UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

PROYECTO FIN DE MÁSTER

"HERRAMIENTA WEB PARA LA EDICIÓN DE NIVELES PARA VIDEOJUEGOS DE TIPO PUZLE BASADOS EN FÍSICAS"

DIRECTOR: José Manuel Redondo López

AUTOR: Aarón Martínez Alvarez

Resumen

Este proyecto consiste en una página web que pone a disposición del usuario una experiencia de juego, creación y compartición de contenidos con otros usuarios dentro de una misma comunidad de jugadores.

Para este proyecto se ha construido una página web en la que se integran videojuegos y una herramienta de creación de contenido para dichos videojuegos, con el objetivo de que los usuarios puedan jugar y compartir sus creaciones con el resto de usuarios de la comunidad.

Este proyecto está dividido en dos secciones, las citadas anteriormente:

1. Videojuegos.
2. Herramienta de creación de contenido.

Videojuegos:

Se han creado dos tipos de videojuegos que los usuarios podrán jugar: Un juego de demolición de construcciones y un juego de habilidad de destrucción de bloques muy similar al clásico Arkanoid. Ambos juegos tienen un estilo *Arcade* y son muy sencillos de jugar.

1. Demolición de Construcciones o *"**Demolition**"*:
   Este juego presenta al jugador una construcción que de derribar usando una serie de proyectiles. El objetivo del juego es conseguir dejar todas las piezas que componen la construcción por debajo de una línea que delimita la altura mínima que hay que conseguir para superar el nivel. La puntuación que el jugador obtenga dependerá del número de proyectiles utilizados para superar el nivel.
2. Destrucción de Bloques o *"**Arkanoid**"*:
   Este juego consiste en destruir una serie de bloques que están colocados en la parte superior de la pantalla, con una bola, que va rebotando en una paleta que se desliza lateralmente controlada por el jugador. El objetivo del juego es destruir todos los bloques sin que la bola toque el borde inferior de la pantalla. La puntuación del jugador varía según el número de bloques destruidos.

Herramienta de Creación de Contenido:

Se proporciona una herramienta web con la que se podrán construir nuevos niveles para los juegos citados anteriormente y que consiste en arrastrar las piezas de una paleta de componentes a una cuadricula para componer la construcción deseada. Una vez terminada la creación, el usuario podrá compartir el nivel creado con el resto de los jugadores de la comunidad.

Todo esto está integrado dentro de una página web donde cada usuario dispondrá de un espacio personal con los niveles que ha elegido para jugar de entre los creados por la comunidad y los niveles propios que haya creado.

Palabras Clave

Videojuego, Creación, Contenido, Comunidad, Compartir, Físicas, Pagina Web.

Abstract

This project is composed by a web portal that offers a gaming, creation and sharing experience to any user within a player's community. We have built a web portal where videogames, and a content creation tool for such games, are integrated in order to allow users to play and share their creations with other users in the community.

This project is divided into two sections, those mentioned above:

1. Videogames.
2. Content creation tool.

Videogames:

Project users can create two types of videogames: A building demolition game and a block destruction game very similar to the classic videogame Arkanoid. Both games are skill-based and a simple and fast playstyle.

1. Demolition game
   This game presents to the player a particular structure composed by blocks. These blocks have to be shot down using a fixed amount of projectiles. The goal is to leave all the components of the construction below a line, which marks the minimum height to be achieved to pass the level. The score that the player gets will depend on the number of shots used to pass the level.
2. Arkanoid
   This game challenges the player to destroy a series of blocks that are placed on top of the screen with a ball. This ball is bouncing on a pallet which slides laterally, controlled by the player. The goal is to destroy all the blocks without letting the ball touch the bottom edge of the screen. The player's score varies according to the number of destroyed blocks.

Content creation tool:

Finally, the project has a content creation web tool to build new levels for the games mentioned above. This tool is based on doing simple drag and drop gestures to a grid-shaped creation area, from a palette of predefined components. After its creation, the user can share a level with other community players.

The website also provides each user a personal space with levels he had chosen to play, composed by those created by the community and its own created levels.

Keywords

Videogame, Creation, Content, Community, Share, Physics, Web Page.

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO 19

1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO 19
1.2 RESUMEN DE TODOS LOS ASPECTOS 20
1.2.1 Memoria del proyecto 20
1.2.2 Introducción 20
1.2.3 Aspectos Teóricos 20
1.2.4 Planificación del Proyecto y resumen de presupuestos 20
1.2.5 Análisis 20
1.2.6 Diseño del sistema 20
1.2.7 Implementación del sistema 21
1.2.8 Desarrollo de pruebas 21
1.2.9 Manuales del sistema 21
1.2.10 Conclusiones y ampliaciones 21
1.2.11 Presupuesto 21
1.2.12 Referencias bibliográficas 21

CAPÍTULO 2. INTRODUCCIÓN 23

2.1 JUSTIFICACIÓN DEL PROYECTO 23
2.2 OBJETIVOS DEL PROYECTO 24
2.3 ESTUDIO DE LA SITUACIÓN ACTUAL 26
2.3.1 Evaluación de Alternativas 27

CAPÍTULO 3. ASPECTOS TEÓRICOS 33

3.1 BOX2D 33
3.2 ARKANOID 34
3.3 PLAYLIST 35

CAPÍTULO 4. PLANIFICACIÓN DEL PROYECTO Y RESUMEN DE PRESUPUESTO 37

4.1 PLANIFICACIÓN 37
4.2 RESUMEN DEL PRESUPUESTO 40

CAPÍTULO 5. ANÁLISIS 41

5.1 DEFINICIÓN DEL SISTEMA 41
5.1.1 Determinación del Alcance del Sistema 41
5.2 REQUISITOS DEL SISTEMA 42
5.2.1 Obtención de los Requisitos del Sistema 44
5.2.2 Identificación de Actores del Sistema 47
5.2.3 Especificación de Casos de Uso 47
5.3 IDENTIFICACIÓN DE LOS SUBSISTEMAS EN LA FASE DE ANÁLISIS 58
5.3.1 Descripción de los Subsistemas 58
5.3.2 Descripción de los Interfaces entre Subsistemas 58
5.4 DIAGRAMA DE CLASES PRELIMINAR DEL ANÁLISIS 59
5.4.1 Diagrama de Clases 59
5.4.2 Descripción de las Clases 60
5.5 ANÁLISIS DE CASOS DE USO Y ESCENARIOS 73

5.5.1 Casos de uso referentes a la página web 73
5.5.2 Casos de uso referentes a los videojuegos en común. 81
5.5.3 Casos de uso referentes al juego "Demolition" 83
5.5.4 Casos de uso referentes al juego "Arkanoid" 85
5.5.5 Casos de uso referentes a la herramienta de edición de niveles en general. 87
5.5.6 Casos de uso referentes a la herramienta de edición de niveles del juego "Demolition" 91
5.6 ANÁLISIS DE INTERFACES DE USUARIO 93
5.6.1 Descripción de la Interfaz 93
5.6.2 Diagrama de Navegabilidad 100
5.7 ESPECIFICACIÓN DEL PLAN DE PRUEBAS 101
5.7.1 Pruebas Unitarias 101
5.7.2 Pruebas de integración y sistema 103

CAPÍTULO 6. DISEÑO DEL SISTEMA 105

6.1 ARQUITECTURA DEL SISTEMA 105
6.1.1 Diagramas de Paquetes 105
6.1.2 Diagramas de Componentes 107
6.1.3 Diagramas de Despliegue 110
6.2 DISEÑO DE CLASES 111
6.2.1 Diagrama de Clases 111
6.3 DIAGRAMAS DE INTERACCIÓN Y ESTADOS 117
6.3.1 Caso de Uso 1 y 4: Login e Ir al home de usuario 117
6.3.2 Caso de Uso 2: Logout 118
6.3.3 Caso de Uso 3: Crear usuario 118
6.3.4 Caso de uso 5 y 6: Ir al editor de niveles: "Demolition", "Arkanoid" 119
6.3.5 Caso de uso 7: Ir a la página "Community" 119
6.3.6 Caso de Uso 8: Ir a la página "About" 120
6.3.7 Caso de Uso 9: Jugar nivel 120
6.3.8 Caso de Uso 10: Editar nivel 121
6.3.9 Caso de Uso 13: Volver a jugar nivel 121
6.3.10 Caso de Uso 11: Añadir nivel a la playlist del usuario 122
6.3.11 Caso de Uso 14: Volver al Home del usuario 123
6.3.12 Caso de Uso 26: Guardar nivel 124
6.4 DIAGRAMAS DE ACTIVIDADES 125
6.5 DISEÑO DE LA BASE DE DATOS 128
6.5.1 Descripción del SGBD Usado 128
6.5.2 Integración del SGBD en Nuestro Sistema 128
6.5.3 Diagrama E-R 129
6.6 DISEÑO DE LA INTERFAZ 130
6.6.1 Ventana de Login 130
6.6.2 Ventana de Registro de un nuevo usuario 131
6.6.3 Ventana User Home 133
6.6.4 Ventana del juego Demolition 134
6.6.5 Ventana del juego Arkanoid 135
6.6.6 Ventana de selección de la herramienta de edición. 136
6.6.7 Ventana de edición de contenido para el juego Demolition. 137
6.6.8 Ventana de edición de contenido para el juego Arkanoid. 138
6.6.9 Ventana Community 139
6.6.10 Ventana About 139
6.7 ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS 140

6.7.1 Pruebas Unitarias ... 140
6.7.2 Pruebas de Integración y del Sistema ... 145
6.7.3 Pruebas de Usabilidad y Accesibilidad ... 147
6.7.4 Pruebas de Rendimiento ... 149

CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA ... 151

7.1 LENGUAJES DE PROGRAMACIÓN ... 151
7.1.1 Java ... 151
7.1.2 JavaScript ... 151
7.2 LENGUAJES DE MARCADO ... 152
7.2.1 HTML5 ... 152
7.2.2 CSS3 ... 152
7.3 LIBRERÍAS ... 153
7.3.1 JQuery ... 153
7.3.2 JQuery-UI ... 153
7.3.3 Box2D ... 153
7.3.4 JQuery-Raty ... 153
7.3.5 JSON-Simple ... 153
7.3.6 Datatables ... 153
7.4 HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO ... 154
7.4.1 Eclipse ... 154
7.4.2 Spring MVC Framework ... 154
7.4.3 Servidor Tomcat ... 154
7.4.4 MySQL Workbench ... 155
7.5 CREACIÓN DEL SISTEMA ... 156
7.5.1 Problemas Encontrados ... 156
7.5.2 Descripción Detallada de las Clases ... 157

CAPÍTULO 8. DESARROLLO DE LAS PRUEBAS ... 171

8.1 PRUEBAS UNITARIAS ... 171
8.1.1 Resultados ... 171
8.2 PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA ... 172
8.3 PRUEBAS DE USABILIDAD Y ACCESIBILIDAD ... 174
8.3.1 Pruebas de Usabilidad ... 174
8.3.2 Pruebas de Accesibilidad ... 187
8.4 PRUEBAS DE RENDIMIENTO ... 195
8.4.1 Inicio de sesión ... 195
8.4.2 Cargar la pantalla de juego de un nivel ... 195
8.4.3 Guardar nivel ... 196
8.4.4 Cargar la ventana Community ... 196
8.4.5 Añadir nivel a la playlist de un jugador ... 196
8.4.6 Conclusión ... 197

CAPÍTULO 9. MANUALES DEL SISTEMA ... 199

9.1 MANUAL DE INSTALACIÓN ... 199
9.1.1 Instalación de MySQL: ... 200
9.1.2 Instalación del servidor Tomcat 7 ... 200
9.1.3 Instalación del PHPMyAdmin ... 201
9.1.4 Creación de las tablas de la base de datos ... 203
9.1.5 Instalar la aplicación web en el servidor tomcat ... 205
9.2 MANUAL DE EJECUCIÓN ... 208

9.3 MANUAL DE USUARIO 209
9.3.1 Acceso a la página web 209
9.3.2 User Home 211
9.3.3 Herramienta de edición 213
9.3.4 Community 215
9.4 MANUAL DEL PROGRAMADOR 216

CAPÍTULO 10. CONCLUSIONES Y AMPLIACIONES 217

10.1 CONCLUSIONES 217
10.2 AMPLIACIONES 218

CAPÍTULO 11. PRESUPUESTO 219

11.1 COSTES DE PERSONAL: 219
11.2 COSTES DE SOFTWARE Y LICENCIAS: 219
11.3 OTROS COSTES: 219
11.4 BENEFICIO: 220
11.5 RESUMEN DEL PRESUPUESTO: 220
11.6 PRESUPUESTO PARA EL CLIENTE 220

CAPÍTULO 12. REFERENCIAS BIBLIOGRÁFICAS 221

12.1 REFERENCIAS EN INTERNET 221

CAPÍTULO 13. APÉNDICES 223

13.1 CONTENIDO ENTREGADO 223
13.1.1 Contenidos 223
13.1.2 Instalación 224
13.2 CÓDIGO FUENTE 225
13.2.1 Paquete Views: 225
13.2.2 Paquete Controller 239
13.2.3 Paquete Business 248
13.2.4 Paquete Persistence 252
13.2.5 Paquete Model 262
13.2.6 Paquete Utils 272

Índice de Figuras

Figura 1 Sploder!: Editor de juegos 27
Figura 2 Sploder!: Pantalla de retos 28
Figura 3 Sploder!: Pantalla de edición de juegos de tipo puzle basados en físicas 28
Figura 4 MightyFingers: Pantalla de edición 30
Figura 5 MightyFingers: Pantalla de edición 2 30
Figura 6 Diagrama de Gantt, Planificación: Fase de Análisis 37
Figura 7 Diagrama de Gantt, Planificación: Fase de Diseño 38
Figura 8 Diagrama de Gantt, Planificación: Fase de Implementación 38
Figura 9 Diagrama de Gantt, Planificación: Fase de Implementación 39
Figura 10 Diagrama de Gantt, Planificación: Fase de Documentación 39
Figura 11 Diagrama de Gantt, Planificación: Fase de Defensa del Proyecto 39
Figura 12 Diagrama casos de uso: Acceso al sistema 47
Figura 13 Diagrama casos de uso: Navegabilidad 48
Figura 14 Diagrama casos de uso: User Home 49
Figura 15 Diagrama casos de uso: Community 50
Figura 16 Diagrama casos de uso: Juego Demolition, In Game 50
Figura 17 Diagrama casos de uso: Juego Demolition, Opciones 51
Figura 18 Diagrama casos de uso: Juego Arkanoid, In Game 52
Figura 19 Diagrama casos de uso: Juego Arkanoid, Opciones 53
Figura 20 Diagrama casos de uso: Editor de niveles, Demolition 54
Figura 21 Diagrama casos de uso: Editor de niveles, Arkanoid 56
Figura 22 Diagrama de clases preliminar del análisis 59
Figura 23 Diagrama de robustez 1: Login 73
Figura 24 Diagrama de robustez 2: Logout 74
Figura 25 Diagrama de robustez 3: Crear usuario 74
Figura 26 Diagrama de robustez 4: Ir al home de usuario 75
Figura 27 Diagrama de robustez 5: Ir al editor de niveles, "Demolition" 76
Figura 28 Diagrama de robustez 6: Ir al editor de niveles, "Arkanoid" 77
Figura 29 Diagrama de robustez 7: Ir a la página "Community" 77
Figura 30 Diagrama de robustez 8: Ir a la página "About" 78
Figura 31 Diagrama de robustez 9: Jugar nivel 78
Figura 32 Diagrama de robustez 10: Editar nivel 79
Figura 33 Diagrama de robustez 11: Añadir nivel a la playlist del usuario 80
Figura 34 Diagrama de robustez 12: Valorar nivel 81
Figura 35 Diagrama de robustez 13: Volver a jugar nivel 82
Figura 36 Diagrama de robustez 14: Volver al home del usuario 82
Figura 37 Diagrama de robustez 15: Rotar cañón hacia arriba 83
Figura 38 Diagrama de robustez 16: Rotar cañón hacia abajo 84
Figura 39 Diagrama de robustez 17: Disparar el cañón 84
Figura 40 Diagrama de robustez 18: Mover el stick del jugador hacia la derecha 85
Figura 41 Diagrama de robustez 19: Mover stick del jugador hacia la izquierda 86
Figura 42 Diagrama de robustez 20: Arrastrar pieza de la paleta a la cuadricula 87
Figura 43 Diagrama de robustez 21: Mover pieza por la cuadricula 88
Figura 44 Diagrama de robustez 22: Mover pieza fuera de la cuadricula 88
Figura 45 Diagrama de robustez 23: Eliminar pieza 89

FIGURA 46 DIAGRAMA DE ROBUSTEZ 24: ESCRIBIR EL NOMBRE DEL NIVEL 89
FIGURA 47 DIAGRAMA DE ROBUSTEZ 25: ESCRIBIR LA DESCRIPCIÓN DEL NIVEL 90
FIGURA 48 DIAGRAMA DE ROBUSTEZ 26: GUARDAR NIVEL 90
FIGURA 49 DIAGRAMA DE ROBUSTEZ 27: SELECCIONAR POSICIÓN DEL CAÑÓN....91
FIGURA 50 DIAGRAMA DE ROBUSTEZ 28: SELECCIONAR POSICIÓN DE LA LÍNEA LÍMITE DE ALTURA 92
FIGURA 51 DIAGRAMA DE ROBUSTEZ 29: SELECCIONAR EL NÚMERO DE PROYECTILES....92
FIGURA 52 BOCETO PRELIMINAR DE LA PÁGINA "LOGIN"....93
FIGURA 53 BOCETO PRELIMINAR DE LA PÁGINA "REGISTER" 94
FIGURA 54 BOCETO PRELIMINAR DE LA PÁGINA "USER HOME" 95
FIGURA 55 BOCETO PRELIMINAR DE LA PÁGINA "GAME"....96
FIGURA 56 BOCETO PRELIMINAR DE LA PÁGINA "LEVEL EDITOR"....97
FIGURA 57 BOCETO PRELIMINAR DE LA PÁGINA "COMMUNITY"....98
FIGURA 58 BOCETO PRELIMINAR DE LA PÁGINA "ABOUT"....99
FIGURA 59 DIAGRAMA DE NAVEGABILIDAD, ACCESO A LA PÁGINA WEB 100
FIGURA 60 DIAGRAMA DE NAVEGABILIDAD, DENTRO DE LA PÁGINA WEB 100
FIGURA 61 DIAGRAMA DE PAQUETES DE LA APLICACIÓN 105
FIGURA 62 DIAGRAMA DE COMPONENTES 107
FIGURA 63 DIAGRAMA DE DESPLIEGUE 110
FIGURA 64 DIAGRAMA DE CLASES, DISEÑO 111
FIGURA 65 PAQUETE UTILS 112
FIGURA 66 PAQUETE PERSISTENCE 112
FIGURA 67 PAQUETE BUSINESS 113
FIGURA 68 PAQUETE CONTROLLER 113
FIGURA 69 PAQUETE LEVEL EDITOR 114
FIGURA 70 PAQUETE COMMUNITY 114
FIGURA 71 PAQUETE MENU 114
FIGURA 72 PAQUETE USER HOME 115
FIGURA 73 PAQUETE INTERFACES 115
FIGURA 74 PAQUETE MODEL 116
FIGURA 75 DIAGRAMA DE SECUENCIA 1: CASOS DE USO 1 Y 4 117
FIGURA 76 DIAGRAMA DE SECUENCIA 2: CASOS DE USO 2 118
FIGURA 77 DIAGRAMA DE SECUENCIA 3: CASOS DE USO 3 118
FIGURA 78 DIAGRAMA DE SECUENCIA 4: CASOS DE USO 5 Y 6 119
FIGURA 79 DIAGRAMA DE SECUENCIA 5: CASOS DE USO 7 119
FIGURA 80 DIAGRAMA DE SECUENCIA 6: CASOS DE USO 8 120
FIGURA 81 DIAGRAMA DE SECUENCIA 7: CASOS DE USO 9 120
FIGURA 82 DIAGRAMA DE SECUENCIA 8: CASOS DE USO 10 121
FIGURA 83 DIAGRAMA DE SECUENCIA 9: CASOS DE USO 13 121
FIGURA 84 DIAGRAMA DE SECUENCIA 10: CASOS DE USO 11 122
FIGURA 85 DIAGRAMA DE SECUENCIA 12: CASOS DE USO 14 123
FIGURA 86 DIAGRAMA DE SECUENCIA 13: CASOS DE USO 26 124
FIGURA 87 DIAGRAMA DE ACTIVIDAD 1 125
FIGURA 88 DIAGRAMA DE ACTIVIDAD 2 126
FIGURA 89 DIAGRAMA DE ACTIVIDAD 3 127
FIGURA 90 DIAGRAMA ENTIDAD-RELACIÓN DE LA BASE DE DATOS 129
FIGURA 91 INTERFACES: VENTANA LOGIN 130
FIGURA 92 INTERFACES: VENTANA ERROR 1 131
FIGURA 93 INTERFACES: VENTANA REGISTER 131
FIGURA 94 INTERFACES: VENTANA ERROR 2 132
FIGURA 95 INTERFACES: VENTANA ERROR 3 132

