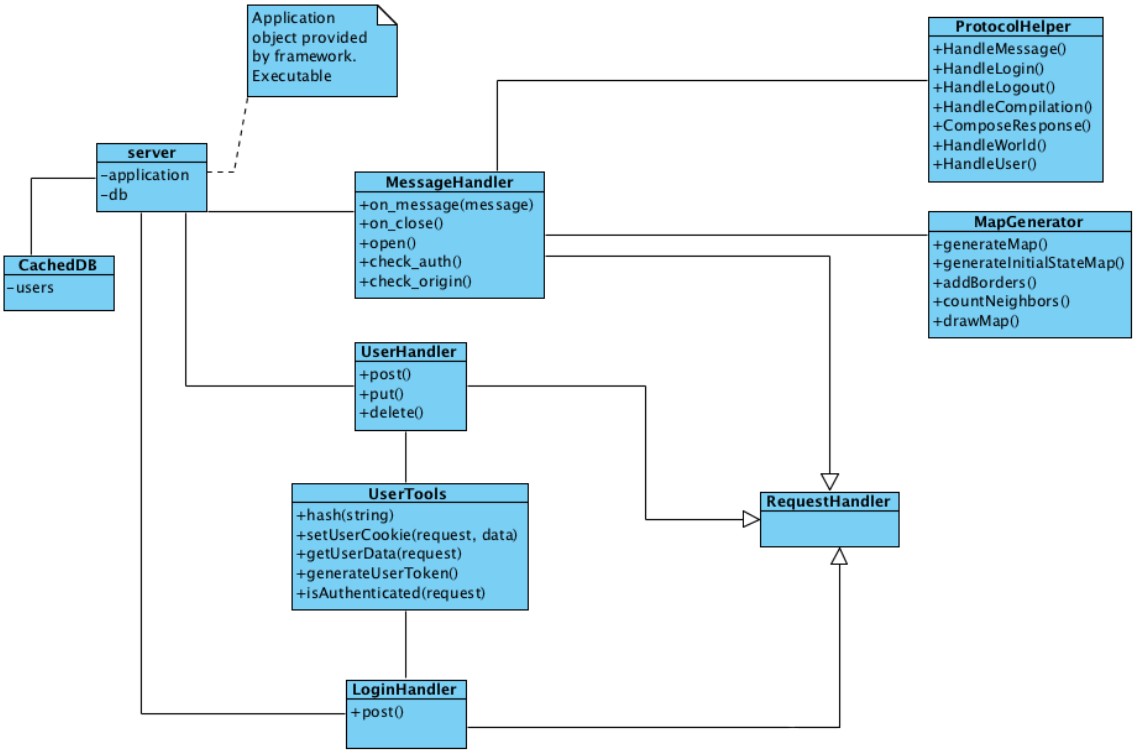


Figura 6.23 Backend: Me

Se ha ampliado el *backend* con nuevos métodos para gestionar el mundo y el usuario.

Nombre de la Clase
ProtocolHelper
Descripción
Encargada de procesar todos los mensajes recibidos en el WebSocket de acuerdo al protocolo establecido, dependiendo de su tipo y de los parámetros incluidos en el mensaje.
Responsabilidades
Clasificar y responder los mensajes del protocolo de forma correcta y ordenada.
Atributos Propuestos
Métodos Propuestos
HandleMessage: Atiende un nuevo mensaje del protocolo clasificándolo según su tipo. **HandleLogin**: Atiende un mensaje de *login* del WebSocket. **HandleLogout**: Atiende un mensaje de *logout* del WebSocket, actualizando la base de datos con el nuevo mundo del usuario y con su nuevo objeto de usuario. **HandleCompilation**: Invoca al analizador léxico-sintáctico y *parsea* el código del usuario, notifica si la compilación ha sido correcta y en caso contrario notifica los errores. **ComposeResponse**: Compone una nueva respuesta del protocolo. **HandleWorld**: Encargado de actualizar el mundo en *backend*. **HandleUser**: Encargado de actualizar el usuario en *backend*.

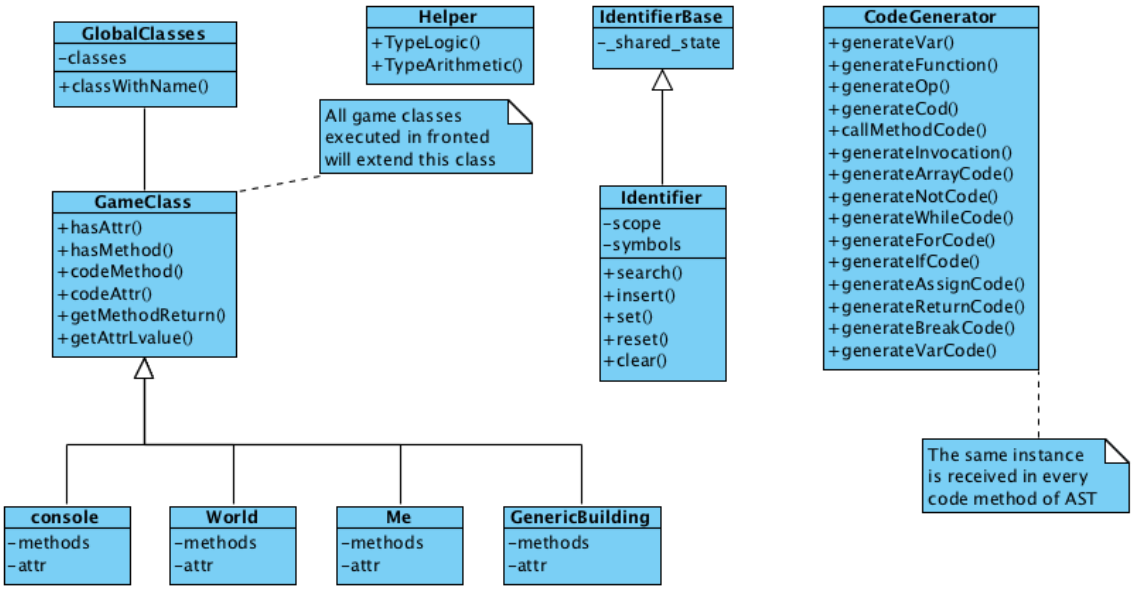


Figura 6.24 Utilidades AST: Edificio Genérico

Nombre de la Clase
GenericBuilding
Descripción
Encargada de representar en *backend* un edificio genérico del juego.
Responsabilidades
Representar los métodos y atributos de un edificio genérico en *backend*.
Atributos Propuestos
methods: Lista de métodos soportados por la aplicación para cualquier edificio del juego. **attr**: Lista de atributos y su posibilidad de ser *lvalue* para cualquier edificio del juego.
Métodos Propuestos

6.8.2 Diagrama de diseño del frontend

El siguiente diagrama se muestra simplificado para facilitar su lectura.

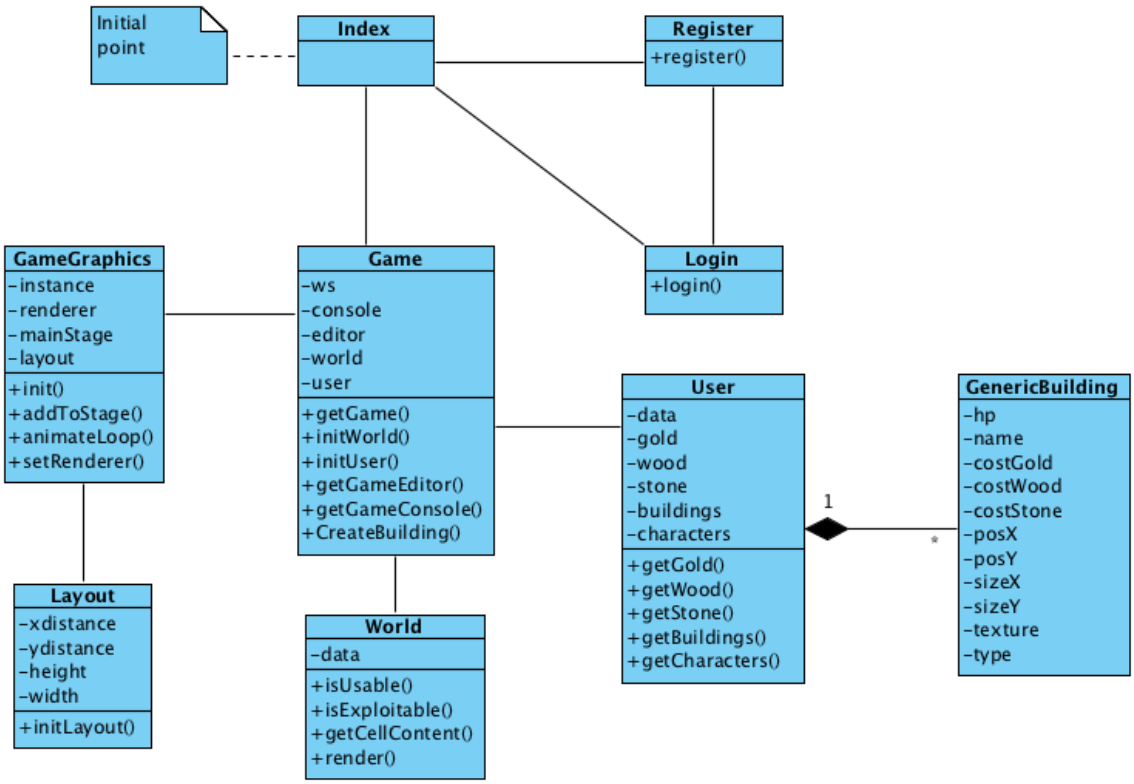


Figura 6.25 Frontend: Edificio Genérico

Nombre de la Clase
Game
Descripción
Vista encargada de la representación del juego y de la interacción con el mismo.
Responsabilidades
Proveer la lógica de visualización del videojuego y de interacción con el mismo.
Atributos Propuestos
ws: WebSocket empleado para la conexión con el servidor. **console**: Consola del usuario en la que aparecerán todos aquellos mensajes que se le deban transmitir. **editor**: Editor con el que interactuará el usuario. **world**: Mundo del usuario con el que podrá interactuar. **user**: Objeto usuario del usuario conectado actualmente.
Métodos Propuestos
getGame: Recupera la instancia de este objeto. **initWorld**: Inicializa el mundo con los datos recibidos. **initUser**: Inicializa el usuario con los datos recibidos. **getGameEditor**: Recupera el objeto editor con el que interactuará el usuario. **getGameConsole**: Recupera el objeto consola que enviará mensajes al usuario. **CreateBuilding**: Crea y devuelve un nuevo edificio del juego del tipo especificado y con los parámetros provistos.

Nombre de la Clase
GenericBuilding
Descripción
Representación en *frontend* de todo edificio genérico.
Responsabilidades
Proveer los métodos y atributos de un edificio.
Atributos Propuestos
hp: Representa los puntos de daño del edificio. **name**: Nombre identificador único del edificio. **costGold**: Coste en oro de la construcción del edificio. **costWood**: Coste en madera de la construcción del edificio. **costStone**: Coste en piedra de la construcción del edificio. **posX**: Posición en X del edificio. **posY**: Posición en Y del edificio. **sizeX**: Tamaño en X del edificio. **sizeY**: Tamaño en Y del edificio. **texture**: Textura a representar para el edificio **type**: Tipo de edificio.
Métodos Propuestos

6.9 Módulo: Personaje Genérico

6.9.1 Diagrama de diseño del frontend

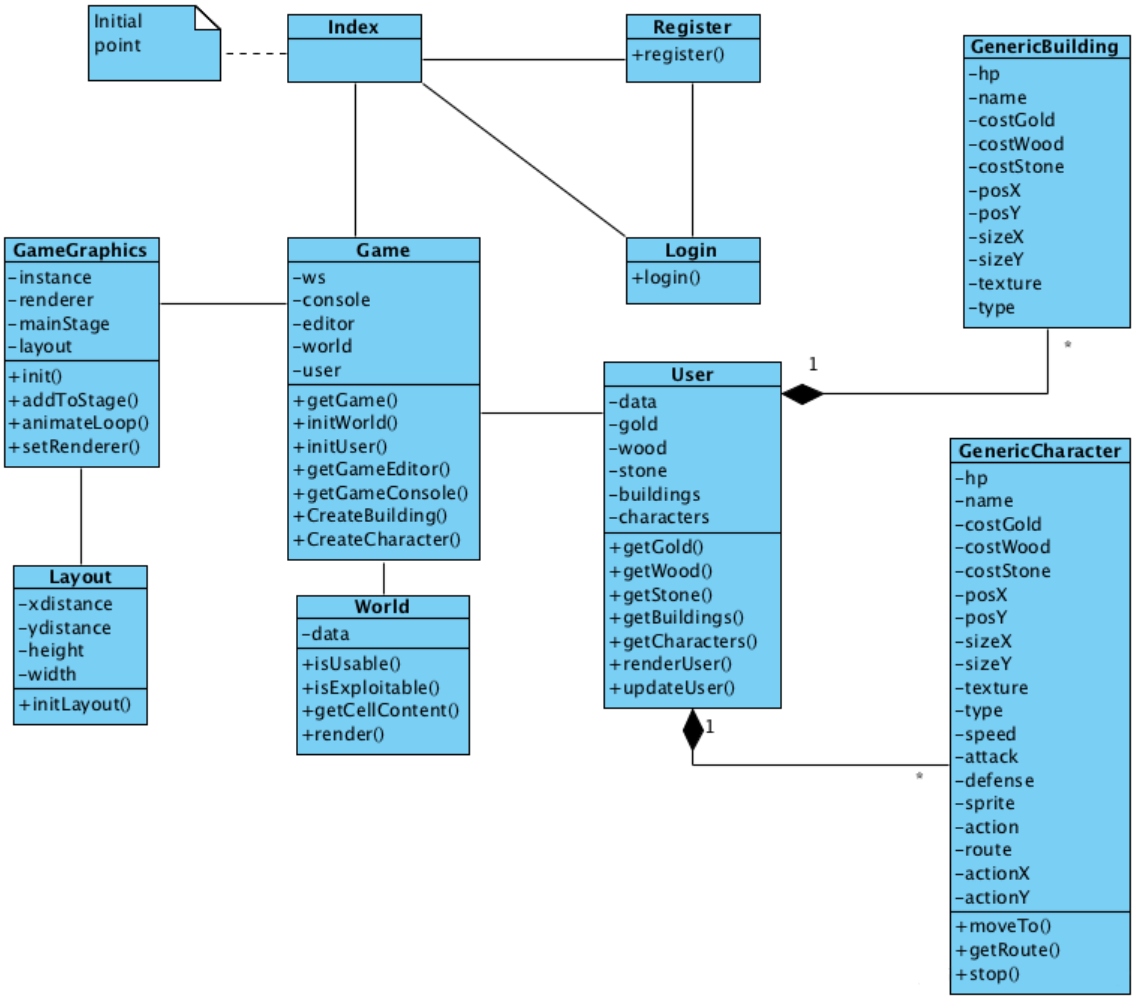


Figura 6.26 Frontend: Personaje Genérico

Nombre de la Clase
GenericCharacter
Descripción
Representación en *frontend* de todo personaje genérico.
Responsabilidades
Proveer los métodos y atributos mínimos de un personaje.
Atributos Propuestos
hp: Representa los puntos de daño del personaje. **name**: Nombre identificador único del personaje. **costGold**: Coste en oro de la construcción del personaje. **costWood**: Coste en madera de la construcción del personaje. **costStone**: Coste en piedra de la construcción del personaje. **posX**: Posición en X del personaje. **posY**: Posición en Y del personaje. **sizeX**: Tamaño en X del personaje. **sizeY**: Tamaño en Y del personaje. **texture**: Textura a representar para el personaje. **type**: Tipo de personaje. **speed**: Velocidad de movimiento del personaje. **attack**: Puntos de ataque del personaje. **defense**: Puntos de defensa del personaje. **sprite**: *Sprite* asignado al personaje. **action**: Acción a realizar por el personaje o que esté realizando. **route**: Ruta a seguir en caso de haber invocado el método de movimiento. **actionX**: Coordenada X en la que realizar la acción. **actionY**: Coordenada Y en la que realizar la acción.
Métodos Propuestos
moveTo: Inicia el movimiento del personaje hacia las coordenadas de destino. **getRoute**: Calcula una ruta optima utilizando el algoritmo A* hasta la casilla destino. **stop**: Detiene la acción en curso del usuario.

Nombre de la Clase
Game
Descripción
Vista encargada de la representación del juego y de la interacción con el mismo.
Responsabilidades
Proveer la lógica de visualización del videojuego y de interacción con el mismo.
Atributos Propuestos
ws: WebSocket empleado para la conexión con el servidor. **console**: Consola del usuario en la que aparecerán todos aquellos mensajes que se le deban transmitir. **editor**: Editor con el que interactuará el usuario. **world**: Mundo del usuario con el que podrá interactuar. **user**: Objeto usuario del usuario conectado actualmente.
Métodos Propuestos
getGame: Recupera la instancia de este objeto. **initWorld**: Inicializa el mundo con los datos recibidos. **initUser**: Inicializa el usuario con los datos recibidos. **getGameEditor**: Recupera el objeto editor con el que interactuará el usuario. **getGameConsole**: Recupera el objeto consola que enviará mensajes al usuario. **CreateBuilding**: Crea y devuelve un nuevo edificio del juego del tipo especificado y con los parámetros provistos. **CreateCharacter**: Crea y devuelve un nuevo personaje del juego del tipo especificado y con los parámetros provistos.

Nombre de la Clase
User
Descripción
Clase que contendrá los datos del usuario en *frontend* para que sean consultados.
Responsabilidades
Contener los datos concretos del objeto usuario del usuario y facilitar su interacción.
Atributos Propuestos
data: Contiene los datos brutos del objeto usuario. **gold**: Cantidad de oro de la que dispone el usuario. **wood**: Cantidad de madera de la que dispone el usuario. **stone**: Cantidad de piedra de la que dispone el usuario. **buildings**: Lista de edificios del usuario. **characters**: Lista de personajes del usuario.
Métodos Propuestos
getGold: Recupera el oro del usuario. **getWood**: Recupera la madera del usuario. **getStone**: Recupera la piedra del usuario. **getBuildings**: Recupera la lista de edificios del usuario. **getCharacters**: Recupera la lista de personajes del usuario. **renderUser**: Dibuja en el mapa todos los edificios y personajes del usuario. **updateUser**: Actualiza el usuario en *backend*.

6.9.2 Diagrama de diseño del backend

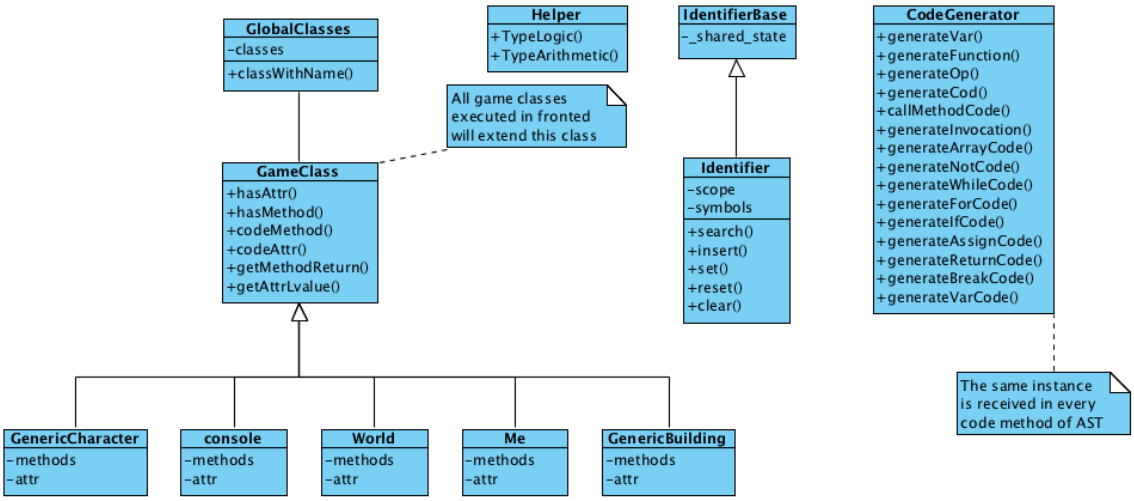


Figura 6.27 Utilidades AST: Personaje Genérico

Se ha añadido una nueva clase que especifica el personaje genérico y sus métodos y atributos.

Nombre de la Clase
GenericCharacter
Descripción
Encargada de representar en *backend* un personaje genérico del juego.
Responsabilidades
Representar los métodos y atributos de un personaje genérico en *backend*.
Atributos Propuestos
methods: Lista de métodos soportados por la aplicación para cualquier personaje. **attr**: Lista de atributos y su posibilidad de ser *lvalue* para cualquier personaje.
Métodos Propuestos

6.10 Módulo: Personaje Aldeano y Edificio Centro del poblado

6.10.1 Diagrama de diseño del frontend

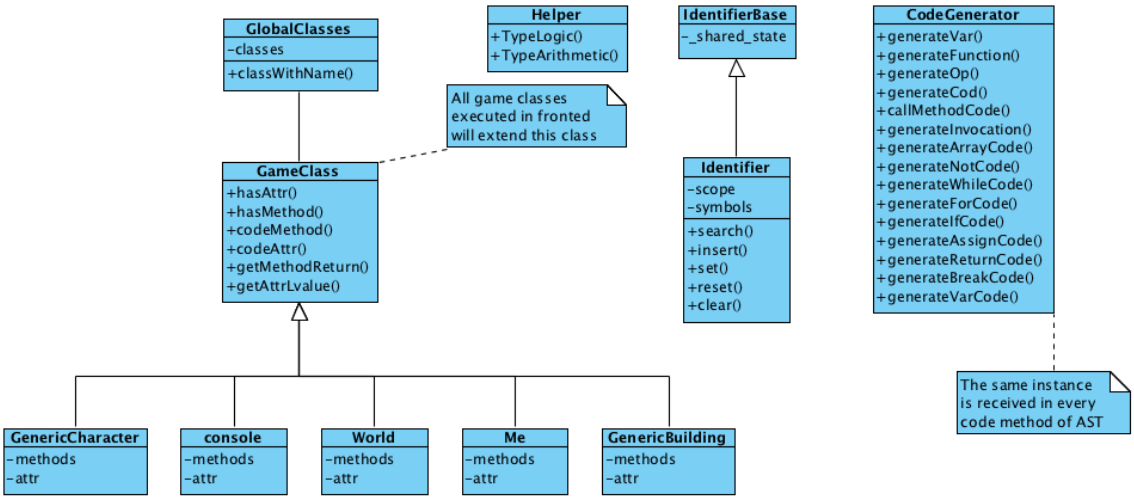


Figura 6.28 Frontend: Aldeano y Centro del poblado

Nombre de la Clase
TownCenter
Descripción
Encargada de representar en *frontend* el edificio del centro del poblado.
Responsabilidades
Representar el edificio del centro del poblado así como su lógica.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
Villager
Descripción
Encargada de representar en *frontend* un aldeano del juego.
Responsabilidades
Representar los métodos y atributos de un aldeano en el juego.
Atributos Propuestos
Métodos Propuestos
exploit: Mina los recursos de una localización indicada.

6.10.2 Diagrama de diseño del backend

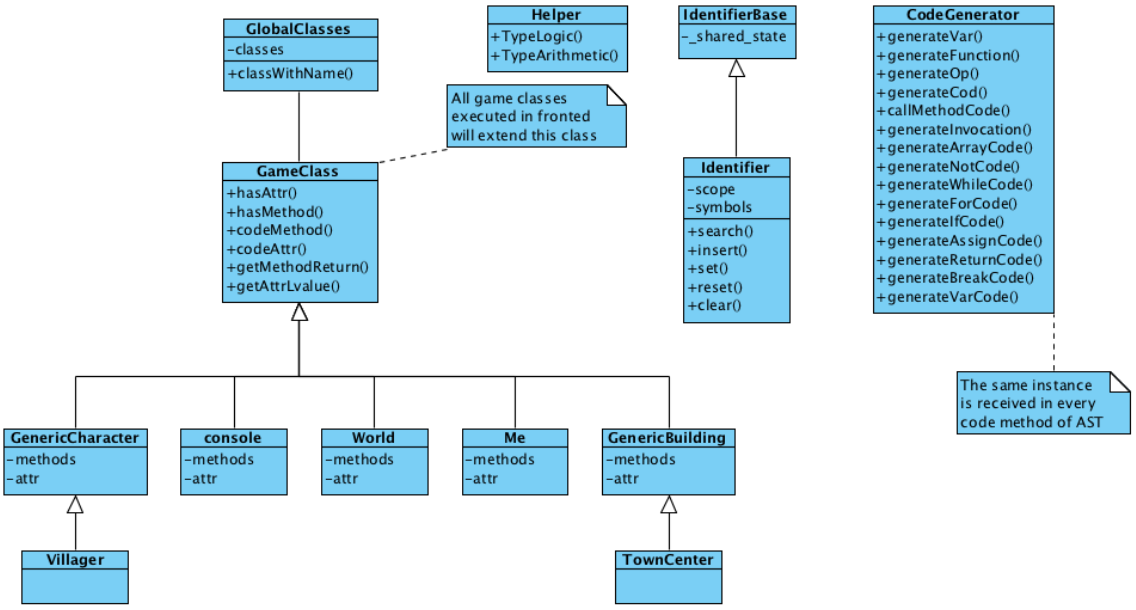


Figura 6.29 Backend: Aldeano y Centro del poblado

Nombre de la Clase
Villager
Descripción
Encargada de representar un aldeano en *backend*.
Responsabilidades
Ejecutar las comprobaciones pertinentes en cuanto a la interacción y creación con un aldeano.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
TownCenter
Descripción
Encargada de representar el edificio del centro del poblado en *backend*.
Responsabilidades
Ejecutar las comprobaciones en cuanto a la creación y uso del edificio del centro del poblado.
Atributos Propuestos
Métodos Propuestos

6.10.3 Diagrama de flujo del módulo

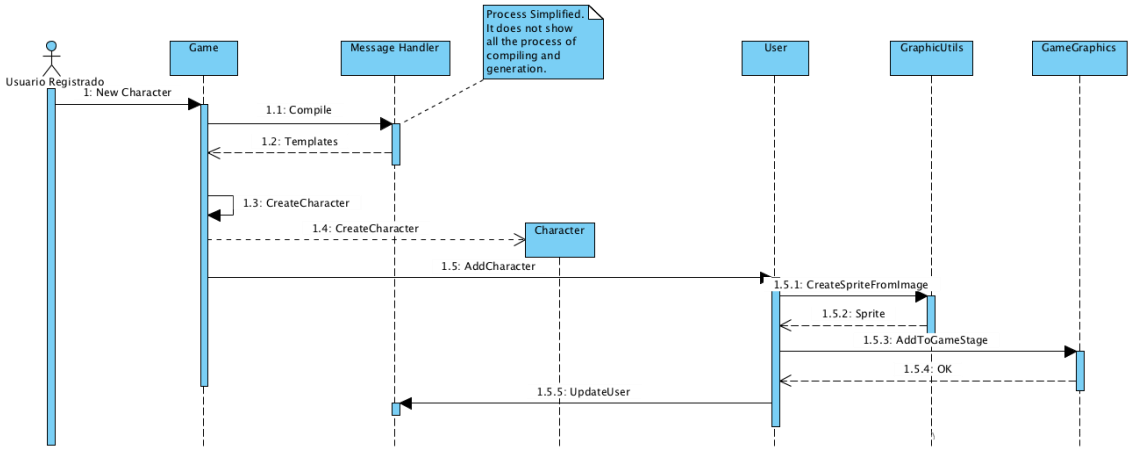


Figura 6.30 Frontend: Creación de personaje

6.11 Módulo: Plantillas de ejecución

6.11.1 Diagrama de diseño del frontend

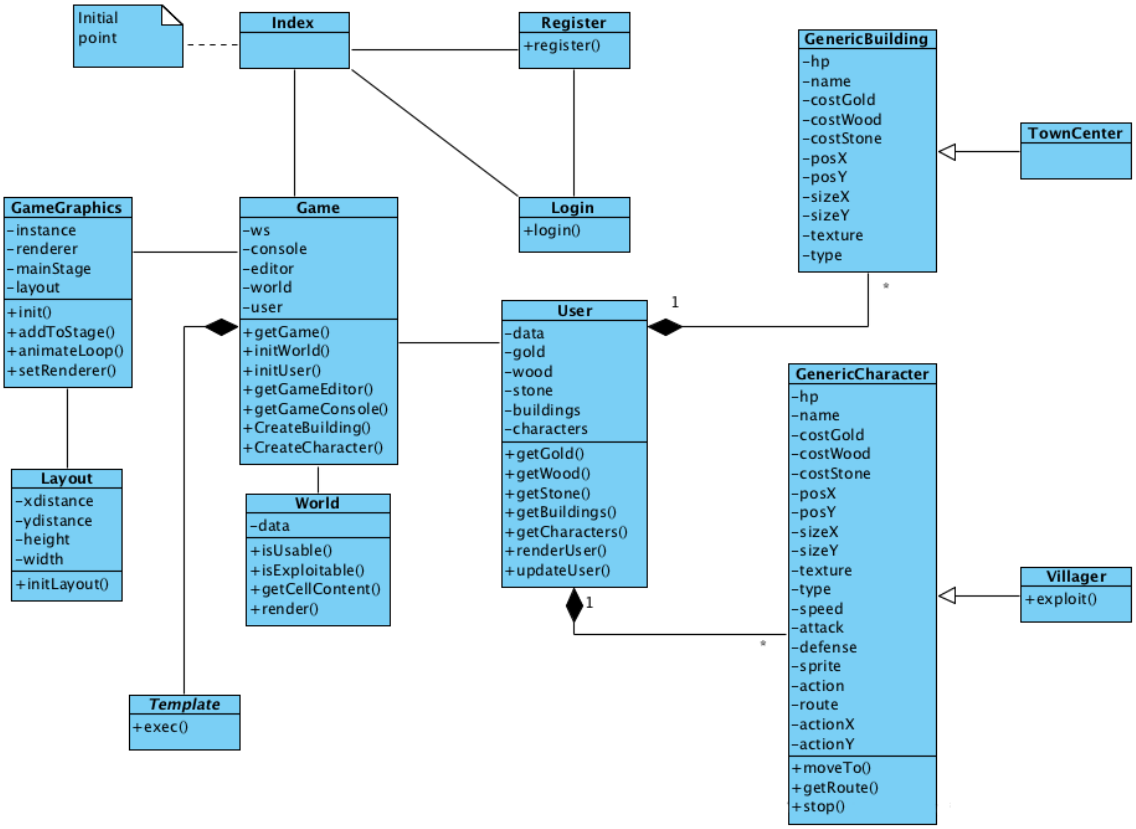


Figura 6.31 Frontend: Plantillas de ejecución

Nombre de la Clase
Template
Descripción
Encargada de actuar como clase base a todas las plantillas del juego.
Responsabilidades
Representar la base de todas las plantillas del juego.
Atributos Propuestos
Métodos Propuestos
exec: Ejecuta la plantilla.