FIGURA 96 INTERFACES: VENTANA USER HOME 133
FIGURA 97 INTERFACES: VENTANA DE JUEGO DEMOLITION 1 134
FIGURA 98 INTERFACES: VENTANA DE JUEGO DEMOLITION 2 134
FIGURA 99 INTERFACES: VENTANA DE JUEGO ARKANOID 1 135
FIGURA 100 INTERFACES: VENTANA DE JUEGO ARKANOID 2 135
FIGURA 101 INTERFACES: VENTANA DE SELECCIÓN DE LA HERRAMIENTA DE EDICIÓN 136
FIGURA 102 INTERFACES: VENTANA DE EDICIÓN DE CONTENIDO PARA EL JUEGO DEMOLITION 137
FIGURA 103 INTERFACES: VENTANA DE EDICIÓN DE CONTENIDO PARA EL JUEGO ARKANOID 138
FIGURA 104 INTERFACES: VENTANA COMMUNITY 139
FIGURA 105 INTERFACES: VENTANA ABOUT 139
FIGURA 106 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO: INICIO DE SESIÓN 195
FIGURA 107 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO: CARGAR LA PANTALLA DE JUEGO DE UN NIVEL 195
FIGURA 108 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO: GUARDAR NIVEL 196
FIGURA 109 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO: CARGAR LA VENTANA COMMUNITY 196
FIGURA 110 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO: AÑADIR NIVEL A LA PLAYLIST DE UN JUGADOR 196
FIGURA 111 MANUAL DE INSTALACIÓN: CONTRASEÑA DEL USUARIO ROOT DE MYSQL 200
FIGURA 112 MANUAL DE INSTALACIÓN: SERVIDOR TOMCAT 201
FIGURA 113 MANUAL DE INSTALACIÓN: PHPMYADMIN 201
FIGURA 114 MANUAL DE INSTALACIÓN: PHPMYADMIN 2 202
FIGURA 115 MANUAL DE INSTALACIÓN: PHPMYADMIN 3 202
FIGURA 116 MANUAL DE INSTALACIÓN: IMPORTAR TABLAS DE LA BBDD 203
FIGURA 117 MANUAL DE INSTALACIÓN: IMPORTAR TABLAS DE LA BBDD 2 203
FIGURA 118 MANUAL DE INSTALACIÓN: IMPORTAR TABLAS DE LA BBDD 3 203
FIGURA 119 MANUAL DE INSTALACIÓN: IMPORTAR TABLAS DE LA BBDD 4 204
FIGURA 120 MANUAL DE INSTALACIÓN: IMPORTAR TABLAS DE LA BBDD 5 204
FIGURA 121 MANUAL DE INSTALACIÓN: IMPORTAR TABLAS DE LA BBDD 6 204
FIGURA 122 MANUAL DE INSTALACIÓN: IMPORTAR TABLAS DE LA BBDD 7 204
FIGURA 123 MANUAL DE INSTALACIÓN: FTP 205
FIGURA 124 MANUAL DE INSTALACIÓN: FTP 2 205
FIGURA 125 MANUAL DE INSTALACIÓN: FTP 3 206
FIGURA 126 MANUAL DE INSTALACIÓN: FTP 4 206
FIGURA 127 MANUAL DE INSTALACIÓN: APLICACIÓN WEB 207
FIGURA 128 MANUAL DE EJECUCIÓN: LOGIN EN EL SISTEMA 208
FIGURA 129 MANUAL DE EJECUCIÓN: ARRANQUE DEL SERVIDOR 208
FIGURA 130 MANUAL DE USUARIO: PANTALLA DE LOGIN 209
FIGURA 131 MANUAL DE USUARIO: PANTALLA DE REGISTRO 210
FIGURA 132 MANUAL DE USUARIO: HOME DEL USUARIO 211
FIGURA 133 MANUAL DE USUARIO: JUEGO DEMOLITION 212
FIGURA 134 MANUAL DE USUARIO: JUEGO ARKANOID 212
FIGURA 135 MANUAL DE USUARIO: EDITOR DE NIVELES DEMOLITION 213
FIGURA 136 MANUAL DE USUARIO: EDITOR DE NIVELES ARKANOID 214
FIGURA 137 MANUAL DE USUARIO: COMMUNITY 215

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La motivación que tiene el desarrollo de este proyecto es el poder aplicar y perfeccionar los conocimientos adquiridos durante los dos años de master y combinarlos con la idea de desarrollar un proyecto vinculado a los videojuegos y el mundo web, dos campos en los que estoy muy interesado.

Con este proyecto pretendo adentrarme un poco más en el desarrollo de proyectos vinculados a los videojuegos y aplicaciones web, profundizar más en las posibilidades que ofrece el mundo web para aplicaciones orientadas al ocio y los videojuegos.

El principal objetivo de este proyecto es construir una plataforma en la que no solo se pueda jugar, sino también se pueda crear. Esta aplicación va destinada a todo aquel que le guste jugar y tenga ganas de ir un paso más allá, siendo parte de la comunidad de creadores de contenido de esta plataforma, aportando sus creaciones para el disfrute de todos los jugadores.

La aplicación propuesta no está concebida inicialmente para fines comerciales, ya que el objetivo de este proyecto es meramente educativo, aunque no se descarta que en un futuro pueda llegar a convertirse en algún producto comercial.

Proyectos como este ya se está llevando a cabo hoy en día en el ámbito de los videojuegos, como por ejemplo en el juego "***Little Big Planet***" de la plataforma Playstation 3, un videojuego de plataformas que incluye una herramienta de creación de niveles para que los jugadores aporten sus creaciones a la comunidad. La idea para realizar el proyecto que se presenta en este documento viene gran parte inspirada por este tipo de ideas.

1.2 Resumen de Todos los Aspectos

1.2.1 Memoria del proyecto

Resumen la motivación del proyecto, los objetivos y el alcance del mismo. También contiene una breve descripción de cada uno de los capítulos de la documentación.

1.2.2 Introducción

Se hace una pequeña introducción al proyecto justificando su realización, se describen los objetivos que se quieren alcanzar con dicho proyecto y se estudia la situación actual en la que se encuentra el ámbito de esta aplicación, incluyendo también el estudio de sistemas similares al presentado en el proyecto.

1.2.3 Aspectos Teóricos

Descripción de las tecnologías utilizadas en el proyecto, terminología del mismo y aspectos de interés referente al proyecto o su ámbito.

1.2.4 Planificación del Proyecto y resumen de presupuestos

Resumen de la planificación seguida para la construcción del proyecto y resumen del presupuesto del mismo.

1.2.5 Análisis

Análisis preliminar del proyecto en el que se determina un primer esbozo de lo que va a ser el proyecto, en él se describen las funcionalidades que ha de tener, los sistemas que lo compondrán y un boceto de lo que podrían ser las interfaces de usuario del sistema. Este análisis sirve como base para realizar el posterior diseño del proyecto.

1.2.6 Diseño del sistema

En esta sección se modela todo lo descrito anteriormente en análisis, se da una estructura a todo lo descrito en la sección anterior ya profundizando en que forma tendrán los distintos sistemas que integran el proyecto y que funcionalidad tendrá cada uno.

1.2.7 Implementación del sistema

Apartado en el que se cuentan aspectos referentes a la construcción del sistema, las dificultades encontradas en dicha construcción y que soluciones se han tomado para paliar estos problemas. También se comentan cosas como las herramientas usadas para desarrollar el proyecto y el lenguaje de programación utilizado.

1.2.8 Desarrollo de pruebas

Descripción de las pruebas realizadas en el sistema y los resultados obtenidos de las mismas.

1.2.9 Manuales del sistema

Compendio de manuales en los que se da toda la información referente a la instalación, ejecución, manejo y pautas para otros desarrolladores interesados en ampliar el proyecto.

1.2.10 Conclusiones y ampliaciones

Resumen que incluye las conclusiones e impresiones acerca de la realización del proyecto, el producto final realizado y los objetivos alcanzados con dicho proyecto. Este capítulo incluye una serie de propuestas de posibles ampliaciones que podrían añadirse al proyecto.

1.2.11 Presupuesto

Sección en la que se detalla el presupuesto necesario para realizar el proyecto.

1.2.12 Referencias bibliográficas

Listado de todas los sitios de los que se ha obtenido información que ha ayudado en la construcción de este proyecto.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Este proyecto se desarrolla con el fin de crear un espacio en el que no solo se pueda jugar, sino que también se pueda crear. Con la herramienta de edición de contenido para la plataforma de juegos se intenta dotar de longevidad al sistema, incitando a los jugadores a crear contenido para los videojuegos presentes y con ello hacer crecer la plataforma. La plataforma presentada no solo proporciona diversión sino que también proporciona una manera sencilla de ampliar dicha diversión con la herramienta de edición de contenido. Esta herramienta tiene un sencillo manejo, muy intuitivo y accesible para que todo el mundo pueda aportar sus creaciones.

Esta plataforma es similar a otras que existen en la red, que ofrecen juegos online, pero esta tiene la particularidad que permite a los usuarios crear los contenidos que pueden jugar y compartir.

2.2 Objetivos del Proyecto

Los objetivos que se quieren alcanzar en el desarrollo de este proyecto son los listados a continuación:

Objetivos generales de la plataforma:

1. Diseñar y construir una plataforma web que permita jugar con una serie de videojuegos y permita crear niveles para dichos videojuegos.
2. Construir un motor de juegos que permita mover todos los niveles creados por la comunidad.
3. Construir una aplicación que permita crear niveles para los juegos presentados en el portal.

Objetivos de la página web donde están alojados los videojuegos y la herramienta de edición:

1. El portal web ha de gestionar usuarios: Crear usuarios, Iniciar y Cerrar sesión.
2. Los niveles que se creen y que sean compartidos han de guardarse en el servidor del portal en algún tipo de formato (JSON, XML, texto).
3. Todos los niveles que se hayan creado dentro de la plataforma han de estar a disposición de todos los jugadores.
4. Se ha de crear un sistema para que los motores de los videojuegos puedan leer los ficheros donde se han guardado los niveles creados y poder reproducirlos.

Objetivos generales referentes a los videojuegos:

1. Utilizar una librería para simular físicas dentro del juego.
2. Todos los niveles creados por los usuarios han de poder reproducirse.
3. Ambos juegos han de contabilizar la puntuación obtenida por el jugador.

Objetivos particulares del videojuego Demolition

1. El juego tendrá un cañón que ha de poder girarse 180º.
2. El cañón ha de disparar los proyectiles en la dirección indicada por el jugador.
3. El juego ha de marcar una línea límite que defina la altura máxima para completar el nivel.
4. Las piezas que conforman la estructura han de perder derribarse con los proyectiles lanzados por el cañón.

Objetivos particulares del videojuego Arkanoid

1. El juego tendrá una plataforma o paleta que ha de moverse horizontalmente (Izquierda y Derecha).
2. El juego tendrá una bola que destruirá los bloques y esta empezara a moverse cuando el jugador pulse la tecla indicada.
3. Los bloques han de desaparecer al entrar en contacto con la bola.
4. El juego ha de terminar la partida si la bola alcanza el borde inferior de la pantalla.

Objetivos referentes a la herramienta de edición:

1. La herramienta ha de presentar una cuadricula que conforme la base donde se colocaran las piezas para crear el nivel.
2. La herramienta ha de mostrar una lista de piezas que los usuarios pueden usar.
3. Las piezas han de poder arrastrarse con el ratón dentro de la cuadricula para construir los niveles.
4. Los niveles han de poder guardarse y compartirse.
5. Los niveles han de poder ser editados por el jugador que los ha creado.

2.3 Estudio de la Situación Actual

En la actualidad existen muchas páginas web que ofrecen una amplia lista de videojuegos o juegos Flash a los que se puede jugar de forma inmediata, solo con entrar y registrarse. Estas webs ofrecen una amplia lista de juegos de todo tipo agrupados por géneros, pero dichas webs solo están centradas en los juegos.

También existen otras webs similares a este proyecto, pero dichas webs utilizan aplicaciones Flash para mover los juegos y presentar los creadores de contenido. Estas webs también permiten al igual que este proyecto, crear contenido para varios tipos de videojuegos como pueden ser: Plataformas, Puzzles o Shooter Clasicos. Pero la diferencia con la web presentada por este proyecto es, que este tipo de plataformas permiten crear y compartir juegos independientes y no contenido para unos juegos concretos.

La forma en la que estas plataformas presentan sus editores de contenido han servido de inspiración para la presentada por este documento, ofrecen una paleta de elementos que se han de colocar en una cuadricula que será el espacio designado como el escenario del juego.

Estos editores aparte de crear la parte visual del juego también permiten crear las mecánicas del mismo, cosa que este proyecto no permite, pero tampoco es el objetivo que persigue.

La tecnología en la que suelen estar implementadas este tipo de herramientas suelen ser por lo general Flash, ya que están orientadas plenamente a la web, sin embargo para este proyecto se ha elegido las tecnologías HTML5, JavaScript y JQuery, que proporcionan la misma potencia que las tecnologías Flash y son totalmente abiertas y llegan a mayor parte del público, sin necesidad de algún tipo de plug-in o software concreto para su funcionamiento.

2.3.1 Evaluación de Alternativas

A continuación se evaluaran algunas de las plataformas y webs que hay en el mercado y que ofrecen experiencias similares a la presentada en este proyecto.

2.3.1.1 Sploder.com

Web oficial: http://www.sploder.com/

Este sitio web permite a los usuarios crear y compartir videojuegos mediante una herramienta flash muy potente con características de edición bastante avanzadas.

Una de las principales características que presenta este sitio web es la variedad de estilos de juegos que se pueden crear dentro de la plataforma web, como por ejemplo: Juegos Arkade con estilo retro, juegos de plataformas, juegos de puzles basados en físicas, shooters clásicos y juegos de ingenio.

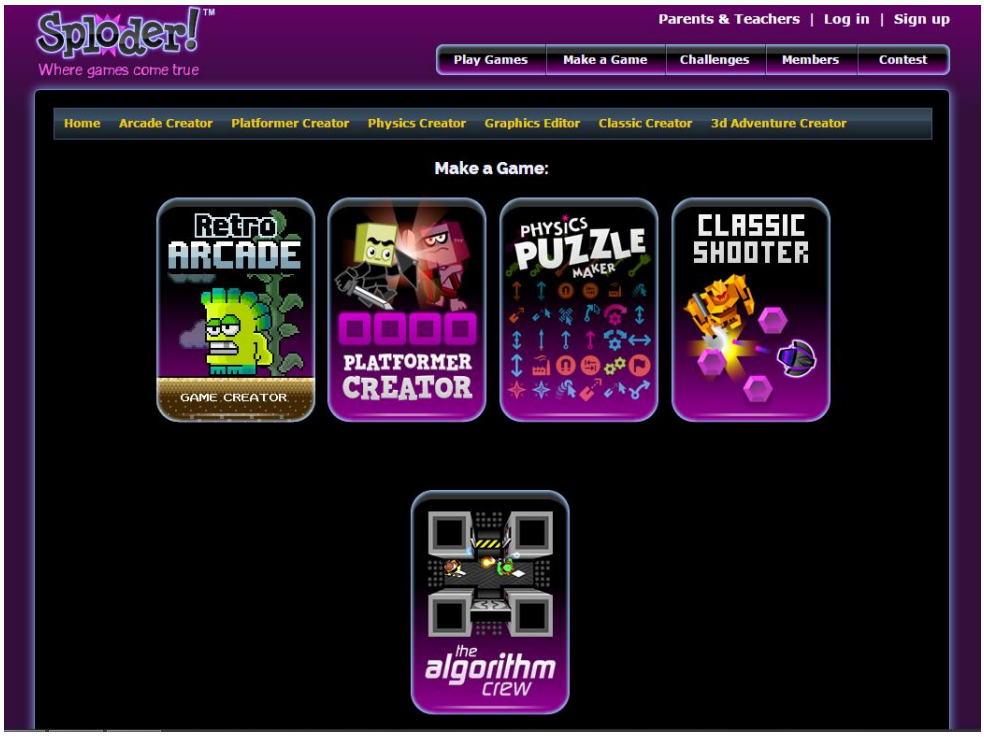


Figura 1 Sploder!: Editor de juegos

Otra de las características que presenta esta web es la posibilidad de competir entre jugadores dentro de los juegos creados mediante retos. Un jugador crea un juego y pone un reto dentro del juego, como por ejemplo conseguir x puntuación y el resto de jugadores tienen que conseguir llegara esa puntuación. Esta es una característica muy interesante.

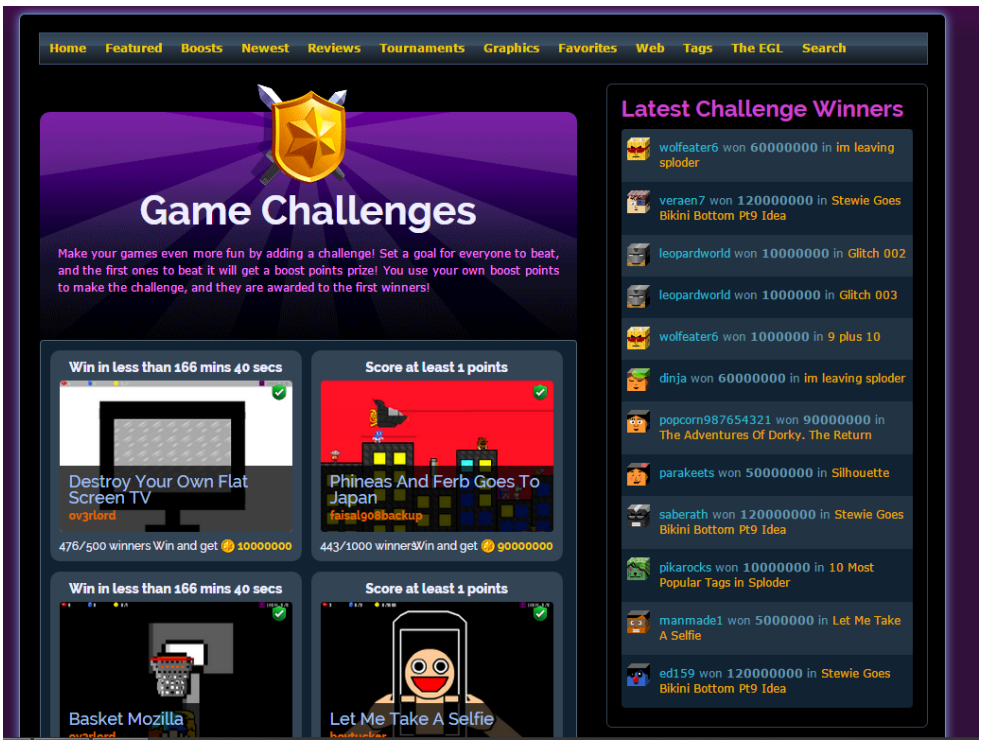


Figura 2 Sploder!: Pantalla de retos

A continuación analizamos la herramienta de edición, más concretamente la herramienta de edición de puzles basados en físicas ya que es la herramienta más similar a la que se presenta en este proyecto:

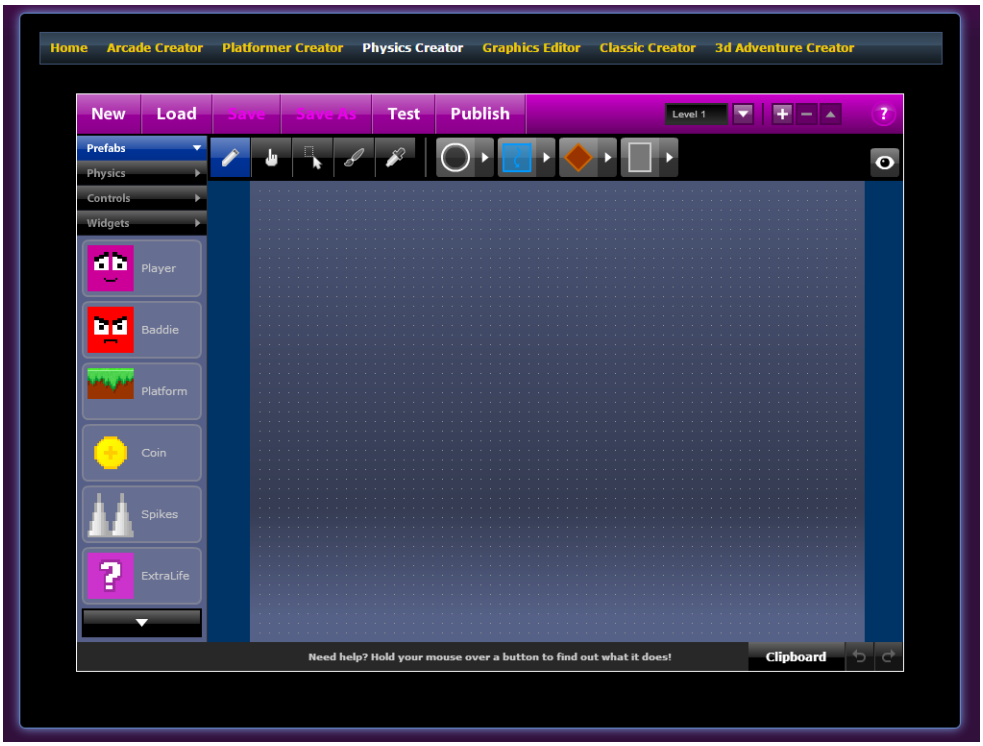


Figura 3 Sploder!: Pantalla de edición de juegos de tipo puzle basados en físicas

En la pantalla de edición que nos presenta esta página web podemos ver dos partes claramente diferenciadas: la parte izquierda donde se encuentran los elementos que van a componer el nivel y la parte central derecha donde se sitúa una cuadricula para colocar los elementos.

En la paleta de elementos encontramos varias secciones donde podemos encontrar elementos de todo tipo:

- **Elementos de construcción:** Elementos que conforman el juego como por ejemplo: plataformas, personajes, elementos coleccionables, etc.
- **Elementos de físicas:** Elementos que se aplican a los objetos de construcción para que se comporten de determinadas maneras: rotaciones, movimientos, etc.
- **Elementos de control:** Elementos que sirven para agregar controles a los objetos.
- **Widgets:** Elementos prefabricados y de control que se pueden añadir a los elementos.

El modo de crear el juego es bastante sencillo, solo basta con arrastrar los distintos elementos de la paleta a la cuadricula central. Alguno de estos elementos hay que arrastrarlos dentro de otros elementos para que adquieran propiedades especiales.

En la parte superior de la cuadricula encontramos una herramientas que sirven para dibujar formas en la cuadricula que pueden actuar como plataformas, o herramientas para arrastrar y seleccionar los elementos de la pantalla.

<u>Diferencias entre Splouder y este proyecto:</u>

- Spluder es un editor de juegos completos mientras que este proyecto solo tiene editores de contenido para los juegos que presenta.
- Splouder tiene un sistema para que los usuarios se reten entre ellos.

Ventajas:

1. Es un sistema muy potente.
2. Permite crear juegos completos y complejos.

Desventajas:

1. Es complejo y difícil de usar.
2. No hay ayuda.

2.3.1.2 MightyFingers.com

Web oficial: http://mightyfingers.com/

Este es otro sitio web que permite crear videojuegos utilizando tecnología HTML5. Esta web presenta una herramienta muy avanzada, muy parecida a las herramientas profesionales de creación de videojuegos. Dicha herramienta permite crear todo tipo de videojuegos y de una complejidad muy alta.

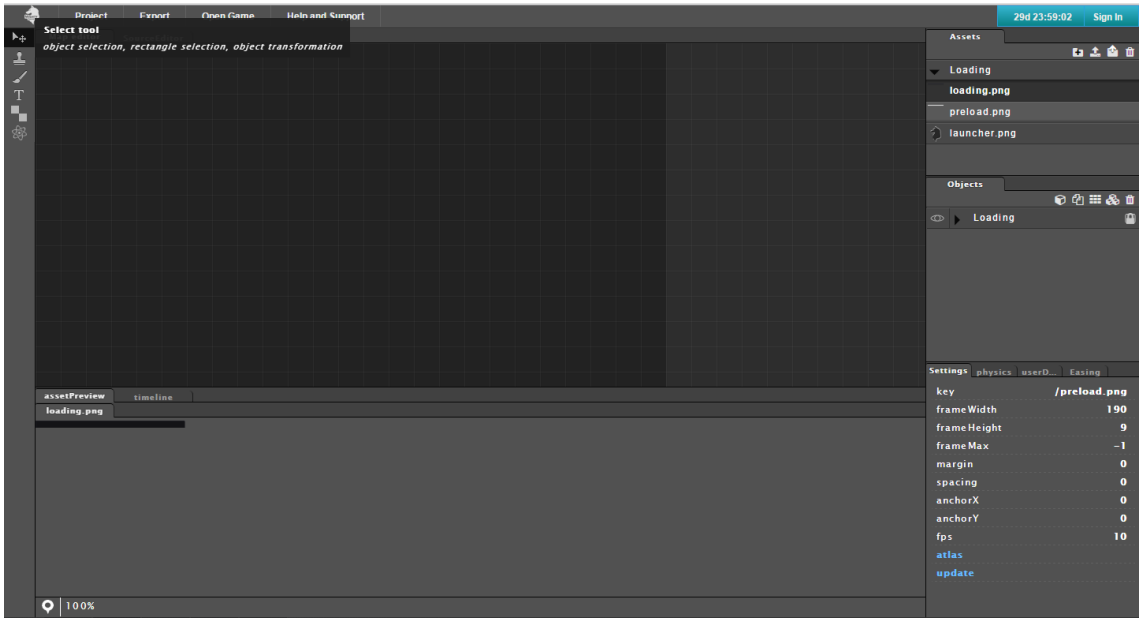


Figura 4 MightyFingers: Pantalla de edición

Esta herramienta permite importar contenidos, distribuir por capas los distintos elementos del juego, incluso programación avanzada mediante código.

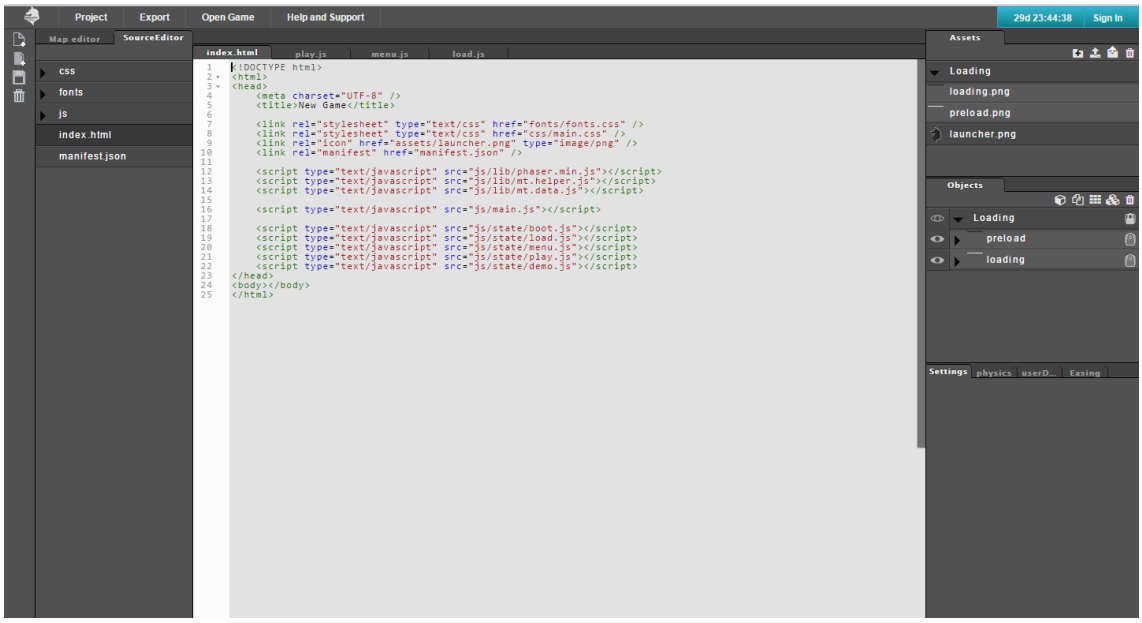


Figura 5 MightyFingers: Pantalla de edición 2

Diferencias entre MightyFingers y este proyecto:

- MightyFingers es un editor de juegos completos mientras que este proyecto solo tiene editores de contenido para los juegos que presenta.
- MightyFingers permite programar los juegos mediante código.
- Esta herramienta no posee un sistema de compartición de juegos con otros usuarios.

Ventajas:

1. Es un sistema muy potente.
2. Permite crear juegos completos y complejos.

Desventajas:

1. Es muy complejo y difícil de usar.
2. Requiere conocimientos de programación.
3. No hay ayuda.

Capítulo 3. Aspectos Teóricos

En esta sección se explican algunos términos y tecnologías utilizadas en el proyecto.

3.1 Box2D

Box2D es una biblioteca libre que implementa un motor físico en dos dimensiones. Está programada en C++ por **Erin Catto**, y se distribuye bajo la licencia zlib.

Esta librería lo que permite es simular un sistema de físicas real en dos dimensiones. Esto quiere decir que podremos crear cuerpos y hacer que se comporten de una manera similar a la realidad en un entorno controlado. Se le podrán aplicar fuerzas, pesos y velocidades a esos cuerpos, podrán interactúan unos con otros mediante colisiones, uniones, etc.

En esta página web se puede consultar información más técnica y detallada de la librería Box2D:

http://box2d.org/documentation.html

3.2 Arkanoid

Es el nombre de un videojuego muy popular desarrollado por la compañía japonesa Taito en 1986 y que está basado en las mecánicas de un videojuego de los años 70 de la compañía norteamericana Atari llamado Breackout.

Arkanoid y Breakout son videojuegos que se describen de la siguiente manera:

En la parte inferior de la pantalla una barra simula una raqueta de front-tenis que el jugador puede desplazar de izquierda a derecha. En la parte superior se sitúa una banda conformada por rectángulos que simulan ser ladrillos. Una pelotita desciende de la nada y el jugador debe golpearla con la raqueta, entonces la pelota asciende hasta pegar en el muro, los ladrillos tocados por la pelota desaparecen. La pelota vuelve a descender y así sucesivamente. El objetivo del juego es terminar con la pared de ladrillos.

El termino Arkanoid con el paso de los años no solo se usaba para dar nombre a este conocido videojuego, sino que también se utiliza de forma extraoficialmente para dar nombre al género al que pertenecen multitud de juegos clónicos y similares a este.

3.3 Playlist

Playlist o lista de reproducción, es un término utilizado generalmente para referirse a listas de canciones que puedes tener en un reproductor. Se ha escogido este término para definir uno de los componentes de la aplicación por dos razones: la primera porque es un concepto similar a la que se quiere representar y segundo porque la palabra "Play" en inglés se puede traducir como jugar, concepto que encaja muy bien en la plataforma ya que está destinada a juegos.

Este término se utiliza dentro del proyecto para designar a una de las partes que componen la zona personal de un usuario dentó de la aplicación. Esta parte hace referencia a la lista de niveles que un usuario ha seleccionado de la lista global de niveles publicados en la página.

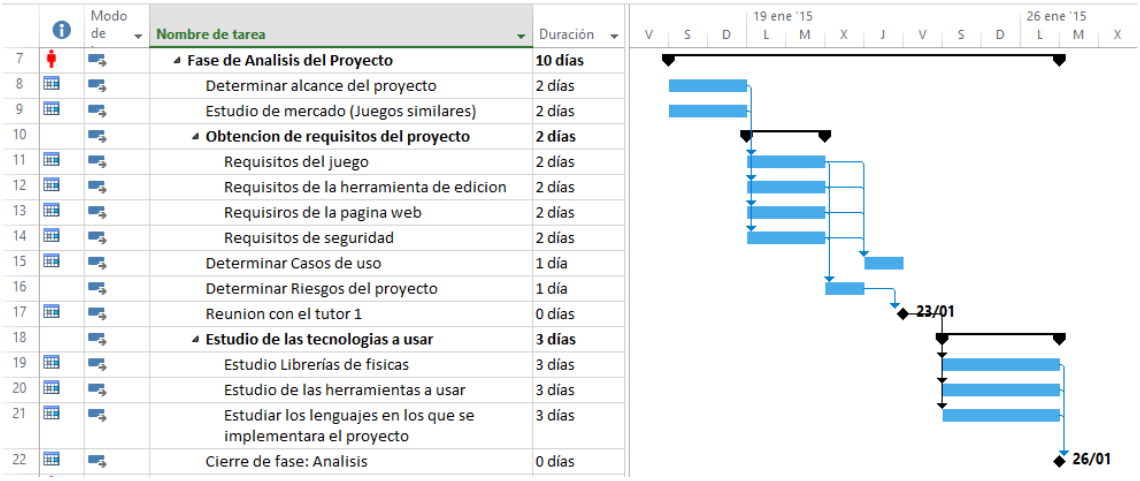
Cada usuario tendrá una playlist con una selección de niveles que podrá jugar y puntuar.

Capítulo 4. Planificación del Proyecto y Resumen de Presupuesto

4.1 Planificación

En esta sección se incluye un diagrama dividido por secciones de la planificación seguida en este proyecto.

El proyecto comienza con una fase de análisis e investigación de la tecnologías que se van a utilizar para desarrollar la aplicación. Esta fase abarca un tercio del primer mes de desarrollo.



	Modo de	Nombre de tarea	Duración
7		**Fase de Analisis del Proyecto**	**10 días**
8		Determinar alcance del proyecto	2 días
9		Estudio de mercado (Juegos similares)	2 días
10		**Obtencion de requisitos del proyecto**	**2 días**
11		Requisitos del juego	2 días
12		Requisitos de la herramienta de edicion	2 días
13		Requisiros de la pagina web	2 días
14		Requisitos de seguridad	2 días
15		Determinar Casos de uso	1 día
16		Determinar Riesgos del proyecto	1 día
17		Reunion con el tutor 1	0 días
18		**Estudio de las tecnologias a usar**	**3 días**
19		Estudio Librerías de fisicas	3 días
20		Estudio de las herramientas a usar	3 días
21		Estudiar los lenguajes en los que se implementara el proyecto	3 días
22		Cierre de fase: Analisis	0 días

Figura 6 Diagrama de Gantt, Planificación: Fase de Análisis

En la siguiente semana tras realizar el análisis se realiza la fase de diseño del proyecto, el tiempo en esta fase se reduce frente a la anterior porque gran parte del trabajo de diseño ha sido reutilizado de trabajos anteriores a la realización de este proyecto, por lo que permite ganar tiempo en la planificación para fases más largas como pueden ser la de Implementación o la fase de Documentación.

Esta fase abarca el diseño de las interfaces y de la página web, el diseño de las mecánicas de juego y los motores, el diseño de la herramienta de edición de niveles y todos sus componentes, el diseño de la arquitectura de la aplicación y de la base de datos.

	Modo de	Nombre de tarea	Duración
23		**Fase de Diseño del proyecto**	**6 días**
24		**Diseño de interfaces**	**2 días**
25		Diseño interfaz Pagina Principal	2 días
26		Diseño interfaz Pagina de Juego	2 días
27		Diseño interfaz Pagina de Edicion	2 días
28		Diseño interfaz Pagina perfil de usuario	2 días
29		Reunion con el tutor 2	0 días
30		**Diseño de los juegos**	**3 días**
31		Diseño de las mecanicas de juego	3 días
32		Diseño del motor del juego	3 días
33		**Diseño de la herramienta de edicion**	**3 días**
34		Diseño de los elementos de edicion	3 días
35		Diseño de la mecanica de edicion	3 días
36		**Diseño de la arquitectura web**	**3 días**
37		Diseño de la estructura del sitio web	3 días
38		Diseño de la base de datos	1 día
39		Reunion con el tutor 3	0 días
40		Cierre de fase: Diseño	0 días

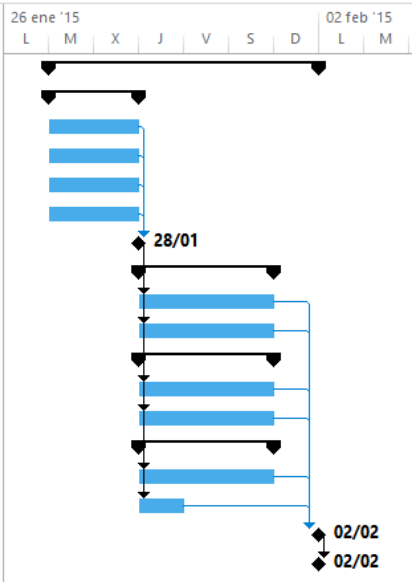


Figura 7 Diagrama de Gantt, Planificación: Fase de Diseño

Tras la fase de diseño llega la fase más larga del proyecto, la fase de implementación, que se alargara durante los próximos tres meses de desarrollo. Esta fase abarca la implementación de los motores de los videojuegos, la herramienta de edición de niveles, las páginas que integran el portal web y la base de datos.

	Modo de	Nombre de tarea	Duración
41		**Fase de Implementación del Proyecto**	**89 días**
42		**Implementacion del juego**	**25 días**
43		Implementacion motor de juego	25 días
44		Implementacion de las mecanicas de juego	25 días
45		**Implementacion herramienta de edicion de niveles**	**40 días**
46		Implementacion de los elementos de edicion	20 días
47		Implementacion de las mecanicas de edicion	20 días
48		Implementacion de paso de mensajes (Juego - Editor)	10 días
49		Implementacion de un parser de mensajes	10 días
50		Reunion con el tutor 4	0 días
51		**Implementacion del portal web**	**17 días**
52		Implementacion Pagina Principal	17 días
53		Implementacion Pagina de Juego	17 días
54		Implementacion Pagina de Edicion	17 días
55		Implementacion Pagina perfil de usuario	17 días
56		Implementacion del modulo de Login	17 días
57		Implementacion de CSS	17 días
58		**Implementacion de la Base de datos**	**7 días**
59		Implementacion e insercion de datos en la BBDD	7 días
60		Reunion con el tutor 5	0 días
61		Cierre de fase: Implementacion	0 días
62		**Fase de Pruebas**	**7 días**

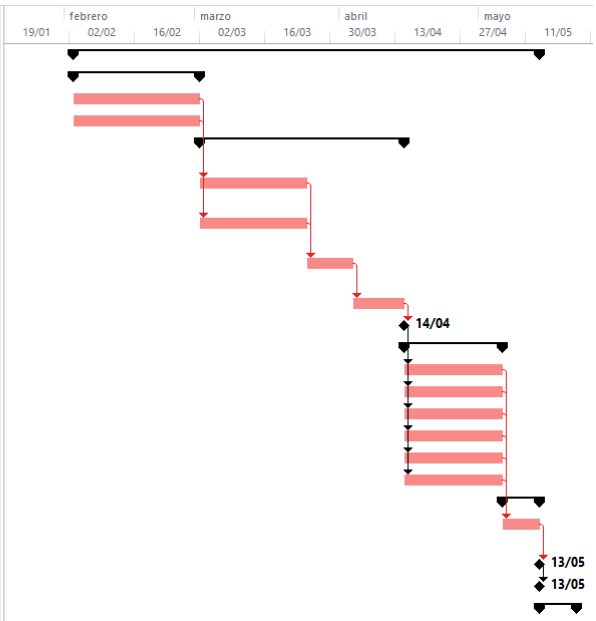


Figura 8 Diagrama de Gantt, Planificación: Fase de Implementación

Tras la fase de implementación viene una fase muy corta de pruebas, en esta fase se realizaran todas las pruebas necesarias para comprobar que el proyecto

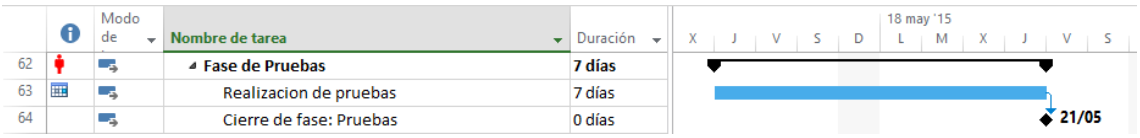


Figura 9 Diagrama de Gantt, Planificación: Fase de Implementación

La fase de documentación abarca todo el desarrollo del proyecto, esta fase, se realiza paralelamente a las fases de análisis, diseño, implementación y pruebas.

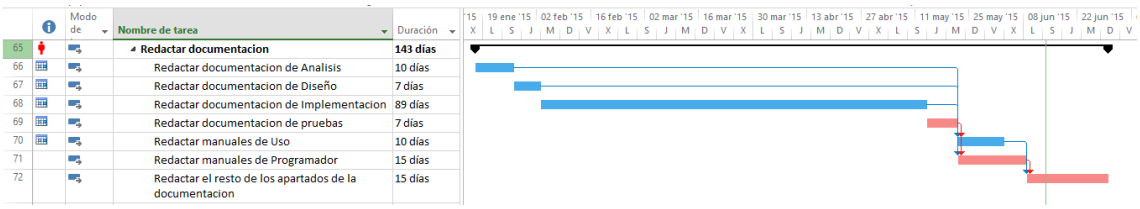


Figura 10 Diagrama de Gantt, Planificación: Fase de Documentación

La última fase del proyecto es la propia defensa del proyecto una vez finalizado. Esta fase abarca la última semana del proyecto que estará principalmente dedicada a hacer y ensayar la defensa del mismo.

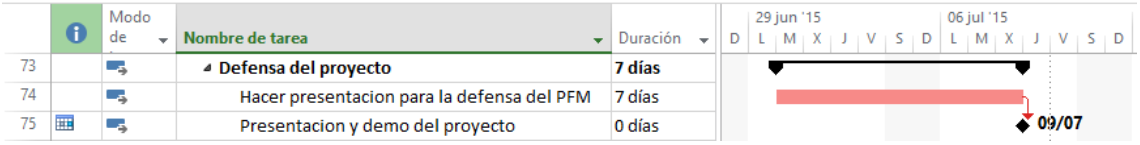


Figura 11 Diagrama de Gantt, Planificación: Fase de Defensa del Proyecto

4.2 Resumen del Presupuesto

Esta tabla representa un resumen a grandes rasgos del presupuesto necesario para acometer el proyecto.

Item	SubItem	Concepto	Cantidad	Precio Unitario	Total
1		*Desarrollo de la aplicación*			**51000 €**
	1.1	Análisis	80	25 €	2000 €
	1.2	Diseño	48	25 €	1200 €
	1.3	Implementación	712	25 €	17800 €
	1.4	Documentación	1144	25 €	28600 €
	1.5	Pruebas	56	25 €	1400 €
2		*Software y Licencias*			**0 €**
	2.1	Microsoft Windows 8.1	1		0 €
	2.2	MySQL	1		0 €
3		*Otros*			**5400 €**
	3.1	Amortización del hardware	1	2000 €	2400 €
	3.2	Beneficios	1	3000 €	3000 €
				Subtotal:	56400 €
				IVA (21%)	11844 €
				Total:	**68244 €**

Capítulo 5. Análisis

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

El punto hasta el que está previsto llegar en este proyecto es tener una web completamente funcional en la que se puedan jugar a dos juegos distintos y se puedan crear contenido para esos dos juegos. También es necesario que estos contenidos puedan ser compartidos con el resto de usuarios de la comunidad.