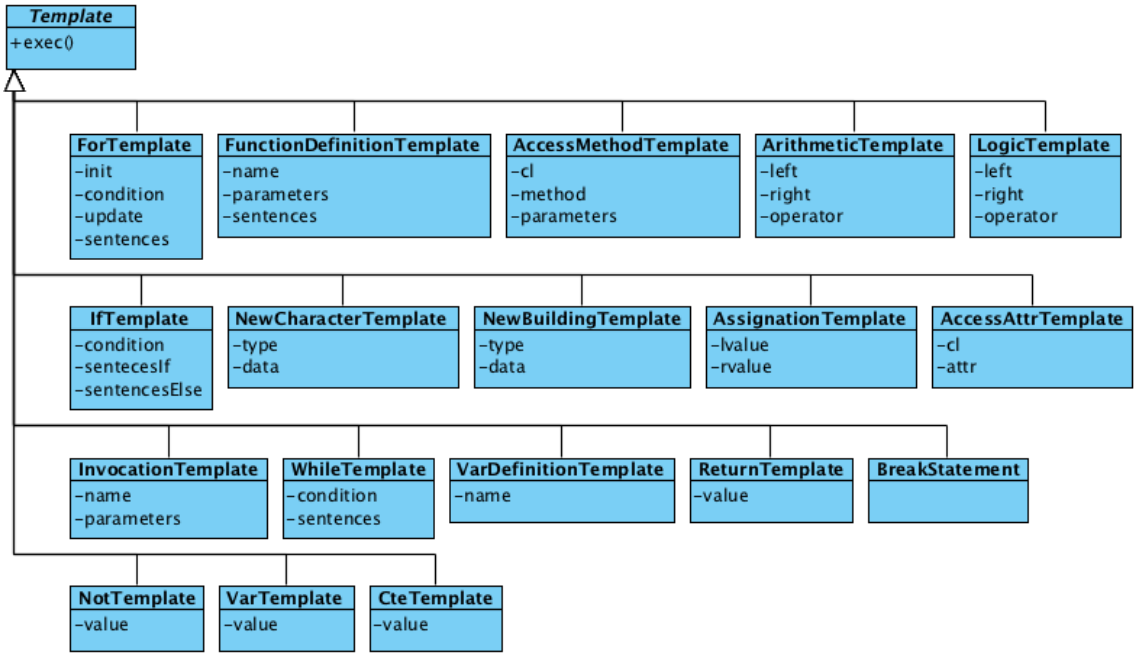


Figura 6.32 Detalle plantillas de ejecución frontend

Nombre de la Clase
ForTemplate
Descripción
Encargada de ejecutar en *frontend* un bucle *for*.
Responsabilidades
Ejecutar un bucle *for* con los parámetros provistos.
Atributos Propuestos
init: Valor de inicialización del bucle. **condition**: Condición de ejecución del bucle. **update**: Condición de actualización del bucle. **sentences**: Sentencias a ejecutar en el bucle *for*.
Métodos Propuestos

Nombre de la Clase
FunctionDefinitionTemplate
Descripción
Encargada de representar en *frontend* una definición de función y ejecutarla.
Responsabilidades
Representar en *frontend* una definición de función y ejecutarla de forma correcta.
Atributos Propuestos
name: Nombre identificativo de la función. **parameters**: Parámetros a recibir por la función. **sentences**: Sentencias a ejecutar en el ámbito de la función.
Métodos Propuestos

Nombre de la Clase
AccessMethodTemplate
Descripción
Encargada de ejecutar en *frontend* un acceso a un método de una clase.
Responsabilidades
Ejecutar el acceso a un método de una clase.
Atributos Propuestos
cl: Clase que contiene el método a invocar. **method**: Método a invocar dentro de la clase. **parameters**: Parámetros con los que se invocara el acceso al método.
Métodos Propuestos

Nombre de la Clase
ArithmeticTemplate
Descripción
Encargada de ejecutar en *frontend* una operación aritmética.
Responsabilidades
Ejecutar una operación aritmética en *frontend*.
Atributos Propuestos
left: Parte izquierda de la operación aritmética. **right**: Parte derecha de la operación aritmética. **operator**: Operador de la operación.
Métodos Propuestos

Nombre de la Clase
LogicTemplate
Descripción
Encargada de ejecutar en *frontend* una operación lógica.
Responsabilidades
Ejecutar una operación lógica.
Atributos Propuestos
left: Parte izquierda de la operación lógica. **right**: Parte derecha de la operación lógica. **operator**: Operador de la operación.
Métodos Propuestos

Nombre de la Clase
IfTemplate
Descripción
Encargada de ejecutar en *frontend* una instrucción *if*.
Responsabilidades
Ejecutar una instrucción *if*.
Atributos Propuestos
condition: Condición del *if*. **sentencesIf**: Sentencias a ejecutar si la condición se cumple. **sentencesElse**: Sentencias a ejecutar si la condición no se cumple.
Métodos Propuestos

Nombre de la Clase
NewCharacterTemplate
Descripción
Encargada de ejecutar en *frontend* la instanciación de un nuevo personaje.
Responsabilidades
Instanciar un nuevo personaje.
Atributos Propuestos
type: Tipo del personaje a instanciar. **data**: Datos a utilizar en la instanciación.
Métodos Propuestos

Nombre de la Clase
NewBuildingTemplate
Descripción
Encargada de ejecutar en *frontend* la instanciación de un nuevo edificio.
Responsabilidades
Instanciar un nuevo edificio.
Atributos Propuestos
type: Tipo del edificio a instanciar. **data**: Datos a utilizar en la instanciación.
Métodos Propuestos

Nombre de la Clase
AssignationTemplate
Descripción
Encargada de ejecutar en *frontend* una operación de asignación de expresiones.
Responsabilidades
Ejecutar una operación de asignación.
Atributos Propuestos
lvalue: Parte izquierda de la operación. **rvalue**: Parte derecha de la operación.
Métodos Propuestos

Nombre de la Clase
AccessAttrTemplate
Descripción
Encargada de ejecutar en *frontend* un acceso a un atributo de una clase.
Responsabilidades
Ejecutar una operación de acceso a atributo.
Atributos Propuestos
cl: Clase que contiene el atributo. **attr**: Atributo de la clase a acceder.
Métodos Propuestos

Nombre de la Clase
InvocationTemplate
Descripción
Encargada de ejecutar en *frontend* una invocación a función.
Responsabilidades
Ejecutar una operación de invocación a función.
Atributos Propuestos
name: Nombre de la función a invocar. **parameters**: Parámetros a usar en la invocación a función.
Métodos Propuestos

Nombre de la Clase
WhileTemplate
Descripción
Encargada de ejecutar en *frontend* un bucle de tipo *While*.
Responsabilidades
Ejecutar un bucle *while*.
Atributos Propuestos
condition: Condición de ejecución del bucle *while*. **sentences**: Sentencias a ejecutar en el bucle si la condición se cumple.
Métodos Propuestos

Nombre de la Clase
VarDefinitionTemplate
Descripción
Encargada de representar una definición de variable en *frontend*.
Responsabilidades
Representar y actuar como una definición de variable en *frontend*.
Atributos Propuestos
name: Nombre de la variable definida.
Métodos Propuestos

Nombre de la Clase
ReturnTemplate
Descripción
Encargada de ejecutar en *frontend* una operación de retorno.
Responsabilidades
Ejecutar una operación de retorno.
Atributos Propuestos
value: Valor a retornar.
Métodos Propuestos

Nombre de la Clase
BreakStatement
Descripción
Encargada de detener la ejecución de un bucle en *frontend*.
Responsabilidades
Detener la ejecución de un bucle en *frontend*.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
NotTemplate
Descripción
Encargada de ejecutar en *frontend* una operación de negación.
Responsabilidades
Ejecutar una operación de negación de valor.
Atributos Propuestos
value: Valor a negar.
Métodos Propuestos

Nombre de la Clase
VarTemplate
Descripción
Encargada de ejecutar un acceso a variable.
Responsabilidades
Ejecutar un acceso a variable.
Atributos Propuestos
value: Variable a acceder.
Métodos Propuestos

Nombre de la Clase
CteTemplate
Descripción
Encargada de representar el valor de una constante.
Responsabilidades
Representar una constante en *frontend*
Atributos Propuestos
value: Valor de la constante.
Métodos Propuestos

6.11.2 Diagrama de diseño del backend

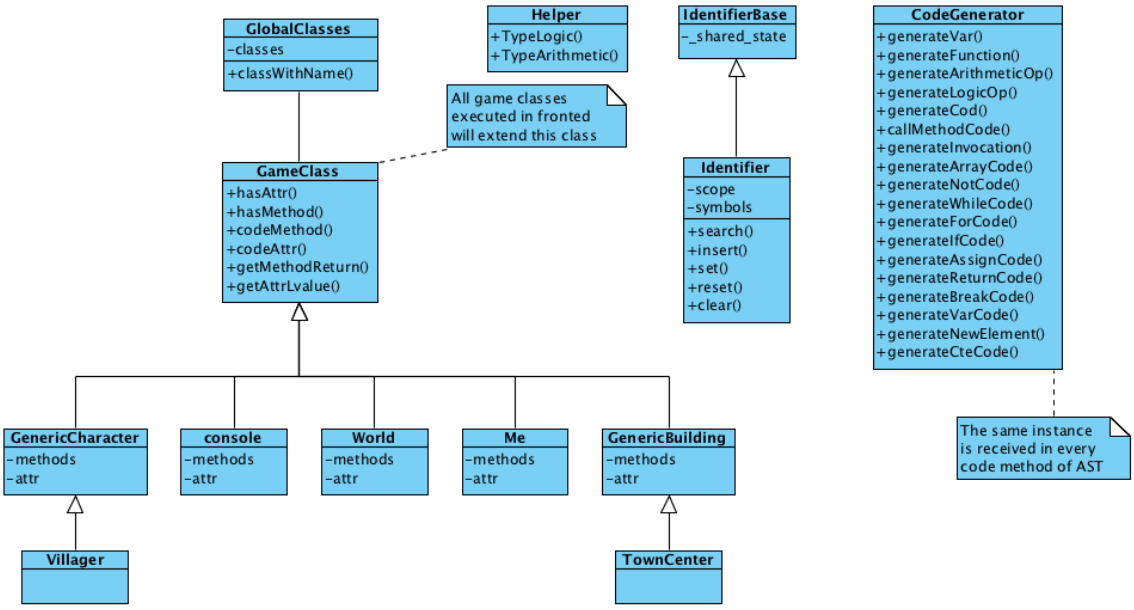


Figura 6.33 Utilidades AST: Plantillas de ejecución

Se ha ampliado la clase CodeGenerator para adaptarla al nuevo sistema de plantillas.

Nombre de la Clase
CodeGenerator
Descripción
Generador de código para cada una de las clases del AST. Una instancia de esta clase es pasada por el árbol en el método *code* de cada nodo.
Responsabilidades
Generar el código equivalente de un AST concreto.
Atributos Propuestos
Métodos Propuestos
generateVar: Genera el código de una definición de variable. **generateFunction**: Genera el código de una definición de función. **generateArithmeticOp**: Genera el código de una operación aritmética. **generateLogicOp**: Genera el código de una operación lógica. **generateCode**: Inserta el código recibido dentro del generador. **callMethodCode**: Genera el código de una llamada a método. **generateInvocation**: Genera el código de una invocación a función. **generateNotCode**: Genera el código de una negación unaria. **generateWhileCode**: Genera el código correspondiente a una sentencia *while*. **generateForCode**: Genera el código correspondiente a un bucle *for*. **generateIfCode**: Genera el código correspondiente a una sentencia *if-else*. **generateAssignCode**: Genera el código correspondiente a una sentencia de asignación. **generateReturnCode**: Genera el código correspondiente a una sentencia *return*. **generateBreakCode**: Genera el código correspondiente a una sentencia *break*. **generateVarCode**: Genera el código correspondiente al valor de una variable. **generateNewElement**: Genera el código correspondiente a una nueva instanciación de clase. **generateCteCode**: Genera el código correspondiente a una constante.

6.12 Módulo: Tutorial de juego

6.12.1 Diagrama de diseño del frontend

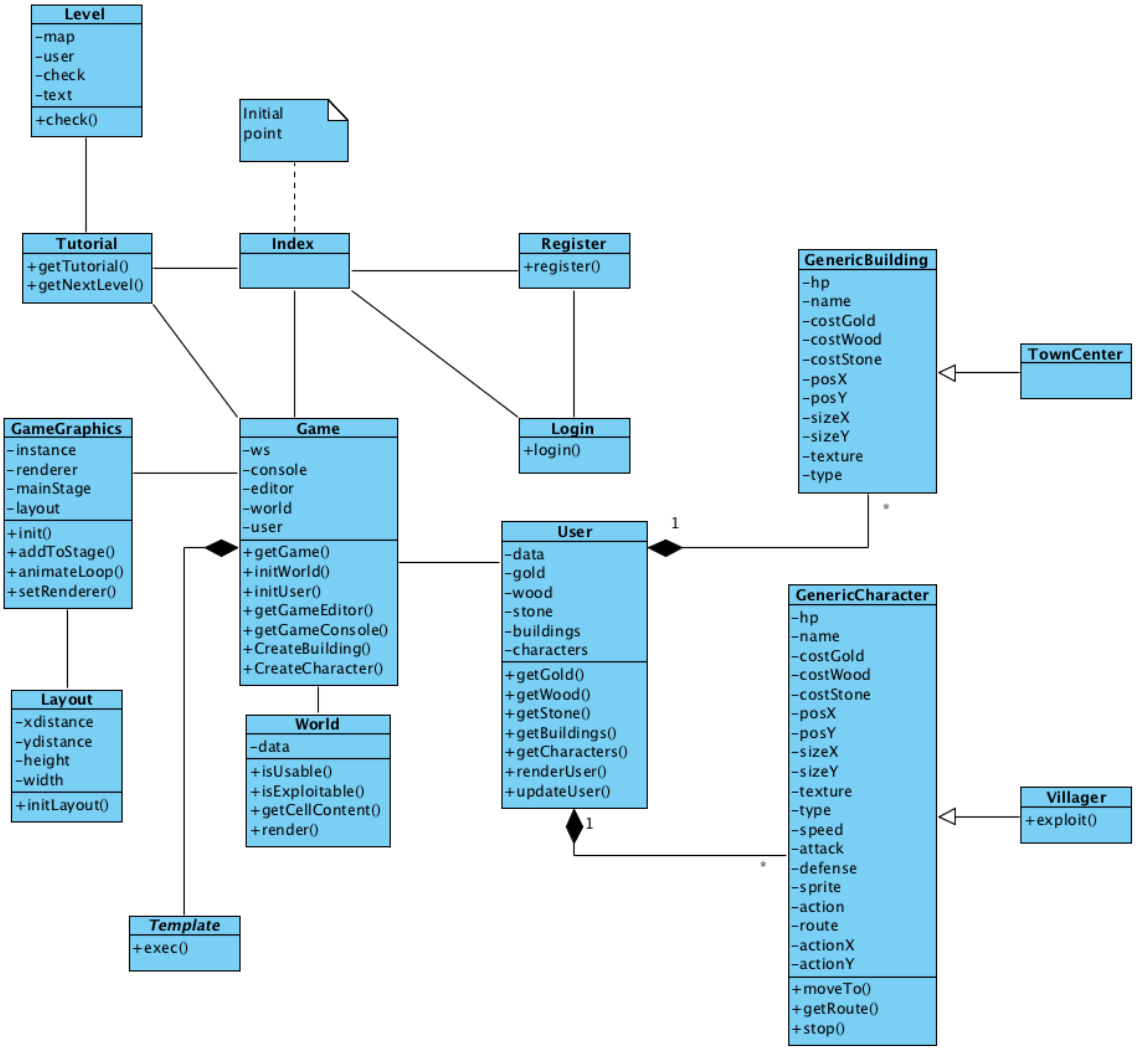


Figura 6.34 Frontend: Tutorial de juego

Nombre de la Clase
Tutorial
Descripción
Encargada de representar la página de tutorial e interactuar limitadamente con el juego.
Responsabilidades
Representar la página del tutorial del juego, sus pantallas y fases.
Atributos Propuestos
Métodos Propuestos
getTutorial: Recupera la instancia del tutorial. **getNextLevel**: Recupera el siguiente nivel del tutorial.

Nombre de la Clase
Level
Descripción
Encargada de representar un nivel de juego del tutorial.
Responsabilidades
Representar un nivel de juego del tutorial.
Atributos Propuestos
map: Mapa del mundo del nivel de tutorial concreto. **user**: Representación del usuario para esta fase del tutorial. **check**: Lista de comprobaciones empleada para confirmar el fin del tutorial. **text**: Diferentes textos a mostrar durante el tutorial.
Métodos Propuestos
check: Ejecuta todas las comprobaciones y retorna si se cumplen todas o no.

6.12.2 Diagrama de diseño del backend

6.12.2.1 Diagrama de la base de datos

Se ha modificado el diagrama de la base de datos para dar cabida al tutorial.

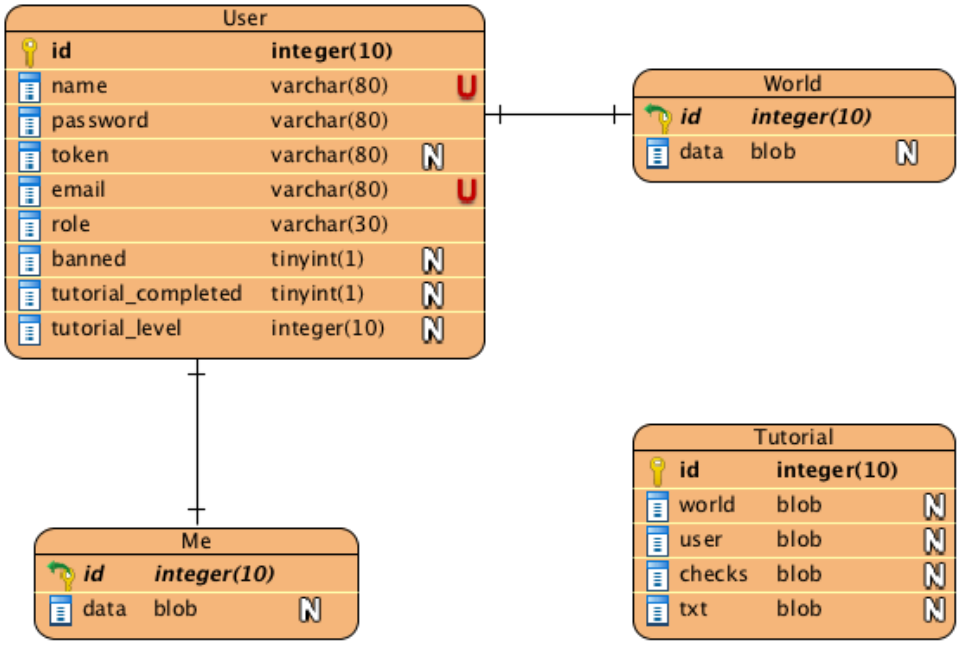


Figura 6.35 Base de datos: Tutorial de juego

Nombre de la Clase
Tutorial
Descripción
Encargada de almacenar en la base de datos todo el contenido de las distintas fases del tutorial.
Responsabilidades
Almacenar los datos de los distintos niveles del tutorial en la base de datos.
Atributos Propuestos
id: Id único del nivel del juego. **world**: Mapa del mundo del nivel de tutorial concreto. **user**: Representación del usuario para esta fase del tutorial. **check**: Lista de comprobaciones empleada para confirmar el fin del tutorial. **text**: Diferentes textos a mostrar durante el tutorial.
Métodos Propuestos

Nombre de la Clase
User
Descripción
Tabla de la base de datos que contendrá los datos concretos del usuario.
Responsabilidades
Contener los datos concretos de cada usuario.
Atributos Propuestos
id: Identificador único del usuario. **name**: Nombre único del usuario. **password**: Hash de la contraseña del usuario. **token**: *Token* aleatorio generado para asegurar la autenticación del usuario. **email**: Email único del usuario. **role**: Rol del usuario en el sistema. Por defecto será *USER*. **banned**: Indicador de si el acceso del usuario se encuentra restringido. **tutorial_completed**: Indicador de si el usuario ha completado el tutorial. **tutorial_level**: Ultimo nivel del tutorial superado por el jugador.
Métodos Propuestos

6.12.2.2 Diagrama de servidor backend

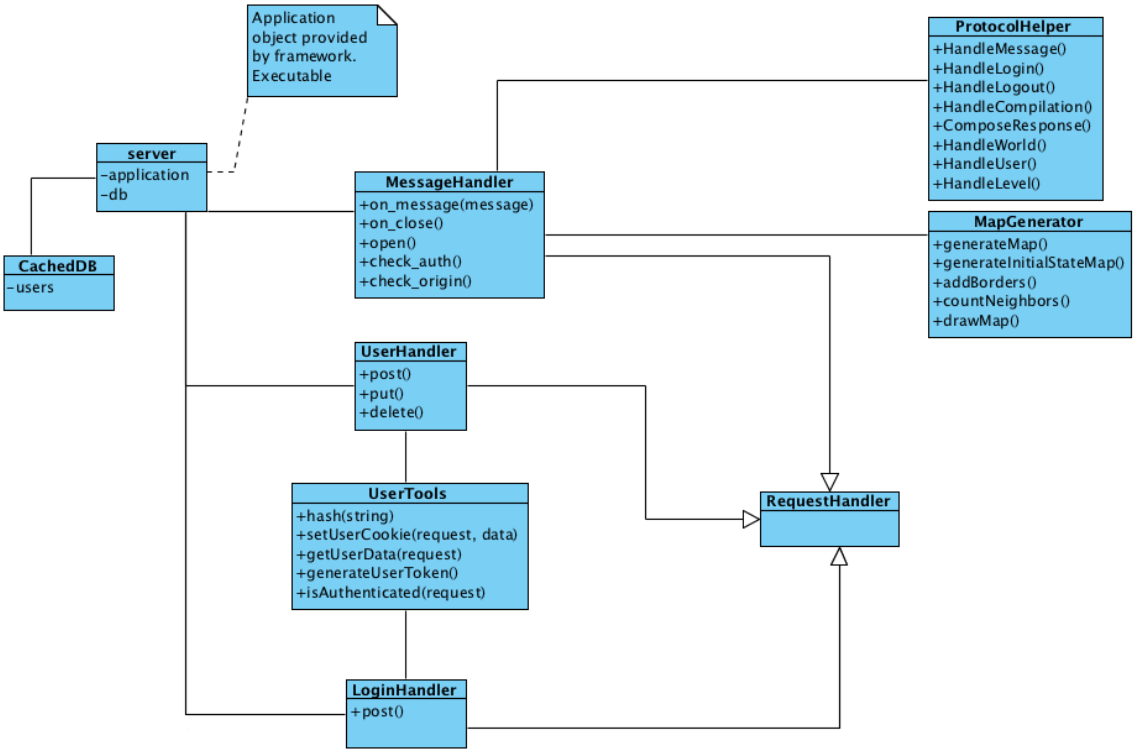


Figura 6.36 Backend: Tutorial de juego

Se ha añadido un nuevo método al protocolo del WebSocket para solicitar nuevos niveles.

Nombre de la Clase
ProtocolHelper
Descripción
Encargada de procesar todos los mensajes recibidos en el WebSocket de acuerdo al protocolo establecido, dependiendo de su tipo y de los parámetros incluidos en el mensaje.
Responsabilidades
Clasificar y responder los mensajes del protocolo de forma correcta y ordenada.
Atributos Propuestos
Métodos Propuestos
HandleMessage: Atiende un nuevo mensaje del protocolo clasificándolo según su tipo. **HandleLogin**: Atiende un mensaje de *login* del WebSocket. **HandleLogout**: Atiende un mensaje de *logout* del WebSocket, actualizando la base de datos con el nuevo mundo del usuario y con su nuevo objeto de usuario. **HandleCompilation**: Invoca al analizador léxico-sintáctico y *parsea* el código del usuario, notifica si la compilación ha sido correcta y en caso contrario notifica los errores. **ComposeResponse**: Compone una nueva respuesta del protocolo. **HandleWorld**: Encargado de actualizar el mundo en *backend*. **HandleUser**: Encargado de actualizar el usuario en *backend*. **HandleLevel**: Encargado de enviar niveles del tutorial a *frontend*.

6.13 Módulo: Administración del tutorial y usuarios

6.13.1 Diagrama de diseño del backend

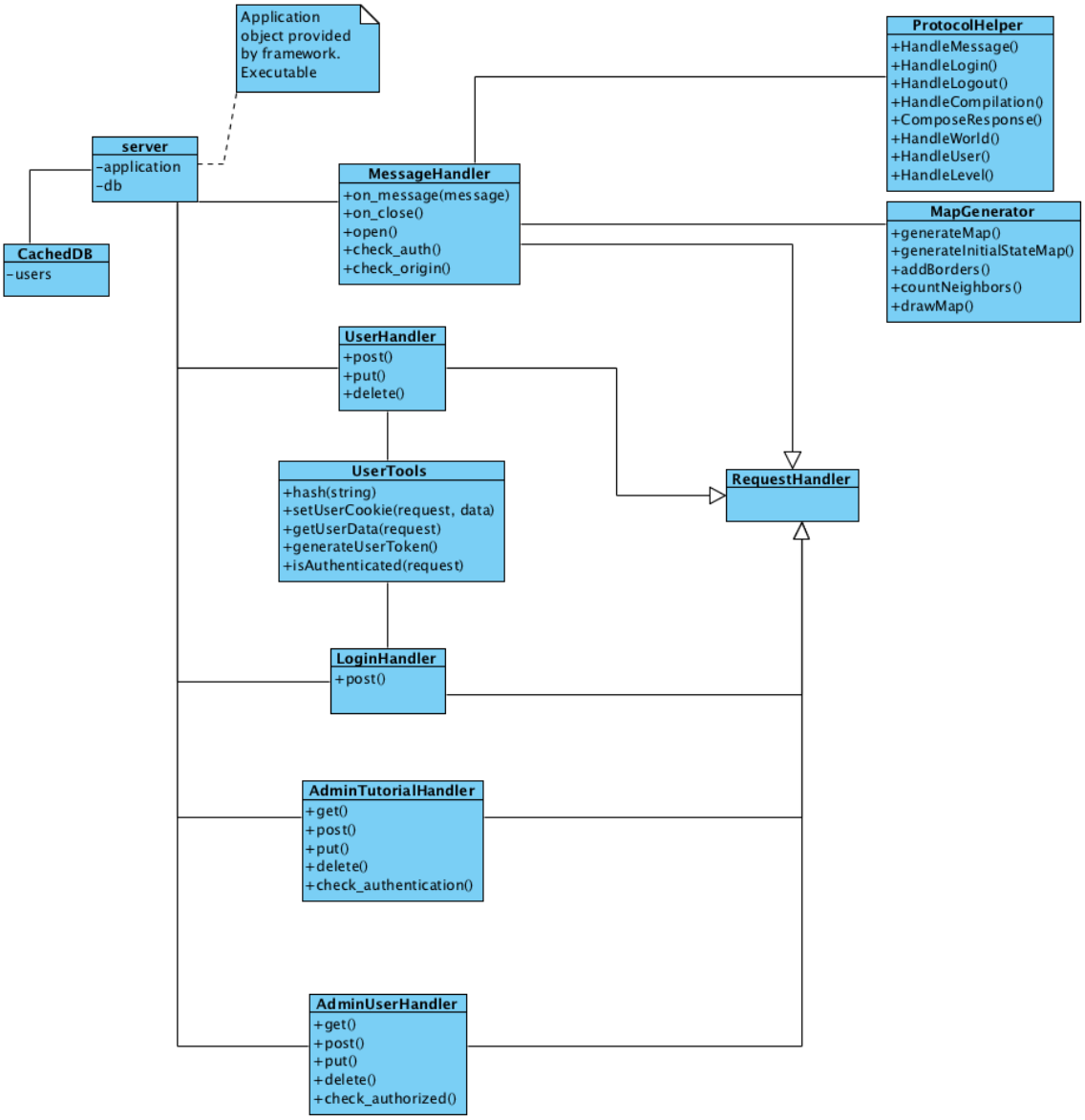


Figura 6.37 Backend: Administración del tutorial y usuarios

Nombre de la Clase
AdminTutorialHandler
Descripción
Encargada de la recuperación, creación, edición y eliminación de fases del tutorial.
Responsabilidades
Gestión de los tutoriales para administradores.
Atributos Propuestos
Métodos Propuestos
get: Recupera una fase de tutorial de la base de datos y la envía a *frontend*. **post**: Crea una nueva fase de tutorial en la base de datos. **put**: Actualiza una fase de tutorial en la base de datos. **delete**: Elimina una fase de tutorial de la base de datos. **check_authentication**: Comprueba que la petición se realiza desde un usuario correctamente autenticado y con privilegios de administración.

Nombre de la Clase
AdminUserHandler
Descripción
Encargada de la recuperación, creación, edición y eliminación de usuarios por parte de un administrador.
Responsabilidades
Gestión de los usuarios para administradores.
Atributos Propuestos
Métodos Propuestos
get: Recupera un usuario y lo envía a *frontend*. **post**: Crea un nuevo usuario en la base de datos. **put**: Actualiza un usuario en la base de datos. **delete**: Elimina un usuario de la base de datos. **check_authorized**: Comprueba que la petición se realiza desde un usuario correctamente autenticado y con privilegios de administración.

6.13.2 Diagrama de diseño del frontend

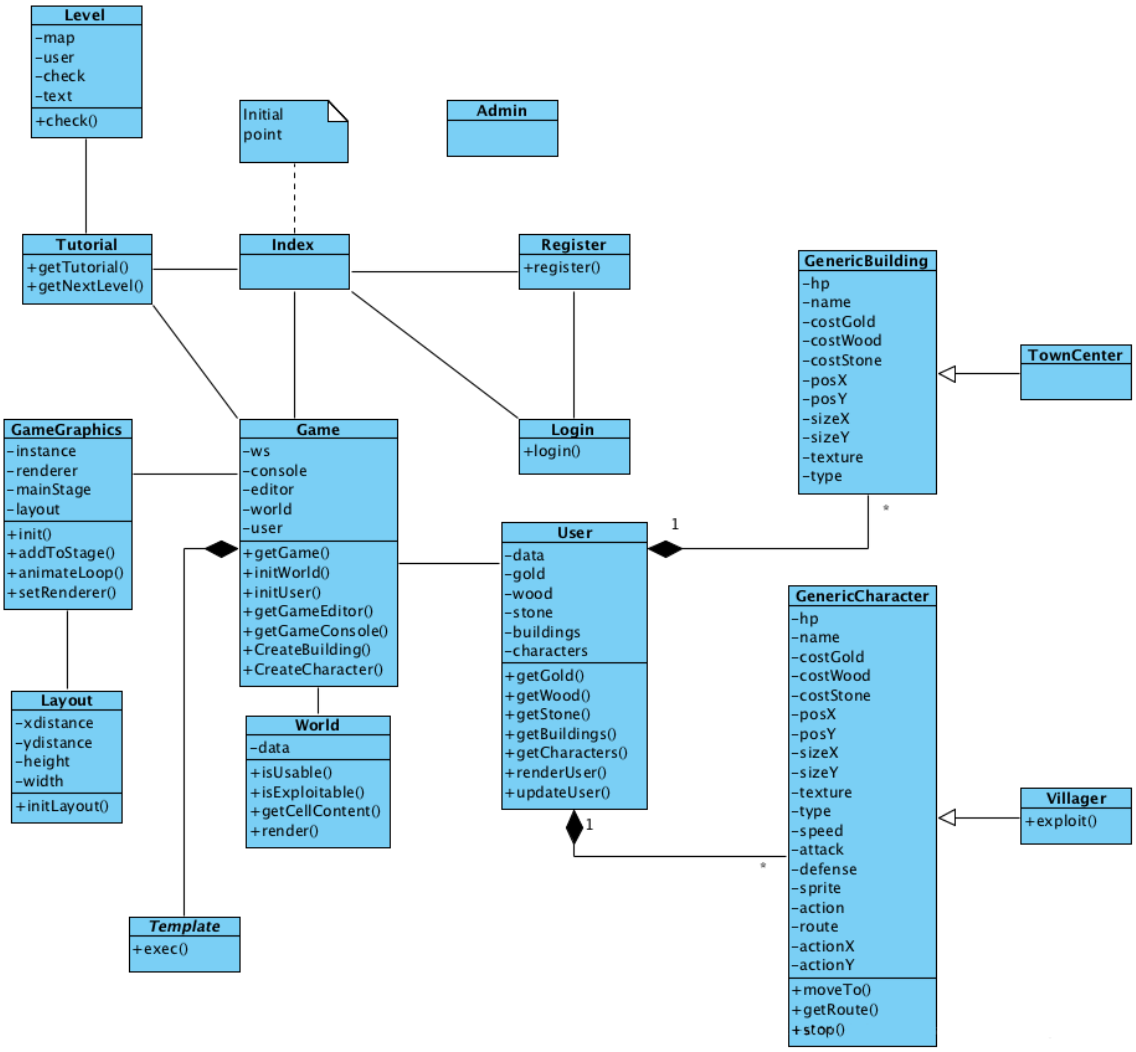


Figura 6.38 Frontend: Administración del tutorial

Nombre de la Clase
Admin
Descripción
Página de administración completamente separada del sitio de juego principal por motivos de seguridad.
Responsabilidades
Interactuar con el punto de entrada de gestión de fases del tutorial.
Atributos Propuestos
Métodos Propuestos

6.14 Módulo: Enemigos

6.14.1 Diagrama de diseño del frontend

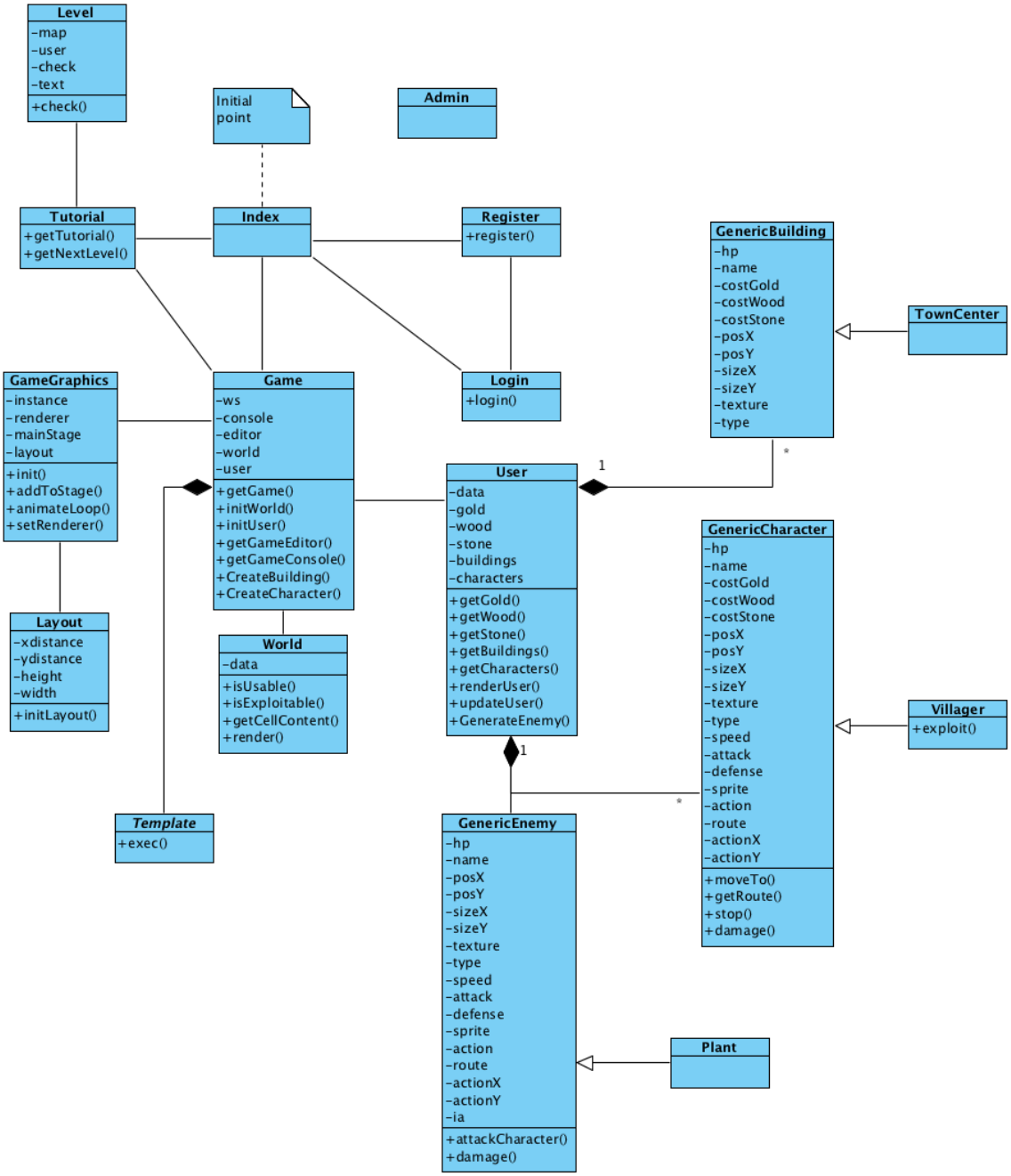


Figura 6.39 Frontend: Enemigos

Nombre de la Clase
GenericCharacter
Descripción
Representación en *frontend* de todo personaje genérico.
Responsabilidades
Proveer los métodos y atributos mínimos de un personaje.
Atributos Propuestos
hp: Representa los puntos de daño del personaje. **name**: Nombre identificador único del personaje. **costGold**: Coste en oro de la construcción del personaje. **costWood**: Coste en madera de la construcción del personaje. **costStone**: Coste en piedra de la construcción del personaje. **posX**: Posición en X del personaje. **posY**: Posición en Y del personaje. **sizeX**: Tamaño en X del personaje. **sizeY**: Tamaño en Y del personaje. **texture**: Textura a representar para el personaje. **type**: Tipo de personaje. **speed**: Velocidad de movimiento del personaje. **attack**: Puntos de ataque del personaje. **defense**: Puntos de defensa del personaje. **sprite**: *Sprite* asignado al personaje. **action**: Acción a realizar por el personaje o que esté realizando. **route**: Ruta a seguir en caso de haber invocado el método de movimiento. **actionX**: Coordenada X en la que realizar la acción. **actionY**: Coordenada Y en la que realizar la acción.
Métodos Propuestos
moveTo: Inicia el movimiento del personaje hacia las coordenadas de destino. **getRoute**: Calcula una ruta optima utilizando el algoritmo A* hasta la casilla destino. **stop**: Detiene la acción en curso del usuario. **damage**: Recibe una cantidad de daño concreta debido a un ataque.