Los dos juegos creados han de ser: un juego de demolición de estructuras y un juego del genero Arkanoid. Ambos juegos han de ser capaces de leer el contenido creado por los usuarios y han de generar automáticamente el nivel según la información leída.

La herramienta de creación de contenido para dichos juegos tiene que ser capaz de guardar la información del nivel creado por el usuario en un fichero JSON para posteriormente ser leída por los motores que moverán los juegos.

Los contenidos creados han de estar disponibles para todos los jugadores, por lo que la aplicación ha de poder guardar dichos contenidos en alguna base de datos para que pueda ser accesible al resto de usuarios de la comunidad.

5.2 Requisitos del Sistema

Se desea crear una página web que integre videojuegos con algún tipo de herramienta para crear contenido para dichos videojuegos. También se desea que los contenidos creados puedan ser compartidos con todos los usuarios de la aplicación, de manera que sean los propios usuarios los que amplíen los contenidos de los videojuegos presentados mediante sus creaciones.

Los jugadores dispondrán de un espacio personal donde podrán encontrar los contenidos que ellos mismos han creado y los contenidos que han seleccionado para jugar de otros usuarios, tendrán una serie de estadísticas como por ejemplo: la puntuación alcanzada o si han superado el nivel o no. También dispondrán de un mecanismo para valorar los contenidos de los demás jugadores.

Requisitos generales de la página web:

- El portal web tiene que poder visualizarse en los principales navegadores: Internet Explorer, Google Chrome, Firefox, Opera y Safari.
- El portal tiene que gestionar usuarios:
    - Login/Logout de usuarios registrados.
    - Creación de nuevos usuarios.
- El portal ha de gestionar los contenidos creados por los usuarios:
    - Un usuario pude subir contenidos creados con la herramienta de edición de niveles.
    - Los usuarios han de poder ver el contenido subido por el resto de usuarios.
    - Los usuarios que han creado un nivel han de poder modificarlo a voluntad.
- El portal web contara con una sección personal de cada usuario:
    - Esta sección contara con una lista de los niveles que el usuario ha creado.
    - Esta sección contara con una lista de los niveles que ha elegido de otros usuarios para poder jugarlos.
    - Esta sección ha de mostrar una serie de estadísticas de los jugadores en los niveles que han jugado como: puntuación máxima obtenida en el nivel, si ha superado el nivel o no, valoración media del nivel.
- El portal web tendrá una sección para que los usuarios construyan y compartan niveles.
- El portal web tendrá una sección con los contenidos creados por la comunidad de usuarios.
- El portal tendrá una sección con la información del autor y del sitio web.

Los videojuegos que se pretenden construir son: un juego de demolición de construcciones y un juego del genero Arkanoid. A continuación se listaran los requerimientos para cada uno de estos juegos.

Requisitos generales de los videojuegos:

- Los juegos tienen que poder ser visualizados en los principales navegadores: Internet Explorer, Google Chrome, Firefox, Opera y Safari.
- Los juegos contarán con una librería que se encargue de simular la física de los elementos del juego.

Juego de demolición de estructuras o "Demolition":

- El juego consistirá en derribar estructuras mediante un cañón que dispara proyectiles.
- El juego mostrará en pantalla el número de disparos de los que dispone el jugador para completar el nivel.
- El juego mostrar en pantalla un marcador con la puntuación obtenía por el jugador en todo momento.
- El juego mostrará una línea que indicara al jugador el límite de altura que pueden alcanzar todas las piezas una vez derribada la construcción.
- El jugador podrá rotar el cañón 180 grados (Hacia arriba y abajo).
- La rotación del cañón se realizara pulsando dos teclas en el teclado. Una para rotar hacia arriba y otra para rotar hacia abajo.
- El disparo se realizara pulsando una tecla del teclado.
- El nivel estará completo o resuelto cuando todas las piezas de la construcción derribada terminen reposando bajo la línea que limita la altura máxima para superar el nivel.
- El motor del juego tendrá que leer los mapas creados por los demás usuarios para reproducirlos.

Juego de destrucción de bloques o "Arkanoid":

- El juego consistirá en hacer rebotar una bola con una barra situada en la parte inferior de la pantalla para destruir unos bloques situados en la parte superior de la pantalla.
- Los bloques de la parte superior del nivel serán estáticos.
- Los bloques han de desaparecer cuando hayan sido impactados por el proyectil.
- El nivel estará completo cuando todos los obstáculos hayan sido destruidos.
- El jugador podrá mover la barra de derecha a izquierda.
- El movimiento de la barra se realizara pulsando dos teclas en el teclado. Una para ir a la derecha y otra a la izquierda.
- El disparo inicial se realizara pulsando una tecla del teclado.
- El motor del juego tendrá que leer los mapas creados por los demás usuarios para reproducirlos.

La sección en la que los jugadores podrán crear contenido para los juegos ofrecidos por la plataforma ha de ser una herramienta sencilla e intuitiva donde los usuarios puedan construir niveles para los videojuegos que ofrece la página web.

Requisitos generales de la herramienta de creación de niveles:

- La aplicación de creación de niveles tiene que poder ser visualizada en los principales navegadores: Internet Explorer, Google Chrome, Firefox, Opera y Safari.
- Se visualizará una cuadricula en la que el usuario colocara las piezas para construir el nivel.
- Se mostrar una lista de piezas que el usuario podrá usar para construir el nivel.
- El usuario podrá arrastrar las piezas de la lista y depositarlas en la cuadricula.
- El usuario podrá arrastrar las piezas ya colocadas en la cuadricula para colocarlas en otro lugar dentro de la cuadricula.
- El usuario podrá configurar el número de disparos disponibles para superar el nivel que está construyendo.
- El usuario podrá ajustar la altura a la que estará colocada la línea para poder superar el nivel.
- El usuario podrá dar un nombre al nivel que ha creado.

- El usuario podrá escribir una pequeña descripción del nivel.
- El constructor de niveles tendrá que guardar los niveles creados en formato JSON.
- Los niveles creados han de almacenarse en algún servidor para mantener la persistencia.

5.2.1 Obtención de los Requisitos del Sistema

5.2.1.1 Requisitos Funcionales

5.2.1.1.1 Requisitos de la página web

Código	Nombre del Requisito	Descripción del Requisito
RF 1.1	Login de usuario	El usuario tiene que acceder a la página web mediante su nombre de usuario y su contraseña.
RF 1.2	Logout de usuario	El usuario abandonara la página tras pulsar el botón de logout.
RF 1.3	Crear usuario	El usuario introducirá un nombre y una contraseña para registrarse como nuevo usuario de la página web.
RF 1.4	Cargar playlist del usuario	La página web ha de cargar la lista de los niveles que un usuario tiene registrados en su playlist.
RF 1.5	Cargar lista de niveles creados por el usuario	La página web ha de cargar la lista de los niveles que un usuario ha creado dentro de la aplicación.
RF 1.6	Cargar lista de todos los niveles disponibles	La página web ha de cargar la lista de todos los niveles existentes dentro de la aplicación.
RF 1.7	Jugar nivel	La página web tiene que cargar el fichero del nivel seleccionado y reproducirlo para que el usuario pueda jugarlo.
RF 1.8	Editar Nivel	La página web tiene que cargar el fichero del nivel seleccionado y mostrarlo en la herramienta de creación del nivel tal y como se guardó para que el usuario pueda modificarlo.
RF 1.9	Añadir nivel a la playlist de un jugador	La página web ha de registrar el nivel seleccionado por el usuario dentro de su playlist personal.

5.2.1.1.2 Requisitos del videojuego Demolition.

Código	Nombre del Requisito	Descripción del Requisito
RF 2.1	Cargar los elementos del juego	El juego ha de cargar todos los elementos que componen el nivel y colocarlos en pantalla.
RF 2.2	Rotación del cañón	El cañón ha de rotar 180 grados, aparecerá inicialmente en posición horizontal y mirando hacia la derecha.
RF 2.3	Disparo del cañón	El cañón ha de lanza proyectiles en cuando se pulse la tecla de disparo.
RF 2.4	Movimiento del proyectil	El proyectil lanzado por el cañón ha de moverse con una velocidad y una fuerza determinadas para derribar los elementos que componen el nivel.
RF 2.5	Eliminación del proyectil lanzado	El proyectil desaparecerá de la pantalla en cuanto entre en estado de reposo.

RF 2.6	Movimiento de los cuerpos que componen la estructura a derribar	Los cuerpos que componen la estructura del nivel han de moverse de forma natural, ya que simularan cuerpos del mundo real, con una determinada masa, gravedad, etc.
RF 2.7	Visualización de la línea límite de altura para superar el nivel	El juego ha de mostrar en pantalla una línea que delimite la altura máxima por la que los cuerpos han de quedar por debajo para superar el nivel.
RF 2.8	Visualización del número de proyectiles utilizados	El juego ha de mostrar un marcador con el número de proyectiles que restan por ser lanzados por el jugador.
RF 2.9	Visualización de la puntuación del jugador	El juego ha de mostrar un marcador con la puntuación obtenida por el jugador.
RF 2.10	Finalizar juego: Nivel superado	El juego finalizará cuando todas las piezas reposen bajo la línea límite. En ese momento la pantalla de juego mostrará un mensaje diciendo que se ha superado el nivel y la puntuación obtenida por el jugador.
RF 2.11	Finalizar juego: Nivel no superado	El juego finalizará cuando todas las piezas estén en reposo y hayan desaparecido todos los proyectiles. En ese momento la pantalla de juego mostrará un mensaje diciendo que no se ha superado el nivel y la puntuación obtenida por el jugador.
RF 2.12	Valorar el nivel	Se ha de mostrar un medidor de estrellas para que el jugador seleccione la valoración que da a dicho nivel.
RF 2.13	Volver a jugar el nivel	En el momento que el juego termina se mostrará un botón que permitirá volver a jugar el nivel. El nivel se reiniciara y comenzara de nuevo.
RF 2.14	Volver a la pantalla de Home del usuario	En el momento que el juego termina se mostrará un botón que permitirá volver a la pantalla de Home del usuario.

5.2.1.1.3 Requisitos del videojuego Arkanoid.

Código	Nombre del Requisito	Descripción del Requisito
RF 3.1	Cargar los elementos del juego	El juego ha de cargar todos los elementos que componen el nivel y colocarlos en pantalla.
RF 3.2	Movimiento de la paleta del jugador	La paleta manejada por el jugador ha de moverse de forma horizontal a izquierda y derecha por la pantalla.
RF 3.3	Movimiento de la bola	La bola tendrá un movimiento libre, con una velocidad y una fuerza determinada, rebotará con una fuerza constante en todos los elementos del juego.
RF 3.4	Destrucción de los bloques	Los bloques que componen el nivel han de ser destruidos en el momento que son impactados por la bola.
RF 3.5	Visualización del número de bloques restantes por destruir	El juego ha de mostrar un marcador con el número de bloques que quedan por ser destruidos.
RF 3.6	Visualización de la puntuación del jugador	El juego ha de mostrar un marcador con la puntuación obtenida por el jugador.
RF 3.7	Finalizar juego: Nivel superado	El juego finalizara cuando todos los bloques hayan sido destruidos. En ese momento la pantalla de juego mostrará un mensaje diciendo que se ha superado el nivel y la puntuación obtenida por el jugador.
RF 3.8	Finalizar juego: Nivel no superado	El juego finalizara cuando la bola toque el borde inferior de la pantalla. En ese momento la pantalla de juego mostrará un mensaje diciendo que no se ha superado el nivel y la puntuación obtenida por el jugador.

RF 3.9	Valorar el nivel	Se ha de mostrar un medidor de estrellas para que el jugador seleccione la valoración que da a dicho nivel.
RF 3.10	Volver a jugar el nivel	En el momento que el juego termina se mostrará un botón que permitirá volver a jugar el nivel. El nivel se reiniciara y comenzara de nuevo.
RF 3.11	Volver a la pantalla de Home del usuario	En el momento que el juego termina se mostrará un botón que permitirá volver a la pantalla de Home del usuario.

5.2.1.1.4 Requisitos de la herramienta de creación de contenido.

Código	Nombre del Requisito	Descripción del Requisito
RF 4.1	Cargar la cuadricula de edición	La herramienta de edición ha de generar una cuadricula del mismo tamaño que la pantalla del juego final donde se colocaran las piezas.
RF 4.2	Cargar lista de elementos de edición	La herramienta de edición ha de mostrar una lista con todos los elementos disponibles para construir los niveles.
RF 4.3	Moverse entre las páginas de la lista de elementos de edición	La lista de elementos de edición ha de mostrar unos botones que permita moverse entre las páginas de elementos disponibles.
RF 4.4	Arrastrar pieza a la cuadricula	Las piezas de la lista de elementos de edición han de poder arrastrase a la cuadricula de la herramienta de edición. Dichas piezas han de ajustarse a las casillas de dicha cuadricula.
RF 4.5	Mover pieza por la cuadricula	Una pieza ya colocada en la cuadricula ha de poder ser arrastrada para colocarse en otra posición de la cuadricula.
RF 4.6	Arrastrar pieza fuera de la cuadricula	Si una pieza es arrastrada fuera de la cuadricula, esta, volverá a la última posición tomada en la cuadricula.
RF 4.7	Eliminar pieza	Cuando se arrastra una pieza de la cuadricula a la papelera, esta es eliminada.
RF 4.8	Escribir nombre del nivel	Se ha de proporcionar un campo para escribir el nombre del nivel.
RF 4.9	Disponibilidad del nombre del nivel	Se ha de comprobar si el nombre elegido para el nivel ya existe en la base de datos. Si el nombre existe se iluminara en rojo y no permitirá guardar el nivel. Si el nombre está disponible, este se iluminara en verde y si permitirá guardar el nivel.
RF 4.10	Escribir descripción del nivel	Se ha de proporcionar un campo para escribir la descripción del nivel.
RF 4.11	Seleccionar altura de la línea limite (Solo juego de Demolición)	Se proporcionará un campo numérico para seleccionar la posición de la línea límite en la herramienta de edición para niveles del juego de Demolición.
RF 4.12	Seleccionar posición del cañón (Solo juego de Demolición)	Se proporcionará un campo numérico para seleccionar la posición del cañón en la herramienta de edición para niveles del juego de Demolición.
RF 4.13	Seleccionar número de proyectiles (Solo juego de Demolición)	Se proporcionará un campo numérico para seleccionar la cantidad de proyectiles a utilizar en la herramienta de edición para niveles del juego de Demolición.

RF 4.14	Cargar nivel para edición	Se ha de cargar el nivel con todos los datos con los que había sido guardado previamente: nombre, descripción, las piezas que lo componen en la posición en la que se encontraban, etc. Se han de poder modificar estos datos, menos el nombre del nivel y se ha de poder guardar otra vez con la nueva información.
RF 4.15	Guardar nivel en fichero JSON (Juego: Demolición)	Se generará un fichero JSON con todos los datos que componen el nivel del jugo de Demolición.
RF 4.16	Guardar nivel en fichero JSON (Juego: Arkanoid)	Se generará un fichero JSON con todos los datos que componen el nivel del jugo Arkanoid.

5.2.1.2 Requisitos tecnológicos:

Código	Nombre del Requisito	Descripción del Requisito
RT 1.1	Navegador Web	Es necesario tener un navegador web en el que visualizar la página web.
RT 1.2	JavaScript activado	El navegador ha de tener activado JavaScript para que las aplicaciones de la página web funcionen.

5.2.2 Identificación de Actores del Sistema

En esta aplicación solo podemos identificar a un tipo de actor, el usuario, es la única entidad que actúa con el sistema dadas las características de la aplicación. No tienen cabida ningún tipo de actor más ya que la aplicación está dirigida a que un único perfil de usuario la use.

5.2.3 Especificación de Casos de Uso

5.2.3.1 Casos de uso referentes a la página web

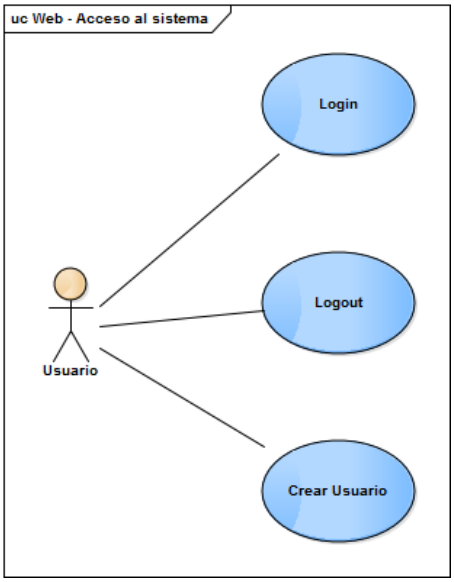


Figura 12 Diagrama casos de uso: Acceso al sistema

Nombre del Caso de Uso
Login
Descripción
El usuario ha de introducir su nombre de usuario y la contraseña para acceder a la página web.

Nombre del Caso de Uso
Logout
Descripción
Cuando se pulsa el enlace para cerrar sesión, el usuario sale de sesión, y se vuelve a la pantalla de Login.

Nombre del Caso de Uso
Crear usuario
Descripción
Se mostrará una pantalla al usuario con 3 campos, uno para el nombre de usuario, otro para la contraseña y el último para repetir la contraseña. Si el usuario mete un nombre ya registrado, no podrá crear el usuario.

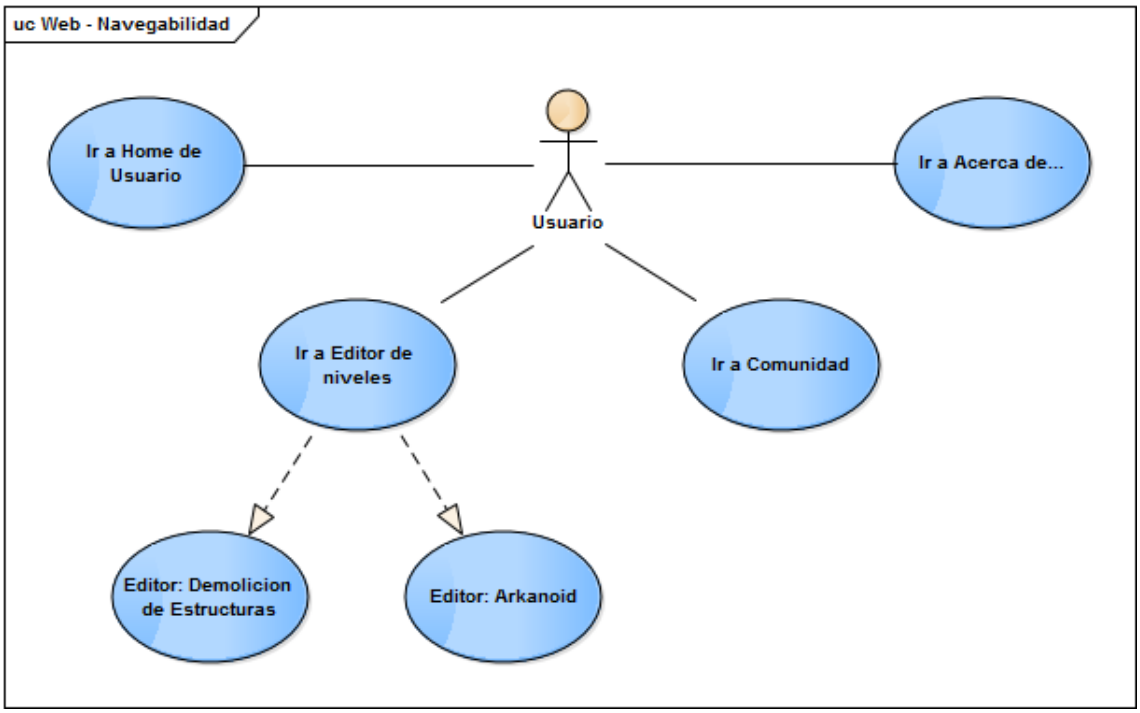


Figura 13 Diagrama casos de uso: Navegabilidad

Nombre del Caso de Uso
Ir al Home de usuario
Descripción
Al pulsar esta opción en el menú de navegación, se carga la página del Home personal del usuario que este en sesión.

Nombre del Caso de Uso
Ir al Editor de niveles: Demolición de estructuras
Descripción
Al pulsar esta opción en el menú de navegación, se carga una página para seleccionar el tipo de nivel que se quiere editar. Al seleccionar la opción *"Editor Demolición"* se carga la página de la herramienta de edición para el juego de demolición de estructuras.

Nombre del Caso de Uso
Ir al Editor de niveles: Arkanoid
Descripción
Al pulsar esta opción en el menú de navegación, se carga una página para seleccionar el tipo de nivel que se quiere editar. Al seleccionar la opción *"Editor Arkanoid"* se carga la página de la herramienta de edición para el juego Arkanoid.

Nombre del Caso de Uso
Ir a Comunidad
Descripción
Al pulsar esta opción en el menú de navegación, se carga la página de la Comunidad, página donde se listarán todos los niveles creados por los usuarios.

Nombre del Caso de Uso
Ir a Acerca de…
Descripción
Al pulsar esta opción en el menú de navegación, se carga la página Acerca de…, página donde hay información sobre el sitio web y el autor.