Nombre de la Clase
User
Descripción
Clase que contendrá los datos del usuario en *frontend* para que sean consultados.
Responsabilidades
Contener los datos concretos del objeto usuario del usuario y facilitar su interacción.
Atributos Propuestos
data: Contiene los datos brutos del objeto usuario. **gold**: Cantidad de oro de la que dispone el usuario. **wood**: Cantidad de madera de la que dispone el usuario. **stone**: Cantidad de piedra de la que dispone el usuario. **buildings**: Lista de edificios del usuario. **characters**: Lista de personajes del usuario.
Métodos Propuestos
getGold: Recupera el oro del usuario. **getWood**: Recupera la madera del usuario. **getStone**: Recupera la piedra del usuario. **getBuildings**: Recupera la lista de edificios del usuario. **getCharacters**: Recupera la lista de personajes del usuario. **renderUser**: Dibuja en el mapa todos los edificios y personajes del usuario. **updateUser**: Actualiza el usuario en *backend*. **GenerateEnemy**: Crea un nuevo enemigo en una posición aleatoria siempre que se encuentre libre.

Nombre de la Clase
GenericEnemy
Descripción
Representación en frontend de todo enemigo genérico.
Responsabilidades
Proveer los métodos y atributos mínimos de un enemigo.
Atributos Propuestos
hp: Representa los puntos de daño del enemigo. **name**: Nombre identificador único del enemigo. **posX**: Posición en X del enemigo. **posY**: Posición en Y del enemigo. **sizeX**: Tamaño en X del enemigo. **sizeY**: Tamaño en Y del enemigo. **texture**: Textura a representar para el enemigo. **type**: Tipo de enemigo. **speed**: Velocidad de movimiento del enemigo. **attack**: Puntos de ataque del enemigo. **defense**: Puntos de defensa del enemigo. **sprite**: *Sprite* asignado al enemigo. **action**: Acción a realizar por el enemigo o que esté realizando. **route**: Ruta a seguir en caso de haber invocado el método de movimiento. **actionX**: Coordenada X en la que realizar la acción. **actionY**: Coordenada Y en la que realizar la acción. **ia**: Inteligencia artificial a aplicar en el enemigo.
Métodos Propuestos
attackCharacter: Ataca al personaje indicado. **damage**: Recibe una cantidad de daño concreta debido a un ataque.

Nombre de la Clase
Plant
Descripción
Representación en *frontend* de una planta enemiga que ataca a aquellos que la rodean.
Responsabilidades
Proveer los métodos y atributos concretos de este enemigo.
Atributos Propuestos
Métodos Propuestos

6.15 Módulo: Ampliación enemigos y combate

6.15.1 Diagrama de diseño del frontend

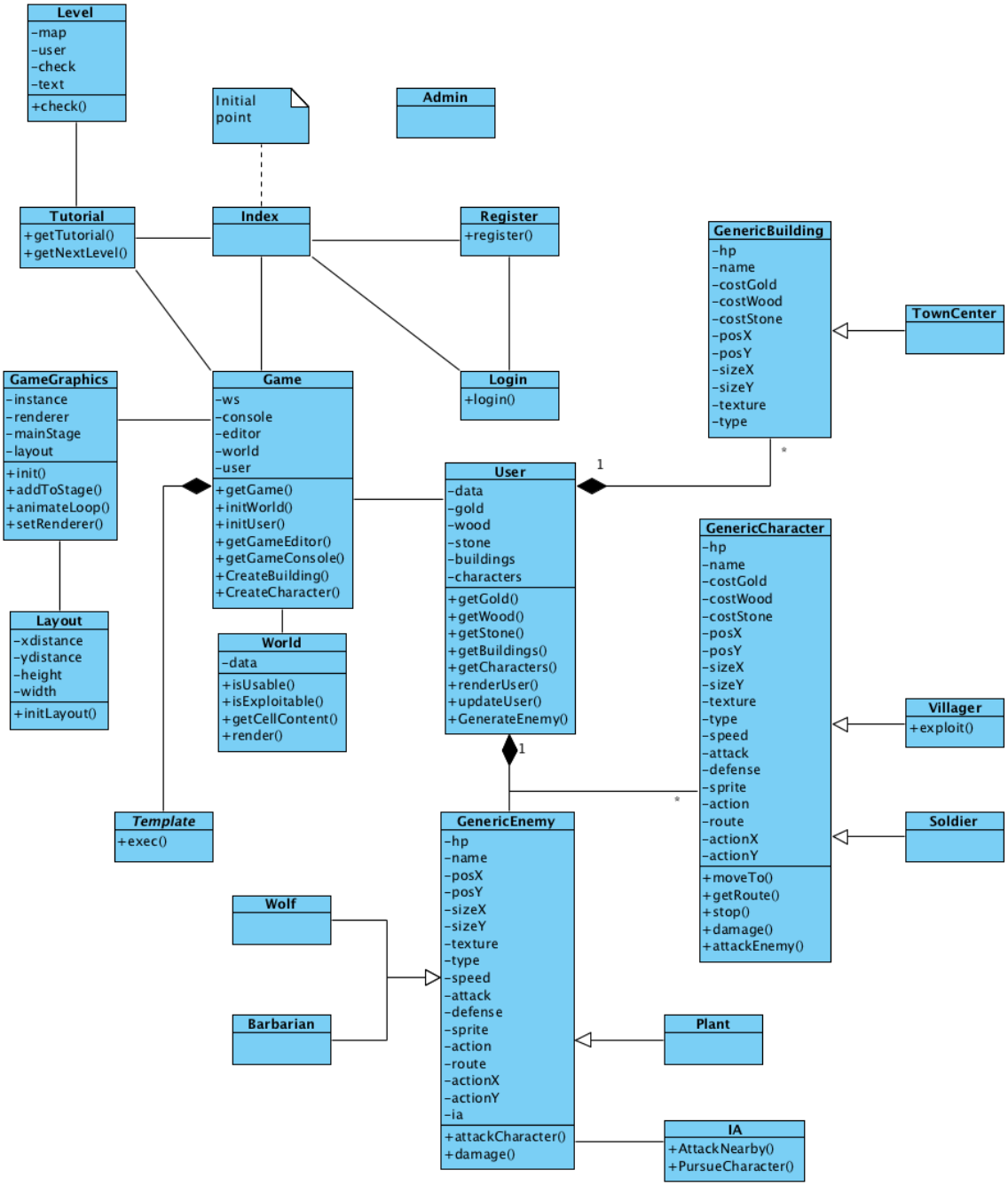


Figura 6.40 Frontend: Ampliación enemigos y combate

Nombre de la Clase
GenericCharacter
Descripción
Representación en *frontend* de todo personaje genérico.
Responsabilidades
Proveer los métodos y atributos mínimos de un personaje.
Atributos Propuestos
hp: Representa los puntos de daño del personaje. **name**: Nombre identificador único del personaje. **costGold**: Coste en oro de la construcción del personaje. **costWood**: Coste en madera de la construcción del personaje. **costStone**: Coste en piedra de la construcción del personaje. **posX**: Posición en X del personaje. **posY**: Posición en Y del personaje. **sizeX**: Tamaño en X del personaje. **sizeY**: Tamaño en Y del personaje. **texture**: Textura a representar para el personaje. **type**: Tipo de personaje. **speed**: Velocidad de movimiento del personaje. **attack**: Puntos de ataque del personaje. **defense**: Puntos de defensa del personaje. **sprite**: *Sprite* asignado al personaje. **action**: Acción a realizar por el personaje o que esté realizando. **route**: Ruta a seguir en caso de haber invocado el método de movimiento. **actionX**: Coordenada X en la que realizar la acción. **actionY**: Coordenada Y en la que realizar la acción.
Métodos Propuestos
moveTo: Inicia el movimiento del personaje hacia las coordenadas de destino. **getRoute**: Calcula una ruta optima utilizando el algoritmo A* hasta la casilla destino. **stop**: Detiene la acción en curso del usuario. **damage**: Recibe una cantidad de daño concreta debido a un ataque. **attackEnemy**: Ataca al enemigo especificado en la casilla indicada.

Nombre de la Clase
Wolf
Descripción
Representación en *frontend* de un lobo enemigo que ataca a aquellos que se encuentran a una casilla de distancia y persigue a todos aquellos que se encuentran a dos o menos.
Responsabilidades
Proveer los métodos y atributos concretos de este enemigo.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
Barbarian
Descripción
Representación en *frontend* de un bárbaro enemigo que ataca a aquellos que se encuentran a una casilla de distancia y persigue a todos aquellos que se encuentran a dos o menos. El bárbaro es más fuerte que el lobo.
Responsabilidades
Proveer los métodos y atributos concretos de este enemigo.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
Soldier
Descripción
Representación en *frontend* de un soldado amistoso, mucho más fuerte que el aldeano pero sin posibilidad de recolectar materiales.
Responsabilidades
Proveer los métodos y atributos concretos de esta unidad.
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
IA
Descripción
Agrupación de los distintos algoritmos de Inteligencia artificial para enemigos implementados en el juego.
Responsabilidades
Agrupar las distintas IAs del juego para los enemigos.
Atributos Propuestos
Métodos Propuestos
AttackNearby: Ataca a todos los personajes que se encuentren en un radio de una casilla del enemigo que tenga esta IA asignada. **PursueCharacter**: Persigue a todos los personajes que se encuentren en un radio de dos casillas del enemigo y si se encuentra a una casilla lo ataca.

6.15.2 Diagrama de diseño del backend

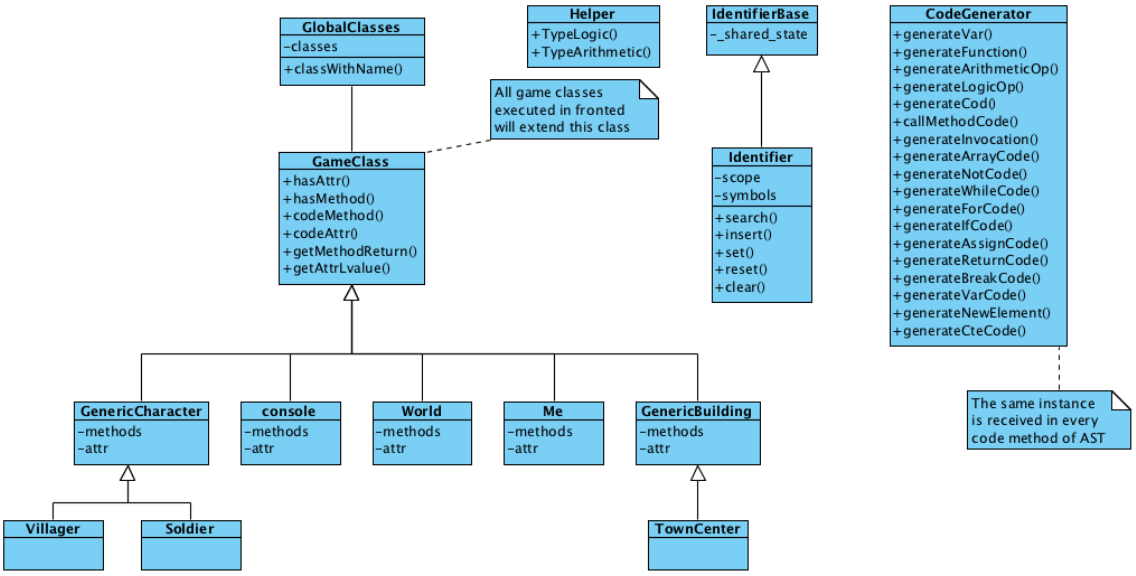


Figura 6.41 Backend: Ampliación de enemigos y combate

Nombre de la Clase
Soldier
Descripción
Representación en *backend* de un soldado amistoso, mucho más fuerte que el aldeano pero sin posibilidad de recolectar materiales.
Responsabilidades
Proveer los métodos y atributos concretos de esta unidad.
Atributos Propuestos
Métodos Propuestos

Capítulo 7. Gramática y analizadores

7.1 Introducción

Se ha creado una gramática de libre contexto que reconoce un lenguaje cuya sintaxis se encuentra basada en JavaScript.

Uno de los objetivos de esta gramática es reconocer símbolos pertenecientes a dicho lenguaje, analizar el código introducido por el usuario tratando de construir un árbol sintáctico y generar una serie de plantillas que se ejecuten de la forma esperada para el usuario.

No obstante, el principal objetivo es reportar la mayor cantidad de información posible al usuario en caso de que se produzca un error con el código que ha introducido, para ello se realiza una gran cantidad de comprobaciones en tiempo de análisis, más allá del mero análisis léxico y sintáctico.

Esta gramática se encuentra localizada en la parte *backend* de la aplicación, principalmente por seguridad ya que si en un futuro se realiza una ampliación de modo multijugador, todo el análisis sintáctico y léxico podría ser modificado por un usuario malicioso.

7.2 Backus Naur Form (BNF) de la gramática desarrollada

A continuación se incluye el *Backus Naur Form* (BNF) de la gramática desarrollada:

```
<program> ::= <program> <instructions>
        |

<instructions> ::= <statement>
        | <definitions>

<definitions> ::= <var_definition>
        | <function_definition>
<var_definition> ::= VAR ID SEMICOLON
        | VAR ID ASSIGN <expression> SEMICOLON

<function_definition> ::= FUNCTION ID LPAREN RPAREN LBRACK <statements> RBRACK
        | FUNCTION ID LPAREN <dec_param> RPAREN LBRACK <statements> RBRACK
<dec_param> ::= <expression> COMMA <dec_param>
        | <expression>

<statements> ::= <statements> <statement>
        |

<statement> ::= BREAK SEMICOLON
        | ID LPAREN RPAREN SEMICOLON
        | ID LPAREN <dec_param> RPAREN SEMICOLON
        | <expression> POINT ID LPAREN RPAREN SEMICOLON
        | <expression> POINT ID LPAREN <dec_param> RPAREN SEMICOLON
        | RETURN <expression> SEMICOLON
        | RETURN SEMICOLON
        | <expression> ASSIGN <expression> SEMICOLON
        | IF LPAREN <expression> RPAREN LBRACK <statements> RBRACK
        | IF LPAREN <expression> RPAREN LBRACK <statements> RBRACK ELSE LBRACK
<statements> RBRACK
        | WHILE LPAREN <expression> RPAREN LBRACK <statements> RBRACK
        | FOR LPAREN <expression> ASSIGN <expression> SEMICOLON <expression> SEMICOLON
<expression> ASSIGN <expression> RPAREN LBRACK <statements> RBRACK

<expression> ::= <expression> PLUS <expression>
        | <expression> MINUS <expression>
        | <expression> TIMES <expression>
        | <expression> DIVIDE <expression>
        | <expression> MODULE <expression>
        | <expression> MINOR <expression>
        | <expression> MAJOR <expression>
        | <expression> MINOREQUAL <expression>
        | <expression> MAJOREQUAL <expression>
        | <expression> AND <expression>
        | <expression> OR <expression>
        | <expression> DIFFERENT <expression>
        | <expression> EQUALS <expression>
        | <expression> POINT ID LPAREN RPAREN
        | <expression> POINT ID LPAREN <dec_param> RPAREN
        | <expression> POINT ID
        | NEW ID LPAREN <dec_param> RPAREN
        | NEW ID LPAREN RPAREN
        | ID LPAREN RPAREN
        | ID LPAREN <dec_param> RPAREN
        | LPAREN <expression> RPAREN
        | NEGATE <expression>
        | ID
        | NUMBER
        | CTESTRING
```

Código fuente 1 BNF de gramática desarrollada

Se pretende que esta gramática sea capaz de reconocer un subconjunto reducido de JavaScript, a continuación se listan las expresiones a reconocer:

- Constantes cadena.
- Constantes numéricas.
- Declaración e inicialización de variables.
- Expresiones *booleanas*.
- Expresiones aritméticas.
- Acceso a métodos de clases.
- Acceso a atributos de clases.
- Definición de funciones.
- Invocaciones a función.
- Asignaciones de variables.
- Sentencias programáticas como:
    - *For*.
    - *If-else*.
    - *While*.
    - *Return*.

7.3 Comprobaciones

Siendo el principal objetivo de la gramática la detección de los errores en código que introduzca el usuario, se ha creado una serie de comprobaciones que intenta aportar la mayor cantidad de información posible para que el usuario pueda solucionarlos.

Estas comprobaciones se ejecutan secuencialmente y de forma recursiva en el árbol sintáctico tras la finalización del análisis sintáctico y antes de la generación de código. La generación de código se ejecuta siempre y cuando ninguna fase anterior haya reportado errores.

7.3.1 Identificación de variables y funciones

La primera comprobación que se detalla es la identificación de las variables y funciones.

Esta comprobación consiste en registrar todas las definiciones de funciones y variables en un objeto identificador global y comprobar cada asignación a variable o invocación a función que dicha variable o función ha sido declarada con anterioridad.

Todas estas declaraciones y usos de las mismas se registran teniendo en cuenta el ámbito de las mismas, siempre pudiendo acceder a variables de un ámbito superior pero no a variables de un ámbito inferior.

En caso de que la declaración no existiese se generaría una excepción y se notificaría el error del uso de una variable o función sin declarar al usuario.

Este mismo procedimiento se emplea con las clases globales del sistema en caso de que se intentara acceder a una clase inexistente o a una propiedad o método incorrecto.

Durante esta comprobación se enlazan las invocaciones, asignaciones de variables y accesos a variables con sus definiciones.

7.3.2 Análisis semántico

Durante el análisis semántico se realizan varias comprobaciones.

Una de ellas es la validación del tipo, en este caso se comprueba que las condiciones de los bucles e *if* sean valores *booleanos*, es decir, verdaderos o falsos.

Igualmente se valida que las expresiones que forman parte de una operación aritmética sean de un tipo operable, en este caso, ambos han de ser numéricos.

También se comprueba que en el caso de una asignación, el elemento a la izquierda de la misma sea un elemento que admita asignaciones, típicamente una variable o un acceso a atributo que pueda ser reescrito.

7.3.3 Comprobación de parámetros

Finalmente se realiza una comprobación de los parámetros pasados a la instanciación de objetos, invocación de funciones e invocación de métodos.

Se trata de comprobar que el número de parámetros recibido es exactamente igual al número de parámetros esperados.

En caso de que esta comprobación no se cumpla en los casos antes descritos se produce la correspondiente excepción y el usuario es notificado.

7.4 Generación de plantillas

La generación de plantillas ocurre una vez se han superado las comprobaciones anteriores, así como el reconocimiento léxico y sintáctico.

Debido a que el código introducido se analiza en la parte *backend* de la aplicación, es necesario generar un código equivalente al introducido de forma segura para que se ejecute en *frontend*.

No obstante debido a que el uso de la palabra reservada *eval* está completamente contraindicado y puede suponer un enorme problema de seguridad, se ha optado por generar una plantilla de código para cada sentencia programática soportada en esta gramática.

Así cada vez que se genera código en realidad se está generando una serie de plantillas *JSON* que son interpretadas posteriormente en *frontend* usando código creado anteriormente que emplea los parámetros introducidos por el usuario dentro de las sentencias programáticas originales, en este caso, de JavaScript.

Estas plantillas se generan mediante un objeto *singleton* encargado de la generación de código. Durante esta fase, se ejecuta un método de generación de código de igual forma que las comprobaciones anteriores, en él, cada nodo del árbol sintáctico genera su propia plantilla y la inserta en la lista general del *singleton*.

A continuación se muestra un ejemplo de la plantilla creada en el generador *backend* en formato *JSON* a partir del código siguiente:

```
while (1<3){}
```

Código fuente 2 Ejemplo gramática: Código introducido

```
{
  "value": {
    "sentences": [],
    "condition": {
      "value": {
        "left": {
          "value": {
            "value": 1
          },
          "type": "cte"
        },
        "operator": "<",
        "right": {
          "value": {
            "value": 3
          },
          "type": "cte"
        }
      },
      "type": "logic"
    }
  },
  "type": "while"
}
```

Código fuente 3 Ejemplo gramática: Plantilla generada

Como puede verse, una sencilla instrucción de código que no realiza prácticamente acción alguna genera un total de 4 plantillas:

- 2 plantillas de constante para el número 1 y el número 3.
- 1 plantilla de operación lógica conteniendo las anteriores y el operador "<".
- 1 plantilla para el bucle *while* conteniendo la operación lógica como condición.

Por supuesto, de haber incluido alguna sentencia en el bucle, estas se habrían almacenado en el apartado *sentences* de la plantilla del bucle.

7.5 Ejecución de plantillas

Una vez las plantillas se han recibido en la parte *frontend* de la aplicación, estas se emplean para generar una serie de objetos de tipo *plantilla* en JavaScript. Estos objetos no son más que la sentencia programática primitiva empleando los campos introducidos por el usuario y analizados en la gramática.

Siendo así y siguiendo con el ejemplo anterior, se generarían cuatro plantillas:

- Dos plantillas constante, que al ser ejecutadas únicamente retornan su valor.
- Una plantilla lógica, que al ser ejecutada retornará el resultado de la operación "<" entre los operandos izquierdo y derecho.
- Una plantilla *while*, que realizará la comprobación de su condición y ejecutará todas aquellas sentencias que tenga en su lista de sentencias mientras su condición se mantenga verdadera.

Capítulo 8. Diseño del Sistema

8.1 Arquitectura del Sistema

8.1.1 Diagramas de Componentes

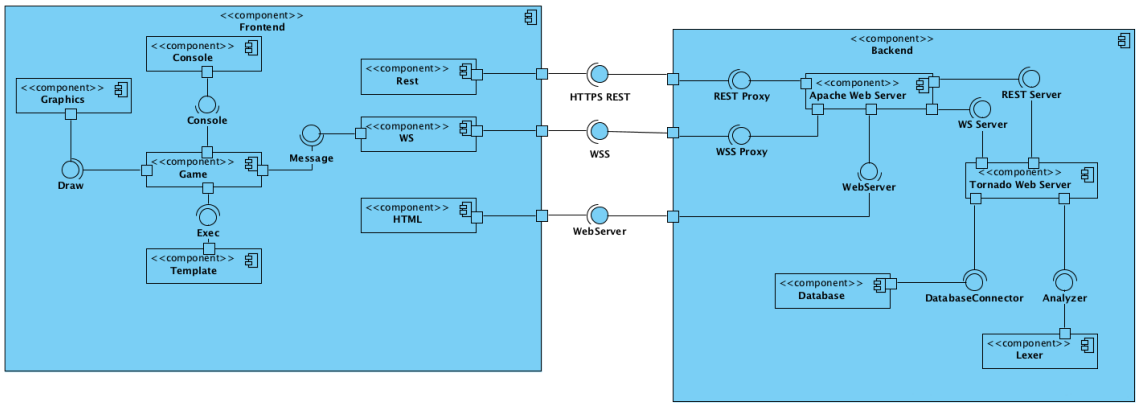


Figura 8.1 Diagrama de componentes del sistema

8.1.2 Diagramas de Despliegue

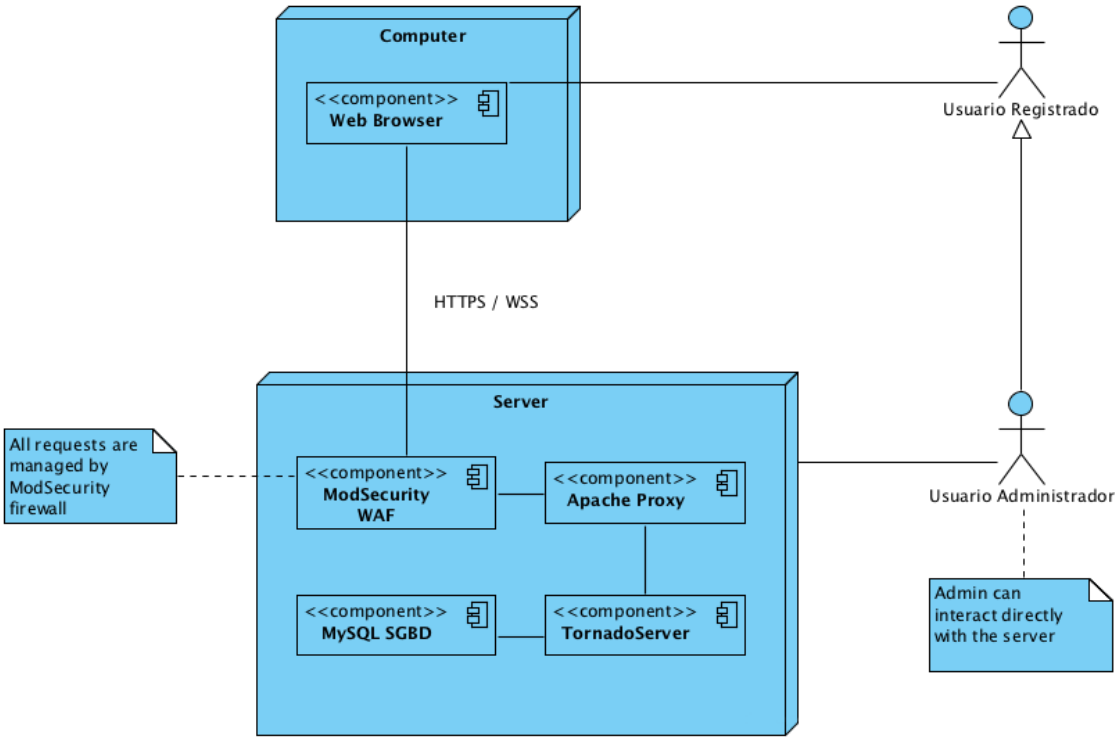


Figura 8.2 Diagrama de despliegue del sistema

8.1.2.1 Computer

Para realizar la conexión con el servidor se requiere un navegador grafico moderno instalado en el ordenador del usuario. Igualmente para poder realizar esta conexión se requiere una conexión activa a internet desde dicho ordenador.

8.1.2.2 Server

El servidor de la aplicación, se encargara de servir y ejecutar el proyecto desarrollado al recibir las peticiones del usuario. Este servidor deberá contar con los siguientes elementos para el correcto funcionamiento seguro de la aplicación:

- Conexión a internet de alta velocidad.
- *Firewall* instalado en el sistema operativo.
- Base de datos MySQL.
- Python versión 3.4 o superior.
- Servidor Web Apache 2.
- Firewall de aplicación de Apache *ModSecurity*.
- Servidor web *Python* Tornado.
- Conector con la base de datos MySQL desde Python.

8.2 Diseño de la Interfaz

A continuación se muestran los distintos diseños de la interfaz de usuario final y se explican los cambios llevados a cabo.

8.2.1 Página principal no autenticado

Tras acceder al sitio web se recibe una página principal como la que se muestra a continuación:

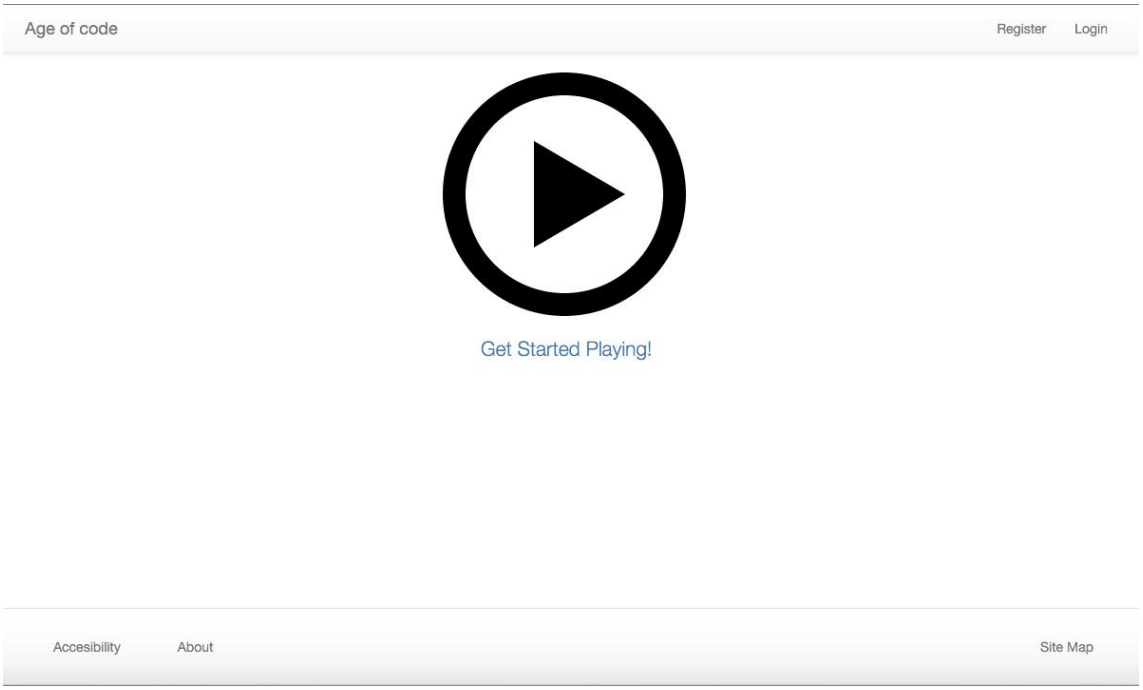


Figura 8.3 Página principal no autenticada

Ha sufrido algunas modificaciones, ocultando el enlace al perfil si no se está autenticado y añadiendo todos los enlaces de la página inferior a páginas de utilidades.

8.2.2 Página principal autenticada

La página principal se encuentra modificada comparada con la página vista antes de la siguiente forma:

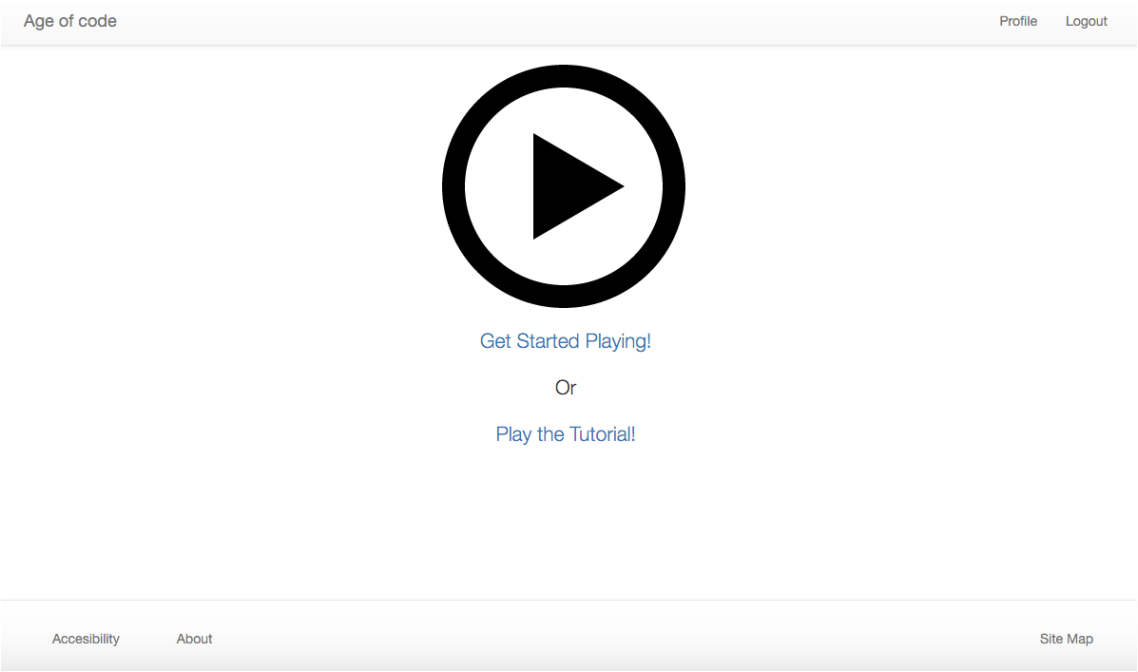


Figura 8.4 Página principal autenticada

Como puede verse en la parte superior de la página se ha incluido los botones "*Profile*" y "*Logout*", así mismo el cuerpo de la página ahora redirige a la página de juego o al tutorial del mismo.

8.2.3 Página de registro

La página de registro permite la creación de un nuevo usuario:

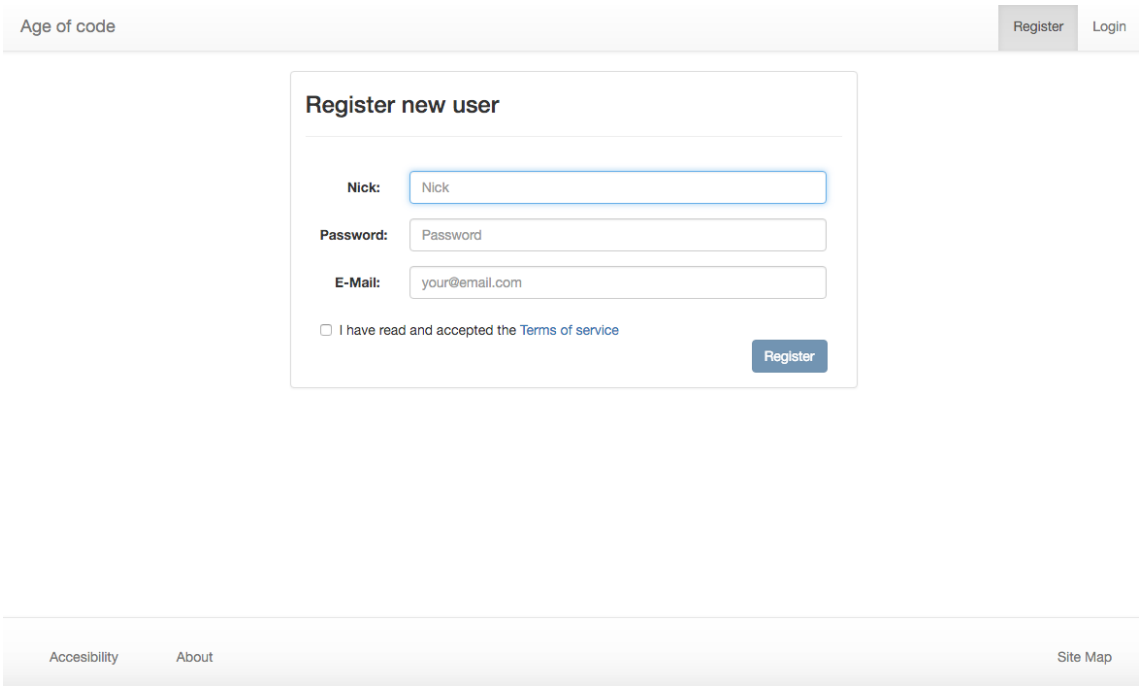


Figura 8.5 Página de registro

Esta página se ha modificado incluyendo las barras superior e inferior del sitio y cambiando el cuadro de los términos del servicio por un enlace a los mismos.