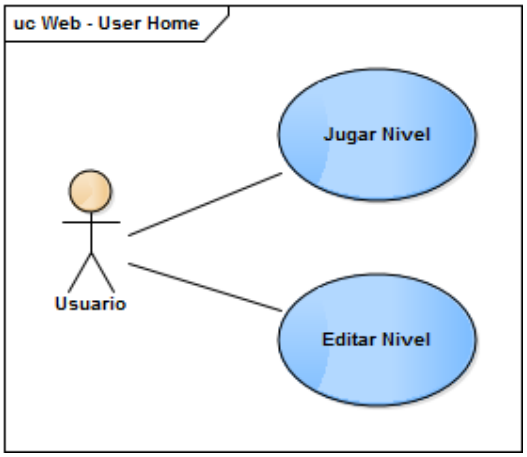


Figura 14 Diagrama casos de uso: User Home

Nombre del Caso de Uso
Jugar nivel
Descripción
Dentro del Home del usuario, cuando se pulsa el botón "Jugar" de un nivel de la lista, se carga la página con el nivel seleccionado para que el usuario pueda jugar dicho nivel.

Nombre del Caso de Uso
Editar nivel
Descripción
Dentro del Home del usuario, cuando se pulsa el botón "Editar" de un nivel de la lista, se carga la herramienta de edición con el nivel seleccionado para que el usuario pueda editarlo.

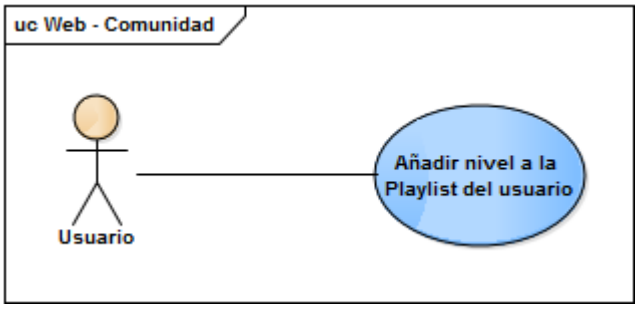


Figura 15 Diagrama casos de uso: Community

Nombre del Caso de Uso
Añadir nivel a la Playlist del usuario
Descripción
Dentro de la página Comunidad, cuando se pulsa el botón "Añadir a Playlist" de uno de los niveles de la lista, este automáticamente se guarda en la Playlist del usuario.

5.2.3.2 Casos de uso referentes al juego Demolición

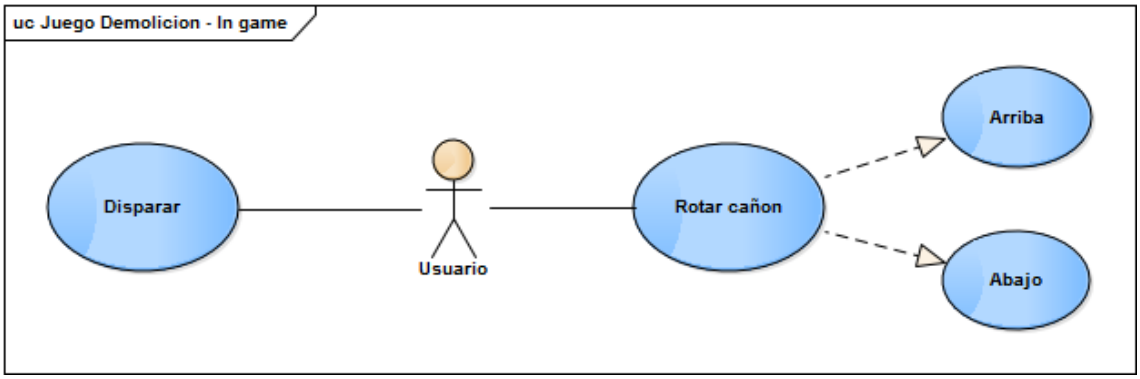


Figura 16 Diagrama casos de uso: Juego Demolition, In Game

Nombre del Caso de Uso
Rotar cañón hacia arriba
Descripción
El usuario mediante el teclado rotará el cañón, al pulsar la tecla asignada el cañón hará un pequeño giro hacia arriba, si se deja pulsada la tecla el cañón girara de continuo hasta llegar al máximo de grados que puede rotar hacia arriba.

Nombre del Caso de Uso
Rotar cañón hacia abajo
Descripción
El usuario mediante el teclado rotará el cañón, al pulsar la tecla asignada el cañón hará un pequeño giro hacia abajo, si se deja pulsada la tecla el cañón girara de continuo hasta llegar al máximo de grados que puede rotar hacia abajo.

Nombre del Caso de Uso
Disparar
Descripción
El usuario mediante el teclado realizará un disparo, al pulsar la tecla asignada, el cañón dispara un proyectil contra la estructura para intentar derribarla.

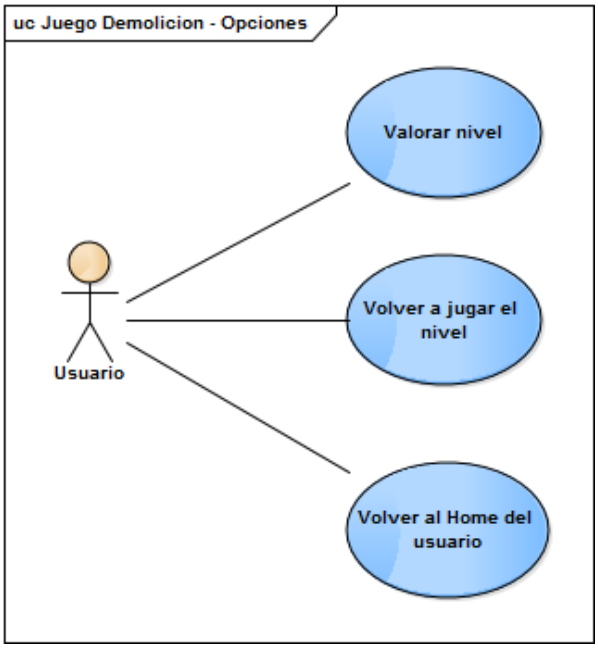


Figura 17 Diagrama casos de uso: Juego Demolition, Opciones

Nombre del Caso de Uso
Valorar nivel
Descripción
En todo momento durante el juego, en la parte derecha de la pantalla se ofrecerá un medidor de estrellas con el que el jugador puede valorar el nivel, puntuándolo entre 1 y 5. Cuando se pulsa en una de las estrellas, estas se colorean para mostrar el valor escogido por el jugador.

Nombre del Caso de Uso
Volver a jugar Nivel
Descripción
Al finalizar el juego, se mostrará un botón en la parte derecha de la pantalla, si se pulsa este botón el nivel se reinicia completamente y el jugador puede volver a jugar el nivel.

Nombre del Caso de Uso
Volver al Home del usuario
Descripción
Al finalizar el juego, se mostrará un botón en la parte derecha de la pantalla, si se pulsa este botón el nivel se carga la página del Home de usuario y se actualizan las puntuaciones obtenidas tras jugar el nivel.

5.2.3.3 Casos de uso referentes al juego Arkanoid

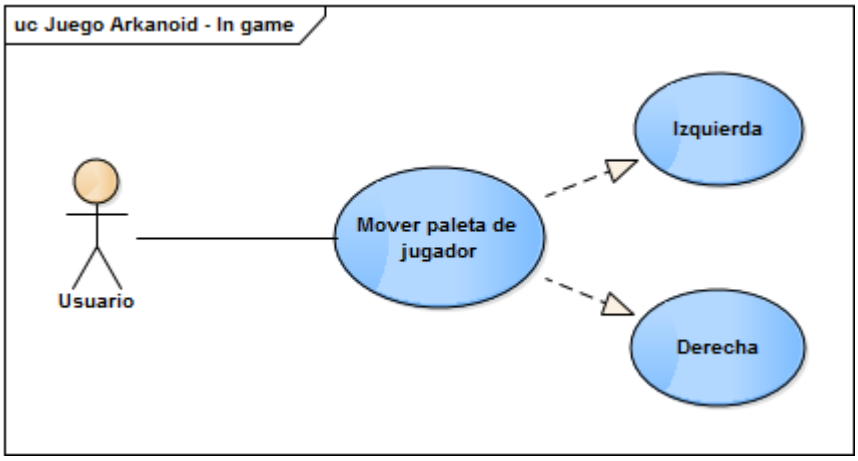


Figura 18 Diagrama casos de uso: Juego Arkanoid, In Game

Nombre del Caso de Uso
Mover paleta del jugador hacia la izquierda
Descripción
El usuario mediante el teclado moverá la paleta del jugador, al pulsar la tecla asignada la paleta hará un pequeño movimiento hacia la izquierda, si se deja pulsada la tecla la paleta se moverá de continuo hasta llegar al extremo izquierdo de la pantalla de juego.

Nombre del Caso de Uso
Mover paleta del jugador hacia la derecha
Descripción
El usuario mediante el teclado moverá la paleta del jugador, al pulsar la tecla asignada la paleta hará un pequeño movimiento hacia la derecha, si se deja pulsada la tecla la paleta se moverá de continuo hasta llegar al extremo derecho de la pantalla de juego.

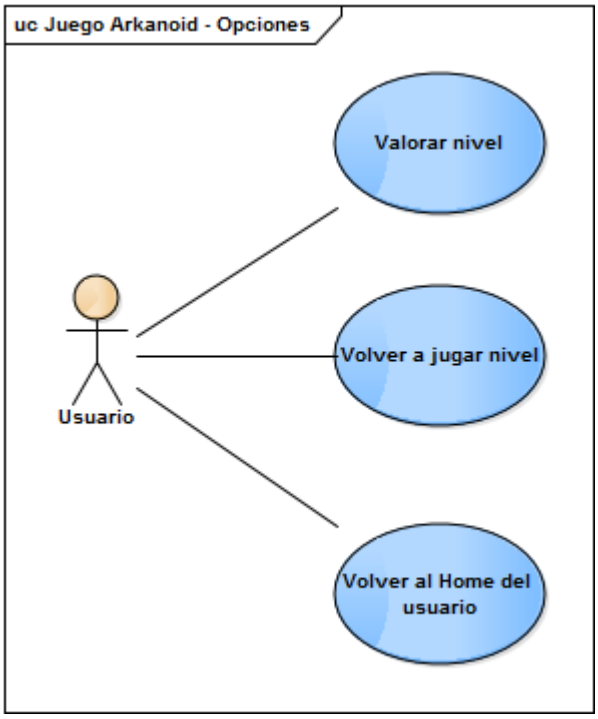


Figura 19 Diagrama casos de uso: Juego Arkanoid, Opciones

Nombre del Caso de Uso
Valorar nivel
Descripción
En todo momento durante el juego, en la parte derecha de la pantalla se ofrecerá un medidor de estrellas con el que el jugador puede valorar el nivel, puntuándolo entre 1 y 5. Cuando se pulsa en una de las estrellas, estas se colorean para mostrar el valor escogido por el jugador.

Nombre del Caso de Uso
Volver a jugar Nivel
Descripción
Al finalizar el juego, se mostrará un botón en la parte derecha de la pantalla, si se pulsa este botón el nivel se reinicia completamente y el jugador puede volver a jugar el nivel.

Nombre del Caso de Uso
Volver al Home del usuario
Descripción
Al finalizar el juego, se mostrará un botón en la parte derecha de la pantalla, si se pulsa este botón el nivel se carga la página del Home de usuario y se actualizan las puntuaciones obtenidas tras jugar el nivel.

5.2.3.4 Casos de uso referentes a la herramienta de edición de niveles

5.2.3.4.1 Editor de niveles Demolición

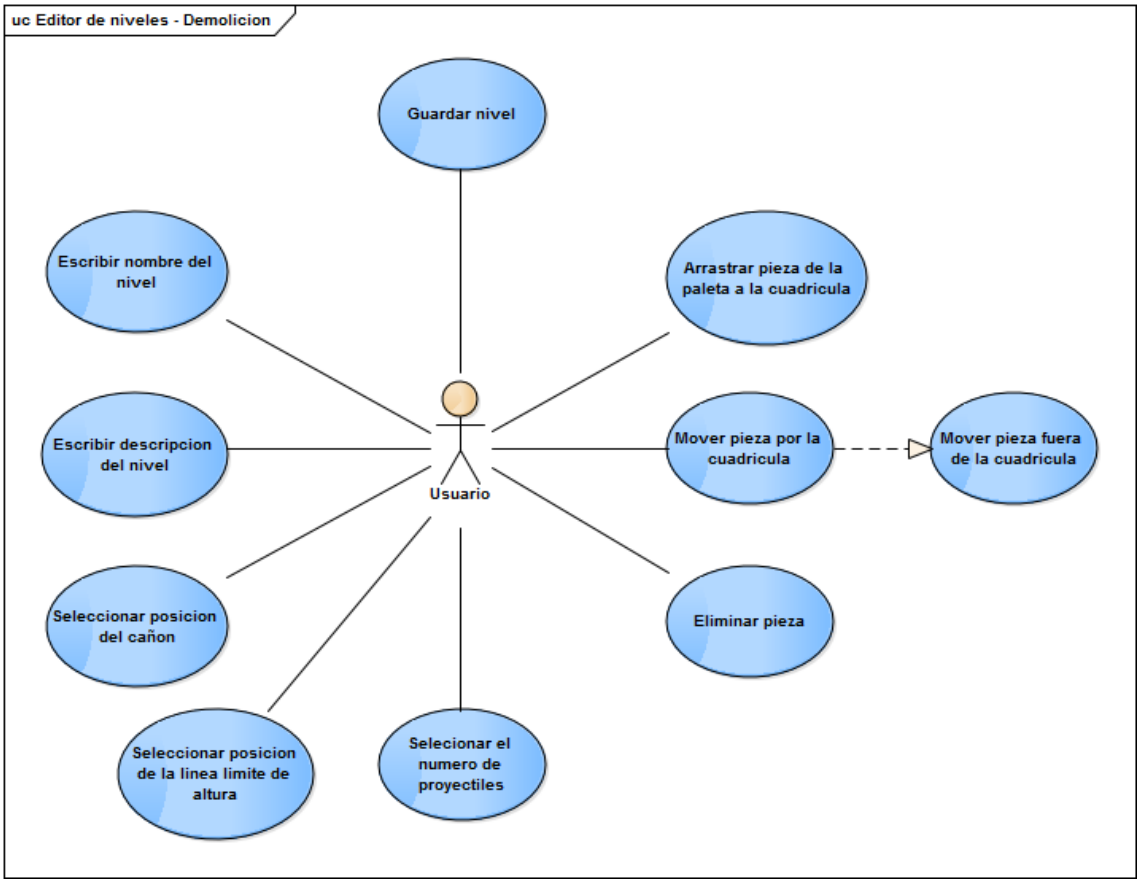


Figura 20 Diagrama casos de uso: Editor de niveles, Demolition

Nombre del Caso de Uso
Arrastrar pieza de la paleta a la cuadricula
Descripción
El usuario coloca el puntero del ratón encima de una de las piezas de la paleta de elementos y arrastra dicha pieza a la cuadricula para colocarla en el nivel. La pieza siempre se ajusta a la cuadricula.

Nombre del Caso de Uso
Mover pieza por la cuadricula
Descripción
El usuario coloca el puntero del ratón encima de una de las piezas que están en la cuadricula y arrastra dicha pieza por la cuadricula para cambiar la posición en la que se encontraba anteriormente. La pieza siempre se ajusta a la cuadricula.

Nombre del Caso de Uso
Mover pieza fuera de la cuadricula
Descripción
El usuario coloca el puntero del ratón encima de una de las piezas que están en la cuadricula y arrastra dicha pieza fuera de la cuadricula, en el momento en el que se suelta dicha pieza fuera de la cuadricula, esta vuelve automáticamente a la última posición tomada dentro de la cuadricula.

Nombre del Caso de Uso
Eliminar pieza
Descripción
El usuario coloca el puntero del ratón encima de una de las piezas que están en la cuadricula y arrastra dicha pieza a la papelera, al soltar dicha pieza en la papelera esta es eliminada.

Nombre del Caso de Uso
Escribir el nombre del nivel
Descripción
El usuario escribe el nombre que quiere darle al nivel en un campo destinado a ello. Si el nombre está disponible, se marcará iluminando en verde el campo de texto, si no está disponible, se ilumina en rojo.

Nombre del Caso de Uso
Escribir descripción del nivel
Descripción
El usuario escribe la descripción que quiere darle al nivel en un campo destinado a ello.

Nombre del Caso de Uso
Seleccionar posición del cañón
Descripción
El usuario selecciona la posición del cañón mediante un selector numérico. Debido a que cada número equivale a una posición dentro de la cuadricula de edición.

Nombre del Caso de Uso
Seleccionar posición de la línea límite de altura
Descripción
El usuario selecciona la posición de la línea límite de altura para superar el nivel mediante un selector numérico. Debido a que cada número equivale a una posición dentro de la cuadricula de edición.

Nombre del Caso de Uso
Seleccionar el número de proyectiles
Descripción
El usuario selecciona el número de proyectiles que puede disparar el cañón mediante un selector numérico. Ha de tener como mínimo 1.

Nombre del Caso de Uso
Guardar nivel
Descripción
Se crea un fichero JSON con el contenido del nivel creado.

5.2.3.4.2 Editor de niveles Arkanoid

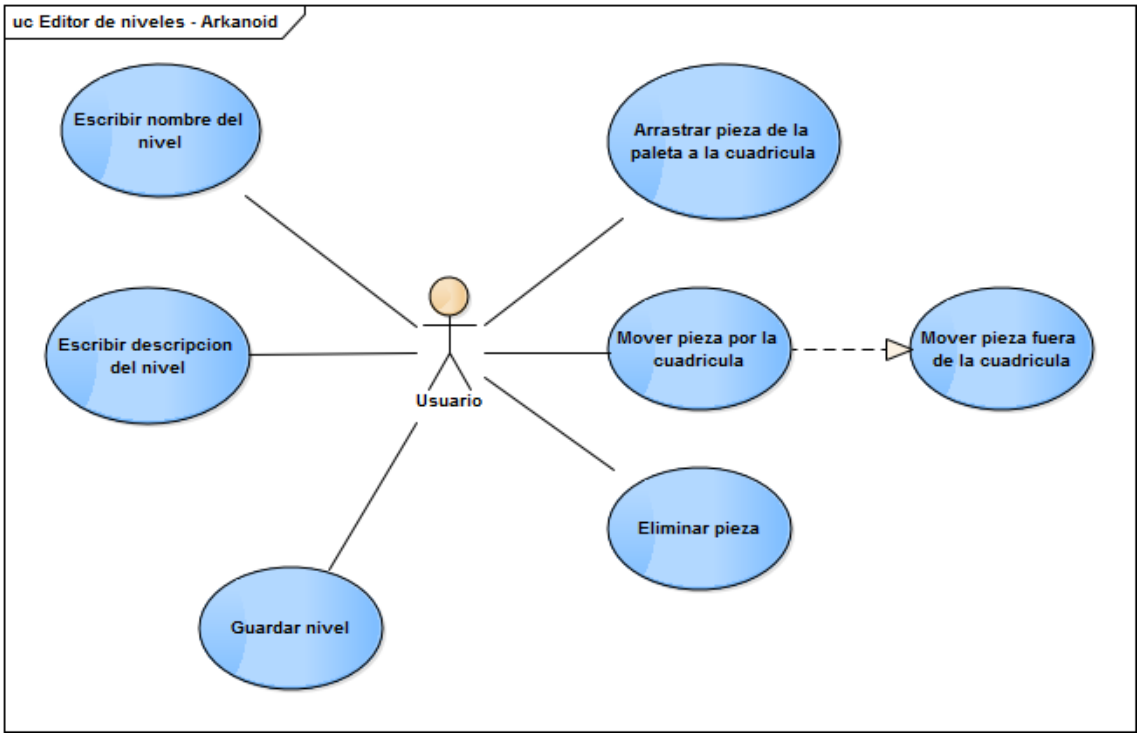


Figura 21 Diagrama casos de uso: Editor de niveles, Arkanoid

Nombre del Caso de Uso
Arrastrar pieza de la paleta a la cuadricula
Descripción
El usuario coloca el puntero del ratón encima de una de las piezas de la paleta de elementos y arrastra dicha pieza a la cuadricula para colocarla en el nivel. La pieza siempre se ajusta a la cuadricula.

Nombre del Caso de Uso
Mover pieza por la cuadricula
Descripción
El usuario coloca el puntero del ratón encima de una de las que están en la cuadricula y arrastra dicha pieza por la cuadricula para cambiar la posición en la que se encontraba anteriormente. La pieza siempre se ajusta a la cuadricula.

Nombre del Caso de Uso
Mover pieza fuera de la cuadricula
Descripción
El usuario coloca el puntero del ratón encima de una de las piezas que están en la cuadricula y arrastra dicha pieza fuera de la cuadricula, en el momento en el que se suelta dicha pieza fuera de la cuadricula, esta vuelve automáticamente a la última posición tomada dentro de la cuadricula.

Nombre del Caso de Uso
Eliminar pieza
Descripción
El usuario coloca el puntero del ratón encima de una de las piezas que están en la cuadricula y arrastra dicha pieza a la papelera, al soltar dicha pieza en la papelera esta es eliminada.

Nombre del Caso de Uso
Escribir el nombre del nivel
Descripción
El usuario escribe el nombre que quiere darle al nivel en un campo destinado a ello. Si el nombre está disponible, se marcará iluminando en verde el campo de texto, si no está disponible, se ilumina en rojo.

Nombre del Caso de Uso
Escribir descripción del nivel
Descripción
El usuario escribe la descripción que quiere darle al nivel en un campo destinado a ello.

Nombre del Caso de Uso
Guardar nivel
Descripción
Se crea un fichero JSON con el contenido del nivel creado.

5.3 Identificación de los Subsistemas en la Fase de Análisis

5.3.1 Descripción de los Subsistemas

5.3.1.1 Página web

Este subsistema es el cuerpo central de la aplicación, ya que es la propia página web que se va a mostrar al usuario final y la que engloba al resto de subsistemas.

5.3.1.2 Videojuego Demolición

Este subsistema es el encargado de mover el juego de demolición de estructuras, de cargar todos los elementos y aplicar las configuraciones pertinentes a cada elemento del juego.

5.3.1.3 Videojuego Arkanoid

Este subsistema es el encargado de mover el juego Arkanoid, de cargar todos los elementos y aplicar las configuraciones pertinentes a cada elemento del juego.

5.3.1.4 Herramienta de edición de niveles

Este subsistema es el encargado de la creación del contenido para los videojuegos presentados, se encarga de generar los ficheros JSON que componen cada nivel.

5.3.2 Descripción de los Interfaces entre Subsistemas

Todas las comunicaciones entre los diferentes subsistemas se realizan con peticiones HTTP y llamadas a las clases controladoras dentro del patrón MVC de Spring.

5.3.2.1 Interfaz entre los distintos subsistemas.

La comunicación entre los distintos subsistemas se realiza a nivel de servidor a través de clases ya que se trata de una aplicación MVC Spring.

La comunicación a nivel de cliente se realiza con llamadas HTTP POST y GET entre las distingas páginas que componen el sistema.

5.4 Diagrama de Clases Preliminar del Análisis

5.4.1 Diagrama de Clases

El diagrama que se presenta a continuación es un diagrama resumido, ya que no se muestran los métodos ni los parámetros de cada clase para poder dar más legibilidad al diagrama. Estos métodos se explicaran a continuación junto a las clases.

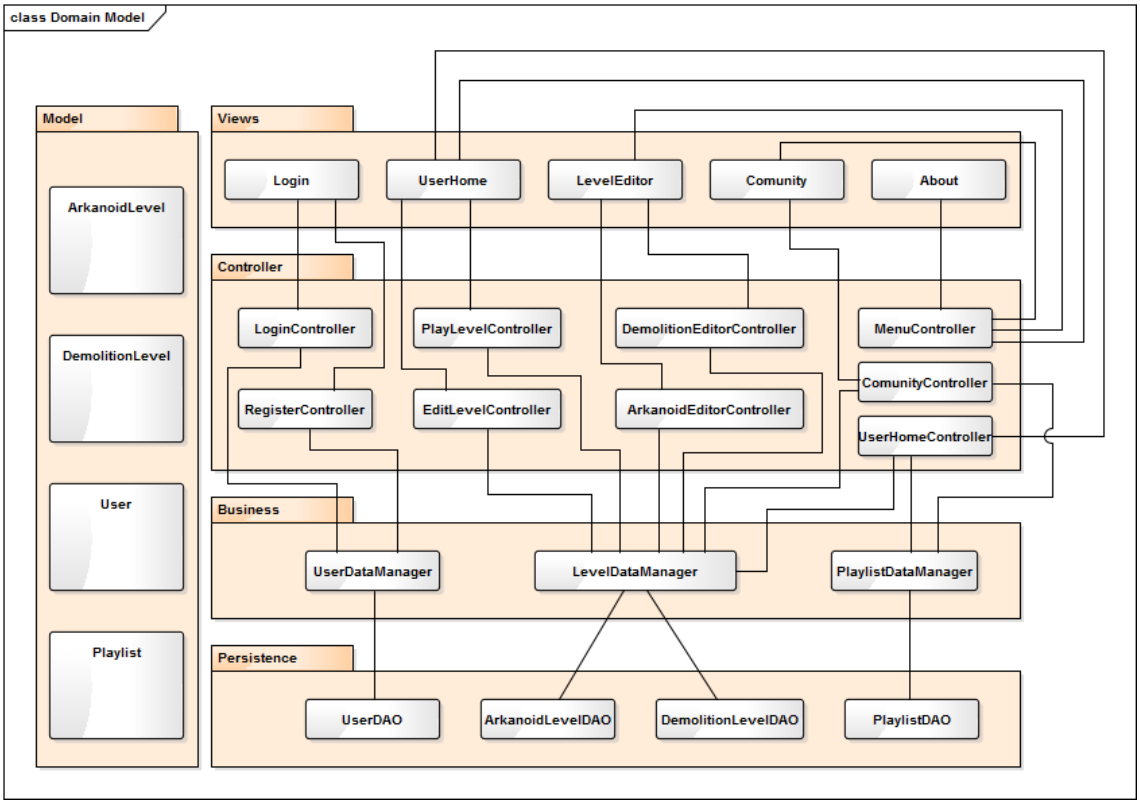


Figura 22 Diagrama de clases preliminar del análisis

El sistema planteado por este diagrama es un patrón Modelo-Vista-Controlador, donde tenemos un modelo transversal para todo el proyecto, un subsistema de persistencia que es el que se comunica con la base de datos y una capa de lógica de negocio que interactúa con el modelo y los controladores.

5.4.2 Descripción de las Clases

5.4.2.1 Subsistema: Model

Nombre de la clase	
User	
Descripción	
Clase que contiene los datos del usuario.	
Responsabilidad	
Crear el objeto User con los datos del usuario.	
Atributos propuestos	
Username	Nombre del usuario del sistema.
Password	Contraseña que el usuario tiene para acceder al sistema.

Nombre de la clase	
DemolitionLevel	
Descripción	
Clase que contiene los datos de un nivel del videojuego “Demolición”.	
Responsabilidad	
Crear el objeto DemolitionLevel con los datos del nivel.	
Atributos propuestos	
LevelName	Nombre del nivel.
LevelDescription	Descripción del nivel.
Author	Usuario creador del nivel.
LevelElementsList	Lista de objetos que componen el nivel.
LimitLinePosition	Posición de la línea de altura máxima para superar el nivel.
CanonPosition	Posición del cañón.
NumBullets	Número de proyectiles que se pueden utilizar en este nivel.

Nombre de la clase	
ArkanoidLevel	
Descripción	
Clase que contiene los datos de un nivel del videojuego “Arkanoid”.	
Responsabilidad	
Crear el objeto ArkanoidLevel con los datos del nivel.	
Atributos propuestos	
LevelName	Nombre del nivel.
LevelDescription	Descripción del nivel.
Author	Usuario creador del nivel.
LevelElementsList	Lista de objetos que componen el nivel.

Nombre de la clase	
Playlist	
Descripción	
Clase que contiene los datos de la lista de niveles, puntuaciones y demás estadísticas de un usuario.	
Responsabilidad	
Crear el objeto Playlist con los datos necesarios.	
Atributos propuestos	
PlaylistUser	Nombre del usuario dueño de la playlist.
DemolitionLevelList	Lista de niveles del videojuego "Demolicion" que el jugador ha seleccionado.
ArkanoidLevelList	Lista de niveles del videojuego "Arkanoid" que el jugador ha seleccionado.

5.4.2.2 Subsistema: Persistence

Nombre de la clase	
UserDAO	
Descripción	
Clase encargada de realizar operaciones con la base de datos.	
Responsabilidad	
Encargada de comunicarse con la base de datos para realizar las operaciones referentes a los usuarios del sistema.	
Atributos propuestos	
getUserQuery	Texto de la consulta para obtener un usuario de la base de datos.
existUserQuery	Texto de la consulta para saber si un usuario existe o no dentro de la base de datos.
insertUserQuery	Texto de la consulta para introducir en la base de datos un nuevo usuario.
Métodos propuestos	
getUser	Método que llama a la base de datos para obtener los datos de un usuario.
existUser	Método que llama a la base de datos para saber si un usuario existe o no en la base de datos.
InsertUser	Método que llama a la base de datos para insertar un nuevo usuario en la base de datos.

Nombre de la clase	
DemolitionLevelDAO	
Descripción	
Clase encargada de realizar operaciones con la base de datos.	
Responsabilidad	
Encargada de comunicarse con la base de datos para realizar las operaciones referentes a los niveles del juego "Demolición" del sistema.	
Atributos propuestos	
getLevelQuery	Texto de la consulta para obtener un nivel de la base de datos.
existLevelQuery	Texto de la consulta para saber si un nivel existe o no dentro de la base de datos.
insertLevelQuery	Texto de la consulta para introducir en la base de datos un nuevo nivel.
updateLevelQuery	Texto de la consulta para actualizar los datos de un nivel.
Métodos propuestos	
getLevel	Método que llama a la base de datos para obtener los datos de un nivel.
existLevel	Método que llama a la base de datos para saber si un nivel existe o no en la base de datos.
insertLevel	Método que llama a la base de datos para insertar un nuevo nivel en la base de datos.
updateLevel	Método que llama a la base de datos para modificar los parámetros de un nivel.

Nombre de la clase	
ArkanoidLevelDAO	
Descripción	
Clase encargada de realizar operaciones con la base de datos.	
Responsabilidad	
Encargada de comunicarse con la base de datos para realizar las operaciones referentes a los niveles del juego "Arkanoid" del sistema.	
Atributos propuestos	
getLevelQuery	Texto de la consulta para obtener un nivel de la base de datos.
existLevelQuery	Texto de la consulta para saber si un nivel existe o no dentro de la base de datos.
insertLevelQuery	Texto de la consulta para introducir en la base de datos un nuevo nivel.
updateLevelQuery	Texto de la consulta para actualizar los datos de un nivel.
Métodos propuestos	
getLevel	Método que llama a la base de datos para obtener los datos de un nivel.
existLevel	Método que llama a la base de datos para saber si un nivel existe o no en la base de datos.
insertLevel	Método que llama a la base de datos para insertar un nuevo nivel en la base de datos.
updateLevel	Método que llama a la base de datos para modificar los parámetros de un nivel.

Nombre de la clase	
PlaylistDAO	
Descripción	
Clase encargada de realizar operaciones con la base de datos.	
Responsabilidad	
Encargada de comunicarse con la base de datos para realizar las operaciones referentes a las playlists de los usuarios.	
Atributos propuestos	
getUserLevelsQuery	Texto de la consulta para obtener todos los niveles de un usuario de la base de datos.
insertLevelQuery	Texto de la consulta para añadir un nivel a la playlist de un usuario.
Métodos propuestos	
getUserLevels	Método que llama a la base de datos para obtener todos los niveles de un usuario.
insertLevel	Método que llama a la base de datos para inserta un nivel en la playlist de un usuario.

5.4.2.3 Subsistema: Business

Nombre de la clase	
UserDataManager	
Descripción	
Clase que actúa como separación entre el módulo de persistencia y el módulo de lógica de negocio.	
Responsabilidad	
Se encarga de comunicarse con la clase DAO que maneja los usuarios.	
Atributos propuestos	
UserDAO	Objeto DAO de la capa de persistencia para los usuarios.
Métodos propuestos	
getUser	Método que llama al getUser de la clase DAO.
existUser	Método que llama al existUser de la clase DAO.
InsertUser	Método que llama al insertUser de la clase DAO.

Nombre de la clase	
LevelDataManager	
Descripción	
Clase que actúa como separación entre el módulo de persistencia y el módulo de lógica de negocio.	
Responsabilidad	
Se encarga de comunicarse con las clases DAO que manejan los niveles de los videojuegos “Demolición” y “Arkanoid”.	
Atributos propuestos	
ArkanoidLevelDAO	Objeto DAO de la capa de persistencia para los niveles del videojuego “Arkanoid”.
DemolitionLevelDAO	Objeto DAO de la capa de persistencia para los niveles del videojuego “Demolición”.
Métodos propuestos	
getLevel	Método que llama al getLevel de la clase DAO que corresponda según el tipo de nivel.
existLevel	Método que llama al existLevel de la clase DAO que corresponda según el tipo de nivel.
insertLevel	Método que llama al insertLevel de la clase DAO que corresponda según el tipo de nivel.
updateLevel	Método que llama al updateLevel de la clase DAO que corresponda según el tipo de nivel.

Nombre de la clase	
PlaylistDataManager	
Descripción	
Clase que actúa como separación entre el módulo de persistencia y el módulo de lógica de negocio.	
Responsabilidad	
Se encarga de comunicarse con la clase DAO que maneja las playlists.	
Atributos propuestos	
PlaylistDAO	Objeto DAO de la capa de persistencia para las playlists.
Métodos propuestos	
getUserLevels	Método que llama al getUserLevels de la clase DAO.
insertLevel	Método que llama al insertLevel de la clase DAO.

5.4.2.4 Subsistema: Controller

Nombre de la clase	
LoginController	
Descripción	
Clase que contiene las operaciones para realizar el acceso de usuario a la página web.	
Responsabilidad	
Se encarga de verificar que el usuario y contraseña introducidos son correctos y en caso afirmativo, cargar la siguiente página.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
User	Objeto con los datos del usuario introducido.
UserDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
Métodos propuestos	
loginUser	Este método comprueba en la base de datos si el usuario introducido es correcto. En caso afirmativo guarda el usuario en la sesión y carga la siguiente página web. En caso negativo muestra un error.

Nombre de la clase	
RegisterController	
Descripción	
Clase que contiene las operaciones para crear un usuario nuevo en la página web.	
Responsabilidad	
Se encarga de verificar el nombre que ha introducido el usuario, si este está disponible, se crea un nuevo usuario y se añade en la base de datos.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
User	Objeto con los datos del usuario introducido.
UserDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
Métodos propuestos	
insertNewUser	Este método primero realiza una serie de comprobaciones: Si el nombre introducido ya está en la base de datos y que las contraseñas introducidas coincidan. Una vez comprobado eso, si todo va bien se crea un nuevo usuario y se inserta en la base de datos. En caso contrario se mostrara un error.

Nombre de la clase	
PlayLevelController	
Descripción	
Clase que contiene las operaciones para jugar con los videojuegos.	
Responsabilidad	
Se encarga de cargar los datos de los niveles a jugar y pasarlos al motor del videojuego para que este se configure debidamente.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario y del nivel jugado mientras dure la sesión.
LevelDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
ArkanoidLevel	Objetos con los datos del nivel jugado (videojuego "Arkanoid").
DemolitionLevel	Objetos con los datos del nivel jugado (videojuego "Demolición").
Métodos propuestos	
PlayDemolitionLevel	Este método accede a la base de datos para obtener los datos del nivel que se va a jugar: Nombre, Ruta donde está el fichero JSON con los contenidos del nivel, etc. Decodifica el contenido del fichero JSON del nivel y lo pasa a la página web que contiene el motor del juego para que construya el nivel.
PlayArkanoidLevel	Este método accede a la base de datos para obtener los datos del nivel que se va a jugar: Nombre, Ruta donde está el fichero JSON con los contenidos del nivel, etc. Decodifica el contenido del fichero JSON del nivel y lo pasa a la página web que contiene el motor del juego para que construya el nivel.

Nombre de la clase	
EditLevelController	
Descripción	
Clase que contiene las operaciones para editar un nivel ya creado.	
Responsabilidad	
Se encarga de cargar los datos del nivel a editar y trasladarlos a la herramienta de edición.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
LevelDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
ArkanoidLevel	Objetos con los datos del nivel jugado (videojuego "Arkanoid").
DemolitionLevel	Objetos con los datos del nivel jugado (videojuego "Demolición").
Métodos propuestos	
EditDemolitionLevel	Método que accede a la base de datos para obtener el nivel a editar del videojuego "Demolición", lee el fichero JSON con el contenido del nivel y lo decodifica para trasladar esa información al editor de niveles.
EditArkanoidLevel	Método que accede a la base de datos para obtener el nivel a editar del videojuego "Arkanoid", lee el fichero JSON con el contenido del nivel y lo decodifica para trasladar esa información al editor de niveles.

Nombre de la clase	
DemolitionEditorController	
Descripción	
Clase que contiene las operaciones para crear un nuevo nivel para el videojuego "Demolición".	
Responsabilidad	
Clase que se encarga de obtener los datos del editor de niveles y crear un fichero JSON con dichos datos e introducir esa información en la base de datos.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario y los datos del nivel creado mientras dure la sesión.
LevelDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
DemolitionLevel	Objetos con los datos del nivel creado para el videojuego "Demolición".
Métodos propuestos	
CreateDemolitionLevel	Este método saca de la sesión los datos del nivel creado por el usuario y crea un fichero JSON con dichos datos. Introduce en la base de datos los datos del nivel junto con la ruta donde se guardó el fichero JSON del nivel.

Nombre de la clase	
ArkanoidEditorController	
Descripción	
Clase que contiene las operaciones para crear un nuevo nivel para el videojuego “Arkanoid”.	
Responsabilidad	
Clase que se encarga de obtener los datos del editor de niveles y crear un fichero JSON con dichos datos e introducir esa información en la base de datos.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario y los datos del nivel creado mientras dure la sesión.
LevelDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
ArkanoidLevel	Objetos con los datos del nivel creado para el videojuego “Arkanoid”.
Métodos propuestos	
CreateArkanoidLevel	Este método saca de la sesión los datos del nivel creado por el usuario y crea un fichero JSON con dichos datos. Introduce en la base de datos los datos del nivel junto con la ruta donde se guardó el fichero JSON del nivel.

Nombre de la clase	
MenuController	
Descripción	
Clase que contiene las operaciones para la navegación entre las páginas web del proyecto.	
Responsabilidad	
Clase encargada de cargar las páginas que aparecen en el menú del portal web.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
Métodos propuestos	
goToUserHome	Método que accede a la página web “User Home”.
goToLevelEditor	Método que accede a la página web “Level Editor”.
goToComunity	Método que accede a la página web “Comunity”.
goToAbout	Método que accede a la página web “About”.

Nombre de la clase	
ComunityController	
Descripción	
Clase que contiene las operaciones para mostrar e interactuar con la lista de niveles creados por los usuarios.	
Responsabilidad	
Se encarga de listar los niveles creados por los usuarios y de añadir los que se seleccionen a las playlist de otros usuarios.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
LevelDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
PlaylistDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
ArkanoidLevelsList	Lista con todos los niveles existentes en la base de datos del videojuego “Arkanoid”.
DemolitionLevelsList	Lista con todos los niveles existentes en la base de datos del videojuego “Demolición”.
Métodos propuestos	
loadAllLevels	Método que carga todos los niveles que hay en la base de datos para los dos juegos que presenta la web y los meten las listas ArkanoidLevelsList y DemolitionLevelsList. Posteriormente estas listas son añadidas a la sesión para que estén disponibles en la página web.
AddLevelToPlaylist	Método que añade el nivel seleccionado a la playlist del usuario en la base de datos.

Nombre de la clase	
UserHomeController	
Descripción	
Clase que contiene las operaciones para mostrar e interactuar con la lista de niveles creados por el usuario en sesión. También contiene las operaciones para mostrar e interactuar con la playlist del usuario.	
Responsabilidad	
Se encarga de listar los niveles creados por el usuario y de mostrar la playlist del usuario.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
PlaylistDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
LevelDataManager	Objeto que se usa para llamar a los métodos de la base de datos.
Métodos propuestos	
LoadUserPlaylist	Método que carga la playlist del usuario, la mete en sesión para que la muestre la página web.
LoadUserLevels	Método que carga todos los niveles creados por el usuario, los mete en la sesión para que los muestre la página web.
PlayLevel	Método que obtiene los datos del nivel y los mete en la sesión para que el videojuego los reproduzca.
EditLevel	Método que obtiene los datos del nivel y los mete en la sesión para que el editor de niveles los reproduzca.

5.4.2.5 Subsistema: Views

Estas clases que aparecen en el diagrama anterior, nos son clases Java, sino que son páginas web JSP con trozos de JavaScript incrustado, se intentara ajustar lo más posible la definición de dichas páginas al formato de las tablas anteriores.

Nombre de la pagina	
Login	
Descripción	
Página web con un formulario para dar acceso al usuario a la página web. También proporciona un formulario para registrar un nuevo usuario.	
Responsabilidad	
Esta página será la encargada de dar acceso al usuario a la página web mediante un formulario de login. También será la encargada de crear nuevos usuarios que no posean cuenta en la página web.	
Métodos propuestos	
loginUser	Método que verifica los datos introducidos por el usuario, si son correctos el usuario accede a la página web, si no es correcto se mostrara un mensaje de error.
createUser	Método que crea un nuevo usuario en la página web. Le pasa los datos del usuario al controlador para que este lo añada a la base de datos.

Nombre de la pagina	
User Home	
Descripción	
Página web donde se muestran los niveles creados por el usuario y la lista de niveles que el usuario ha seleccionado para jugar. La página también tendrá los motores para mover los juegos que los usuarios tienen en sus listas.	
Responsabilidad	
Esta página es la encargada de listar los niveles creados por el jugador logeado y la playlist del usuario. También será la responsable de mover los juegos.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
Métodos propuestos	
PlayLevel	Método que lee los datos del nivel guardados en la sesión y los carga en el juego.
EditLevel	Método que llama a la herramienta de edición de niveles para cargar el nivel seleccionado en ella y poder editarlo.
MoveCanon	Método que mueve el cañón del juego “Demolición” según la tecla pulsada.
Shot	Método que dispara el cañón del juego “Demolición”.
MoveStick	Método que mueve el stick del juego “Arkanoid” según la tecla pulsada.
ReplayLevel	Método que reinicia el nivel que se está jugando en ese momento.
BackToHome	Método que sale del juego y vuelve a la página USer Home.