8.2.4 Página de perfil

En la página de perfil se pueden visualizar y editar algunos datos del usuario:

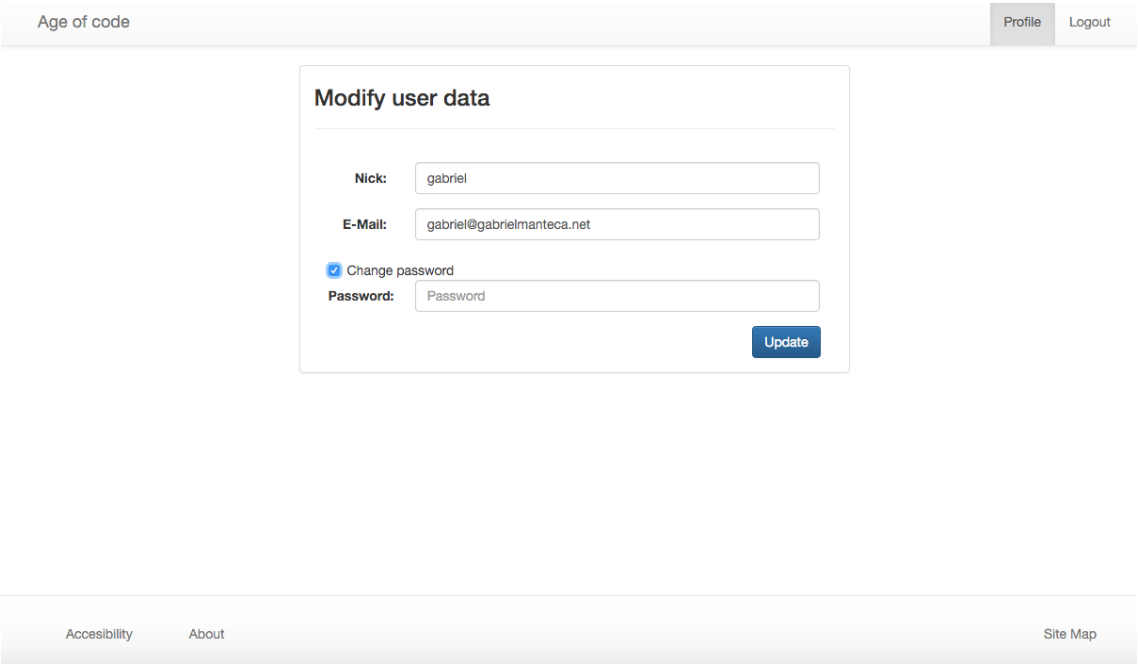


Figura 8.6 Página del perfil

8.2.5 Página de juego

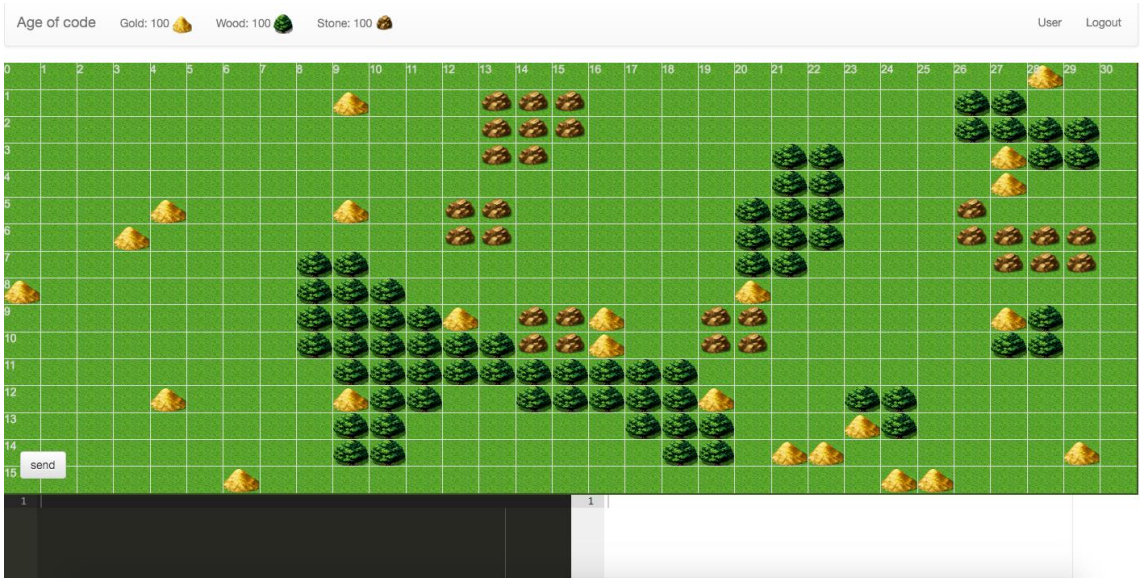


Figura 8.7 Página de juego

La página de juego se ha visto ligeramente modificada añadiendo un botón de *logout* y mostrando los recursos en la parte superior, igualmente se ocupa todo el ancho de la pantalla y se ha modificado el *dropdown* original por un *drawer* en la parte derecha de la pantalla tras pulsar el enlace *User*.

8.2.6 Página de tutorial

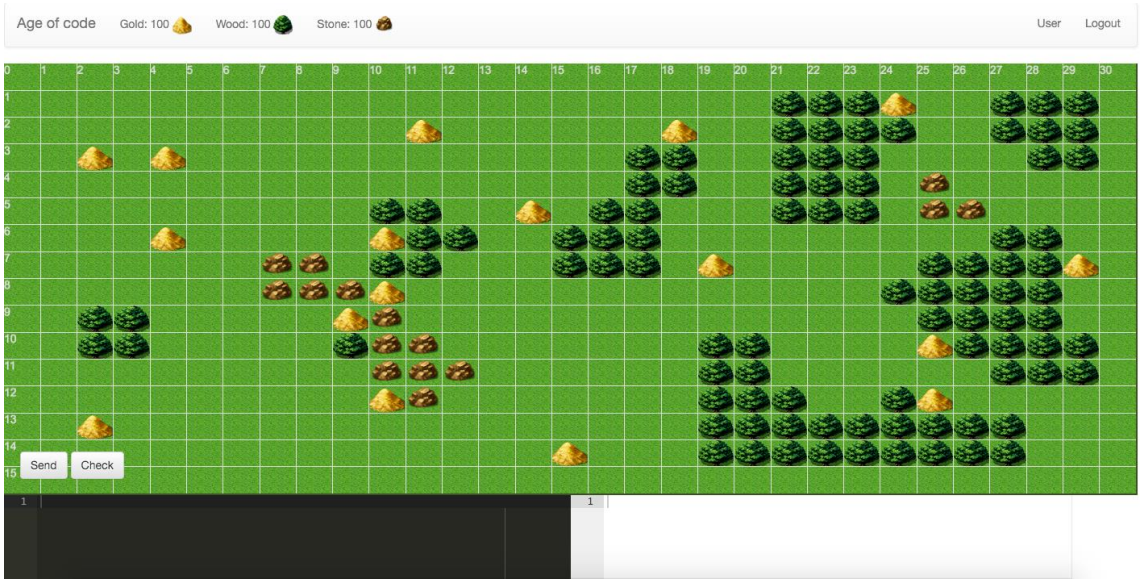


Figura 8.8 Página de tutorial

La página de tutorial no se encontraba especificada pues se pensaba reutilizar la página de juego, no obstante se optó al final por su creación a parte incluyendo los elementos distintivos de la misma como un botón de comprobación de la fase de tutorial.

8.2.7 Página de login

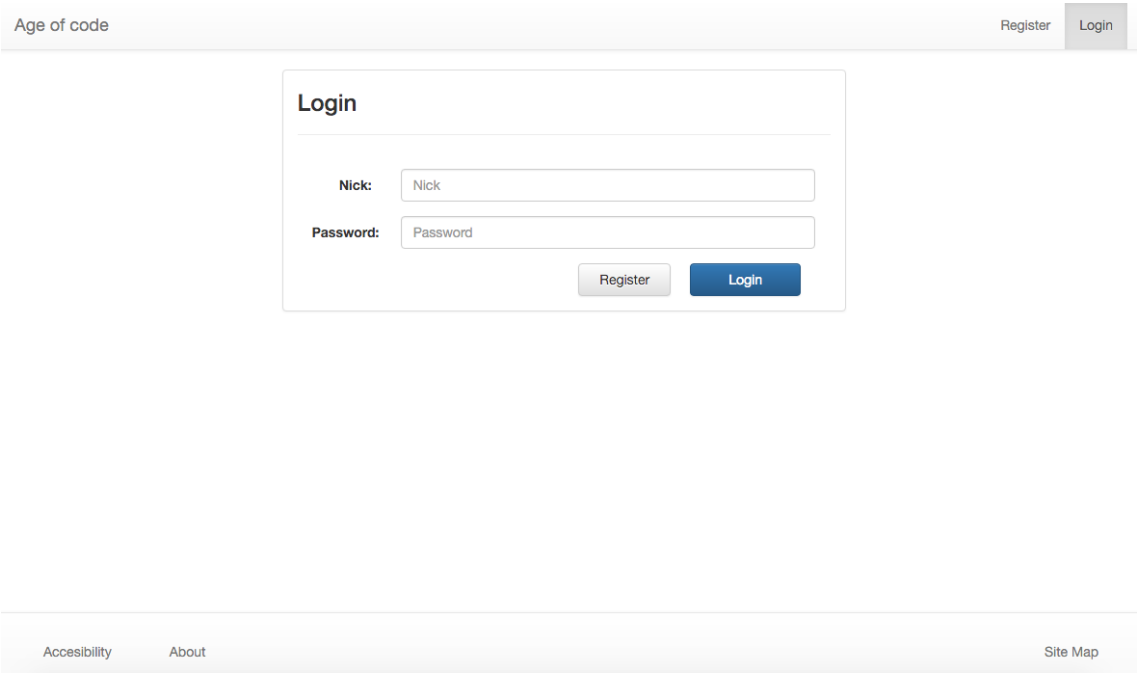


Figura 8.9 Página de login

Se ha modificado la página de *login* con respecto al diseño original por mantener la estética general del sitio, esto ha implicado mantener las barras superior e inferior del diseño y eliminar el botón de volver por la marca Brand del sitio.

8.3 Especificación Técnica del Plan de Pruebas

A continuación se describe el ordenador cliente desde el que se han realizado las pruebas y los datos del servidor donde se está ejecutando el proyecto.

Figura 8.10 Especificaciones del equipo de pruebas Windows

Las pruebas se realizarán en su mayor parte desde un equipo Surface Pro 2 con Windows 10 y 8 Gb de RAM y un equipo Mac Book Pro ejecutando OS X El Capitán con 8 Gb de RAM.

Figura 8.11 Especificaciones del equipo de pruebas OS X

El servidor que ejecuta el proyecto se encuentra alojado en un *DataCenter* de la empresa Kimsufi. El servidor dispone de un procesador Intel Atom N2800 de 4 núcleos y 4 Gb de RAM. El equipo se encuentra conectado a una línea simétrica de alta velocidad.

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    2
Core(s) per socket:    2
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 54
Stepping:              1
CPU MHz:               798.000
BogoMIPS:              3733.33
L1d cache:             24K
L1i cache:             32K
L2 cache:              512K
NUMA node0 CPU(s):     0-3
```

Figura 8.12 Especificaciones CPU

```
             total       used       free     shared    buffers     cached
Mem:          3936        646       3289          5         51        403
-/+ buffers/cache:        191       3744
Swap:         2046          0       2046
```

Figura 8.13 Especificaciones Memoria

8.3.1 Pruebas Unitarias

Las pruebas unitarias se realizarán en su mayor parte en la gramática del *backend* dado que es la parte más fácilmente testeable y la que puede ocasionar mayores problemas. Con éste propósito se han de diseñar una serie de pruebas que testeen de forma exhaustiva sus comprobaciones para evitar falsos positivos y errores en la medida de lo posible.

Estas pruebas se realizaran mediante un *framework* de pruebas para tal efecto en el servidor y deberán ser pasadas tras la finalización de cada módulo, en caso de que alguna prueba no sea superada, se deberá solucionar y hacer que dicha prueba sea superada junto a todas las demás antes de continuar hasta el siguiente módulo.

8.3.2 Pruebas de Integración y del Sistema

Al igual que las pruebas unitarias, se realizarán estas pruebas al finalizar cada módulo y no se continuará el desarrollo hasta que el resultado de las mismas sea el esperado.

Tras las pruebas unitarias se realizarán estas pruebas, antes de poder continuar hasta el siguiente módulo han de ser superadas. Si estas pruebas no son superadas, se otorgará prioridad a retrasar el módulo en la medida de lo posible antes que continuar con el desarrollo sin haber superado alguna prueba.

8.3.3 Pruebas de Accesibilidad

Se establece el mínimo cumplimiento de una accesibilidad de WCAG 2 de nivel AA. Este nivel será validado mediante validadores a tal efecto como *TAW* y *AChecker* y comprobaciones de tipo manual.

Se espera cumplir estas normas en la medida de lo posible siempre que no interfiera con los *frameworks* empleados y el objetivo principal del sitio.

8.3.4 Pruebas de Usabilidad

Se han detectado 3 tipos de usuarios potenciales a la hora de realizar estas pruebas:

- **Usuarios Novatos**: Con pocos conocimientos de informática y tecnología en general.
- **Usuarios Avanzados**: Con un conocimiento de informática a nivel usuario y capaz de desenvolverse en nuevos entornos con relativa facilidad.
- **Usuarios Expertos**: Capaz de desenvolverse en nuevos entornos tecnológicos con una facilidad notable. No es necesario indicarle pasos y no requiere ayuda externa.

Se usará una muestra de cada uno de los estratos anteriores y se les explicará mínimamente el objetivo del juego así como lo que se espera que realicen. Tras esto se procederá a una observación de las pruebas y a rellenar los cuestionarios a continuación.

8.3.4.1 Cuestionarios

Se presentan los siguientes cuestionarios para evaluar a los usuarios:

8.3.4.1.1 Preguntas de carácter general

A continuación se muestra el cuestionario que clasificará al usuario en uno de los tipos anteriores:

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez un juego como el de esta prueba?
1. Sí, he usado juegos similares 2. No, aunque tengo experiencia previa con videojuegos 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Tiene Vd. Alguna experiencia previa con videojuegos?
1. Si 2. No
En caso afirmativo, ¿Cuánta experiencia diría tener?
1. Mucha. Juego bastante tiempo prácticamente a diario. 2. Amplia. Juego cada semana algunas horas. 3. Esporádica. Juego semanalmente de forma esporádica. 4. Poca. Juego muy de vez en cuando. 5. Muy poca. Alguna vez he jugado a videojuegos.

8.3.4.1.2 Actividades guiadas

A continuación se detallan las actividades guiadas que se desarrollaran con los usuarios de prueba. En la tabla se indica la actividad, el tiempo esperado, la dificultad estimada y finalmente si se obtuvo éxito en las mismas:

Actividad	**Tiempo**	**Dificultad**	**Éxito**
Registrar usuario	2 minutos	Media	
Iniciar sesión	1 minuto	Baja	
Modificar datos	2 minutos	Media	
Ver el mapa del sitio	1 minuto	Baja	
Iniciar el tutorial	1 minuto	Baja	
Completar un nivel del tutorial	3 minutos	Alta	
Iniciar el juego	1 minuto	Baja	
Interactuar con el juego	5 minutos	Alta	
Cerrar sesión	1 minuto	Baja	

8.3.4.1.3 Preguntas sobre la aplicación

Se propone el siguiente y sencillo cuestionario para evaluar al usuario sobre la usabilidad:

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
Observaciones				

8.3.4.1.4 Cuestionario para el responsable

A continuación se presenta el cuestionario para el responsable de las pruebas que deberá completar tras cada prueba con los usuarios.

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	
Errores graves cometidos	
Comentarios del usuario	
El usuario entiende lo que hace	

Capítulo 9. Implementación del Sistema

9.1 Estándares y Normas Seguidos

9.1.1 Estándar W3C

Figura 9.1 Logo W3C

El proyecto se desarrolló siguiendo los estándares del W3C (World Wide Web Consortium) en cuanto a HTML5. Para asegurar que estos estándares se cumplen se han empleado los distintos validadores ofertados por este mismo organismo, obteniendo su aprobado en ellos.

Igualmente se ha mantenido un nivel de accesibilidad suficiente en la página principal del sitio acorde a las normas WCAG 2.0, no obstante, como se ha indicado en la propia página de accesibilidad del sitio, ha resultado imposible alcanzar este nivel en la página de juego ya que interferiría con el propósito de la aplicación en sí.

9.1.2 ECMAScript 5

Todo el código JavaScript generado mediante TypeScript ha sido compilado al estándar ECMAScript 5.

ECMAScript es un estándar general basado en la implementación de JavaScript propuesta por NetScape. El desarrollo comenzó en 1996.

Todos los navegadores web modernos, entre ellos Google Chrome y Mozilla Firefox incluyen una implementación de este estándar.

Se eligió la versión 5 de este estándar debido a que es la última versión que el compilador de TypeScript soporta de manera oficial, dado que la versión 6 y actual de ECMAScript se encuentra aún en fase experimental.

9.2 Lenguajes de Programación

Se han empleado distintos lenguajes y tecnologías en las distintas partes del desarrollo de este proyecto. Estos se explican a continuación.

9.2.1 HTML5

Figura 9.2 Logo HTML5

Hyper Text Markup Language en su quinta revisión se publicó en octubre de 2014 tras varios años de desarrollo por parte de WHATWG y el equipo de W3C HTML WG. Dentro de esta revisión, entre otras muchas cosas, se introdujeron nuevos elementos en la especificación general de HTML en la forma de nuevas etiquetas y también en la forma de nuevas *APIs* de desarrollo mediante el lenguaje JavaScript.

También cabe destacar en esta nueva revisión el intento de añadir semántica a la web con nuevas etiquetas estándares que describen su contenido.

9.2.1.1 Ventajas de HTML5

La mayor ventaja de HTML5 con respecto a las otras especificaciones disponibles es el gran número de APIs de programación que ofrece, así como los elementos que se usan para el desarrollo de videojuegos basados en web como *WebGL* y el *Canvas* de esta última especificación.

9.2.1.2 Desventajas

No todas las APIs se encuentran desarrolladas en todos los navegadores, igual que algunas aun solo tienen soporte parcial.

Lo mismo sucede con algunos elementos como es el caso de *WebGL* en navegadores antiguos.

9.2.1.3 Elección

Se ha elegido HTML5 por las citadas *APIs* de desarrollo y para poder usar bibliotecas que usen el *canvas* o *WebGL* para poder hacer que el juego sea fluido y dinámico.

9.2.2 CSS3

Figura 9.3 Logo CSS 3

Cascading Style Sheets en su tercera revisión es un DSL (*Domain Specific Language*) textual empleado en tecnologías web para definir los estilos y diseño de un sitio web. Los borradores de esta especificación se liberaron en 1999, no obstante debido a su modularización no se encuentra aún completamente desarrollado y se siguen añadiendo elementos a la misma.

Las hojas de estilo pueden introducirse en un documento HTML mediante ficheros externos enlazados con una etiqueta *link*, directamente entre etiquetas *style* o dentro de las etiquetas HTML usando el atributo *style*.

Esta última especificación tuvo como objetivo principal incluir gran parte de lo que los usuarios introducían en sus sitios web mediante JavaScript o con imágenes diseñadas a propósito, ejemplos de estos casos son las animaciones, transiciones y los bordes redondeados y degradados.

9.2.2.1 Ventajas de CSS3

CSS presenta varias ventajas:

- Posibilidad de separar la estructura del documento del estilo del mismo.
- Soporte a distintas resoluciones y dispositivos.
- Nuevos elementos y estilos de representación.
- Soporte a la accesibilidad.

9.2.2.2 Desventajas

No obstante CSS también presenta desventajas y limitaciones:

- Las pseudoclases no son controlables desde cliente, no se pueden bloquear.
- No toda la especificación de CSS se encuentra soportada por todos los navegadores aunque la mayor parte sí que lo está, de forma directa o utilizando los prefijos de cada navegador.
- No se puede incluir contenido de una regla en otra.

9.2.2.3 Elección

Se emplea CSS3 porque es el estándar de facto en el estilo de sitios web, además de aportar todos los estilos que son prácticamente obligatorios en el desarrollo de un sitio web.

Igualmente dado que no existe una alternativa, aparte de emplear una versión antigua del mismo, no se considera que haya otra elección posible.

9.2.3 TypeScript

TypeScript es un superconjunto de JavaScript de código abierto desarrollado por Microsoft que añade soporte a clases, interfaces y *tipado* estático junto a todas las comprobaciones que esto conlleva.

Figura 9.4 Logo TypeScript

Dicho de otra forma es un intento de llevar el *tipado* estático y la orientación a objetos a JavaScript, compilando a una versión de ECMAScript compatible, siendo posible incluso especificar la versión del mismo a la que traducir el código TypeScript.

Este lenguaje fue presentado por Microsoft en 2012 y desde entonces se mantiene actualizándose y con una comunidad que va en aumento.

9.2.3.1 Ventajas

- *Tipado* estático y comprobaciones en tiempo de compilación.
- Clases y módulos que facilitan la organización del código.
- Interfaces y herencia que facilitan el desarrollo.
- Compatibilidad con todas las aplicaciones JavaScript desarrolladas.
- Soporte para módulos externos escritos en JavaScript con comprobaciones de tipo por parte de la comunidad.
- Soporte para los IDEs de desarrollo más populares.

9.2.3.2 Desventajas

- No todos los módulos se encuentran soportados por la comunidad.
- El soporte de estos módulos en la mayor parte de las ocasiones no es oficial, igualmente no existe un repositorio oficial donde encontrar las declaraciones de los mismos.

9.2.3.3 Elección

A pesar de las limitaciones antes especificadas con los módulos señalados, TypeScript facilita la codificación y previene una gran cantidad de errores, igualmente facilita terriblemente la ampliación del código desarrollado y las buenas practicas del mismo otorgando la orientación a objetos de la que no dispone JavaScript.

9.2.4 Python 3

Figura 9.5 Logo Python

Python es un lenguaje interpretado multiplataforma de código abierto caracterizado por su orientación a objetos y su *tipado* dinámico.

El principal objetivo de Python es hacer un lenguaje que permita un código fácilmente legible y muy potente.

La primera versión de Python pública salió en 1991 y desde entonces no ha hecho más que crecer, hasta que en 2009 surgió una bifurcación en el lenguaje, naciendo la versión 3 de Python por donde continuaría la innovación del lenguaje pero manteniendo la versión 2.7 para que los módulos desarrollados anteriormente aun pudiesen ser usados.

Python cuenta con su propio repositorio de paquetes y varias herramientas que consultan dicho repositorio para facilitar la instalación de los mismos.

9.2.4.1 Ventajas

- Gran cantidad de módulos ya desarrollados.
- Gran comunidad que facilita el soporte al desarrollo.
- Facilidad de desarrollo.
- Gran cantidad de ejemplos de uso disponibles.
- Facilidad de aprendizaje.
- Multiplataforma.
- Diversidad de entornos de desarrollo.
- Integración con otras herramientas.

9.2.4.2 Desventajas

- Menos módulos en la versión 3 que en la 2.
- No existe retro compatibilidad entre ambas.
- Tipado dinámico muchas veces ocasiona errores.
- Lenguaje interpretado por lo que pueden suceder errores en tiempo de ejecución que en un lenguaje estático se detectarían en compilación.

9.2.4.3 Elección

Se ha elegido Python en su versión 3 por la facilidad que el lenguaje otorga al desarrollo haciendo que se desarrolle muy rápido y se obtengan unos tiempos de ejecución rápidos, así como la disponibilidad de una enorme cantidad de módulos que facilitan el desarrollo de cualquier proyecto. Tampoco es nada despreciable la gran cantidad de expertos que emplean el lenguaje y pueden dar soporte sobre el mismo.

9.3 Herramientas y Programas Usados para el Desarrollo

9.3.1 Servidor Web Apache 2

Figura 9.6 Logo Apache

Apache es un servidor web lanzado en 1995 que destaca por ser multiplataforma y no disponer de interfaz gráfica de administración, siendo toda su configuración obtenida a través de archivos XML.

Desde su lanzamiento en 1995, apache ha tenido una tendencia a la alza llegando a situarse como el servidor web más usado en apenas un año desde esta fecha de lanzamiento.

Apache esta mantenido por una comunidad de usuarios bajo la supervisión de la *Apache Foundation* y cuenta con una enorme cantidad de módulos para permitir que Apache ejerza otros roles además del de simple servidor web.

Apache también es bastante seguro de por sí, permitiendo configurarlo para adaptarlo a los niveles de seguridad que creamos convenientes e incluso pudiendo instalar *Mods* específicos para mejorar la seguridad.

9.3.1.1 Ventajas

- Gratuito y mantenido.
- Muy usado.
- Seguro.
- Multiplataforma.
- Gran cantidad de *Mods* disponibles.

9.3.1.2 Desventajas

- Configuración mediante archivos XML.
- Configuración inicial poco segura.
- Una vulnerabilidad grave descubierta en apache o sus módulos afecta a gran parte de los servidores web.

9.3.1.3 Elección

Se ha elegido apache para actuar de proxy inverso entre la aplicación Python desarrollada y el *frontend* de la misma, evitando tener puertos extra abiertos y añadiendo una capa de seguridad

extra mediante la instalación de *Mods* específicos para mejorar la seguridad. Igualmente se usa apache para servir las páginas estáticas del sitio y cifrar las conexiones mediante HTTPS y WSS.

9.3.2 MySQL

Figura 9.7 Logo MySQL

MySQL es un sistema de gestión de bases de datos relacionales multihilo y multiusuario. Se ofrece con dos tipos de licencia GNU GPL para todos aquellos que puedan cumplir dicha licencia y una licencia privativa que ha de ser comprada en caso de que una empresa concreta quiera incluir *MySQL* en sus productos.

MySQL es propiedad última de *Oracle Corporation* desde 2009 y cuenta con más de 6 millones de instalaciones en todo el mundo.

Ha sido desarrollado mayormente en ANSI C y se encuentra disponible de forma multiplataforma.

Esta es la base de datos más usada en la actualidad, si bien no es la más potente.

9.3.2.1 Ventajas

- SGBD más usado en la actualidad.
- Licencia GPL.
- Bajo consumo.
- Multiusuario.
- Conectores a esta base de datos desde prácticamente cualquier lenguaje.
- Soporte a SQL.

9.3.2.2 Desventajas

- El mantenimiento no lo lleva la comunidad por lo que a veces puede ralentizarse.
- Con grandes volúmenes de peticiones el rendimiento disminuye.
- No es la base de datos más avanzada del momento.

9.3.2.3 Elección

Se ha elegido *MySQL* como SGBD por su licencia, soporte, comunidad y extensión, si bien esta no es la base de datos más avanzada del momento, siendo Oracle la que ganaría en ese campo con mucha diferencia, se considera que esta base de datos cumple el propósito y el objetivo de este desarrollo con creces.

Enlazado a los argumentos anteriores, la licencia de una base de datos *Oracle* es completamente imposible de abarcar en este ámbito y la configuración de la misma supondría un tiempo excesivo para el objetivo que ha de desempeñar.

9.3.3 Atom

Figura 9.8 Logo Atom

Atom es un editor de texto creado por *GitHub* y liberado en 2014 de forma open source y multiplataforma.

Atom cuenta con soporte para *plugins* escritos en *NodeJS* y dispone de un motor de git integrado en el propio editor. Desde el primer momento este editor contó con un gran soporte para lenguajes y la comunidad lo ha ido expandiendo mediante *plugins* convirtiéndolo en uno de los editores más usados en la actualidad.

9.3.3.1 Ventajas

- Gratuito y *open source*.
- Gran cantidad de *plugins*.
- Expansión y mantenimiento.
- Muy usado.
- Multiplataforma.
- Puede emplearse como IDE.

9.3.3.2 Desventajas

- Lento en comparación con otros editores de texto plano.

9.3.3.3 Elección

Se ha elegido Atom como editor de texto para emplearlo como IDE de desarrollo *frontend* junto a sus *plugins* de TypeScript y Emmet. El disponer de estos *plugins* facilita considerablemente el desarrollo web y permite el desarrollo desde cualquier sistema operativo.

Igualmente su integración con git facilita la sincronización del repositorio usado.

9.3.4 PyCharm

Figura 9.9 Logo PyCharm

PyCharm es un IDE para desarrollar en Python integrado dentro de la plataforma de desarrollo de JetBrains. Este IDE incluye análisis de código, un depurador gráfico e integración con sistemas de control de versiones.

Este IDE detecta fallos algunos fallos de código en tiempo de codificación, ahorrando parte de las dificultades de depuración que se tienen siempre desarrollando en lenguajes interpretados.

Al estar integrado dentro de JetBrains, muchos *plugins* de *IntelliJ* son compatibles con PyCharm, haciendo que parte de las funcionalidades se encuentren desarrolladas.

9.3.4.1 Ventajas

- Detección de errores en tiempo de codificación.
- Gran depurador gráfico.
- Soporte para distintos *frameworks*.
- Soporte para *plugins* de la plataforma JetBrains.
- Licencia académica gratuita.

9.3.4.2 Desventajas

- Licencia comercial de pago.
- No detecta todos los errores en tiempo de codificación.
- No funcionan todos los *plugins* de la plataforma JetBrains.

9.3.4.3 Elección

Se ha elegido PyCharm por ser el IDE que más facilidades otorga en cuanto al desarrollo de código y proyectos Python. Además de esto el contar con una licencia académica gratuita ha facilitado que pueda emplearse sin coste extra para el desarrollo.

9.3.5 Navegadores Web

Un navegador web es una aplicación que interpreta código, típicamente recibido desde un servidor, lo interpreta y lo muestra de forma legible para el usuario, bien sea mediante la visualización de un sitio web o mediante la ejecución de código JavaScript que genere un resultado visible.

El navegador típicamente emplea el protocolo HTTP aunque últimamente se está extendiendo el uso de su variante cifrada, HTTPS.

Estas aplicaciones cumplen los estándares del W3C así como algunos soportan borradores y características experimentales, disponibles únicamente usando las características concretas del navegador, también cabe destacar que no todos los navegadores realizan las implementaciones de estos estándares de la misma forma por lo que puede resultar y resulta que un mismo sitio web se visualiza de forma distinta en distintos navegadores.

Se han empleado distintos navegadores web durante el desarrollo y pruebas del sitio web desarrollado. Se destacan los siguientes por ser los más empleados.

9.3.5.1 Google Chrome

Figura 9.10 Logo Google Chrome

Google Chrome fue realizado por Google utilizando componentes de código abierto y se encuentra disponible para su descarga gratuita en el sitio del desarrollador.

Este navegador emplea el motor de *renderizado Blink*, un *fork* del motor de código abierto *WebKit*. Este navegador también tiene una versión de código abierto llamada *Chromium*.

Desde que este navegador salió al mercado en 2008 su uso se ha ido extendiendo hasta ser el navegador más usado en la actualidad, siendo portado a Windows y Mac OS X y disponible en Linux a través de *Chromium*, incluso está siendo portado a iOS y cada vez se añaden más características del navegador completo a su versión de Android.

9.3.5.1.1 Ventajas

- Navegador más usado.
- Rapidez.
- Gratuito.
- Multiplataforma.
- Actualizaciones automáticas.
- Estupendas herramientas de desarrollo.
- Cumple casi todos los estándares.

9.3.5.1.2 Desventajas

- La versión de código abierto normalmente se encuentra un poco atrasada.

9.3.5.1.3 Elección

La elección de Chrome no es una posibilidad sino una obligación, al tener la mayor cuota de mercado de todos los demás navegadores, es obligatorio probarlo para demostrar que no existen problemas con él.

9.3.5.2 Mozilla Firefox

Figura 9.11 Logo Mozilla Firefox

Mozilla Firefox o más conocido como Firefox es el segundo navegador más usado actualmente, debido al declive de Internet Explorer.

Firefox se desarrolló por la fundación Mozilla y se lanzó al mercado en el año 2004, usa un motor de *renderizado* propio llamado *gecko*, en lugar de *WebKit* o un *fork* del mismo.

Entre las características a destacar, se trata de un navegador de código abierto y completamente multiplataforma disponible para la mayor parte de sistemas, soporta una gran cantidad de extensiones y su velocidad de *renderizado* es muy elevada. Al igual que Chrome este navegador se actualiza de forma automática evitando grandes brechas de seguridad a diferencia de otros navegadores.

9.3.5.2.1 Ventajas

- Multiplataforma.
- Código abierto.
- Gratuito.
- Extensiones.
- Actualizaciones automáticas.
- Cumple los estándares.

9.3.5.2.2 Desventajas

- No es el navegador más usado.

9.3.5.2.3 Elección

Al igual que en el caso anterior es casi obligatorio probar este navegador, más aun siendo completamente multiplataforma y teniendo una cuota de mercado elevada.

9.3.5.3 Opera

Figura 9.12 Logo Ópera

Opera es un navegador web creado por la empresa Opera Software lanzado en 1995 que utiliza el motor de renderizado *Blink*, dado que su propio motor, presto, les supuso demasiados problemas.

Este navegador es *freeware* siendo software propietario con algunos componentes de código abierto y fue el precursor de muchas características que posteriormente se introdujeron en los demás navegadores como la navegación por pestañas.

9.3.5.3.1 Ventajas

- Sigue siendo usado.
- Precursor de características.
- Cumple estándares.
- Gratuito.
- Multiplataforma.

9.3.5.3.2 Desventajas

- Al usar el mismo motor que Chrome, es inevitable que Opera siempre vaya unos pasos por detrás.

9.3.5.3.3 Elección

Al igual que en los casos anteriores se estima que es necesario realizar las pruebas con este navegador, no obstante al usar *Blink* no se esperan mayores diferencias entre Opera y Chrome.

9.3.5.4 Edge

Figura 9.13 Logo Edge

Microsoft Edge está destinado a ser el sucesor del navegador clásico de la plataforma Windows, Internet Explorer, fue liberado en 2015 por lo que es uno de los navegadores con menor cuota actualmente, no obstante se espera que esta aumente conforme pase el tiempo.

El motor de Edge, *Chakra*, se ha desarrollado de forma *open source* y se encuentra disponible para descarga en el repositorio de código de Microsoft. Edge soporta gran parte de los nuevos estándares y se espera su ampliación en un futuro cercano, con soporte para extensiones y otras características.

9.3.5.4.1 Ventajas

- Sucesor de internet explorer.
- Soporte a los nuevos estándares.

- *Open source*.

9.3.5.4.2 Desventajas

- No es multiplataforma.
- Poco tiempo en el mercado.
- Escasa base de usuarios.

9.3.5.4.3 Elección

Dado que es el sucesor de Internet Explorer y se encuentra instalado en todos los equipos Windows por defecto, se espera que su base de usuarios aumente considerablemente por lo que es importante realizar el desarrollo siendo soportado en Edge.

9.3.6 Microsoft Office

Figura 9.14 Logo Microsoft Office

Microsoft Office más conocido como Office es una suite ofimática creada y lanzada por Microsoft en 1989, disponible para los sistemas operativos Windows y Mac OS X, aunque recientemente se está empezando a migrar a dispositivos móviles como pueden ser Android, iOS y Windows Phone.

Esta suite es de naturaleza comercial y de pago, no obstante se dispone de licencia académica para su uso completo.

Si bien no se emplea toda la suite completa en el desarrollo de este proyecto sí que se emplea gran parte de ella, siendo destacables:

9.3.6.1 Microsoft Word

Figura 9.15 Logo Microsoft Word

Más conocido como Word, es el procesador de textos incluido dentro de esta suite. Siendo probablemente uno de los procesadores de textos más conocidos.

Word soporta una multitud considerable de características como la creación de índices, referencias cruzadas, formateos de texto, distintos tipos de fuentes, estilos, etc.

Durante este proyecto se empleará para crear la presente documentación.

9.3.6.2 Microsoft Excel

Figura 9.16 Logo Microsoft Excel

Microsoft Excel, más conocido como Excel, es el programa encargado de las hojas de cálculo en la suite de Microsoft Office.

Esta herramienta soporta interacción entre las distintas casillas de cada hoja para poder realizar calendarios, presupuestos, estimaciones y cálculos en general.

En este proyecto se ha empleado para la elaboración de los presupuestos y la gestión de los distintos cálculos.

9.3.6.3 Microsoft PowerPoint

Figura 9.17 Logo PowerPoint

Conocido como PowerPoint, es el encargado de la realización, visualización y representación de presentaciones. Ofrece distintos tipos de diapositivas así como plantillas de presentaciones, animaciones, elementos predefinidos e incluso transiciones entre diapositivas.

En este proyecto esta herramienta será empleada para la presentación del mismo.

9.3.6.4 Microsoft Project

Figura 9.18 Logo Project

Microsoft Project es la herramienta para la planificación de proyectos creada por Microsoft, entre sus características se destaca la planificación por distintas unidades temporales, crear y asignar recursos, creación y modificación de calendarios, asignación de costes y generación de presupuestos.

En este proyecto su uso será llevar la mayor parte de la gestión del proyecto.

9.3.6.5 Ventajas

- Una plataforma muy conocida y usada.
- Amplia experiencia de uso.
- Alta calidad.
- Licencia académica gratuita.
- Disponible en Mac OS X y Windows y siendo portado a plataformas móviles.

9.3.6.6 Desventajas

- La licencia es privada aunque disponga de licencia académica.
- No es completamente multiplataforma.

9.3.6.7 Elección

Se ha elegido esta suite por ser una de las más empleadas en ofimática en los tiempos actuales, igualmente se dispone de una buena experiencia previa en la misma y se dispone de todas las herramientas requeridas integradas en el mismo paquete.

9.3.7 Visual Paradigm

Figura 9.19 Logo Visual Paradigm

Visual Paradigm es una herramienta case para UML 2 desarrollada por el Hong Kong Institute of Vocational Education, dispone de una licencia comercial, no obstante dispone de una edición *community* gratuita.

Entre sus características se incluyen utilidades de generación de diagramas a partir de ejecutables para determinados lenguajes como Java y programas .NET y generación de código Java y C# a partir de diagramas.

Por supuesto, ofrece la posibilidad de diseñar manualmente todos los diagramas UML que puedan ser necesarios para el ciclo de vida de un proyecto.

9.3.7.1 Ventajas

- Multiplataforma.
- Versión *community* gratuita.
- Capacidad de creación de gran cantidad de diagramas.
- Exportación de diagramas a distintos formatos.
- Experiencia trabajando en el mismo.
- Características de generación de código.
- Características de generación de diagramas.

9.3.7.2 Desventajas

- Edición pro de pago.
- Edición *community* limitada con marca de agua y a proyectos no comerciales.
- Lentitud de ejecución.

9.3.7.3 Elección

Se ha elegido esta herramienta por la familiaridad previa que se tiene con la misma y por cumplir todos los requisitos necesarios para desarrollar los diagramas UML del proyecto. Igualmente se disponía una licencia académica para su uso.

9.3.8 Sphinx

Figura 9.20 Logo Sphinx

Sphinx es una herramienta empleada para generar documentación de código Python a partir de la documentación integrada en los distintos ficheros de un proyecto *Python*. Sphinx genera documentación en distintos formatos como HTML y PDF, habiendo sido el primero el elegido por la facilidad y cantidad de información que provee.

Cuenta con soporte a extensiones por parte de la comunidad de usuarios y una facilidad de uso considerable.

9.3.8.1 Ventajas

- Facilidad de uso.
- Gratuita.
- Extensiones.
- Comunidad.

9.3.8.2 Desventajas

- Difícil crear formatos propios de documentación.

9.3.8.3 Elección

Se ha elegido este sistema de documentación por la facilidad de uso que otorgaba, así como la buena calidad de la documentación generada por el mismo y las extensiones provistas por la comunidad.

9.3.9 YuiDocs

Figura 9.21 Logo YUI

YuiDocs es una herramienta escrita en *NodeJS* que genera documentación en distintos formatos a partir de documentación en formato de comentarios en código JavaScript. Lamentablemente este sistema de documentación no funciona en TypeScript por lo que se han tenido que incluir los comentarios de documentación en JavaScript para que este sistema funcione.

Igualmente este sistema no incluye *snippets* de comentarios y es necesario incluir gran parte de los mismos a través de código.

9.3.9.1 Ventajas

- Amplitud de documentación, permitiendo documentar como si fuera TypeScript.
- Muy usado.
- Gran calidad de documentación generada.
- Gran facilidad de uso.

9.3.9.2 Desventajas

- Falta de *snippets*.
- Leves fallos en la documentación generada.
- Sin soporte para TypeScript.
- Sin soporte para funciones no incluidas en clases.

9.3.9.3 Elección

Se ha elegido este sistema de documentación porque a pesar de sus desventajas, era el que más se ajustaba a lo que se necesitaba de todas las alternativas examinadas.

9.4 Frameworks de desarrollo

Durante el desarrollo de este proyecto se han usado una gran cantidad de *frameworks*, bibliotecas y utilidades que faciliten parte del desarrollo. A continuación se destacan aquellas más importantes.

9.4.1 Bootstrap

Bootstrap es un *framework frontend* creado por twitter en 2011 de forma *open source* con intención de facilitar herramientas para crear sitios web y aplicaciones.

Figura 9.22 Logo Bootstrap

Bootstrap aporta elementos HTML, CSS, tipográficos y JavaScript para facilitar el desarrollo y el diseño de distintos sitios web y desde su tercera versión se pretende centrar su diseño a móviles, haciendo que los sitios sean responsivos y adaptables.

Este *framework* esta principalmente basado en clases CSS que pueden ser aplicadas a los elementos HTML del documento, no obstante también existe la posibilidad de emplear los elementos y funciones JavaScript que otorga su complemento en este lenguaje para determinadas acciones como Modales o Alertas.

Finalmente destacar que la documentación de Bootstrap es excelente, otorgando una gran cantidad de ejemplos en los que basar un diseño y una comunidad enorme que aporta ideas propias a situaciones concretas de los desarrollos.

9.4.1.1 Ventajas

- *Open source* y gratuito.
- Fácilmente integrable en aplicaciones desarrolladas.
- Gran comunidad de desarrollo.
- Excelente documentación con ejemplos de uso concretos.
- Elementos reutilizables.

9.4.1.2 Desventajas

- Puede colisionar con hojas de estilo propias.
- Puede colisionar con *frameworks* JavaScript *frontend* como Angular JS.
- A veces puede resultar difícil modificar su comportamiento.

9.4.1.3 Elección

Bootstrap es actualmente uno de los *frameworks frontend* más empleados y se encuentra integrado en gran parte de otros *frameworks* y CMS, debido a la posible utilización de otros *frameworks* se ha considerado necesario emplear algo conocido y usado por si hubiese que buscar soluciones en la comunidad. Igualmente las características de este *framework* cumplen con creces los requisitos necesarios para el desarrollo.

9.4.2 AngularJS

Figura 9.23 Logo AngularJS

AngularJS conocido normalmente como Angular, es un *framework* web *open source* mantenido principalmente por google y una comunidad de desarrolladores que intenta facilitar el desarrollo de las aplicaciones web de una sola página soportando un enfoque Modelo-Vista-Controlador (MVC) y Modelo-Vista-VistaModelo (MVVM).

Para usar este *framework* se actualiza el HTML con etiquetas propias que posteriormente son leídas por angular y realiza enlaces de forma dinámica entre los atributos y las variables estándar de JavaScript definidas en el controlador y el *scope* de la vista.

Angular actualiza los datos de la vista de forma directa al ser cambiados en el controlador y facilita todos los desarrollos de forma dinámica.

9.4.2.1 Ventajas

- Enlaces dinámicos de datos entre controladores y vista.
- Herramientas incluidas en el *framework* para realizar peticiones http.
- Gran comunidad en expansión.
- Gran cantidad de *plugins* desarrollados.

9.4.2.2 Desventajas

- Colisión con otras bibliotecas JavaScript, como la biblioteca JavaScript de Bootstrap.
- Enormes cambios entre las distintas versiones de la biblioteca, haciendo difícil la retro compatibilidad.
- Documentación poco clara.

9.4.2.3 Elección

Se ha elegido usar Angular para facilitar el enlazado de datos entre *frontend* y *backend*, no obstante se han encontrado diversos problemas que ha sido necesario solventar durante el desarrollo, usando otras bibliotecas o consultando a la comunidad posibles soluciones.

9.4.3 PLY

PLY (*Python lex-yacc*) es una biblioteca que implementa *lex* y *yacc* en Python y otorga soporte en dicho lenguaje, en otras palabras, se trata de una biblioteca que facilita el *parseo* de texto y la definición de gramáticas para, en este caso, el reconocimiento de un lenguaje de programación.

PLY fue diseñado con un objetivo puramente académico en 2001, por lo que su gestión de errores es muy clara y facilita solucionar aquellos errores típicos que surgen durante la definición de una gramática.

9.4.3.1 Ventajas

- Fácil depuración.
- Rendimiento elevado.
- Implementación en Python.
- *Open Source* y gratuito.

9.4.3.2 Desventajas

- Poco mantenimiento.
- Pocos ejemplos.

9.4.3.3 Elección

Debido a la poca experiencia con la construcción de gramáticas de la que se dispone, se optó por emplear esta biblioteca que facilita considerablemente la depuración de errores en la gramática, además como se espera un número elevado de peticiones se requiere que el *parseo* y ejecución de las sentencias sea lo más rápido posible, siendo esta herramienta en su modo optimizado una buena solución.

9.4.4 TornadoWeb

Figura 9.24 Logo TornadoWeb

Tornado es un *framework* web *open source* diseñado en Python y completamente multiplataforma diseñado en 2009 y actualmente mantenido por Facebook.

Su principal característica es su escalabilidad y el hecho de ser un servidor web no bloqueante dirigido por un único bucle de eventos, en las comparativas puede verse que el rendimiento de tornado es considerablemente elevado, a la par con otros *frameworks*.

Además de admitir soporte para aplicaciones web, Tornado también tiene utilidades asíncronas de cliente y soporte para WebSockets, entre otras cosas.

En cuanto a tornado como servidor de páginas web, se dispone de un lenguaje de plantillas que permite insertar datos en la página antes de servirla.

9.4.4.1 Ventajas

- Gratuito y *open source*.
- Escalable.
- Mantenido.
- Soporte para REST y WebSockets.
- Gran rendimiento.
- Soporte como cliente.
- Lenguaje de plantillas.

9.4.4.2 Desventajas

- No tiene soporte multihilo en algunas circunstancias.

9.4.4.3 Elección

Dado que tornado ofrece un rendimiento considerable y soporta aquellas funcionalidades que se requieren sin necesidad de módulos externos, se ha elegido como *framework* del *backend* del servidor. Además se cuenta con experiencia previa en su uso que recortará el tiempo de aprendizaje.

9.4.5 Pixi JS

Figura 9.25 Logo Pixi JS

Pixi JS es un motor de *renderizado* 2D escrito en JavaScript y disponible de forma *open source*.

Este motor de *renderizado* se caracteriza por su rapidez y por tener soporte tanto para *canvas* en *WebGL* como para *canvas* básico en HTML5. El uso de esta biblioteca facilita terriblemente el uso de este elemento para dibujar en una aplicación web.

Si bien Pixi está considerado un motor de *renderizado* en lugar de un motor de juegos, sí que puede emplearse para desarrollar juegos en web, ya que soporta animaciones, *sprites*, interacción y muchas otras características que serían necesario implementar manualmente de no contar con ella.

9.4.5.1 Ventajas

- Gran rapidez.
- Soporte para animaciones y *sprites*.
- Gratuito y *open Source*.
- Soporte para *WebGL* y *Canvas*.
- Fácil de desarrollar en JavaScript plano.
- Buena documentación.

9.4.5.2 Desventajas

- A pesar de que la documentación es buena resulta difícil desarrollar con este motor debido a que los ejemplos son demasiado sencillos y no muestran una arquitectura completa para su uso.
- El consumo de CPU y memoria puede ser elevado en algunas circunstancias.

9.4.5.3 Elección

De entre todos los motores de *renderizado* y motores de juego probados, Pixi destacó por su velocidad y facilidad de uso inicial, cumpliendo todos los requisitos necesarios, con una comunidad activa de desarrolladores y un soporte relativamente fuerte.

9.5 Creación del Sistema

A continuación se relatan todos aquellos aspectos encontrados durante el desarrollo del proyecto.

9.5.1 Problemas Encontrados

En este apartado se relatan los problemas encontrados durante el desarrollo y las soluciones que se fueron aportando a los mismos.

9.5.1.1 ModSecurity

ModSecurity del servidor *Apache* usado para mejorar la seguridad del sistema ha sido una fuente inagotable de problemas en el desarrollo de este proyecto, a continuación se destacan los principales:

- Conflicto entre *ModDeflate* y *ModSecurity*, algunas páginas comprimidas con *ModDeflate* provocaban que *ModSecurity* encontrase caracteres típicos de código ASP.NET o JSP como <% o %> deteniendo la entrega de la página a *frontend* y sirviendo un error 403. Se solucionó este error desactivando *ModDeflate*.
- Se ha encontrado un problema con las cookies cifradas debido a que por este cifrado durante el redireccionado de páginas *ModSecurity* encontraba cadenas que podían representar vectores de ataque de *SQL injection*. Dado que el contenido de las cookies es posteriormente descifrado y jamás en ningún caso se accede al contenido sin antes parsear, se ha optado por desactivar esta regla.
- Ha sido necesario solicitar a *ModSecurity* que no filtre los verbos HTTP PUT y DELETE, dado que ambos se emplean en la API *RESTful* de gestión de usuarios.

9.5.1.2 Angular y Bootstrap

Angular y Bootstrap presentan considerables problemas si se usan las características JavaScript de Bootstrap, debido a que ambas bibliotecas colisionan, impidiendo la ejecución de la biblioteca de Bootstrap.

Este problema se ha solventado utilizando una implementación de la comunidad de esta misma biblioteca sobre angular.

9.5.1.3 YuiDocs

YuiDocs ha sido el generador de documentación elegido para realizar la documentación del *frontend* de la aplicación, no obstante se han encontrado que a veces el enlace desde el índice de una clase al correspondiente elemento no funciona.

9.5.1.4 Dibujado en Chrome

Durante un breve periodo de varios días en abril, se detectaron ciertos problemas con el dibujado de la interfaz gráfica del juego en el navegador Google Chrome, únicamente sucedía en este navegador bajo Mac OS X, siendo su visualizado correcto en Mozilla Firefox o en Google Chrome en su versión de Windows.

Actualmente dicho defecto no ha vuelto a ocurrir y se presupone que fue una versión defectuosa de esta plataforma.

9.5.2 Descripción Detallada de las Clases

La documentación de las clases y métodos desarrollados se ha dividido en dos partes, una dedicada al servidor Python de *backend* y otra dedicada al TypeScript empleado en *frontend*.

Se ruega consultar el capítulo 15.2 para más información acerca de la documentación técnica de las clases.

Capítulo 10. Desarrollo de las Pruebas

10.1 Pruebas Unitarias

Al final de cada módulo se han ido ejecutando las pruebas unitarias desarrolladas en la parte de la gramática y se han ido solucionando las distintas partes de la misma siempre que estas han dado fallos.

A continuación se muestra el resultado de la ejecución de los distintos módulos de la gramática:

```
[Telperion:age-of-code gabriel$ nosetests tests/LexerTests.py tests/SyntacticTest]
s.py tests/IdentifierTests.py tests/SemanticTests.py tests/OffsetTests.py
.....
.
----------------------------------------------------------------------
Ran 81 tests in 0.044s

OK
```

Figura 10.1 Pruebas unitarias

10.2 Pruebas de Integración y del Sistema

Además de las pruebas definidas anteriormente y superadas de forma exitosa, por lo que se omiten las tablas descritas antes, se detallan a continuación un número de pruebas realizadas como ampliación del plan de pruebas anterior así como el resultado obtenido en la ejecución de las mismas.

Prueba	**Resultado Esperado**
Crear un usuario con los datos en blanco	El sistema notifica el error
	Resultado Obtenido
	El sistema notifica el error

Prueba	**Resultado Esperado**
Introducir una secuencia de inyección SQL	El sistema notifica un error en la información introducida no siendo vulnerable
	Resultado Obtenido
	El resultado esperado

Prueba	**Resultado Esperado**
Crear una nueva unidad en el juego	El sistema crea una nueva unidad en las coordenadas especificadas con el nombre introducido.
	Resultado Obtenido
	El resultado esperado

Prueba	**Resultado Esperado**
Seleccionar y mover una unidad por el mundo	El sistema retorna la unidad indicada por el nombre especificado y la mueve dependiendo de su velocidad hasta el punto indicado
	Resultado Obtenido
	El resultado esperado

Prueba	**Resultado Esperado**
Seleccionar y mover una unidad hasta un punto inaccesible	El sistema retorna la unidad indicada por el nombre especificado y trata de calcular una ruta hasta el punto indicado, sino existe ruta se notifica el error al usuario.
	Resultado Obtenido
	El resultado esperado

Prueba	**Resultado Esperado**
Seleccionar una unidad, moverla hasta una casilla contigua a un recurso y	El sistema retorna la unidad indicada por el nombre especificado y calcula la ruta, si existe moverá al personaje hasta dicho punto, a continuación procederá a la extracción de dicho material.

solicitar la recuperación de dicho recurso.	
	Resultado Obtenido
	El resultado esperado

10.3 Pruebas de Usabilidad

10.3.1 Preguntas de carácter general

A continuación se presentan las preguntas de carácter general que respondieron los distintos usuarios que realizaron las pruebas:

10.3.1.1 Usuario 1

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez un juego como el de esta prueba?
1. Sí, he usado juegos similares **2. No, aunque tengo experiencia previa con videojuegos** 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido **4. Que tenga todas las funciones necesarias**
¿Tiene Vd. Alguna experiencia previa con videojuegos?
1. Si 2. No
En caso afirmativo, ¿Cuánta experiencia diría tener?
1. Mucha. Juego bastante tiempo prácticamente a diario. 2. Amplia. Juego cada semana algunas horas. 3. Esporádica. Juego semanalmente de forma esporádica. 4. Poca. Juego muy de vez en cuando. 5. Muy poca. Alguna vez he jugado a videojuegos.

10.3.1.2 Usuario 2

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office **4. Únicamente leo el correo y navego ocasionalmente**
¿Ha usado alguna vez un juego como el de esta prueba?
1. Sí, he usado juegos similares 2. No, aunque tengo experiencia previa con videojuegos **3. No, nunca**
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Tiene Vd. Alguna experiencia previa con videojuegos?
1. Si **2. No**
En caso afirmativo, ¿Cuánta experiencia diría tener?
1. Mucha. Juego bastante tiempo prácticamente a diario. 2. Amplia. Juego cada semana algunas horas. 3. Esporádica. Juego semanalmente de forma esporádica. 4. Poca. Juego muy de vez en cuando. 5. Muy poca. Alguna vez he jugado a videojuegos.

10.3.1.3 Usuario 3

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente **4. Nunca o casi nunca**
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión **2. Lo uso básicamente para ocio** 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez un juego como el de esta prueba?
1. Sí, he usado juegos similares **2. No, aunque tengo experiencia previa con videojuegos** 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar **2. Que sea intuitivo** 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Tiene Vd. Alguna experiencia previa con videojuegos?
1. Si 2. No
En caso afirmativo, ¿Cuánta experiencia diría tener?
1. Mucha. Juego bastante tiempo prácticamente a diario. 2. Amplia. Juego cada semana algunas horas. 3. Esporádica. Juego semanalmente de forma esporádica. 4. Poca. Juego muy de vez en cuando. 5. Muy poca. Alguna vez he jugado a videojuegos.

10.3.2 Actividades guiadas

10.3.2.1 Usuario 1

Actividad	**Tiempo**	**Dificultad**	**Éxito**
Registrar usuario	2 minutos	Media	S
Iniciar sesión	1 minuto	Baja	S
Modificar datos	2 minutos	Media	S
Ver el mapa del sitio	1 minuto	Baja	S
Iniciar el tutorial	1 minuto	Baja	S
Completar un nivel del tutorial	3 minutos	Alta	S
Iniciar el juego	1 minuto	Baja	S
Interactuar con el juego	5 minutos	Alta	S
Cerrar sesión	1 minuto	Baja	S

10.3.2.2 Usuario 2

Actividad	**Tiempo**	**Dificultad**	**Éxito**
Registrar usuario	2 minutos	Media	S
Iniciar sesión	1 minuto	Baja	S
Modificar datos	2 minutos	Media	S
Ver el mapa del sitio	1 minuto	Baja	S
Iniciar el tutorial	1 minuto	Baja	S
Completar un nivel del tutorial	3 minutos	Alta	N
Iniciar el juego	1 minuto	Baja	S
Interactuar con el juego	5 minutos	Alta	S
Cerrar sesión	1 minuto	Baja	S

10.3.2.3 Usuario 3

Actividad	**Tiempo**	**Dificultad**	**Éxito**
Registrar usuario	2 minutos	Media	S
Iniciar sesión	1 minuto	Baja	S
Modificar datos	2 minutos	Media	S
Ver el mapa del sitio	1 minuto	Baja	S
Iniciar el tutorial	1 minuto	Baja	S
Completar un nivel del tutorial	3 minutos	Alta	S
Iniciar el juego	1 minuto	Baja	S
Interactuar con el juego	5 minutos	Alta	S
Cerrar sesión	1 minuto	Baja	S

10.3.3 Preguntas sobre la aplicación

10.3.3.1 Usuario 1

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?		X		
¿Existe ayuda para las funciones en caso de que tenga dudas?			X	
¿Le resulta sencillo el uso de la aplicación?		X		
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?		X		
¿El tiempo de respuesta de la aplicación es muy grande?				X

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es		X		
Los iconos e imágenes usados son		X		
Los colores empleados son		X		
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?		X		
¿Cree que el programa está bien estructurado?		X		

Observaciones
Los enlaces de tutorial y juego no están claros. No está clara la documentación del lenguaje. No se soportan los métodos encadenados (Clase.metodo().metodo2()…). El juego es demasiado abierto. El juego no tiene animaciones.

10.3.3.2 Usuario 2

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?		X		
¿Existe ayuda para las funciones en caso de que tenga dudas?			X	
¿Le resulta sencillo el uso de la aplicación?			X	
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?			X	
¿El tiempo de respuesta de la aplicación es muy grande?				X

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son		X		
Los colores empleados son		X		
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				X
¿El diseño de las pantallas es claro y atractivo?		X		
¿Cree que el programa está bien estructurado?		X		

Observaciones
El objetivo del juego no está claro. No sé qué hacer durante el juego y el tutorial. El idioma de la página no es español.

10.3.3.3 Usuario 3

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?		X		
¿Existe ayuda para las funciones en caso de que tenga dudas?			X	
¿Le resulta sencillo el uso de la aplicación?			X	
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?			X	
¿El tiempo de respuesta de la aplicación es muy grande?				X

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es		X		
Los iconos e imágenes usados son			X	
Los colores empleados son		X		
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				X
¿El diseño de las pantallas es claro y atractivo?				X
¿Cree que el programa está bien estructurado?				X

Observaciones
No se cómo moverme por la página. A veces no sé cómo ir a los sitios. Todo lo que se hacer en el juego es crear personajes. Si me atacan no me da tiempo a reaccionar. No sé si me están atacando.

10.3.4 Cuestionario para el responsable

10.3.4.1 Usuario 1

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	Si
Tiempo en realizar cada tarea	Poco
Errores leves cometidos	Pocos
Errores graves cometidos	Pocos
Comentarios del usuario	Principalmente preguntas sobre la documentación de los objetos.
El usuario entiende lo que hace	Parece que si

10.3.4.2 Usuario 2

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	No
Tiempo en realizar cada tarea	Medio
Errores leves cometidos	Medio
Errores graves cometidos	Tantos como leves
Comentarios del usuario	El usuario no entiende la aplicación, necesita traducciones continuas, le resulta complicado realizar las tareas de programación.
El usuario entiende lo que hace	No

10.3.4.3 Usuario 3

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	No
Tiempo en realizar cada tarea	Mucho
Errores leves cometidos	Bastantes
Errores graves cometidos	Tantos como leves
Comentarios del usuario	Le falta información para jugar, preguntando por las posibilidades.
El usuario entiende lo que hace	Se le ve habituado a los juegos

10.4 Pruebas de Accesibilidad

10.4.1 Revisión Preliminar

Paso 1. Selección de un grupo de páginas representativo de la aplicación

Para pasar las pruebas de accesibilidad se han escogido todas las páginas del sitio que no hacen un uso exhaustivo de gráficos, es decir, todas aquellas menos la página del tutorial y del juego, dado que estas dos no pueden cumplir los criterios de accesibilidad de tiempo, entre otros, ya que interfiere completamente con el objetivo de las mismas.

Paso 2. Examinar las páginas usando un navegador gráfico

Se han realizado las siguientes operaciones:

- Desactivar las imágenes. El sitio seguía siendo usable dado que las imágenes ocultadas eran, o bien simple decoración que no aportaba nada al funcionamiento del sitio o bien disponían del correspondiente atributo “alt” que indicaba su propósito.
- Cambiar el tamaño del texto y comprobar que sigue siendo usable. El sitio sigue siendo usable no obstante se añade la barra de *scroll* vertical y a ciertas resoluciones resulta difícil navegar por el sitio.
- Pruebas de visualización en distintas resoluciones. Se ha comprobado que el sitio es usable hasta resoluciones de 800x600 y 600x800, por supuesto resoluciones superiores no ocasionan problemas. Por los objetivos de este proyecto, se descarta cualquier resolución menor a las indicadas.
- Pruebas en distintos tamaños de pantalla. El sitio ha sido probado en equipos de 10, 15 y 21 pulgadas, no detectando ningún tipo de problema de visualización con el mismo.
- Probar el sitio sin estilos CSS. El sitio aun es navegable y legible, no obstante se detectan ciertos campos vacíos que no deberían aparecer por el funcionamiento de la biblioteca AngularJS empleada en el desarrollo.
- Visualizar la página mediante un simulador de daltonismo. Las páginas no ocasionan dificultades de visualizado.
- Usar el teclado para navegar a través de los enlaces y controles de formularios, usando *Tab* para desplazarse. El sitio es navegable mediante teclado.

Paso 3. Examinar las páginas usando uno o varios navegadores especializados

Se han empleado los siguientes navegadores para visualizar el sitio web:

- Google Chrome 49.
- Safari 9.1.
- Mozilla Firefox 45.
- Internet Explorer 11.
- Links (modo texto) 2.8.

Habiendo detectado únicamente incongruencias en el navegador en modo texto debido a que AngularJS no se ejecuta y no puede ocultar algunos campos.

Paso 4. Utilizar herramientas automáticas de evaluación de accesibilidad

Se han empleado las herramientas automatizadas de evaluación *AChecker* y *TAW*, superando exitosamente la validación.

Paso 5. Resumen de resultados

Como conclusión al informe de accesibilidad, el TAW detectó algunas anomalías, no obstante debido al uso de Bootstrap y AngularJS estos no pueden cumplirse sin impedir el correcto funcionamiento del sitio.

10.4.2 Evaluación de Conformidad

Paso 1. Determinar el alcance de la evaluación

Para pasar las pruebas de accesibilidad se han escogido todas las páginas del sitio que no hacen un uso exhaustivo de gráficos, es decir, todas aquellas menos la página del tutorial y del juego, dado que estas dos no pueden cumplir los criterios de accesibilidad de tiempo, entre otros, ya que interfiere completamente con el objetivo de las mismas.

Paso 2. Utilizar herramientas de evaluación automática de accesibilidad

Se han empleado las siguientes herramientas:

- *AChecker*.
- *W3C Validator.*

Los navegadores empleados han sido:

- Google Chrome 49.
- Safari 9.1.
- Mozilla Firefox 45.
- Internet Explorer 11.
- Links (modo texto) 2.8.

Paso 3. Evaluar manualmente la muestra de páginas

- Desactivar las imágenes. El sitio seguía siendo usable dado que las imágenes ocultadas eran, o bien simple decoración que no aportaba nada al funcionamiento del sitio o bien disponían del correspondiente atributo “alt” que indicaba su propósito.
- Cambiar el tamaño del texto y comprobar que sigue siendo usable. El sitio sigue siendo usable no obstante se añade la barra de *scroll* vertical y a ciertas resoluciones resulta difícil navegar por el sitio.
- Pruebas de visualización en distintas resoluciones. Se ha comprobado que el sitio es usable hasta resoluciones de 800x600 y 600x800, por supuesto resoluciones superiores no ocasionan problemas. Por los objetivos de este proyecto, se descarta cualquier resolución menor a las indicadas.
- Pruebas en distintos tamaños de pantalla. El sitio ha sido probado en equipos de 10, 15 y 21 pulgadas, no detectando ningún tipo de problema de visualización con el mismo.
- Probar el sitio sin estilos CSS. El sitio aun es navegable y legible, no obstante se detectan ciertos campos vacíos que no deberían aparecer por el funcionamiento de la biblioteca AngularJS empleada en el desarrollo.
- Visualizar la página mediante un simulador de daltonismo. Las páginas no ocasionan dificultades de visualizado.
- Usar el teclado para navegar a través de los enlaces y controles de formularios, usando *Tab* para desplazarse. El sitio es navegable mediante teclado.

Paso 4. Elaborar el informe de conclusiones

El validador AChecker no ha encontrado ningún problema de accesibilidad en las páginas evaluadas.

El validador del W3C encontró ciertos problemas con los atributos *ng* forzados en el código HTML debido al uso de AngularJS. Estos atributos no se encuentran en el estándar por lo que no pasaban la validación. Se ha propuesto una solución a este problema que consistía en la sustitución de los atributos *ng* por atributos *data-ng*, soportados tanto por el estándar como por la biblioteca.

Se han detectado errores del mismo validador del W3C en las páginas de *login* y registro, debido a que emplea etiquetas que no son parte del estándar, por el uso de una biblioteca externa que aporta funcionalidades de visualización a AngularJS.

10.4.3 Checklist del WCAG 2.0

Nivel A

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
1.1.1	Contenido no textual		X		
1.2.1	Solo audio y solo vídeo (grabado)				X
1.2.2	Subtítulos (grabados)				X
1.2.3	Audiodescripción o medio alternativo (grabado)				X
1.3.1	Información y relaciones		X		
1.3.2	Secuencia significativa				X
1.3.3	Características sensoriales				X
1.4.1	Uso del color				X
1.4.2	Control del audio				X
2.1.1	Teclado	Todo el sitio esta accesible mediante tabulaciones	X		
2.1.2	Sin trampas para el foco del teclado		X		
2.2.1	Tiempo ajustable				X
2.2.2	Poner en pausa, detener, ocultar				X
2.3.1	Umbral de tres destellos o menos				X
2.4.1	Evitar bloques	No se dispone de bloques que evitar			X
2.4.2	Titulado de páginas		X		
2.4.3	Orden del foco		X		
2.4.4	Propósito de los enlaces (en contexto)		X		
3.1.1	Idioma de la página		X		
3.2.1	Al recibir el foco		X		
3.2.2	Al recibir entradas		X		

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
3.3.1	Identificación de errores		X		
3.3.2	Etiquetas o instrucciones		X		
4.1.1	Procesamiento	Menos la excepción descrita anteriormente	X		
4.1.2	Nombre, función, valor		X		

Nivel AA

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
1.2.4	Sub títulos (en directo)				X
1.2.5	Audiodescripción (grabado)				X
1.4.3	Contraste (mínimo)		X		
1.4.4	Cambio de tamaño del texto		X		
1.4.5	Imágenes de texto				X
2.4.5	Múltiples vías		X		
2.4.6	Encabezados y etiquetas		X		
2.4.7	Foco visible		X		
3.1.2	Idioma de las partes				X
3.2.3	Navegación coherente		X		
3.2.4	Identificación coherente		X		
3.3.3	Sugerencias ante errores		X		
3.3.4	Prevención de errores (legales, financieros, datos)				X

Capítulo 11. Manuales del Sistema

11.1 Manual de Instalación

A continuación se describe el proceso de instalación del servidor que alojará el sistema. Se parte de un sistema Linux *Ubuntu Server* 14.04 LTS *NetInst*, con los siguientes paquetes por defecto:

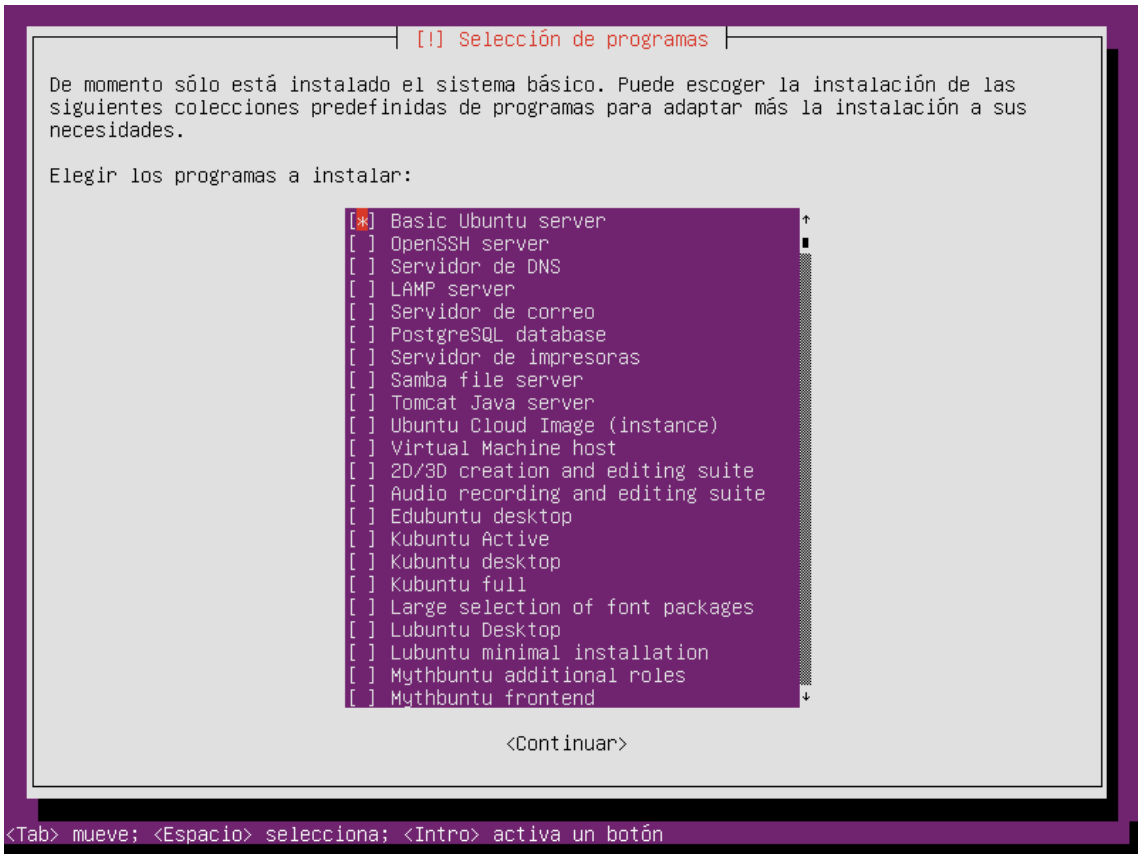


Figura 11.1 Instalación del sistema base

11.1.1 Instalación del servidor de páginas web

Las páginas web HTML estáticas se servirán desde un servidor *Apache2* estándar, instalable mediante la siguiente instrucción:

```
gmanteca@miw-tfm:~$ sudo aptitude install apache2
Se instalarán los siguiente paquetes NUEVOS:
  apache2 apache2-bin{a} apache2-data{a} libapr1{a} libaprutil1{a}
  libaprutil1-dbd-sqlite3{a} libaprutil1-ldap{a} ssl-cert{a}
0 paquetes actualizados, 8 nuevos instalados, 0 para eliminar y 0 sin actualizar
.
Necesito descargar 1.283 kB de archivos. Después de desempaquetar se usarán 5.34
8 kB.
¿Quiere continuar? [Y/n/?]
```

Figura 11.2 Instalación del servidor Apache

Tras la instalación, el servicio se encontrará automáticamente ejecutado, no obstante se requieren un número de paquetes extra del servicio para que la instalación sea correcta. El primero de ellos será instalar el cortafuegos de aplicación de apache, *ModSecurity*:

```
gmanteca@miw-tfm:~$ sudo aptitude install libapache2-modsecurity
Se instalarán los siguiente paquetes NUEVOS:
  libapache2-mod-security2{a} libapache2-modsecurity liblua5.1-0{a}
  modsecurity-crs{a}
0 paquetes actualizados, 4 nuevos instalados, 0 para eliminar y 0 sin actualizar
.
Necesito descargar 565 kB de archivos. Después de desempaquetar se usarán 4.143
kB.
```

Figura 11.3 Instalación de ModSecurity

Tras la instalación se renombra su configuración recomendada que será usada como base.