Nombre de la pagina	
Level Editor	
Descripción	
Página con una herramienta para crear niveles para los videojuegos “Demolición” y “Arkanoid” presentados en este proyecto. La creación de estos niveles se basara en mecánicas Drag & Drop.	
Responsabilidad	
Esta página se encarga de presentar una herramienta de creación de niveles, los niveles creados por esta herramienta se introducirán en la base de datos y se guardaran en formato JSON.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
Métodos propuestos	
CreateGrid	Método para generar una cuadricula que sea la zona donde se colocaran los elementos para crear el nivel.
DeletePiece	Método para eliminar una pieza colocada en la zona de edición.
ValidateLevelName	Método que verifica si el nombre del nivel está en la base de datos o no.
SaveLevel	Método que empaqueta todos los datos del nivel creado y pasarlo al controlador para que lo meta en la base de datos y generar el fichero JSON.

Nombre de la pagina	
Comunity	
Descripción	
Página que muestra una lista con todos los niveles creados para los dos juegos presentes en la página web.	
Responsabilidad	
Esta página se encarga de listar los niveles creados por los demás usuarios y de añadirlos a las playlist.	
Atributos propuestos	
Session	Objeto que se utiliza en toda la web para mantener los datos del usuario mientras dure la sesión.
Métodos propuestos	
LoadAllLevels	Obtiene de la sesión la lista de niveles que se han cargado de la base de datos y los muestra en la página web.
AddLevelToPlaylist	Llama al controlador para pasarle el nivel que ha seleccionado el usuario, para que sea introducido en su playlist en la base de datos.

Nombre de la pagina	
About	
Descripción	
Página que muestra información sobre la propia página web y sobre el autor del proyecto.	
Responsabilidad	
Mostrar información al usuario.	

5.5 Análisis de Casos de Uso y Escenarios

5.5.1 Casos de uso referentes a la página web

5.5.1.1 Caso de Uso 1: Login

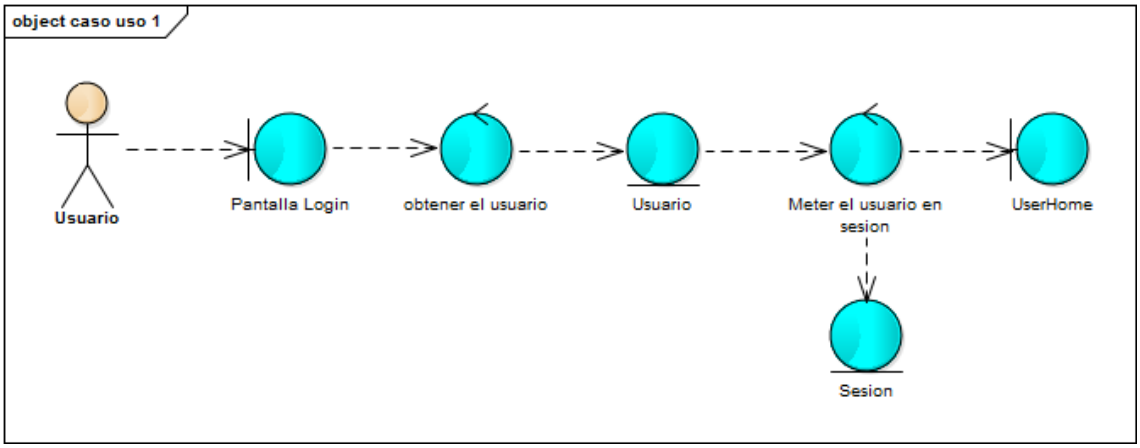


Figura 23 Diagrama de robustez 1: Login

Login	
Postcondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Actores	Usuario
Descripción	1. Se muestra al usuario una página con un formulario para acceder a la web. 2. El usuario escribe su nombre de usuario. 3. El usuario escribe su contraseña. 4. Se pulsa el botón "Login". 5. Se comprueban los datos introducidos con los guardados en la base de datos. 6. Se crea un objeto usuario y se introduce en sesión. 7. El usuario accede a la página web.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El usuario pulsa el botón "Login" y alguno de los campos necesario para acceder están vacíos. Se notifica al usuario que los campos han de estar rellenados para poder acceder.

Excepciones	• **El usuario ha introducido mal el nombre o la contraseña:** El nombre o la contraseña introducidos han de coincidir con los guardados en la base de datos.
	  ○ Notificar el error.
	• **El usuario ha introducido un nombre que no existe:** El nombre introducido ha de existir en la base de datos.
	  ○ Notificar el error.

5.5.1.2 Caso de Uso 2: Logout

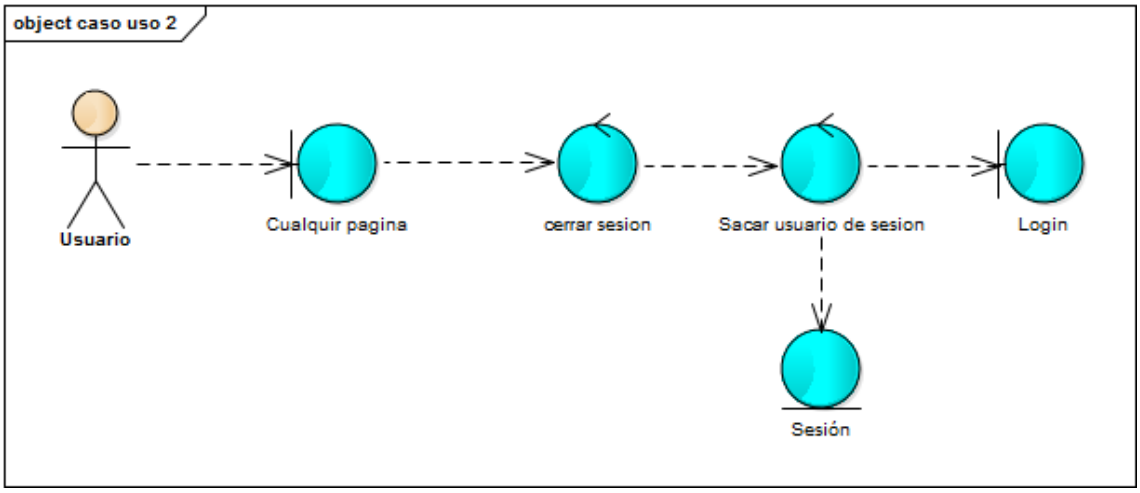


Figura 24 Diagrama de robustez 2: Logout

Logout	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Postcondiciones	Ya no hay ningún objeto usuario dentro de la sesión.
Actores	Usuario
Descripción	1. El usuario pulsa el enlace “Logout”. 2. Se elimina el objeto usuario de la sesión. 3. Se regresa a la pantalla “Login”.

5.5.1.3 Caso de Uso 3: Crear usuario

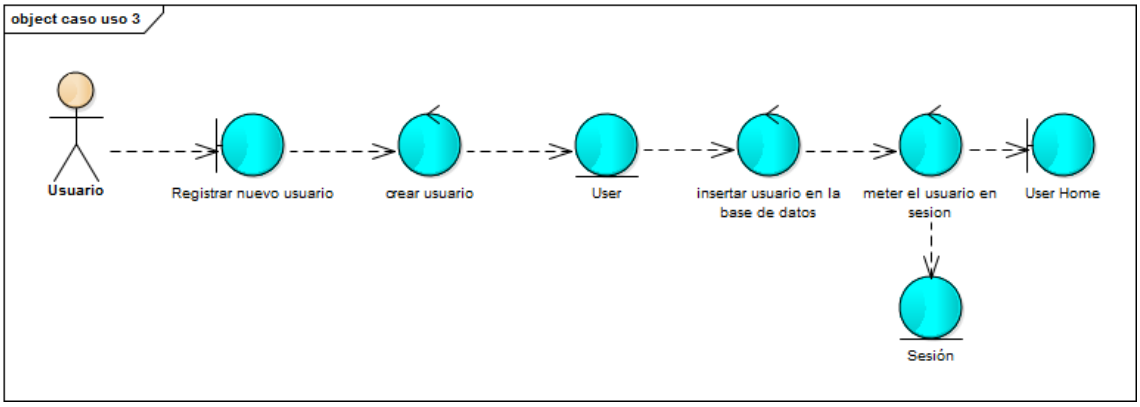


Figura 25 Diagrama de robustez 3: Crear usuario

Crear usuario	
Postcondiciones	Ha de existir un nuevo usuario dentro del sistema.
Actores	Usuario.
Descripción	1. Se muestra al usuario una página con un formulario para que introduzca los datos del nuevo usuario. 2. El usuario escribe el nombre de usuario. 3. El usuario escriba la contraseña. 4. El usuario escribe otra vez la contraseña. 5. Pulsa el botón "Register". 6. Se comprueba que el nombre de usuario no exista en la base de datos. 7. Se crea un objeto usuario y se mete en sesión. 8. El usuario accede a la página web.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El usuario pulsa el botón "Cancel". Se regresa a la página "Login". • **Escenario Alternativo 2:** El usuario pulsa el botón "Register" con alguno de los campos obligatorios vacío. Se notifica al usuario que los campos han de estar rellenados para poder crear el usuario.

Excepciones	• El usuario ha introducido un nombre que ya existe: No pueden existir dos usuarios con el mismo nombre.
	  ○ Notificar el error.
	• Las contraseñas introducidas no coinciden: La contraseña y la repetición han de ser iguales para poder crear el usuario.
	  ○ Notificar el error.

5.5.1.4 Caso de Uso 4: Ir al home de usuario

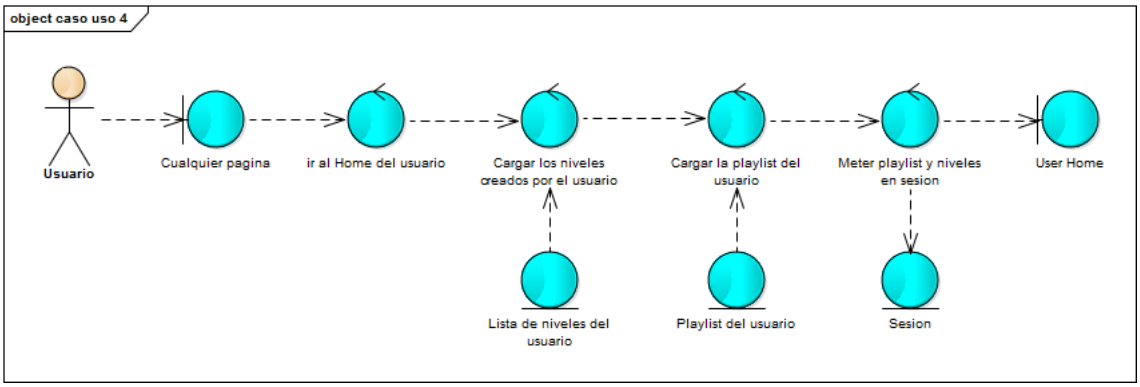


Figura 26 Diagrama de robustez 4: Ir al home de usuario

Ir al Home del usuario	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Postcondiciones	Han de existir dentro de la sesión el objeto Playlist y la lista de niveles creados por el usuario.
Actores	Usuario
Descripción	1. Se pulsa el botón "Home" del menú superior. 2. Se obtiene de la base de datos la lista de niveles creados por el usuario y se añade a la sesión. 3. Se obtiene de la base de datos la lista de niveles que el usuario tiene en su playlist. 4. Se crea el objeto playlist con esa información y se añade a sesión. 5. Se accede a la página home del usuario y esta muestra la lista de niveles creados por el usuario y la playlist.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** Acceder a través del logo. En el paso 1 en vez de acceder desde el menú, se accede pulsando sobre el logotipo de la página web en el menú superior.

5.5.1.5 Caso de Uso 5: Ir a editor de niveles, "Demolition"

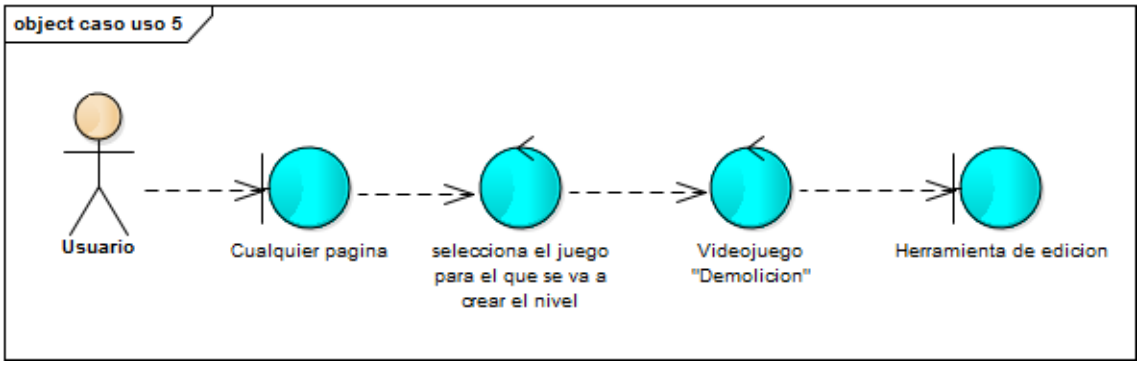


Figura 27 Diagrama de robustez 5: Ir al editor de niveles, "Demolition"

Ir al editor de niveles: Demolición	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Actores	Usuario
Descripción	1. Se pulsa el botón "Level Creator" del menú superior. 2. Se accede a la página para seleccionar el juego para el que se va a crear el nivel. 3. Se selecciona el juego "Demolition". 4. Se carga la página con la herramienta de edición.

5.5.1.6 Caso de Uso 6: Ir al editor de niveles, "Arkanoid"

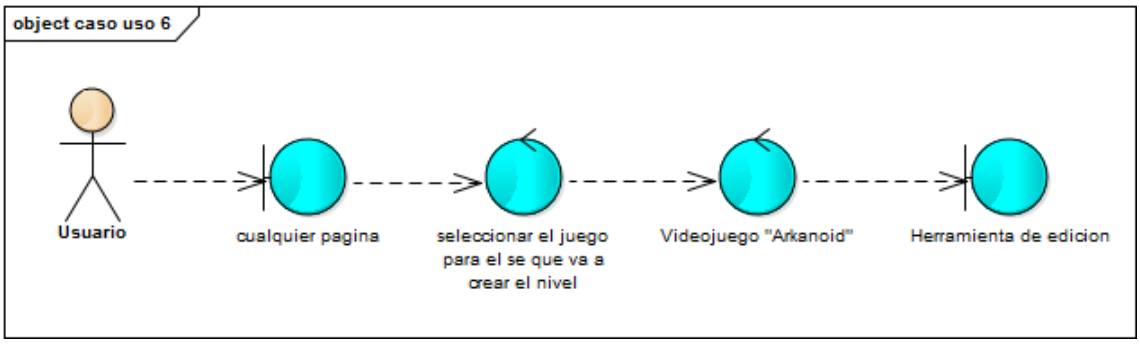


Figura 28 Diagrama de robustez 6: Ir al editor de niveles, "Arkanoid"

Ir al editor de niveles: Arkanoid	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Actores	Usuario
Descripción	1. Se pulsa el botón "Level Creator" del menú superior. 2. Se accede a la página para seleccionar el juego para el que se va a crear el nivel. 3. Se selecciona el juego "Arkanoid". 4. Se carga la página con la herramienta de edición.

5.5.1.7 Caso de Uso 7: Ir a la página "Community"

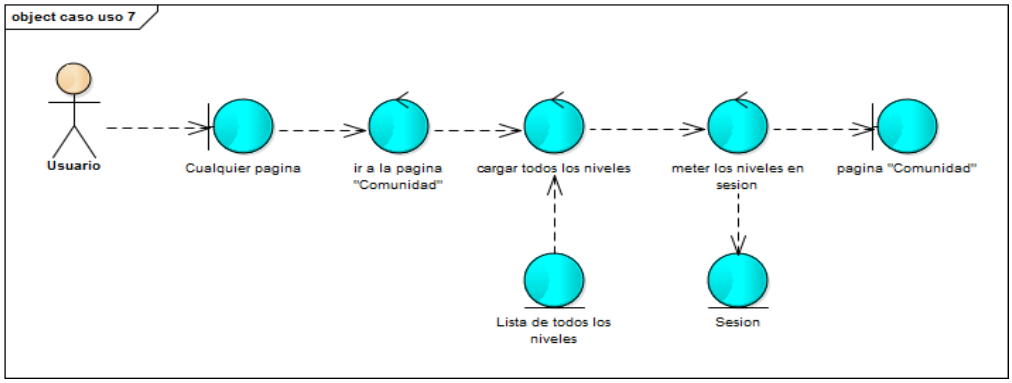


Figura 29 Diagrama de robustez 7: Ir a la página "Community"

Ir a la página "Community"	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Postcondiciones	Han de existir dentro de la sesión una la lista con todos los niveles existentes.
Actores	Usuario
Descripción	1. Se pulsa el botón "Community" del menú superior. 2. Se accede a la base de datos para obtener una lista de todos los niveles existentes. 3. Se añade a la sesión la lista de niveles. 4. Se carga la página "Community" en la que se muestran todo los niveles cargados.

5.5.1.8 Caso de Uso 8: Ir a la página "About"

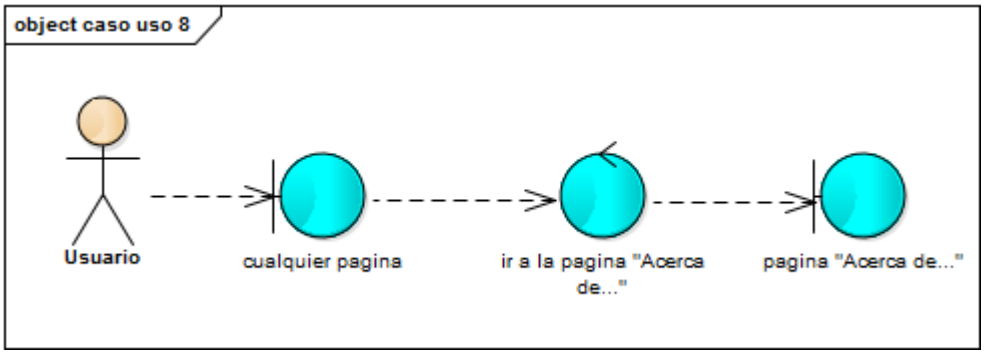


Figura 30 Diagrama de robustez 8: Ir a la página "About"

Ir a la página "About"	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Actores	Usuario
Descripción	1. Se pulsa el botón "About" del menú superior. 2. Se carga la página "About".

5.5.1.9 Caso de Uso 9: Jugar nivel

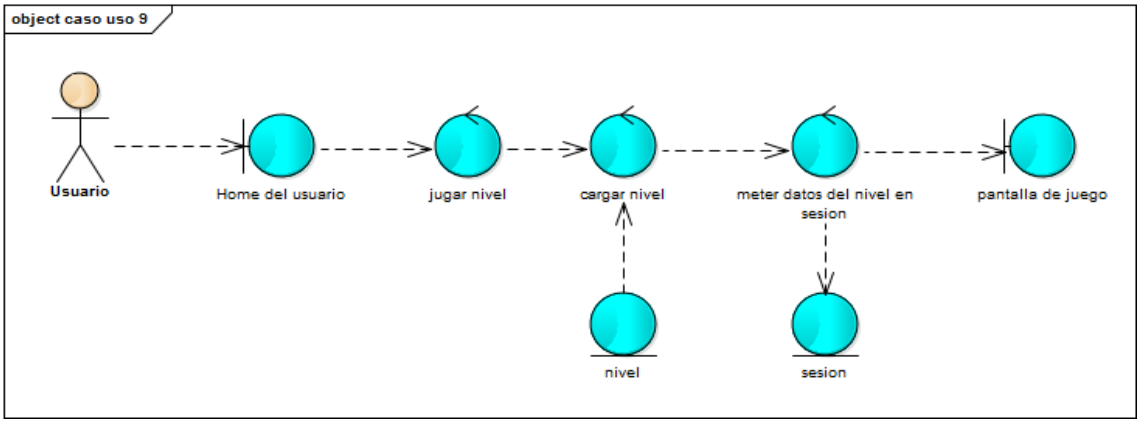


Figura 31 Diagrama de robustez 9: Jugar nivel

Jugar nivel	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Postcondiciones	Ha de existir dentro de la sesión el nivel que se vaya a jugar.
Actores	Usuario
Descripción	1. Se pulsa el botón "Play" de uno de los niveles dentro del "Home" del usuario. 2. Se crea un objeto nivel vacío. 3. Se busca en la base de datos la información sobre el nivel y la ruta donde está guardado el fichero JSON que lo contiene. 4. Se lee el fichero JSON y se decodifica. 5. Se rellena el objeto nivel creado anteriormente con la información decodificada del fichero. 6. Se añade el objeto nivel a la sesión. 7. Se carga la pantalla del juego y el código JavaScript de esta página obtiene los datos del nivel de la sesión para construirlo.

Excepciones	• No existe el fichero: El fichero ha de estar en la dirección indicada.
	  ○ Notificar el error.
	• El formato del fichero no es el correcto: El formato del fichero ha de ser un fichero JSON específico para los niveles.
	  ○ Notificar el error.