```
gmanteca@miw-tfm:~$ sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/m
odsecurity/modsecurity.conf
```

Figura 11.4 Configuración de ModSecurity

Y se activa el motor de reglas.

```
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine On


# -- Request body handling ---------------------------------------------------

# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
#
SecRequestBodyAccess On


# Enable XML request body parser.
# Initiate XML Processor in case of xml content-type
#
SecRule REQUEST_HEADERS:Content-Type "text/xml" \
     "id:'200000',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor
=XML"


# Maximum request body size we will accept for buffering. If you support
# file uploads then the value given on the first line has to be as large
# as the largest file you are willing to accept. The second value refers
"/etc/modsecurity/modsecurity.conf" 213L, 7773C escritos
```

Figura 11.5 Activación del Motor de Reglas

Tras esto se habilitan las distintas reglas a usar en el *Mod*.

```
<IfModule security2_module>
        # Default Debian dir for modsecurity's persistent data
        SecDataDir /var/cache/modsecurity

        # Include all the *.conf files in /etc/modsecurity.
        # Keeping your local configuration in that directory
        # will allow for an easy upgrade of THIS file and
        # make your life easier
        IncludeOptional /etc/modsecurity/*.conf
        Include /usr/share/modsecurity-crs/*conf
        Include /usr/share/modsecurity-crs/activated_rules/base/*.conf
        Include /usr/share/modsecurity-crs/activated_rules/optional/*.conf
</IfModule>
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"/etc/apache2/mods-enabled/security2.conf" 13L, 537C          1,1           Todo
```

Figura 11.6 Activación de reglas

Finalmente se reinicia *Apache* y el *Mod* se encontrará instalado.

A continuación se preparará el sistema de gestión de certificados HTTPS. Para ello se usará la herramienta *Let's Encrypt* del *ISRG* (*Internet Security Research Group*):

```
gmanteca@miw-tfm:~$ git clone https://github.com/letsencrypt/letsencrypt
Clonar en «letsencrypt»...
remote: Counting objects: 34166, done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 34166 (delta 11), reused 0 (delta 0), pack-reused 34142
Receiving objects: 100% (34166/34166), 9.06 MiB | 699.00 KiB/s, done.
Resolving deltas: 100% (24305/24305), done.
Checking connectivity... hecho.
gmanteca@miw-tfm:~$ cd letsencrypt/
```

Figura 11.7 Instalación de la CA de certificados

Tras la descarga del cliente se puede ejecutar automáticamente el script para que solicite el certificado y lo instale automáticamente en nuestro apache mediante la instrucción:

```
gmanteca@miw-tfm:~/letsencrypt$ ./letsencrypt-auto --apache
Bootstrapping dependencies for Debian-based OSes...
```

Figura 11.8 Instalación de certificados https

El cliente solicitará distinta información acerca del certificado y el dominio por lo que no es posible reflejarlo en este manual, no obstante es necesario que para la creación del certificado se emplee un dominio válido y ser el propietario del mismo.

A continuación se incluirán en el servidor las páginas del *frontend* y los archivos JavaScript a servir desde apache, disponibles adjuntos a esta documentación o a través de git con la siguiente instrucción:

```
gmanteca@miw-tfm:~$ git clone https://github.com/gmanteca/age-of-code.git
Clonar en «age-of-code»...
Username for 'https://github.com': gmanteca
Password for 'https://gmanteca@github.com':
remote: Counting objects: 2250, done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 2250 (delta 2), reused 0 (delta 0), pack-reused 2238
Receiving objects: 100% (2250/2250), 850.56 KiB | 267.00 KiB/s, done.
Resolving deltas: 100% (1660/1660), done.
Checking connectivity... hecho.
```

Figura 11.9 Descarga del proyecto desarrollado

Y moverlo posteriormente a la carpeta desde donde será servido:

```
gmanteca@miw-tfm:~$ sudo cp -r age-of-code/frontend/ /var/www/html/age-of-code
[sudo] password for gmanteca:
gmanteca@miw-tfm:~$ ls /var/www/html/
age-of-code/ index.html
gmanteca@miw-tfm:~$ ls /var/www/html/age-of-code/
about.html          icons/              sass/
accessibility.html  index.html          texts/
adm.html            js/                 ts/
config.rb           login.html          tsconfig.json
css/                map.html            tutorial.html
fonts/              profile.html
game.html           register.html
```

Figura 11.10 Puesta en público del proyecto

Debido a los problemas descritos anteriormente es necesario desactivar algunos *Mods* de Apache y algunas reglas de *ModSecurity*:

```
gmanteca@miw-tfm:~$ sudo a2dismod deflate
[sudo] password for gmanteca:
Module deflate disabled.
To activate the new configuration, you need to run:
  service apache2 restart
```

Figura 11.11 Deshabilitar Mods

En primer lugar desactivado *ModDeflate*.

```
  pass"


#
# Set the following policy settings here and they will be propagated to the 30 r
ules
# file (modsecurity_crs_30_http_policy.conf) by using macro expansion.
# If you run into false positves, you can adjust the settings here.
#
SecAction \
  "id:'900012', \
  phase:1, \
  t:none, \
  setvar:'tx.allowed_methods=GET HEAD POST PUT DELETE OPTIONS', \
  setvar:'tx.allowed_request_content_type=application/x-www-form-urlencoded|mult
ipart/form-data|text/xml|application/xml|application/x-amf|application/json', \
  setvar:'tx.allowed_http_versions=HTTP/0.9 HTTP/1.0 HTTP/1.1', \
  setvar:'tx.restricted_extensions=.asa/ .asax/ .ascx/ .axd/ .backup/ .bak/ .bat
/ .cdx/ .cer/ .cfg/ .cmd/ .com/ .config/ .conf/ .cs/ .csproj/ .csr/ .dat/ .db/ .
dbf/ .dll/ .dos/ .htr/ .htw/ .ida/ .idc/ .idq/ .inc/ .ini/ .key/ .licx/ .lnk/ .l
og/ .mdb/ .old/ .pass/ .pdb/ .pol/ .printer/ .pwd/ .resources/ .resx/ .sql/ .sys
/ .vb/ .vbs/ .vbproj/ .vsdisco/ .webinfo/ .xsd/ .xsx/', \
  setvar:'tx.restricted_headers=/Proxy-Connection/ /Lock-Token/ /Content-Range/
/Translate/ /via/ /if/', \
  nolog, \
  pass"


<ity-crs/modsecurity_crs_10_setup.conf" 435L, 13785C escritos
```

Figura 11.12 Habilitar métodos REST

En segundo lugar permitir los métodos HTTP PUT y DELETE.

Tras esto, deshabilitar las siguientes reglas por id en las distintas localizaciones:

```
<Location "/aocodewss">
    <IfModule security2_module>
        SecRuleRemoveById 981318
        SecRuleRemoveById 981172
        SecRuleRemoveById 981243
        #accept
        SecRuleRemoveById 960015
    </IfModule>
</Location>
```

Figura 11.13 Deshabilitar reglas en conflicto

En la dirección empleada para acceder al WebSocket mediante un proxy.

```
<Location "/aocode">
    <IfModule security2_module>
        SecRuleRemoveById 981318
        SecRuleRemoveById 981172
        SecRuleRemoveById 981243
    </IfModule>
</Location>

<Directory "/var/www/html/age-of-code">
    DirectoryIndex index.html
    <IfModule security2_module>
        SecRuleRemoveById 981318
        SecRuleRemoveById 981172
        SecRuleRemoveById 981243
    </IfModule>
</Directory>
```

Figura 11.14 Deshabilitar reglas en localizaciones

En el acceso a la api y en el acceso al propio directorio HTML.

Finalmente se configuran los *Reverse ProxyPass* para la API y el WebSocket:

```
ProxyPass /aocode http://127.0.0.1:8080
ProxyPassReverse /aocode http://127.0.0.1:8080
```

Figura 11.15 Creación de proxy para la API

```
ProxyPass /aocodewss ws://127.0.0.1:8080/wss
ProxyPassReverse /aocodewss ws://127.0.0.1:8080/wss
```

Figura 11.16 Creación de proxy para el WebSocket

Para que los proxys funcionen se habilitarán los *Mods* de *Apache*: *ModProxy*, *ModProxyHttp*, *ModProxyHtml*, *ModProxyWstunnel*.

Con esto el servidor apache se encontrará instalado y configurado para funcionar.

11.1.2 Instalación del servidor de WebSocket y REST

Se ha empleado Python en su versión 3 junto al *framework* web Tornado y PLY. Tornado estará encargado de la parte REST y WebSocket de la aplicación, mientras que PLY se encargará de la gramática y los analizadores.

Se procede a instalar *Python* y su gestor de paquetes, pip:

```
gmanteca@miw-tfm:~$ sudo aptitude install python3 python3-pip
[sudo] password for gmanteca:
Se instalarán los siguiente paquetes NUEVOS:
  binutils{a} build-essential{a} cpp{a} cpp-4.8{a} dpkg-dev{a} fakeroot{a}
  g++{a} g++-4.8{a} gcc{a} gcc-4.8{a} libalgorithm-diff-perl{a}
  libalgorithm-diff-xs-perl{a} libalgorithm-merge-perl{a} libasan0{a}
  libatomic1{a} libc-dev-bin{a} libc6-dev{a} libcloog-isl4{a}
  libdpkg-perl{a} libexpat1-dev{a} libfakeroot{a} libfile-fcntllock-perl{a}
  libgcc-4.8-dev{a} libgmp10{a} libgomp1{a} libisl10{a} libitm1{a}
  libmpc3{a} libmpfr4{a} libpython3-dev{a} libpython3.4{a}
  libpython3.4-dev{a} libquadmath0{a} libstdc++-4.8-dev{a} libtsan0{a}
  linux-libc-dev{a} make{a} manpages-dev{a} python-chardet-whl{a}
  python-colorama-whl{a} python-distlib-whl{a} python-html5lib-whl{a}
  python-pip-whl{a} python-requests-whl{a} python-setuptools-whl{a}
  python-six-whl{a} python-urllib3-whl{a} python3-chardet{a}
  python3-colorama{a} python3-dev{a} python3-distlib{a} python3-html5lib{a}
  python3-pip python3-pkg-resources{a} python3-requests{a}
  python3-setuptools{a} python3-six{a} python3-urllib3{a} python3-wheel{a}
  python3.4-dev{a}
0 paquetes actualizados, 60 nuevos instalados, 0 para eliminar y 0 sin actualiza
r.
Necesito descargar 61,1 MB de archivos. Después de desempaquetar se usarán 156 M
B.
¿Quiere continuar? [Y/n/?] _
```

Figura 11.17 Instalación de Python

A continuación se instalarán los paquetes de Tornado y PLY mediante la orden *pip3 install tornado ply*. Se requiere la instalación de un conector con la base de datos MySQL empleada, por lo que se instalará a parte el paquete *mysql-connector-python* 2.1.3, no obstante este paquete no funciona en los repositorios de pip por lo que es necesario descargarlo de la página oficial de MySQL e instalarlo siguiendo sus instrucciones.

11.1.3 Instalación de la base de datos

Se ha empleado MySQL como base de datos por lo que se requiere su instalación en el sistema:

```
gmanteca@miw-tfm:~$ sudo aptitude install mysql-client mysql-server
Se instalarán los siguiente paquetes NUEVOS:
  libaio1{a} libdbd-mysql-perl{a} libdbi-perl{a} libhtml-template-perl{a}
  libmysqlclient18{a} libterm-readkey-perl{a} libwrap0{a} mysql-client
  mysql-client-5.5{a} mysql-client-core-5.5{a} mysql-common{a} mysql-server
  mysql-server-5.5{a} mysql-server-core-5.5{a} tcpd{a}
0 paquetes actualizados, 15 nuevos instalados, 0 para eliminar y 0 sin actualiza
r.
Necesito descargar 9.410 kB de archivos. Después de desempaquetar se usarán 97,1
 MB.
```

Figura 11.18 Instalación de MySQL

Durante la instalación se introducirá la contraseña del usuario *root* que se usará a continuación para crear la base de datos:

```
gmanteca@miw-tfm:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 42
Server version: 5.5.47-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database aocode;
Query OK, 1 row affected (0.00 sec)
```

Figura 11.19 Creación de la base de datos

Tras esto se restaura la copia de la base de datos de la siguiente forma:

```
gmanteca@miw-tfm:~$ mysql -u root -p aocode < db.sql
Enter password:
```

Figura 11.20 Restauración de los datos

Una vez restaurada la base de datos, se crea el usuario y los privilegios necesarios. Por facilitar la reproducción de las instrucciones, se facilitan a continuación:

```
create user 'aocodeuser'@'localhost' identified by
'0e848aac671731710656a2c7047bc8ac15d13ad2671e0bea';
grant all on aocode.* to 'aocodeuser'@'localhost';
```

Código fuente 4 Creación de base de datos

11.2 Manual de Ejecución

Se detallan a continuación los pasos para ejecutar la aplicación tras haber seguido el manual anterior de instalación del sistema:

1. Se ejecutará la orden siguiente orden para ejecutar el servidor apache en caso de que el mismo no se encuentre en ejecución:

```
gmanteca@miw-tfm:~$ sudo service apache2 start
 * Starting web server apache2
 *
gmanteca@miw-tfm:~$
```

Figura 11.21 Ejecución de Apache

2. En la carpeta del proyecto se ejecutará la orden:

```
Python3 server.py
```

Código fuente 5 Inicio del servidor backend

Tras esto, el sitio web podrá ser accedido a través de la dirección de nuestro servidor /age-of-code. Por ejemplo, la dirección actual de ejecución: https://eru.gabrielmanteca.net/age-of-code.

11.3 Manual de Usuario

A continuación se describe el uso de las páginas y las distintas funcionalidades que presentan. Este manual ha sido realizado usando Google Chrome como navegador grafico para visitar dichos sitios.

11.3.1 Página principal no autenticado

Tras acceder al sitio web se recibe una página principal como la que se muestra a continuación:

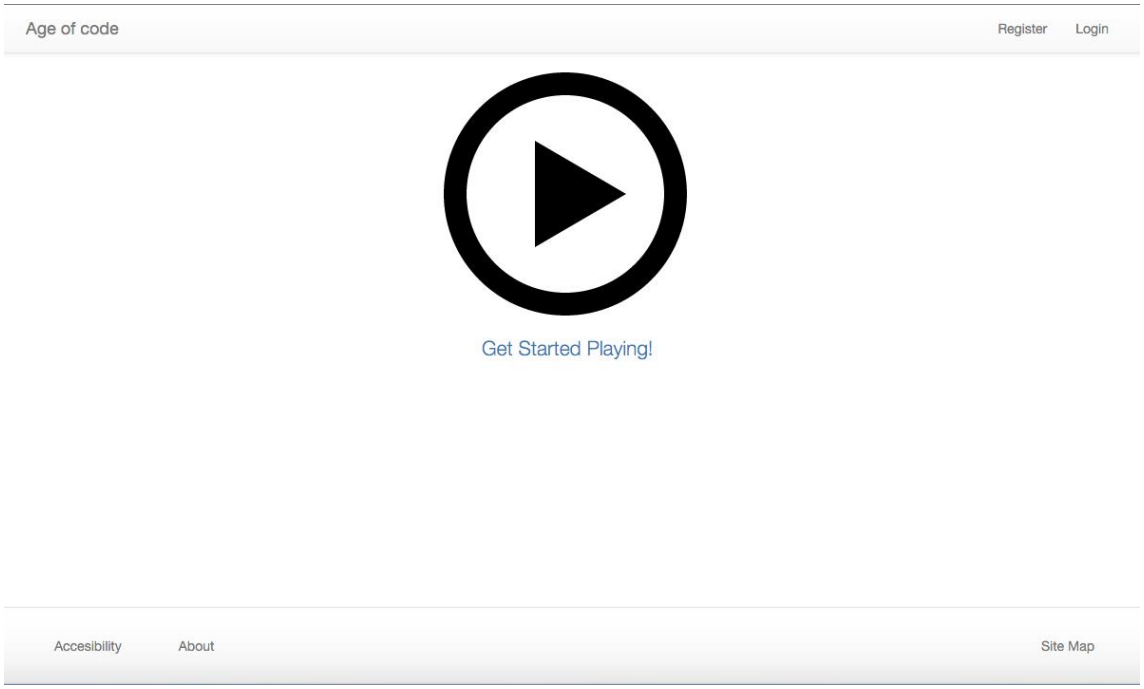


Figura 11.22 Página principal del sitio sin autenticación

El sitio web tiene 3 partes claramente diferenciadas: Una barra superior, una barra inferior y el cuerpo de la página.

En la parte superior un enlace que dirige siempre a la página principal del sitio, un enlace de registro para crear una nueva cuenta y un enlace de *login* para iniciar sesión si ya se dispone de una cuenta.

En el cuerpo se dispone de un enlace al juego si se encuentra autenticado en el sistema o a la página de inicio de sesión en caso contrario.

Finalmente en la parte inferior se muestran enlaces a distinta información sobre el sitio que puede resultar relevante.

El procedimiento natural para un nuevo usuario consistirá en crear una nueva cuenta. Para ello se pulsa sobre el enlace "*Register*".

11.3.2 Página de registro

La página de registro permite la creación de un nuevo usuario:

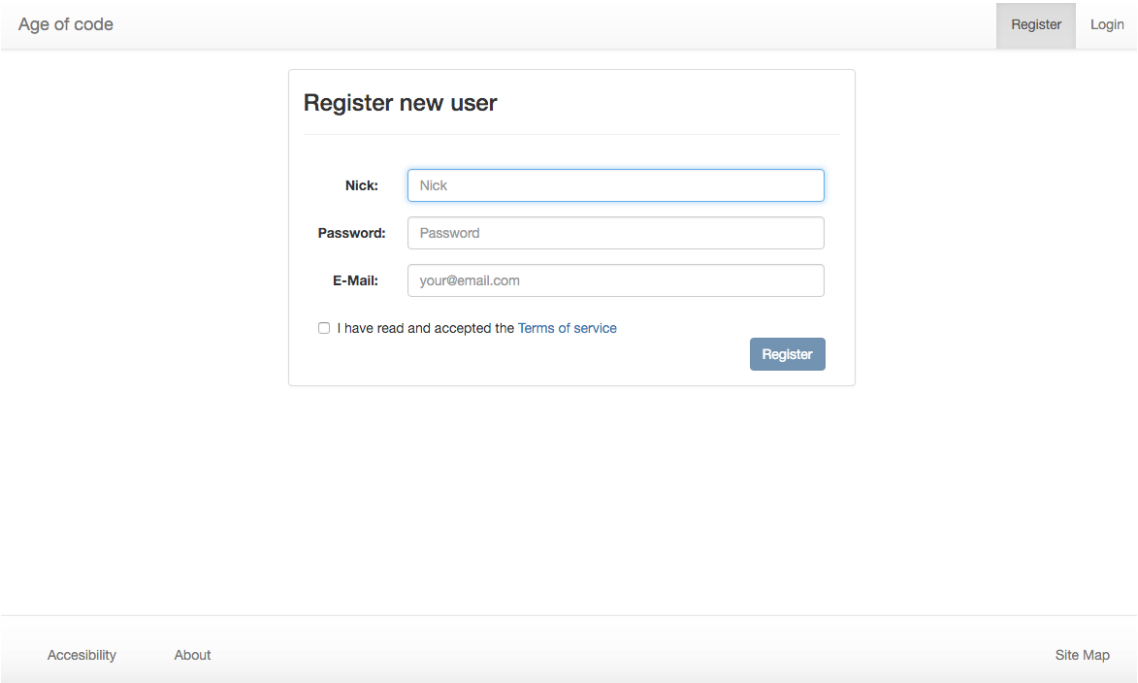


Figura 11.23 Página de registro del sitio

Un usuario nuevo debe ingresar todos los campos, un nombre único, una contraseña segura y una dirección de correo única. Tras esto debe aceptar los términos del servicio y una vez lo haya hecho podrá pasar a registrarse.

Para este ejemplo se ha creado un usuario de prueba, tras pulsar el botón de registro, se redirigirá a la página principal.

11.3.3 Página de login

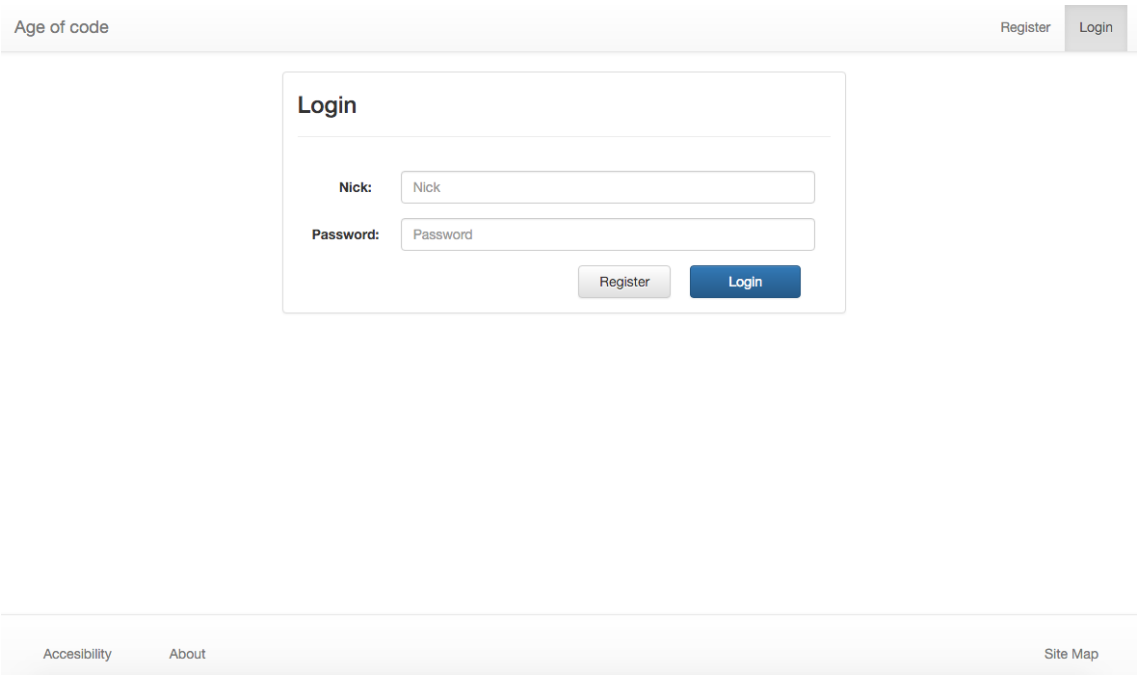


Figura 11.24 Página de login del sitio

En la página de *login* se dispone de un enlace a la página de registro por si el usuario no dispusiera de un usuario con el que iniciar sesión, igualmente se espera que se introduzca el usuario y contraseña y se pulse el botón de *login* para iniciar sesión.

Si la información es correcta se redirigirá a la página principal, en caso contrario se mostrará un mensaje de error y se permitirá reintentar el *login*.

11.3.4 Página principal autenticada

La página principal se encuentra modificada comparada con la página vista antes de la siguiente forma:

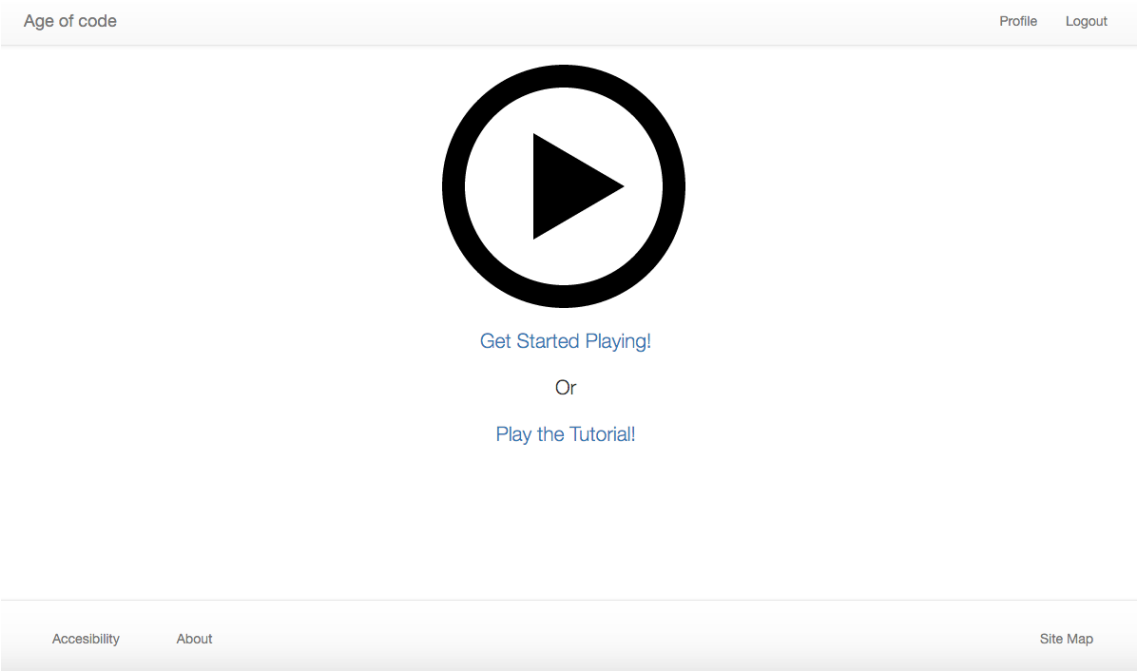


Figura 11.25 Página principal autenticada

Como puede verse en la parte superior de la página se ha incluido los botones "*Profile*" y "*Logout*", así mismo el cuerpo de la página ahora redirige a la página de juego o al tutorial del mismo.

A continuación se visualizarán los distintos cambios que pueden realizarse en el perfil.

11.3.5 Página de perfil

En la página de perfil se pueden visualizar y editar algunos datos del usuario:

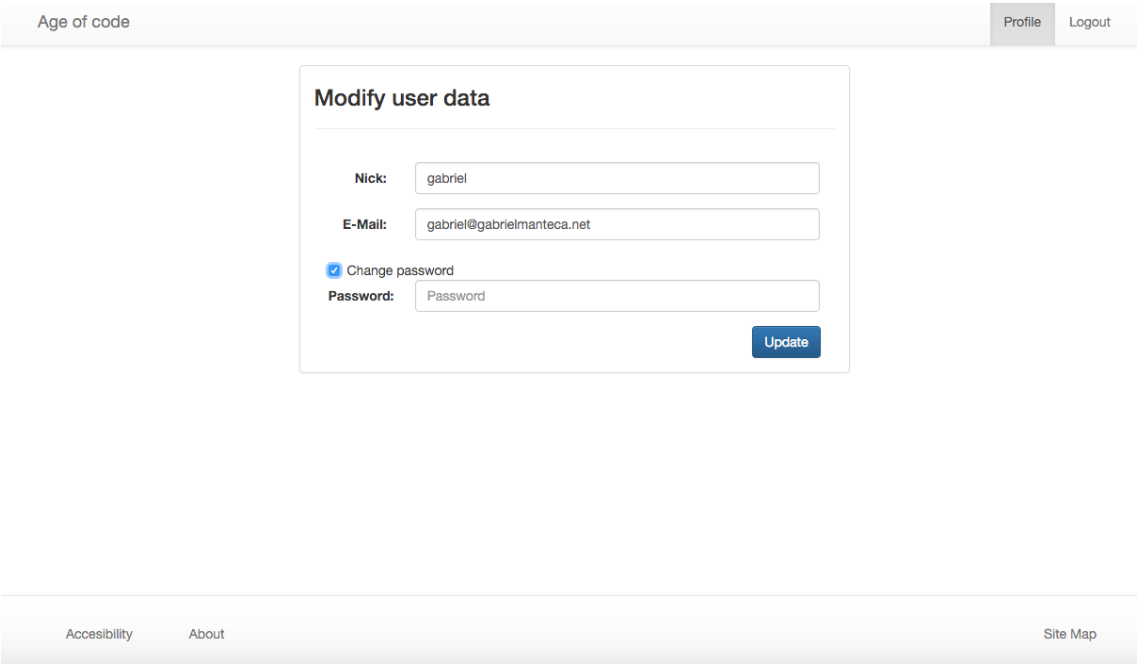


Figura 11.26 Página de perfil

Si se desea cambiar la información disponible, solo ha de rellenarse y a continuación pulsar el botón enviar.

11.3.6 Página de juego

Tras volver a la página inicial se puede pulsar sobre el enlace de juego para empezar a jugar.

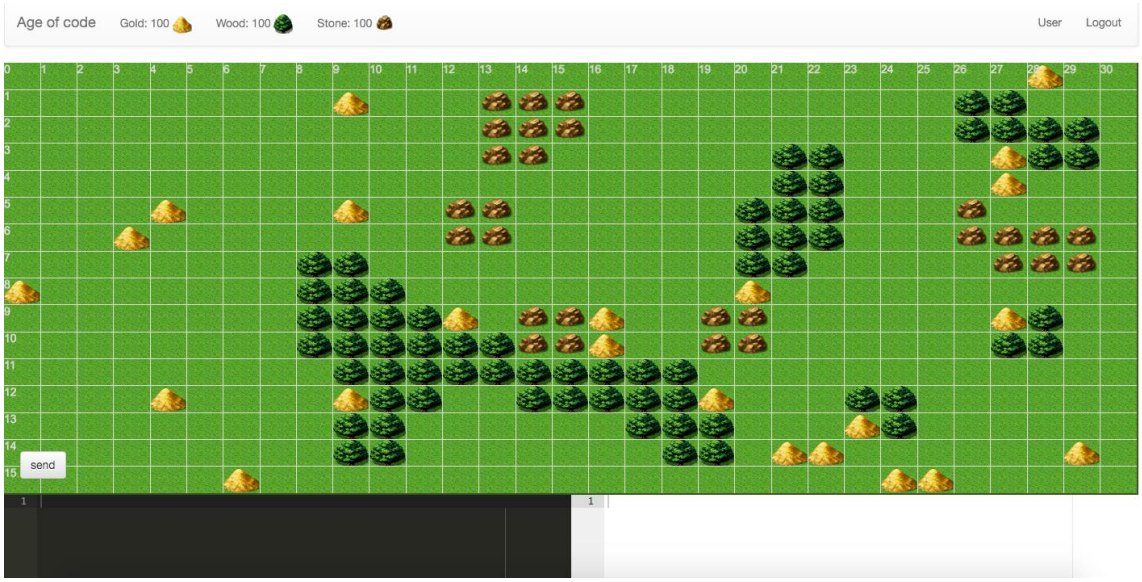


Figura 11.27 Página de juego

El mapa se mostrará de forma distinta para cada usuario ya que se genera de forma aleatoria, pero en esta pantalla se pueden diferenciar distintas partes: la primera de ellas en la parte superior indicando los recursos de los que dispone el usuario, un enlace a su propio usuario con información sobre los personajes y edificios y un botón de *logout*.

En el centro de la página puede observarse el mundo de juego representado de forma gráfica.

Finalmente en la parte inferior izquierda se muestra la consola donde el usuario introducirá el código y sobre esta el botón de enviar dicho código, a la derecha se encuentra la consola de errores del usuario.

Debido a que los elementos disponibles en el lenguaje de programación son numerosos, es considerablemente recomendable visitar el tutorial en primer lugar.

11.3.7 Página de tutorial

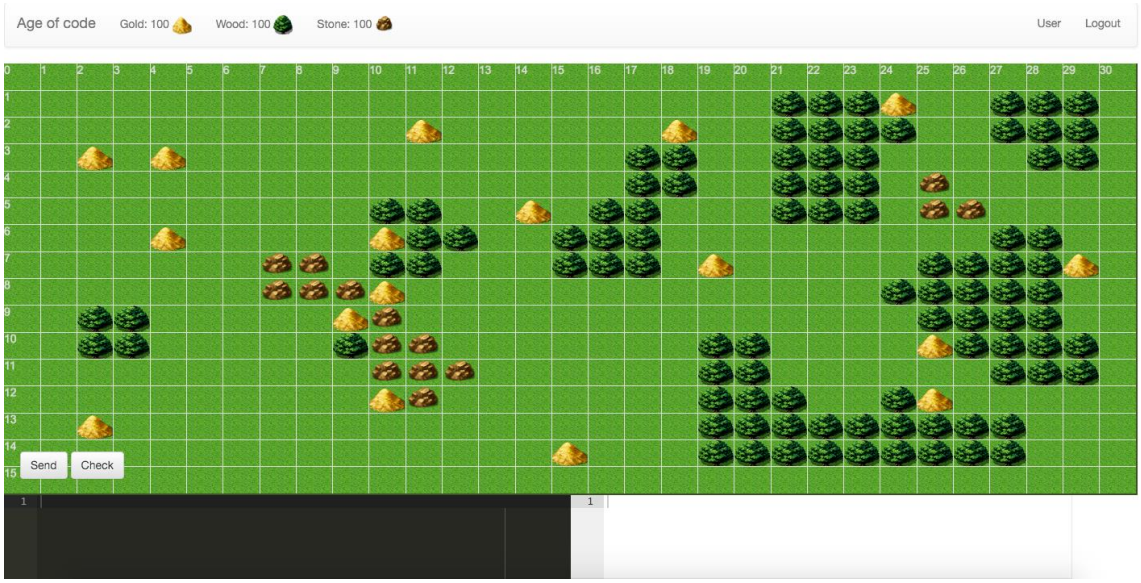


Figura 11.28 Página de tutorial

La página de tutorial es idéntica a la página de juego, no obstante cuenta con un botón a mayores para comprobar los objetivos del nivel y comprobar si estos se han cumplido. Además de esto cuenta con pantallas de ayuda que van indicando al usuario las posibilidades de la aplicación como juego.

11.4 Manual del Programador

Se distinguen distintas partes del proyecto que pueden ser ampliadas y por lo tanto susceptibles de aparecer en el siguiente manual, se detallan todas y cada una de ellas en sus distintos apartados.

11.4.1 Servicios REST de la aplicación

Se definen distintos servicios *REST* en la carpeta con nombre homónimo dentro del *backend* de la aplicación, cada punto de entrada del servicio está definido como una clase que implementa los distintos métodos a soportar. Para mayor información sobre el funcionamiento de estos puntos de entrada, se recomienda consultar la documentación oficial del *framework* web empleado, Tornado.

Además de la creación de dicha clase será necesario incluir una nueva entrada en el archivo *server.py* que refleje la existencia de dicho punto de entrada.

11.4.2 WebSocket de la aplicación

Al igual que en el punto anterior se ha definido un punto de entrada llamado *MessageHandler*, este punto de entrada recibe los mensajes recibidos por el websocket y los procesa empleando una clase auxiliar, denominada *ProtocolHelper*.

Se recomienda que si se van a hacer ampliaciones a los mensajes o al protocolo, se edite esta clase para evitar complicar la arquitectura.

11.4.3 Utilidades de backend

Se ha dispuesto un número de utilidades en el *backend* para tratar de facilitar las tareas comunes, estas clases son entre otras, la creación de mapas, la conexión con la base de datos y la gestión de la autenticación de los usuarios.

Estas clases pueden encontrarse en el módulo *tools* del proyecto y se recomienda encarecidamente su uso para en la ampliación del proyecto.

11.4.4 Analizador léxico y sintáctico

Todas las clases y objetos analizados se encuentran definidos en el módulo *lexer* del proyecto. Este analizador se encuentra dividido en un analizador léxico, un analizador sintáctico y distintas clases abstractas que conforman un árbol sintáctico abstracto (AST).

El analizador léxico definido en el archivo *lexer*, define un conjunto de expresiones regulares que define todos aquellos conjuntos de caracteres que pertenecen al lenguaje, puede ampliarse ampliando dichas reglas y añadiendo nuevas reglas de precedencia en caso de ser necesario.

El analizador sintáctico definido en el archivo *yacc*, reúne los elementos definidos anteriormente y con ellos va construyendo el árbol sintáctico definido al principio de este apartado. Está formado por métodos cuya documentación especifica la regla gramatical que ha de cumplir.

Para más información acerca de estos dos elementos, se recomienda revisar la documentación del módulo PLY usado.

Finalmente las clases del AST definidas se clasifican en varios tipos principales: Expresiones, que retornan un valor, Sentencias, que no lo hacen y pueden estar compuestas por expresiones, Definiciones del lenguaje y finalmente clases, tanto instanciables como clases del sistema.

Todos estos elementos implementan varios métodos de verificación que comprueban, si el elemento se encuentra definido, si el elemento es semánticamente correcto, si los parámetros provistos coinciden con la definición y finalmente el código que es necesario generar para el elemento del AST concreto.

Para ampliar este analizador es necesario modificar el analizador léxico y sintáctico en la medida que sea necesario y finalmente implementar las clases oportunas con los correspondientes métodos antes citados.

11.4.5 Plantillas de ejecución frontend

En *frontend* se ha creado una serie de plantillas que ejecutan el código generado en *backend* sin necesidad de utilizar funciones como *eval*. Estas plantillas tratan de ser lo más seguras posibles y se encuentran definidas en el archivo *Template*.

En caso de realizarse alguna ampliación a la hora de generar código muy posiblemente estas plantillas habrán de ser editadas también, creando una nueva o bien contemplando una modificación de alguna plantilla actual. Para editar estas plantillas sencillamente ha de implementarse la interfaz *Template* provista en el archivo descrito.

11.4.6 Ampliación de personajes

Crear un nuevo personaje en el sistema influye en *backend* y en *frontend*, por lo que se explican de forma separadas.

En *backend* es necesario crear una nueva clase que herede de la clase *GenericCharacter*, en el archivo *GlobalClasses*. En esta clase será necesario implementar sus métodos y atributos de forma idéntica a aquellos ya implementados.

En *frontend* es necesario crear una nueva clase heredando de la clase *GenericCharacter* e implementar sus diferencias, esta clase puede encontrarse en el archivo *Characters*.

11.4.7 Ampliación de Edificios

Los edificios se implementan de forma análoga a los personajes, tanto en *backend* como en *frontend*.

11.4.8 Ampliación de enemigos

En esta versión de desarrollo los enemigos se encuentran únicamente en *frontend* debido a que el juego es principalmente de un jugador, en caso de que esto varíe será necesario implementar el generador de enemigos en *backend* así como su clase correspondiente.

En este caso la implementación de los enemigos es idéntica a los edificios y personajes, solo se requiere la creación de una subclase de la clase *GenericEnemy* en el archivo de mismo nombre y una ligera edición del generador de enemigos para que contemple la posibilidad de usar un nuevo enemigo en lugar de los ya desarrollados.

Capítulo 12. Conclusiones y Ampliaciones

12.1 Conclusiones

Se ha elaborado un sistema amplio y de sencilla expansión que cumple e incluso expande las expectativas originales.

En este contexto me siento bastante orgulloso del trabajo realizado a lo largo de estos meses, permitiéndome aprender nuevas tecnologías y mejorar como desarrollador en aquellos campos en los que no tenía experiencia previa como es el caso de los videojuegos web y obteniendo experiencia en herramientas de última tecnología como es AngularJS. También me gustaría destacar el haber obtenido experiencia desarrollando una gramática y sus comprobaciones, algo de lo que no tenía gran experiencia antes de iniciar este máster.

Si bien soy consciente de que aún queda un gran camino por delante, el proyecto inicial desarrollado admite una gran expansión y su crecimiento puede ser bastante elevado. He intentado llevar a cabo todas las buenas prácticas de las que disponía de forma que esta expansión pueda llevarse a cabo con las mayores facilidades posibles.

También aprendí bastante sobre seguridad en este proyecto, campo que personalmente me interesa, intentando hacer que el proyecto sea lo más seguro posible y aplicando todas aquellas técnicas que conozco y que dimos en el máster.

12.2 Ampliaciones

Como se destacó anteriormente, este proyecto puede tener una enorme cantidad de ampliaciones que mayormente por falta de tiempo no se pudieron incluir en la versión final. A continuación se destacan algunas de ellas y sus posibles aportaciones:

- Mapa dinámico: El mapa podría ser ampliable e incluso virtualmente ilimitado gracias al generador de mapas que ya se provee en la actual versión, siendo así podría ampliarse mediante regiones del tamaño a la que ya se aporta y que se permita que los personajes y el usuario visualicen distintas regiones a petición.
- Nuevos edificios y personajes: Los personajes y los edificios se han diseñado de forma que sean fáciles de crear por lo que mejoras que ampliarían su uso seria añadir nuevos personajes o edificios.
- Modo multijugador: El juego se ha diseñado de forma que pueda funcionar de forma multijugador sin excesivas ampliaciones, de ahí el uso de un WebSocket para comunicar *backend* y *frontend* en lugar de peticiones http.
- Internacionalización: Dado que se usan páginas estáticas sería necesario usar un *plugin* o duplicar las páginas para internacionalizarla, es una opción recomendable a futuro.
- Páginas *man* del sistema: Sería recomendable incluir un manual de referencia de las funciones implementadas y las clases del sistema para facilitar su uso a los usuarios.
- Creación de una gramática en *frontend*: Para acelerar la detección de errores se podría incluir una gramática en *frontend* que evite un exceso de mensajes a *backend*.
- Incluir un sistema de niveles: De tal forma que los personajes mejoren mediante el combate y el usuario pueda invertir recursos en la mejora de los edificios.
- Inclusión de un sistema de misiones: Diariamente se darían una serie de objetivos a los usuarios a cumplir durante el juego para obtener recursos u otras recompensas.
- Sistema de alianzas: Seria posible incluir un sistema de alianzas entre usuarios una vez el juego sea multijugador.
- Sistema de logros: Permitir otorgar recompensas por determinadas acciones a los usuarios mediante logros.
- Niebla de exploración: Al iniciar el juego se muestra el mapa cubierto por niebla que se va retirando a medida que se explore.
- Sistema de civilizaciones: Se puede otorgar al usuario la opción de seleccionar una civilización al registrarse y permitir que dicha civilización vaya avanzando y mejorando tecnológicamente.

Capítulo 13. Planificación del Proyecto y Presupuesto finales

13.1 Planificación Final

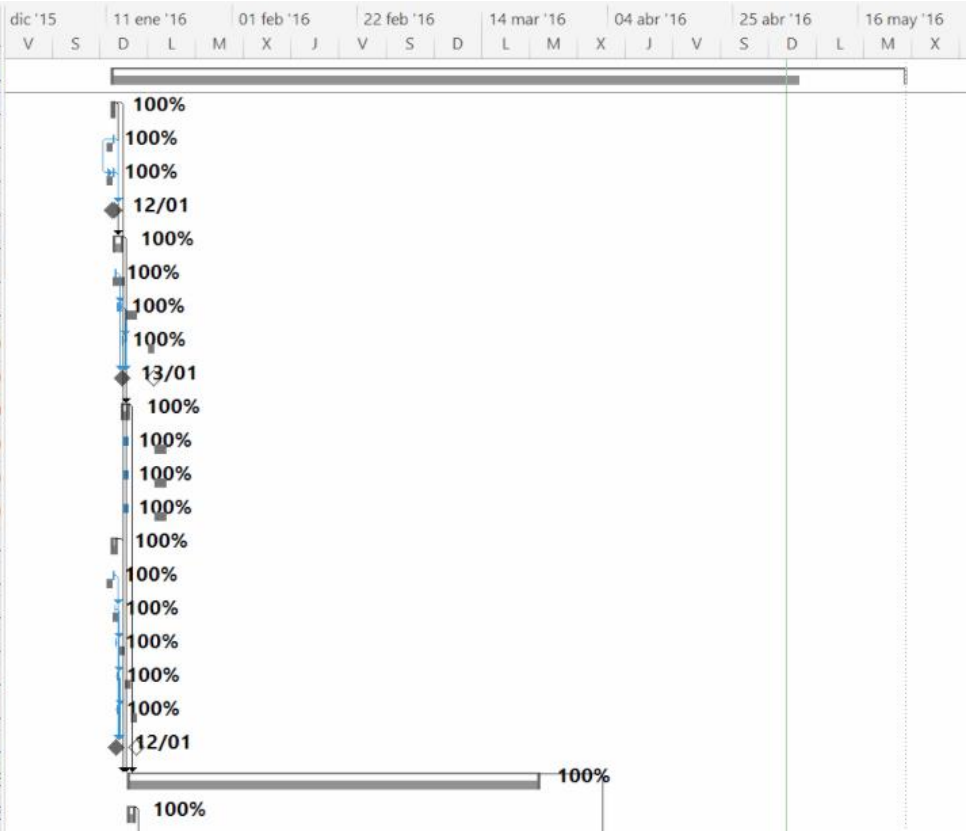
La planificación final se ha visto modificada recortando los tiempos de aquellos módulos y apartados que por facilidad de programación y conocimientos previos han resultado ser más rápidos de desarrollar de lo que se tenía planificado originalmente.

Gracias a este tiempo que se dispuso de margen, se optó por realizar una parada en el desarrollo desde el día 23 de Marzo al día 4 de Abril, previa consulta y aprobación del Director del proyecto como queda reflejada en el acta correspondiente.

Igualmente fue necesario incluir en la planificación una tarea de creación de la memoria, pues se fue documentando cada módulo tras la finalización del mismo, no obstante, no se tuvo en cuenta el tiempo que llevaría realizar la escritura de los demás apartados menos técnicos.

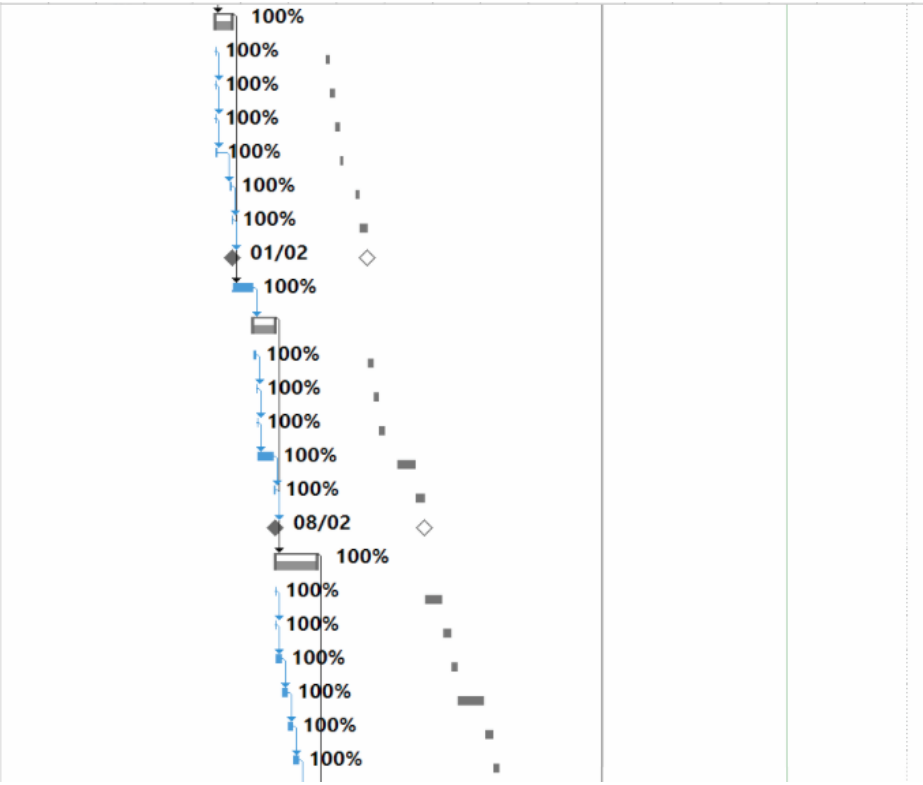
Finalmente, con el tiempo que se dispuso de margen final tras el desarrollo original se desarrollaron distintas ampliaciones sobre el desarrollo original como se refleja en la planificación.

	Nombre de tarea	Duración	% completado	Comienzo
1	**Inicio del proyecto**	**760 horas**	**100%**	**mar 12/01/1**
2	**Procesos de inicio**	**1 hora**	**100%**	**mar 12/01/1**
3	Establecer las comunicaciones entre	1 hora	100%	mar 12/01/1
4	Establecer los requisitos del proyecto	1 hora	100%	mar 12/01/1
5	Validar los requisitos del proyecto	0 horas	100%	mar 12/01/1
6	**Investigación preliminar**	**12 horas**	**100%**	**mar 12/01/1**
7	Busqueda de aplicaciones similares	6 horas	100%	mar 12/01/1
8	Búsqueda y selección de un framew	4 horas	100%	mar 12/01/1
9	Búsqueda y selección de un SGBD	2 horas	100%	mié 13/01/1
10	Documentación: Aplicaciones relacio	0 horas	100%	mié 13/01/1
11	**Planificacion inicial de riesgos**	**8 horas**	**100%**	**mié 13/01/1**
12	Tormenta de ideas para la identifica	8 horas	100%	mié 13/01/1
13	Cuantificación y priorización de ries	8 horas	100%	mié 13/01/1
14	Toma de decisiones de riesgos	8 horas	100%	mié 13/01/1
15	**Preparación del entorno de desarrollo**	**6 horas**	**100%**	**mar 12/01/1**
16	Instalación del entorno de ejecucion	1 hora	100%	mar 12/01/1
17	Instalación de IDEs y compiladores e	1 hora	100%	mar 12/01/1
18	Securización del servidor	2 horas	100%	mar 12/01/1
19	Preparacion del entorno de pruebas	1 hora	100%	mar 12/01/1
20	Preparacion del control de versiones	1 hora	100%	mar 12/01/1
21	Entorno de desarrollo	0 horas	100%	mar 12/01/1
22	**Desarrollo**	**390 horas**	**100%**	**jue 14/01/16**
23	**Construcción del servidor Backend**	**9 horas**	**100%**	**jue 14/01/16**

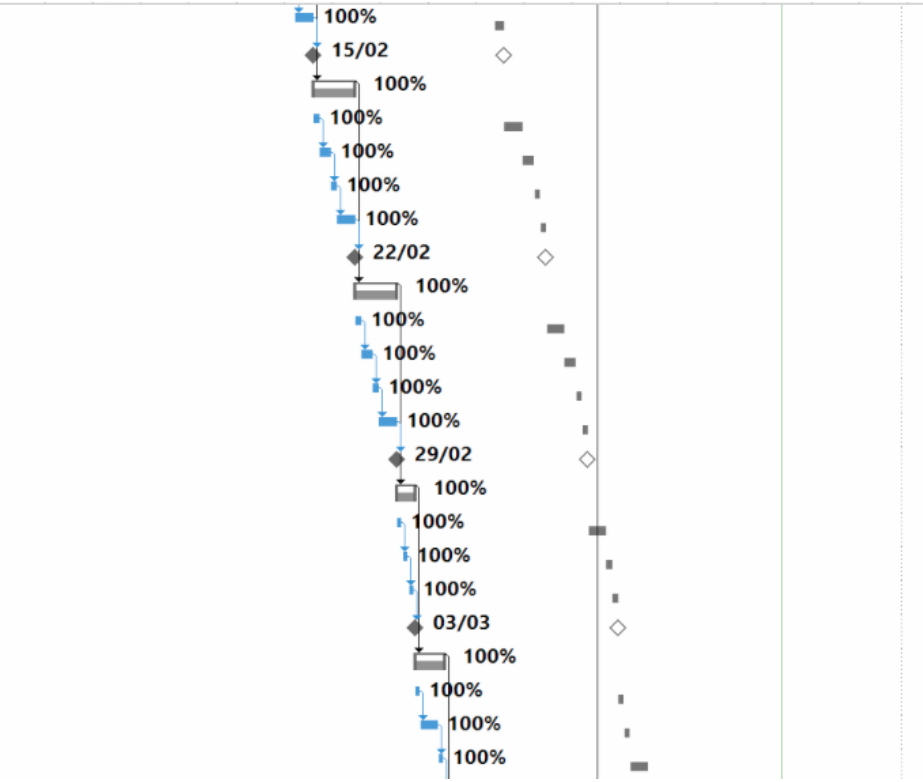


24	✓	Definición de los puntos de entrac	2 horas	100%	jue 14/01/16	
25	✓	Definicion de los puntos de entrac	2 horas	100%	jue 14/01/16	
26	✓	Creación de la base de datos de la	1 hora	100%	vie 15/01/16	
27	✓	Conexión del backend con la base	4 horas	100%	vie 15/01/16	
28	✓	Servidor Backend	0 horas	100%	vie 15/01/16	
29	✓	**Construccion del frontend**	**14 horas**	**100%**	**vie 15/01/16**	
30	✓	Codificación del HTML	6 horas	100%	vie 15/01/16	
31	✓	Creación de las hojas de estilo	8 horas	100%	lun 18/01/16	
32	✓	Entrega del Frontend	0 horas	100%	mar 19/01/1	
33	✓	**Definir el analizador léxico y sintactico**	**48 horas**	**100%**	**mar 19/01/1**	
34	✓	Definicion de las expresiones regu	2 horas	100%	mar 19/01/1	
35	✓	Definicion de la gramatica del leng	8 horas	100%	mar 19/01/1	
36	✓	Construcción del AST del lenguaje	8 horas	100%	mié 20/01/1	
37	✓	Almacenar las expresiones en el A	6 horas	100%	jue 21/01/16	
38	✓	Construccion del analizador semar	20 horas	100%	vie 22/01/16	
39	✓	Prueba del analizador	4 horas	100%	mié 27/01/1	
40	✓	Entrega del analizador	0 horas	100%	mié 27/01/1	
41	✓	**Definir el protocolo del websocket**	**16 horas**	**100%**	**mié 27/01/1**	
42	✓	Definir los mensajes a tratar	2 horas	100%	mié 27/01/1	
43	✓	Definir los campos necesarios en c	2 horas	100%	mié 27/01/1	
44	✓	Codificar la creación de mensajes	6 horas	100%	jue 28/01/16	
45	✓	Codificar la creación de mensajes	6 horas	100%	jue 28/01/16	
46	✓	Protocolo del websocket	0 horas	100%	vie 29/01/16	

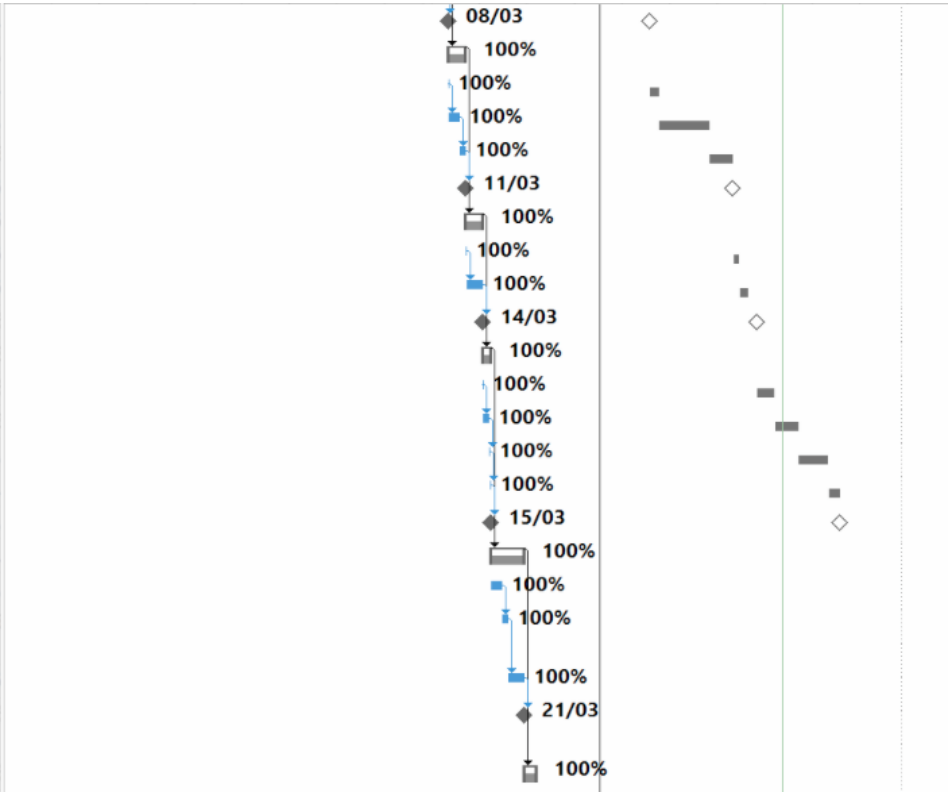
47	✓	**Gestión de usuarios**	**8 horas**	**100%**	**vie 29/01/16**
48	✓	Creación de la tabla de usuarios	1 hora	100%	vie 29/01/16
49	✓	Recuperación de datos	1 hora	100%	vie 29/01/16
50	✓	Creación de usuarios	1 hora	100%	vie 29/01/16
51	✓	Actualización de usuarios	1 hora	100%	vie 29/01/16
52	✓	Eliminación de usuarios	1 hora	100%	lun 01/02/16
53	✓	Codificacion del login	3 horas	100%	lun 01/02/16
54	✓	Gestión de usuarios	0 horas	100%	lun 01/02/16
55	✓	Documentar tareas	28 horas	100%	lun 01/02/16
56	✓	**Objeto World**	**13 horas**	**100%**	**vie 05/02/16**
57	✓	Definición del objeto World	2 horas	100%	vie 05/02/16
58	✓	Definición de la representación de	4 horas	100%	vie 05/02/16
59	✓	Crear la tabla Mundo en el SGBD	1 hora	100%	vie 05/02/16
60	✓	Codificar el objeto en Backend	3 horas	100%	vie 05/02/16
61	✓	Documentar el objeto	3 horas	100%	lun 08/02/16
62	✓	Entrega: objeto World	0 horas	100%	lun 08/02/16
63	✓	**Objeto Me**	**40 horas**	**100%**	**lun 08/02/16**
64	✓	Definición del objeto Me	1 hora	100%	lun 08/02/16
65	✓	Creación de la tabla en el SGBD	1 hora	100%	lun 08/02/16
66	✓	Codificar la lista de edificios	8 horas	100%	lun 08/02/16
67	✓	Codificar la lista de personajes	8 horas	100%	mar 09/02/1
68	✓	Codificar los recursos	6 horas	100%	mié 10/02/1
69	✓	Codificar la creación de edificios	8 horas	100%	jue 11/02/16



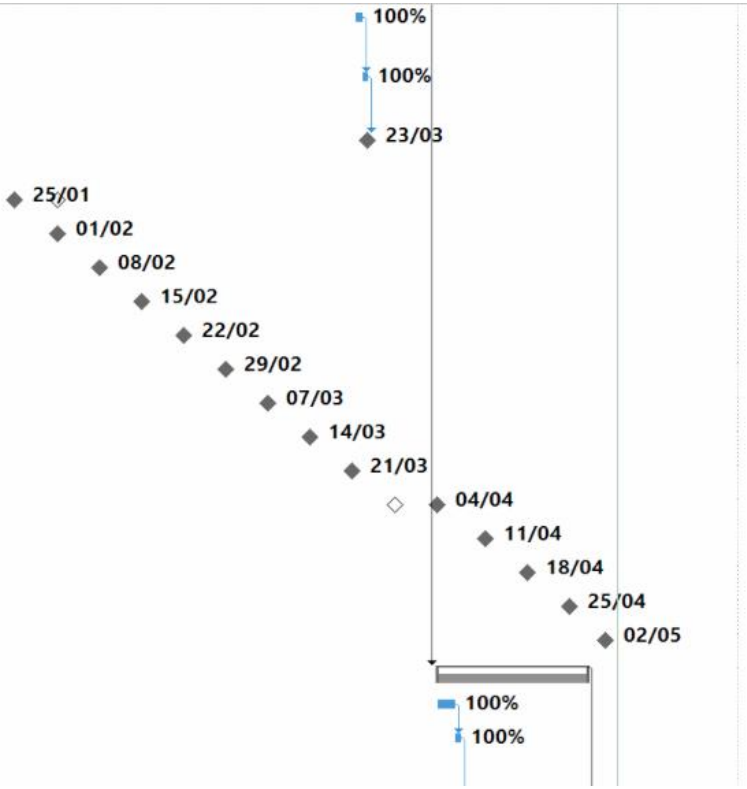
70	✓	Documentar el objeto	8 horas	100%	vie 12/02/16	
71	✓	Entrega: objeto Me	0 horas	100%	lun 15/02/16	
72	✓	**Personaje generico**	**40 horas**	**100%**	**lun 15/02/16**	
73	✓	Definir el personaje generico	8 horas	100%	lun 15/02/16	
74	✓	Codificar las acciones del personaj	16 horas	100%	mar 16/02/1	
75	✓	Crear la tabla personaje	8 horas	100%	jue 18/02/16	
76	✓	Documentar la creación de person	8 horas	100%	vie 19/02/16	
77	✓	Entrega: Personaje Generico	0 horas	100%	lun 22/02/16	
78	✓	**Edificio generico**	**40 horas**	**100%**	**lun 22/02/16**	
79	✓	Definir el edificio generico	8 horas	100%	lun 22/02/16	
80	✓	Codificar las acciones del edificio g	16 horas	100%	mar 23/02/1	
81	✓	Crear la tabla edificio	8 horas	100%	jue 25/02/16	
82	✓	Documentar la creación de edificic	8 horas	100%	vie 26/02/16	
83	✓	Entrega: Edificio Generico	0 horas	100%	lun 29/02/16	
84	✓	**Edificio: centro del poblado**	**24 horas**	**100%**	**lun 29/02/16**	
85	✓	Definir el objeto centro del poblad	4 horas	100%	lun 29/02/16	
86	✓	Codificar las funciones y atributos	4 horas	100%	mar 01/03/1	
87	✓	Documentar el edificio	4 horas	100%	mié 02/03/1	
88	✓	Entrega: Edificio Centro del poblac	0 horas	100%	jue 03/03/16	
89	✓	**Personaje: Aldeano**	**24 horas**	**100%**	**jue 03/03/16**	
90	✓	Definir el objeto aldeano	4 horas	100%	jue 03/03/16	
91	✓	Codificar funciones y atributos del	4 horas	100%	vie 04/03/16	
92	✓	Documentar el personaje	4 horas	100%	lun 07/03/16	



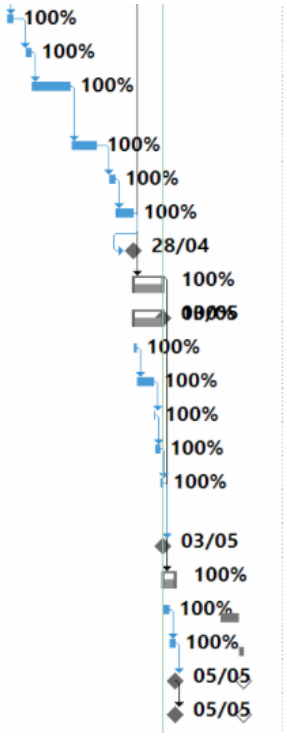
93	✓	Entrega: personaje Aldeano	0 horas	100%	mar 08/03/1
94	✓	**Plantillas de ejecucion**	**22 horas**	**100%**	**mar 08/03/1**
95	✓	Definicion de las plantillas	2 horas	100%	mar 08/03/1
96	✓	Codificacion de las plantillas	12 horas	100%	mar 08/03/1
97	✓	Pruebas sobre las plantillas	8 horas	100%	jue 10/03/16
98	✓	Entrega: Plantillas de ejecucion	0 horas	100%	vie 11/03/16
99	✓	**Instrucciones de ejecucion backend**	**6 horas**	**100%**	**vie 11/03/16**
100	✓	Definicion de mensajes	2 horas	100%	vie 11/03/16
101	✓	Codificacion del generador de mer	4 horas	100%	vie 11/03/16
102	✓	Entrega: Instrucciones de ejecucio	0 horas	100%	lun 14/03/16
103	✓	**Tutorial de juego**	**14 horas**	**100%**	**lun 14/03/16**
104	✓	Definicion de los pasos	2 horas	100%	lun 14/03/16
105	✓	Codificacion de las fases	6 horas	100%	lun 14/03/16
106	✓	Creación de pruebas unitarias	4 horas	100%	mar 15/03/1
107	✓	Persistir el tutorial en el SGBD	2 horas	100%	mar 15/03/1
108	✓	Entrega: Tutorial de juego	0 horas	100%	mar 15/03/1
109	✓	**Administración del tutorial**	**26 horas**	**100%**	**mar 15/03/1**
110	✓	Creación del frontend	16 horas	100%	mar 15/03/1
111	✓	Creación del punto de entrada Backend REST	6 horas	100%	jue 17/03/16
112	✓	Securización del panel	4 horas	100%	vie 18/03/16
113	✓	Entrega: Administración del tutorial	0 horas	100%	lun 21/03/16
114	✓	**Administración de los usuarios**	**18 horas**	**100%**	**lun 21/03/16**



115	✓	Ampliación del panel de administración	12 horas	100%	lun 21/03/16	
116	✓	Creación del punto de entrada Backend REST	6 horas	100%	mar 22/03/1	
117	✓	Entrega: Administración de los usuarios	0 horas	100%	mié 23/03/1	
118	✓	Reunion Scrum	0 horas	100%	lun 25/01/16	
119	✓	Reunion Scrum	0 horas	100%	lun 01/02/16	
120	✓	Reunion Scrum	0 horas	100%	lun 08/02/16	
121	✓	Reunion Scrum	0 horas	100%	lun 15/02/16	
122	✓	Reunion Scrum	0 horas	100%	lun 22/02/16	
123	✓	Reunion Scrum	0 horas	100%	lun 29/02/16	
124	✓	Reunion Scrum	0 horas	100%	lun 07/03/16	
125	✓	Reunion Scrum	0 horas	100%	lun 14/03/16	
126	✓	Reunion Scrum	0 horas	100%	lun 21/03/16	
127	✓	Reunion Scrum	0 horas	100%	lun 04/04/16	
128	✓	Reunion Scrum	0 horas	100%	lun 11/04/16	
129	✓	Reunion Scrum	0 horas	100%	lun 18/04/16	
130	✓	Reunion Scrum	0 horas	100%	lun 25/04/16	
131	✓	Reunion Scrum	0 horas	100%	lun 02/05/16	
132	✓	**Creación de la memoria del proyecto**	**152 horas**	**100%**	**lun 04/04/16**	
133	✓	Escritura de aspectos teoricos	24 horas	100%	lun 04/04/16	
134	✓	Revisión de planificación y presupuesto	8 horas	100%	jue 07/04/16	



135	✓	Descripción de las interfaces	8 horas	100%	vie 08/04/16
136	✓	Explicación de pruebas	8 horas	100%	lun 11/04/16
137	✓	Descripción de las tecnologias empleadas	36 horas	100%	mar 12/04/1
138	✓	Creación de manuales	36 horas	100%	lun 18/04/16
139	✓	Escritura de gestión de proyectos	8 horas	100%	lun 25/04/16
140	✓	Formato y detalles	24 horas	100%	mar 26/04/1
141	✓	Entrega: Memoria	0 horas	100%	jue 28/04/16
142	✓	**Ampliaciones**	**24 horas**	**100%**	**vie 29/04/16**
143	✓	**Ampliación: Enemigos**	**24 horas**	**100%**	**vie 29/04/16**
144	✓	Definir el enemigo generico	4 horas	100%	vie 29/04/16
145	✓	Codificar las acciones y la IA	6 horas	100%	vie 29/04/16
146	✓	Documentar el enemigo	4 horas	100%	lun 02/05/16
147	✓	Crear el enemigo Planta	4 horas	100%	lun 02/05/16
148	✓	Codificar el generador de enemigos	6 horas	100%	mar 03/05/1
149	✓	Entrega: Enemigos	0 horas	100%	mar 03/05/1
150	✓	**Cierre del proyecto**	**16 horas**	**100%**	**mié 04/05/1**
151	✓	Despliegue a produccion	8 horas	100%	mié 04/05/1
152	✓	Pruebas en produccion	8 horas	100%	jue 05/05/16
153	✓	Validación final	0 horas	100%	jue 05/05/16
154	✓	Cierre del proyecto	0 horas	100%	jue 05/05/16



13.2 Presupuesto Final

A continuación se presenta el presupuesto final detallado tras haber realizado el desarrollo completo del sistema.

13.2.1 Desarrollo de Presupuesto Detallado (Empresa)

A continuación se muestra el presupuesto final por parte de la empresa incluyendo los recortes en tiempo por la facilidad de programación y los retrasos ocasionados por nuevas tareas. Igualmente se detallan nuevos módulos que no estaban previstos incluidos dentro del presupuesto original.

Igualmente se incluye la compra de 4 licencias mensuales de la herramienta *CASE* Visual Paradigm, con una amortización del 100% de las mismas debido a que se ha usado a lo largo de todo el desarrollo.

Tras la realización del mismo, se nota una diferencia de 2.009,81 € con respecto al presupuesto inicial debido a que el recorte en horas es mayor que el coste introducido de las nuevas tareas.

Este beneficio a mayores se suma al beneficio calculado de este presupuesto de 861,17€, haciendo que el beneficio total sea de 2.870,98 €.

Ítem	Concepto	Cantidad	Amortización	Precio Unitario (€)	Total (€)
1	*Investigación Inicial*				
1.1	Procesos de inicio	1h	100%	10	10
1.2	Investigación preliminar	12h	100%	10	120
1.3	Planificación de riesgos	8h	100%	10	80
1.4	Preparación del entorno de desarrollo	6h	100%	10	60
2	*Recursos Software*				
2.1	Microsoft Windows 10	2	10%	135	27
2.2	Microsoft Office Professional 2013	2	7%	539	75,46
2.3	Ubuntu Server 14.04 LTS	1	10%	0	0
2.4	Licencia PyCharm	1	40%	199	79,60
2.5	Vim	1	5%	0	0

2.6	Apache Server	1	100%	0	0
2.7	Python 3	3	100%	0	0
2.8	Servidor Tornado	1	100%	0	0
2.9	Framework *Frontend*	1	100%	0	0
2.10	Analizador léxico y sintáctico Python	1	100%	0	0
2.11	SGBD MySQL	1	100%	0	0
2.12	Herramienta *CASE* Visual Paradigm	4	100%	30	120
3	*Recursos Hardware*				
3.1	Servidor Dedicado	1	100%	84,62	84,62
3.2	Equipo Sobremesa	1	10%	1400	140
3.3	Equipo portátil	1	10%	1200	120
4	*Módulos del proyecto*				
4.1	Construcción del servidor *backend*	9h	100%	10	90
4.2	Construcción del *frontend*	14h	100%	10	140
4.3	Definir el analizador léxico y sintáctico	48h	100%	10	480
4.4	Definir el protocolo del WebSocket	16h	100%	10	160
4.5	Gestión de usuarios	8h	100%	10	80
4.6	Documentación de Tareas	28h	100%	10	280
4.7	Objeto *World*	13h	100%	10	130
4.8	Objeto *Me*	40h	100%	10	400
4.9	Personaje genérico	40h	100%	10	400
4.10	Edificio genérico	40h	100%	10	400
4.11	Edificio: Centro del poblado	24h	100%	10	240
4.12	Personaje: Aldeano	24h	100%	10	240
4.13	Plantillas de ejecución	22h	100%	10	220
4.14	Instrucciones de ejecución *backend*	6h	100%	10	60
4.15	Tutorial de juego	14h	100%	10	140

4.16	Administración del tutorial	26h	100%	10	260
4.17	Administración de los usuarios	18h	100%	10	180
4.18	Elaboración de la memoria	152h	100%	10	1520
4.19	Ampliación Enemigos	24h	100%	10	240
4.20	Cierre y despliegue	16h	100%	10	160
5	*Otros Gastos*				
5.1	Desplazamientos	-	100%	130	130
5.2	Electricidad	-	100%	120	120
5.3	Gestión	-	100%	100	100
5.4	Gastos de oficina	-	100%	125	125
5.5	Contingencias	-	100%	600	600
5.6	Alojamiento	-	100%	800	800
Subtotal					8.611,68
Beneficio (10%)					861,17
IVA (21%)					1.989,30
TOTAL					**11.462,15**

13.3 Gestión del proyecto

A lo largo del desarrollo de este proyecto se han empleado distintos procesos de gestión del mismo, que se detallan a continuación. Estos procesos se han empleado para asegurar principalmente el cumplimiento del plazo de entrega, la calidad del trabajo desarrollado y minimizar las posibilidades de que un riesgo imprevisto pusiera en peligro su entrega.

A continuación se detallan los procesos empleados y como se han usado.

13.3.1 Gestión de riesgos

Inicialmente se realizó una gestión de riesgos identificativa y se cuantificaron para tratar de enfocar su peligrosidad e impacto. Estos documentos se adjuntan a la presente memoria.

Tras esta gestión de riesgos inicial se realizó una gestión de riesgos semanal identificando aquellos riesgos que se detecten cada semana en la reunión semanal. Esta última gestión se encuentra reflejada en las actas de reuniones adjuntas al presente documento.

13.3.2 Alcance del proyecto

Ha sido necesario definir y establecer el alcance previsto del proyecto, dicho alcance se encuentra reflejado en la presente memoria así como la justificación de todo el trabajo desarrollado.

13.3.3 Gestión del tiempo

Se ha desarrollado un cronograma y una planificación a seguir que se indica en esta memoria mediante el programa Project de Microsoft. En todo momento se ha tratado de no sobrepasar la fecha límite de ninguna tarea.

13.3.4 Gestión del cambio

Todos los cambios que han sido detectados y sea necesario aplicar para la aprobación del proyecto final se han llevado a cabo en las ya citadas reuniones semanales. Dicha prueba de la gestión del cambio se encuentra en las actas de reuniones adjuntas a este documento.

13.3.5 Gestión de comunicaciones e interesados

Todas aquellas comunicaciones llevadas a cabo con los interesados se han realizado de forma semanal, planificando las tareas a desarrollar a lo largo de la semana y validando el trabajo desarrollado hasta el momento o rechazándolo e indicando los correspondientes cambios.

Estas reuniones se encuentran descritas en las adjuntas actas de reuniones.

13.3.6 Cierre del proyecto

Tras la última aprobación del proyecto, despliegue del mismo y finalización de la presente memoria, esta será entregada junto a todos sus anexos y adjuntos para completar el cierre del actual proyecto.

Capítulo 14. Referencias Bibliográficas

14.1 Libros y Artículos

A continuación se relatan los libros consultados durante el desarrollo del proyecto.

[Cueva01] Cueva Lovelle, Juan Manuel. “Lenguajes gramáticas y autómatas”. 2001.

[Kniberg07] Kniberg, Henrik. “Scrum&XP from the Trenches”. 2007.

[Meucci14] Meucci, Matteo; Muller, Andrew. “OWASP Testing Guide 4.0”. 2014.

[Schwaber11] Schwaber, Ken; Sutherland, Jeff. “La guía de Scrum”. 2011.

[Stuttard11] Stuttard, Dafydd; Pinto, Marcus. “The Web Application Hacker’s Handbook”. Wiley Publishing. 2011.

14.2 Referencias en Internet

[AC16] The Ace Community. "Ace The High Performance Code Editor for The Web". https://ace.c9.io/. 2016.

[AUIT16] The AngularUI Team. "Angular Directives for Bootstrap". http://angular-ui.github.io/bootstrap/. 2016.

[Brandl16] Brandl, George; Sphinx Team. "Sphinx Documentation". http://www.sphinx-doc.org/en/stable/contents.html. 2016.

[Beazley15] Beazley, David M. "PLY (Python Lex-Yacc)". http://www.dabeaz.com/ply/ply.html. 2015.

[CC16] CodeCombat. "CodeCombat – Learn how to code by playing a game". https://codecombat.com/. 2016.

[CO16] Code.org. "The Hour of Code". https://hourofcode.com/es. 2016.

[Google16] Google. "AngularJS API: API Reference". https://code.angularjs.org/1.4.9/docs/api. 2015.

[Grins15] Grins, Brian. "A* Search / Pathfinding Algorithm in Javascript". http://github.com/bgrins/javascript-astar. 2015.

[Groves15] Groves, Mat. "Pixi.js Documentation". http://pixijs.github.io/docs/. 2015.

[Kelley13] Kelley, Jacob. "A Library for creating beautiful mobile shelfs in Javascript". https://github.com/jakiestfu/Snap.js/. 2013.

[Microsoft16] Microsoft. "TypeScript Documentation". http://www.typescriptlang.org/docs/. 2016.

[Mott13] Mott, Jeff. "CryptoJS". https://code.google.com/archive/p/crypto-js/. 2013.

[NoAuthor16] No Autor. "LightBot". 2016.

[NoAuthor] No Author. "YUIDoc Syntax Reference". http://yui.github.io/yuidoc/syntax/index.html.

[Ostrovsky09] Ostrovsky, Igor. "RoboZZle online puzzle game". 2009.

[PSF16] Python Software Foundation. "Python 3.4.4 Documentation". https://docs.python.org/3.4/. 2016.

[TTA16] The Tornado Authors, "Tornado Web Server – Documentation 4.3". http://www.tornadoweb.org/en/stable/. 2016.

[VC16] Valve Corporation. "Steam Store". http://store.steampowered.com/. 2016.

Capítulo 15. Apéndices

15.1 Glosario y Diccionario de Datos

- **Backend**: Parte del sistema ejecutándose en el equipo servidor.
- **Base de datos**: Conjunto de datos organizado y dividido por tablas persistente.
- **Cliente**: Equipo desde el cual un usuario se conecta al sistema.
- **Cookie**: Pequeño fragmento de información que envía un servidor web al usuario que lo visita con información de sus preferencias o datos.
- **Framework**: Conjunto de herramientas y utilidades empleadas en el desarrollo del proyecto.
- **Frontend**: Parte del sistema ejecutándose en el equipo cliente.
- **JavaScript**: Lenguaje de programación incluido en un sitio web que se ejecuta en el navegador del usuario.
- **Navegador web**: Programa ejecutado en el equipo del usuario que realiza peticiones a un servidor web e interpreta el código devuelto por el mismo de forma gráfica para visualizar un sitio web correcto.
- **Parsear**: Proceso de transformación de un conjunto de datos a otro mediante una serie de reglas.
- **Python**: Lenguaje de programación interpretado ejecutado en este caso en el lado servidor.
- **Renderizar**: Proceso digital de dibujado en una pantalla.
- **Servidor**: Equipo en el cual se ejecuta la aplicación y atiende las peticiones de los clientes, devolviendo el código correspondiente para cada petición.
- **Servidor Web**: Aplicación corriendo en el equipo servidor encargada de atender las peticiones de los clientes y servir las páginas correspondientes.
- **Token**: Identificador único otorgado al usuario al realizar un inicio de sesión.

15.2 Contenido Entregado en el Archivo adjunto

15.2.1 Contenidos

15.2.1.1.1 Estructura general de directorios del proyecto

Directorio	Contenido
./ Directorio raíz del Archivo adjunto	Contiene un fichero leeme.txt explicando toda esta estructura.
./age-of-code	Contiene toda la estructura de directorios del proyecto para desarrollo.
./instalacion	Ficheros utilizados para la instalación del proyecto.
./documentación	Contiene toda la documentación asociada al proyecto.
./documentación/img	Directorio que contiene las imágenes utilizadas en la documentación.
./documentación/uml	Ficheros que genera la herramienta Visual Paradigm empleada.
./herramientas	Contiene los ficheros de instalación de las herramientas utilizadas para el desarrollo o puesta en marcha del proyecto.
./herramientas/desarrollo	Ficheros de instalación de las herramientas utilizadas en el desarrollo
./sql	Ficheros de respaldo de la base de datos.

15.2.1.1.2 Estructura de Directorios de desarrollo

Directorio	Contenido
./ Directorio raíz de "desarrollo"	Contiene la raíz del servidor *backend*.
./lexer	Contiene todos los archivos de código de los analizadores léxico y sintáctico, así como las clases del AST.
./rest	Contiene los puntos de entrada REST del servidor.
./tests	Contiene los *tests* unitarios del proyecto.
./tools	Contiene las clases utilidad del proyecto.
./websocket	Contiene las clases y utilidades del control del WebSocket.
./doc	Contiene toda la documentación relativa al proyecto.
./doc/backend	Contiene la documentación relativa al *backend* del proyecto.
./doc/frontend	Contiene la documentación relativa al *frontend* del proyecto.
./frontend	Contiene la raíz del contenido *frontend* que ha de ser introducido en el servidor apache.
./frontend/js	Contiene los scripts JavaScript de bibliotecas externas empleadas.
./frontend/texts	Contiene las texturas y *sprites* de los objetos del juego.
./frontend/ts	Contiene los archivos TypeScript y JavaScript desarrollados para el proyecto.
./frontend/css	Contiene aquellas hojas de estilo creadas y usadas en el proyecto.

15.2.2 Código Ejecutable e Instalación

Para ejecutar este proyecto solo se requiere un servidor apache en el que se incluirán las paginas HTML a servir y disponer de python3 y MySQL en el servidor, disponiendo de los paquetes de Python ply, tornado y el conector con MySQL. Para más información con respecto a este punto consúltese el manual de instalación y el manual de ejecución.

Capítulo 16. Anexo I: Actas de Reunión

16.1 Acta día 25 de Enero

Autor: Gabriel Manteca Muñoz **Fecha:** 25 enero 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.1.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Construcción del servidor ***backend***	100%	Si	
Construcción del ***frontend***	100%	Si	
Definir el analizador léxico y sintáctico	80%	Si	

16.1.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Definir el analizador léxico y sintáctico	100%	
Definir el protocolo del WebSocket	100%	
Gestión de usuarios	100%	

16.1.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.1.4 Control de cambios

16.1.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.1.5 Observaciones

No se incluyen observaciones en esta reunión.

16.1.6 Valoración de los Supervisores

16.1.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.1.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.2 Acta día 1 de febrero

Autor: Gabriel Manteca Muñoz **Fecha:** 1 febrero 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.2.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Definir el analizador léxico y sintáctico	100%	Si	
Definir el protocolo del WebSocket	100%	Si	
Gestión de usuarios	100%	Si	

16.2.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Documentación formal de lo anterior	100%	
Objeto ***World***	100%	

16.2.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.2.4 Control de cambios

16.2.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.2.5 Observaciones

No se incluyen observaciones en esta reunión.

16.2.6 Valoración de los Supervisores

16.2.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.2.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.3 Acta día 8 de febrero

Autor: Gabriel Manteca Muñoz **Fecha:** 8 febrero 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.3.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Documentación formal de lo anterior	50%	Si	Faltan diagramas de actividad, secuencia
Objeto ***World***	90%	Si	Añadir más métodos

16.3.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Objeto ***World***	100%	Terminar
Documentación	100%	Terminar
Objeto ***Me***	20%	Empezar

16.3.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.3.4 Control de cambios

16.3.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.3.5 Observaciones

No se incluyen observaciones en esta reunión.

16.3.6 Valoración de los Supervisores

16.3.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.3.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.4 Acta día 15 de febrero

Autor: Gabriel Manteca Muñoz **Fecha:** 15 febrero 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.4.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Documentación	100%	Si	
Objeto ***World***	100%	Si	
Objeto ***Me***	100%	Si	

16.4.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Edificio genérico	100%	

16.4.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.4.4 Control de cambios

16.4.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.4.5 Observaciones

No se incluyen observaciones en esta reunión.

16.4.6 Valoración de los Supervisores

16.4.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.4.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.5 Acta día 21 de febrero

Autor: Gabriel Manteca Muñoz **Fecha:** 21 febrero 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.5.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Edificio genérico	100%	Si	

16.5.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Personaje genérico	100%	

16.5.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.5.4 Control de cambios

16.5.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.5.5 Observaciones

No se incluyen observaciones en esta reunión.

16.5.6 Valoración de los Supervisores

16.5.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.5.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.6 Acta día 29 de febrero

Autor: Gabriel Manteca Muñoz **Fecha:** 29 febrero 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.6.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Personaje genérico	100%	Si	
Edificio genérico	100%	Si	

16.6.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Personaje aldeano	90%	
Edificio Centro del poblado	100%	

16.6.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.6.4 Control de cambios

16.6.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.6.5 Observaciones

No se incluyen observaciones en esta reunión.

16.6.6 Valoración de los Supervisores

16.6.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.6.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.7 Acta día 7 de marzo

Autor: Gabriel Manteca Muñoz **Fecha:** 7 marzo 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.7.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Personaje Aldeano	90%	Si	
Edificio Centro del poblado	100%	Si	

16.7.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Plantillas de ejecución	100%	
Personaje aldeano	100%	

16.7.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.7.4 Control de cambios

16.7.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.7.5 Observaciones

No se incluyen observaciones en esta reunión.

16.7.6 Valoración de los Supervisores

16.7.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.7.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.8 Acta día 14 de marzo

Autor: Gabriel Manteca Muñoz **Fecha:** 14 marzo 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.8.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Plantillas de ejecución	100%	Si	
Personaje aldeano	100%	Si	

16.8.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Tutorial de juego	100%	
Administración del tutorial	100%	

16.8.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.8.4 Control de cambios

16.8.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción
1	Gestión del tutorial	Si	Director	Para facilitar la gestión de los tutoriales se solicita la creación de un panel de administración que aligere este trabajo.

16.8.5 Observaciones

No se incluyen observaciones en esta reunión.

16.8.6 Valoración de los Supervisores

16.8.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.8.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.9 Acta día 21 de marzo

Autor: Gabriel Manteca Muñoz **Fecha:** 21 marzo 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.9.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Tutorial de juego	100%	Si	
Administración del tutorial	100%	Si	

16.9.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Administración de usuarios	100%	
Creación de la memoria del proyecto	25%	

16.9.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.9.4 Control de cambios

16.9.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción
1	Introducción de la memoria del proyecto en la planificación	Si	Director	Incluir una estimación del tiempo que llevará escribir los apartados de la misma.
2	Descanso de desarrollo	Si	Desarrollador	Detener el desarrollo del proyecto durante las vacaciones que inician el día 23 y continuar al volver.

16.9.5 Observaciones

No se incluyen observaciones en esta reunión.

16.9.6 Valoración de los Supervisores

16.9.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.9.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.10 Acta día 4 de abril

Autor: Gabriel Manteca Muñoz **Fecha:** 4 abril 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.10.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Administración de usuarios	100%	Si	
Creación de la memoria del proyecto	25%	Si	

16.10.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Creación de la memoria del proyecto	50%	

16.10.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.10.4 Control de cambios

16.10.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.10.5 Observaciones

No se incluyen observaciones en esta reunión.

16.10.6 Valoración de los Supervisores

16.10.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.10.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.11 Acta día 11 de abril

Autor: Gabriel Manteca Muñoz **Fecha:** 11 abril 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.11.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Creación de la memoria del proyecto	50%	Si	

16.11.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Creación de la memoria del proyecto	75%	

16.11.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.11.4 Control de cambios

16.11.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.11.5 Observaciones

No se incluyen observaciones en esta reunión.

16.11.6 Valoración de los Supervisores

16.11.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.11.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.12 Acta día 18 de abril

Autor: Gabriel Manteca Muñoz **Fecha:** 18 abril 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.12.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Creación de la memoria del proyecto	75%	Si	

16.12.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Creación de la memoria del proyecto	90%	

16.12.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.12.4 Control de cambios

16.12.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.12.5 Observaciones

No se incluyen observaciones en esta reunión.

16.12.6 Valoración de los Supervisores

16.12.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.12.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.13 Acta día 25 de abril

Autor: Gabriel Manteca Muñoz **Fecha:** 25 abril 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.13.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Creación de la memoria del proyecto	90%	Si	

16.13.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Creación de la memoria del proyecto	100%	
Ampliación: Enemigos	100%	

16.13.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.13.4 Control de cambios

16.13.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.13.5 Observaciones

No se incluyen observaciones en esta reunión.

16.13.6 Valoración de los Supervisores

16.13.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.13.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.14 Acta día 3 de Mayo

Autor: Gabriel Manteca Muñoz **Fecha:** 3 mayo 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.14.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Creación de la memoria del proyecto	100%	Si	
Ampliación: Enemigos	100%	Si	

16.14.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Ampliación: Más enemigos y combate	100%	Añadir más IA y combate para el usuario.

16.14.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.14.4 Control de cambios

16.14.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.14.5 Observaciones

No se incluyen observaciones en esta reunión.

16.14.6 Valoración de los Supervisores

16.14.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.14.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

16.15 Acta día 9 de mayo

Autor: Gabriel Manteca Muñoz **Fecha:** 9 mayo 2016

Nombre del proyecto: Age of Code: Desarrollo de un Videojuego para aprender a Programar

16.15.1 Tareas Iteración pasada

Nombre	Progreso	Aprobada	Comentarios
Ampliación: Más enemigos y combate	100%	Si	

16.15.2 Tareas a desarrollar

Nombre	Avance esperado	Descripción
Cierre de proyecto	100%	
Revisión de memoria	100%	

16.15.3 Gestión de Riesgos

ID	Nombre	Probabilidad	Impacto	Respuesta

16.15.4 Control de cambios

16.15.4.1 Cambios solicitados

ID	Nombre	Aprobado	Proponente	Descripción

16.15.5 Observaciones

No se incluyen observaciones en esta reunión.

16.15.6 Valoración de los Supervisores

16.15.6.1 Validación del trabajo actual

☒ Aprobado ☐ Denegado

16.15.6.2 Notas de los supervisores

No hay notas.

Firma del desarrollador:	Firma del supervisor:
Fecha: _____	Fecha: _____
Nombre: _____	Nombre: _____

Capítulo 17. Anexo II: Gestión de riesgos

17.1 Plan de gestión de riesgos

17.1.1 Metodología

Durante el desarrollo del proyecto se plantea una metodología de identificación y gestión de riesgos basada en trabajo en equipo y reuniones periódicas.

Cada vez que se detecte un riesgo este será notificado en una reunión periódica (SCRUM), se calculará y cuantificará su alcance así como las medidas a tomar sobre el riesgo y será registrado en el correspondiente acta de reunión.

En cuanto a la identificación de riesgos inicial, se plantea una consulta con el director del proyecto para ayudar a identificar dichos riesgos. Así como una tormenta de ideas para intentar detectar la mayor cantidad de riesgos posible.

Una vez detectados se calculará y cuantificará su alcance así como se discutirán las medidas a tomar sobre el mismo.

Finalmente en cuanto a los riesgos que no puedan eliminarse o mitigarse, el desarrollador cumplirá el papel de monitorizar y controlar los riesgos intentando mitigar su efecto en el proceso de desarrollo, así como de controlar los demás riesgos y dirigir la identificación de los nuevos.

17.1.2 Herramientas y tecnologías

A la hora de gestionar los riesgos se emplearan las siguientes técnicas:

Reuniones periódicas: El desarrollador y el director se reunirán periódicamente como mínimo una vez a la semana y discutirán la posibilidad de nuevos riesgos que se hayan podido detectar.

Tormenta de ideas: a la hora de empezar a identificar riesgos se plantea ejecutar una tormenta de ideas entre el desarrollador y el director para intentar identificar los posibles riesgos inherentes al proyecto.

Consulta de expertos: Para empezar a identificar los riesgos se plantea realizar una consulta al director con mayor experiencia para que colabore y detecte posibles riesgos que el desarrollador no haya detectado.

17.1.3 Roles y Responsabilidades

- Desarrollador: Dirigir las reuniones SCRUM, participación en las reuniones de identificación de riesgos, identificación de riesgos, control y monitorización de riesgos evitables.
- Director: Experto al que consultar en materia de riesgos y director de las reuniones SCRUM.

17.1.4 Presupuesto

Se ha calculado un presupuesto de 1.000 € para paliar los posibles riesgos detectados en el desarrollo del proyecto.

17.1.5 Calendario

La gestión de riesgos se realizará de la siguiente forma:

- Gestión inicial de riesgos: En el momento de iniciar el proyecto se realizará una reunión que servirá para determinar los riesgos iniciales que puedan encontrarse.
- Gestión general de riesgos: Se realizará de forma semanal durante las reuniones del sprint de SCRUM, realizados típicamente cada lunes.

17.1.6 Categorías de Riesgo

Se presentan las siguientes categorías de riesgo basadas en aquellas presentadas en el PMBoK:

- Técnico: Requisitos, Tecnología, Complejidad, Fiabilidad.
- Externo: Proveedores, Usuario, Tiempo, Legislativo.
- Organizacional: Dependencias, Priorización.
- Gestión de proyecto: Estimación, Planificación, Control.

17.1.7 Definiciones de probabilidad

Muy Baja	(0%..20%] El valor usado en la matriz de probabilidad e impacto es: 10%
Baja	(20%..40%] El valor usado en la matriz de probabilidad e impacto es: 30%
Media	(40%..60%] El valor usado en la matriz de probabilidad e impacto es: 50%
Alta	(60%..80%] El valor usado en la matriz de probabilidad e impacto es: 70%
Muy Alta	(80%..100%) El valor usado en la matriz de probabilidad e impacto es: 90%

17.1.8 Definiciones de impacto por objetivos

Impacto sobre los objetivos principales Definido sobre amenazas					
Objetivos	*Muy bajo*	*Bajo*	*Medio*	*Alto*	*Crítico*
Presupuesto	Aumento del coste insignificante	Aumento del coste <10%	Aumento del coste de 10-20%	Aumento del coste de 20-40%	Aumento del coste >40%
Tiempo	El plazo de entrega no se ha visto alterado	El plazo de entrega se ve alterado muy ligeramente	El plazo de entrega se ha alterado de forma notable	El plazo de entrega se ha alterado de forma muy notable	El proyecto corre peligro de no poder completarse
Alcance	El alcance del proyecto no se ha visto alterado	El alcance del proyecto se ve alterado muy ligeramente	El alcance del proyecto se ha alterado de forma notable	El alcance del proyecto se ha alterado de forma muy notable	El proyecto corre peligro de no ser útil
Calidad	La calidad del proyecto no se ha visto alterado	La calidad del proyecto se ve alterado muy ligeramente	La calidad del proyecto se ha alterado de forma notable	La calidad del proyecto se ha alterado de forma muy notable	El proyecto corre peligro de no ser usable
Impacto sobre los objetivos principales Definido sobre oportunidades					
Objetivos	*Muy bajo*	*Bajo*	*Medio*	*Alto*	*Crítico*
Presupuesto	Reducción del coste insignificante	Reducción del coste <10%	Reducción del coste de 10-20%	Reducción del coste de 20-40%	Reducción del coste >40%
Tiempo	No se ha ahorrado tiempo de forma apreciable	Se han ahorrado < 5 días de desarrollo	Se han ahorrado 5-10 días de desarrollo	Se han ahorrado 10-20 días de desarrollo	Se han ahorrado > 20 días de desarrollo
Alcance	El alcance del proyecto no se ha visto realmente alterado	El alcance del proyecto se ve alterado muy ligeramente	El alcance del proyecto se ha alterado de forma notable	El alcance del proyecto se ha alterado de forma muy notable	El alcance del proyecto ha superado las expectativas del cliente
Calidad	La calidad del proyecto no se ha visto alterada realmente	La calidad del proyecto aumenta muy ligeramente	La calidad del proyecto aumenta de forma notable	La calidad del proyecto aumenta de forma muy notable	La calidad del proyecto ha superado las expectativas del cliente

17.1.9 Matriz de probabilidad e impacto

Definido sobre oportunidades

Muy Alta	***0.90***	0.05	0.14	0.27	0.50	0.81
Alta	***0.70***	0.04	0.11	0.21	0.39	0.63
Media	***0.50***	0.03	0.08	0.15	0.28	0.45
Baja	***0.30***	0.02	0.05	0.09	0.17	0.27
Muy baja	***0.10***	0.01	0.02	0.03	0.06	0.09
		0.05	***0.15***	***0.30***	***0.55***	***0.90***

		Muy bajo	Bajo	Medio	Alto	Crítico
		Definido sobre amenazas				
Muy Alta	***0.90***	0.05	0.14	0.27	0.50	0.81
Alta	***0.70***	0.04	0.11	0.21	0.39	0.63
Media	***0.50***	0.03	0.08	0.15	0.28	0.45
Baja	***0.30***	0.02	0.05	0.09	0.17	0.27
Muy baja	***0.10***	0.01	0.02	0.03	0.06	0.09
		0.05	***0.15***	***0.30***	***0.55***	***0.90***
		Muy bajo	**Bajo**	**Medio**	**Alto**	**Crítico**

17.1.10 Niveles de Tolerancia

La tolerancia sobre las amenazas se encuentra en 0,25.

La tolerancia sobre las oportunidades se encuentra en 0,25.

17.1.11 Planes de contingencia

- En cuanto a la planificación:
  - La planificación tiene una fecha límite que no debe superarse
  - Si comienzan a acumularse retrasos que puedan afectar dicha fecha limite habrá que re planificar las tareas para que estas sean más eficientes sin comprometer la fecha limite
  - En caso de que esta re planificación no sea suficiente, se considera la opción de comprometer parte del alcance siempre y cuando se cuente con la aprobación del supervisor y siga siendo un proyecto apto para lo que se pretende conseguir
- En cuanto al presupuesto:
  - El presupuesto admite una variación bastante más considerable que la planificación.
  - En este aspecto se admiten variaciones leves de presupuesto siempre que estas no comprometan la integridad del proyecto.
- En cuanto al alcance:
  - El proyecto tiene un estándar mínimo que cumplir que no ha de reducirse.
  - Todo cambio en el alcance debe estar aprobado por el supervisor del proyecto y nunca sin motivo.
  - Estos motivos pueden ser desde un error de planificación a un riesgo inesperado.
  - El alcance del proyecto siempre debe ser como mínimo suficiente para superar la convocatoria de este proyecto.

17.1.12 Plan de seguimiento

Se define a continuación el procedimiento para seguir el desarrollo de un riesgo:

- En primer lugar se identifica el riesgo.
- En las reuniones definidas en el calendario anteriormente se evalúa y comprueban los identificadores del riesgo.
- Se redefine su probabilidad y calcula su impacto.
- Trata de evitarse que el riesgo suceda.

17.2 Riesgos Negativos detectados

ID	Nombre	Responsable	Probabilidad	Impacto				Impacto	0,25	Respuesta
				Presup.	Planific.	Alcance	Calidad		Priorización	
1	Dificultad de uso y adaptación de los *frameworks* elegidos.	Equipo de desarrollo	Media	Muy Bajo	Medio	Alto	Medio	0,28		Evitar, tratar de elegir el *framework* de forma correcta previendo lo que se desea realizar con el mismo. En caso de que la adaptación o el uso sea demasiado costosa en tiempo plantear el uso de otro *framework*
2	Gramática y analizador léxico demasiado complejos.	Equipo de desarrollo	Alta	Medio	Alto	Crítico	Alto	0,63		Intentar obtener alguna gramática o analizado *open source* como muestra. Consultar a algún experto.
3	Incumplimiento del plazo de entrega	Equipo de desarrollo	Media	Bajo	Crítico	Alto	Medio	0,45		Evitar, Es necesario monitorizar el proyecto para que este riesgo jamás llegue a producirse. Si se viese imposible de cumplir sería preferible reducir el alcance del proyecto o incluso la calidad del mismo.
4	Perdida del trabajo desarrollado por dificultades técnicas	Equipo de desarrollo	Baja	Alto	Crítico	Crítico	Crítico	0,27		Tratar de evitar el riesgo empleando sistemas de copia de seguridad y de control de versiones en nube.
5	Dificultades durante la definición del protocolo de comunicaciones *backend-frontend*	Equipo de desarrollo	Baja	Muy Bajo	Alto	Medio	Bajo	0,17		Consulta con el supervisor y en caso de ser necesario consulta con un experto en la materia.
6	Entregable que no supera la validación	Equipo de desarrollo	Baja	Muy Bajo	Medio	Medio	Medio	0,09		Adaptación del desarrollo para cumplir la validación.
7	Ataques a la seguridad e integridad de la aplicación	Equipo de desarrollo	Baja	Muy Bajo	Medio	Medio	Alto	0,17		Aseguración del servidor y aplicación. Mitigación de los daños causados

8	Actualización de versión de la plataforma	Equipo de desarrollo	Media	Bajo	Medio	Medio	Medio	0,15		Formación sobre los cambios, adaptación del trabajo desarrollado
9	Variación imprevista en el alcance	Equipo de desarrollo	Baja	Bajo	Alto	Crítico	Medio	0,27		Replanteamiento del alcance del proyecto o ajuste de la planificación.
10	Imposibilidad de trabajo durante cierto tiempo	Equipo de desarrollo	Baja	Medio	Alto	Alto	Alto	0,17		Adaptación del alcance.
11	Planificación realizada de forma incorrecta necesitando más tiempo	Equipo de desarrollo	Baja	Medio	Alto	Alto	Alto	0,17		Modificación en el alcance para que pueda realizarse a tiempo.

17.3 Riesgos Positivos detectados

ID	Nombre	Responsable	Probabilidad	Impacto				Impacto	0,25	Respuesta
				Presup.	Planific.	Alcance	Calidad		Priorización	
1	Conocimiento de las tecnologías empleadas por todas las partes del proyecto	Equipo de desarrollo	Muy Alta	Bajo	Medio	Bajo	Medio	0,27		Ahorro de actividades formativas, adelanto de la planificación, mejora de la calidad por trabajar con tecnologías familiares.
2	Planificación realizada de forma incorrecta necesitando menos tiempo	Equipo de desarrollo	Media	Medio	Alto	Alto	Alto	0,28		Adelanto de la planificación, ahorro de presupuesto en recursos laborales, posibilidad de mejorar el alcance y la calidad del proyecto