5.5.1.10 Caso de Uso 10: Editar nivel

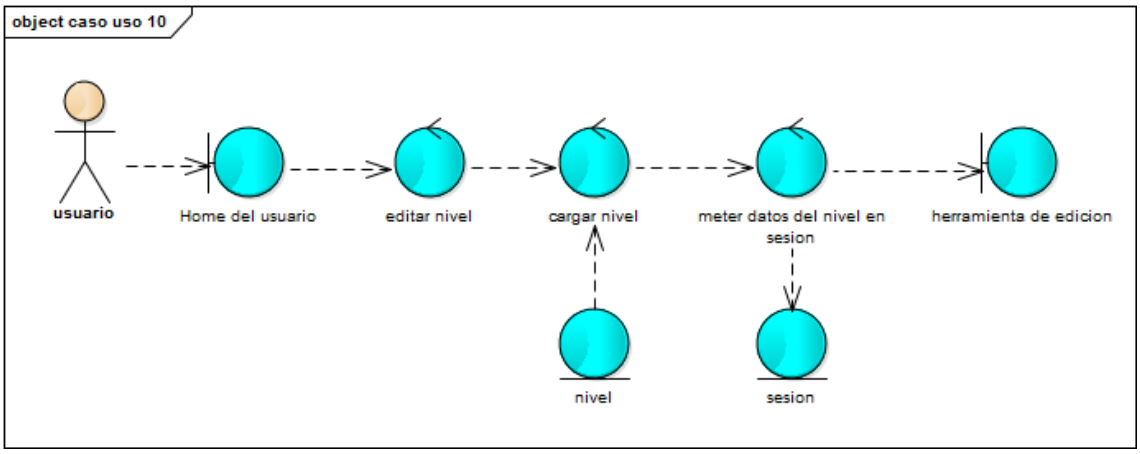


Figura 32 Diagrama de robustez 10: Editar nivel

Editar nivel	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Postcondiciones	Ha de existir dentro de la sesión el nivel que se vaya a editar.
Actores	Usuario
Descripción	1. Se pulsa el botón "Edit" de uno de los niveles dentro del "Home" del usuario. 2. Se crea un objeto nivel vacío. 3. Se busca en la base de datos la información sobre el nivel y la ruta donde está guardado el fichero JSON que lo contiene. 4. Se lee el fichero JSON y se decodifica. 5. Se rellena el objeto nivel creado anteriormente con la información decodificada del fichero. 6. Se añade el objeto nivel a la sesión. 7. Se carga la herramienta de edición y se colocan en ella todas las piezas y los datos del nivel creado para su edición.

Excepciones	• No existe el fichero: El fichero ha de estar en la dirección indicada.
	  ○ Notificar el error.
	• El formato del fichero no es el correcto: El formato del fichero ha de ser un fichero JSON específico para los niveles.
	  ○ Notificar el error.

5.5.1.11 Caso de Uso 11: Añadir nivel a la playlist del usuario

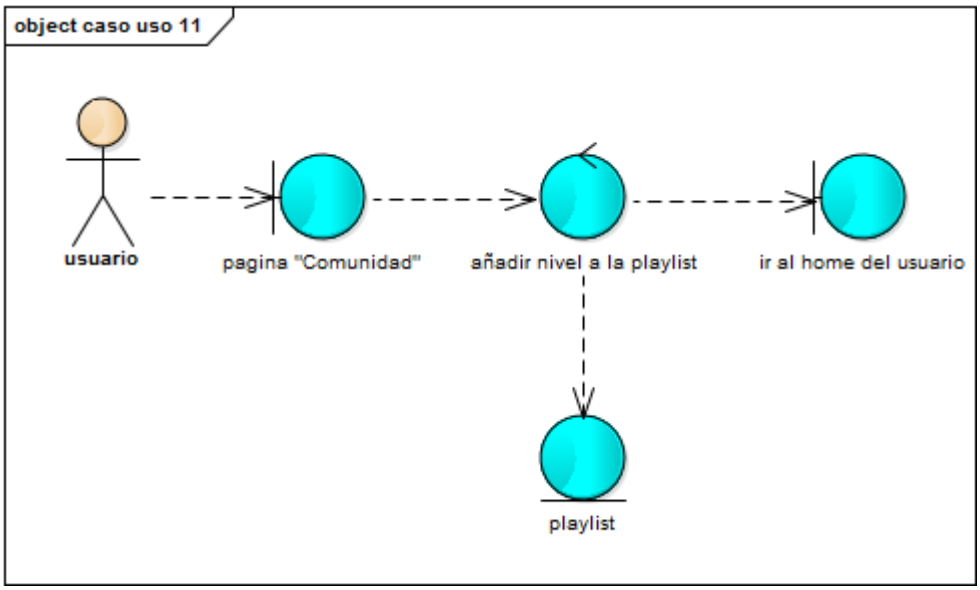


Figura 33 Diagrama de robustez 11: Añadir nivel a la playlist del usuario

Añadir nivel a la playlist del usuario	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Postcondiciones	Tiene que existir un nivel más en la playlist del usuario.
Actores	Usuario
Descripción	1. Se pulsa el botón "Add to playlist" de uno de los niveles dentro de la página "Community". 2. Se añade el nivel seleccionado a la playlist del usuario dentro de la base de datos. 3. Se carga la página "Home" del usuario con el nuevo nivel añadido en su playlist.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El usuario añade un nivel que ya estaba en sui playlist. Cuando esto ocurre el proceso completo fluye con normalidad ya que la propia restricción de la base de datos no permite añadir un nivel que ya estaba.

5.5.2 Casos de uso referentes a los videojuegos en común.

5.5.2.1 Caso de Uso 12: Valorar nivel

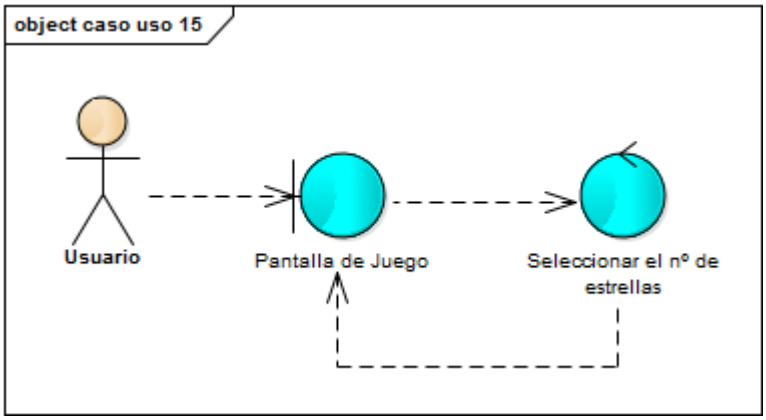


Figura 34 Diagrama de robustez 12: Valorar nivel

Valorar nivel	
Actores	Usuario
Descripción	1. Seleccionar la cantidad de estrellas (1 a 5).

5.5.2.2 Caso de Uso 13: Volver a jugar nivel

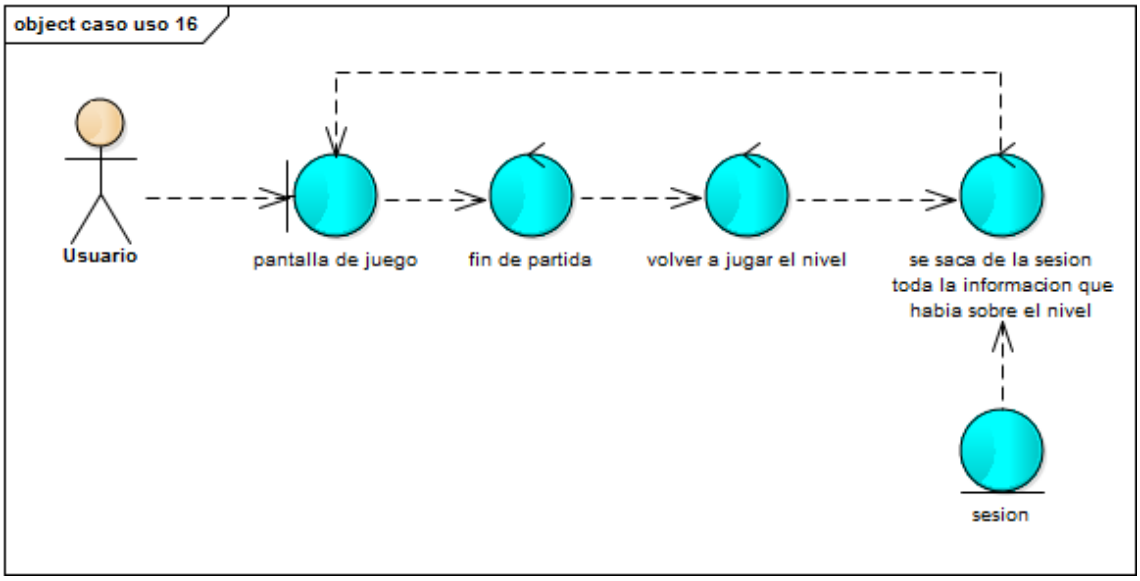


Figura 35 Diagrama de robustez 13: Volver a jugar nivel

Volver a jugar el nivel	
Precondiciones	Ha de haber un objeto nivel en la sesión.
Actores	Usuario
Descripción	1. Fin de partida. 2. Se pulsa el botón “Play again”. 3. Se saca la información del nivel de la sesión. 4. Se recarga la pantalla de juego con el nivel reiniciado.

5.5.2.3 Caso de Uso 14: Volver al home del usuario

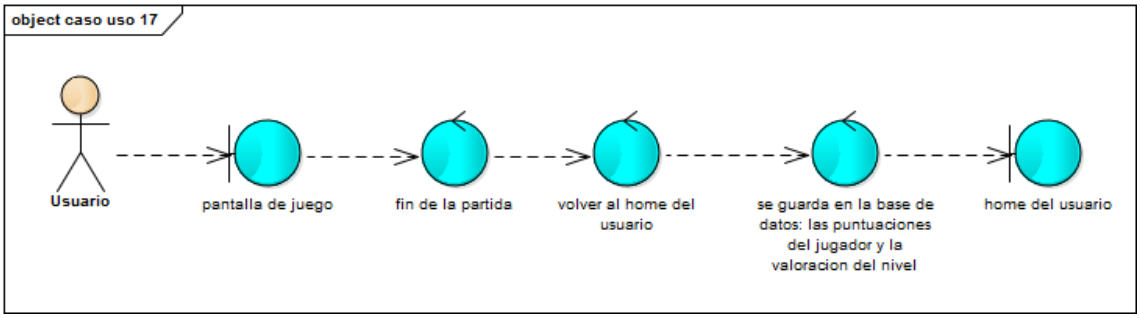


Figura 36 Diagrama de robustez 14: Volver al home del usuario

Volver al Home del usuario	
Precondiciones	Tiene que haber un objeto usuario dentro de la sesión.
Postcondiciones	La información del jugador sobre el nivel tiene que haberse modificado.
Actores	Usuario
Descripción	1. Fin de partida. 2. Se pulsa el botón "Back to user home". 3. Se modifican las estadísticas del jugador y la valoración del nivel en la base de datos. 4. Se carga el home del usuario con los datos actualizados.

5.5.3 Casos de uso referentes al juego "Demolition"

5.5.3.1 Caso de Uso 15: Rotar cañón hacia arriba

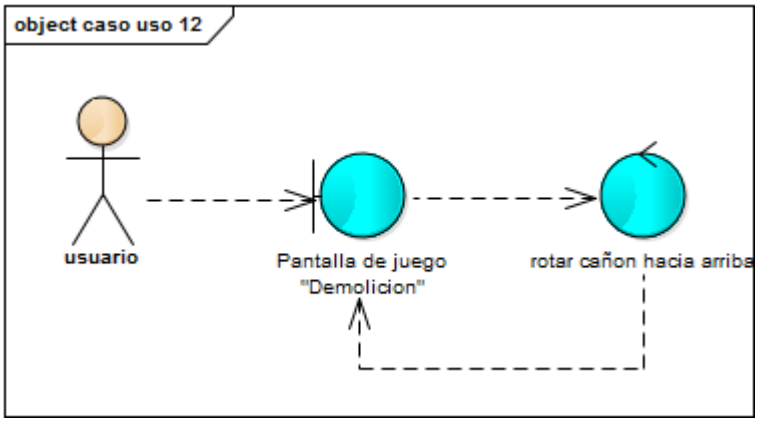


Figura 37 Diagrama de robustez 15: Rotar cañón hacia arriba

Rotar cañón hacia arriba	
Actores	Usuario
Descripción	1. Se pulsa la tecla asignada para rotar el cañón hacia arriba. 2. El cañón hace un pequeño movimiento de rotación hacia arriba.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El jugador deja pulsado la tecla de rotación. En este caso el movimiento del cañón es continuo hasta llegar al tope de rotación que puede alcanzar. • **Escenario Alternativo 2:** El jugador Rota el cañón estando este rotado al máximo. Cuando esto ocurre el cañón deja de moverse, ya que ha alcanzado el máximo de rotación.

5.5.3.2 Caso de Uso 16: Rotar cañón hacia abajo

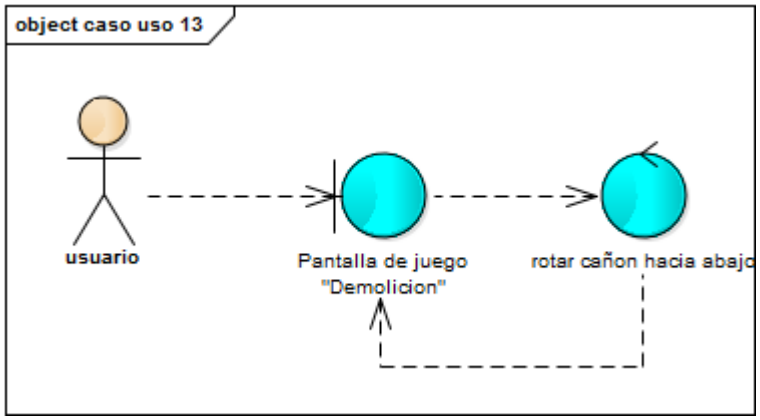


Figura 38 Diagrama de robustez 16: Rotar cañón hacia abajo

Rotar cañón hacia abajo	
Actores	Usuario
Descripción	1. Se pulsa la tecla asignada para rotar el cañón hacia abajo. 2. El cañón hace un pequeño movimiento de rotación hacia abajo.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El jugador deja pulsado la tecla de rotación. En este caso el movimiento del cañón es continuo hasta llegar al tope de rotación que puede alcanzar. • **Escenario Alternativo 2:** El jugador Rota el cañón estando este rotado al mínimo. Cuando esto ocurre el cañón deja de moverse, ya que ha alcanzado el mínimo de rotación.

5.5.3.3 Caso de Uso 17: Disparar el cañon

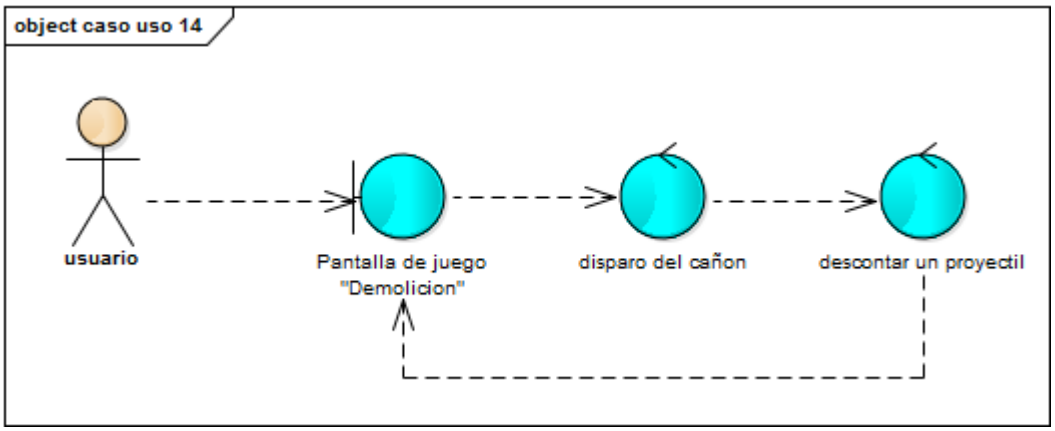


Figura 39 Diagrama de robustez 17: Disparar el cañón

Disparar el cañón	
Precondiciones	El contador de proyectiles restantes ha de estar como mínimo a 1.
Postcondiciones	El contador de proyectiles tiene un proyectil menos.
Actores	Usuario
Descripción	1. Se pulsa la tecla asignada para disparar el cañón. 2. Un proyectil sale disparado des de la punta del cañón. 3. Se descuenta una unidad el contador de proyectiles restantes.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** Disparar con el contador a 0. No sale ningún proyectil por el cañón.

5.5.4 Casos de uso referentes al juego "Arkanoid"

5.5.4.1 Caso de Uso 18: Mover el stick del jugador hacia la derecha

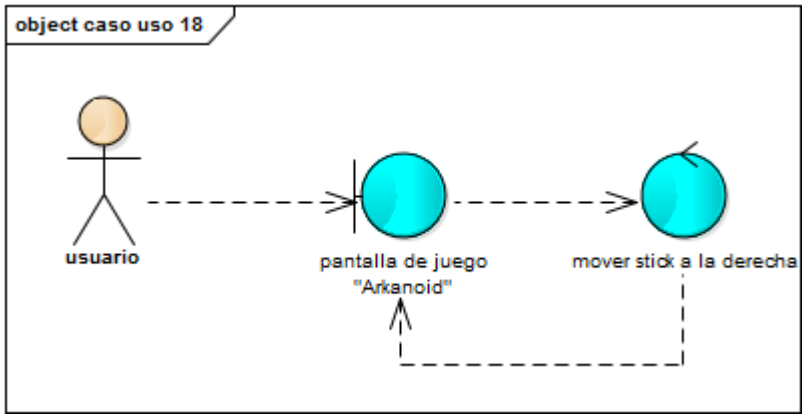


Figura 40 Diagrama de robustez 18: Mover el stick del jugador hacia la derecha

Mover stick del jugador hacia la derecha	
Actores	Usuario
Descripción	1. Se pulsa la tecla asignada para mover el stick hacia la derecha. 2. El stick hace un pequeño movimiento hacia la derecha.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El jugador deja pulsado la tecla de movimiento. En este caso el movimiento del stick es continuo hasta llegar al límite derecho de la pantalla de juego. • **Escenario Alternativo 2:** El jugador mueve el stick estando este en el límite de la pantalla. Cuando esto ocurre el stick deja de moverse, ya que ha alcanzado el límite de la pantalla.

5.5.4.2 Caso de Uso 19: Mover el stick del jugador hacia la izquierda

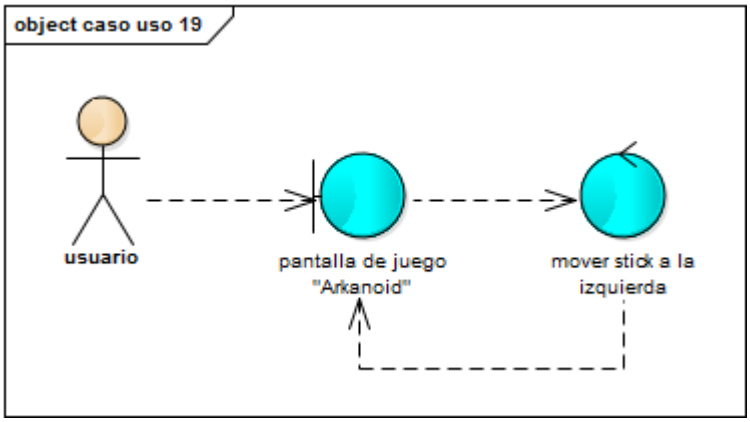


Figura 41 Diagrama de robustez 19: Mover stick del jugador hacia la izquierda

Mover stick del jugador hacia la izquierda	
Actores	Usuario
Descripción	1. Se pulsa la tecla asignada para mover el stick hacia la izquierda. 2. El stick hace un pequeño movimiento hacia la izquierda.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El jugador deja pulsado la tecla de movimiento. En este caso el movimiento del stick es continuo hasta llegar al límite izquierdo de la pantalla de juego. • **Escenario Alternativo 2:** El jugador mueve el stick estando este en el límite de la pantalla. Cuando esto ocurre el stick deja de moverse, ya que ha alcanzado el límite de la pantalla.

5.5.5 Casos de uso referentes a la herramienta de edición de niveles en general.

5.5.5.1 Caso de Uso 20: Arrastrar pieza de la paleta a la cuadricula

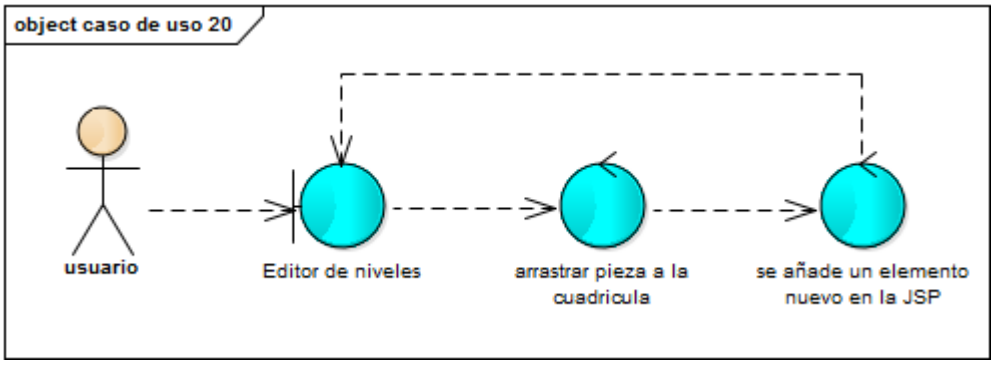


Figura 42 Diagrama de robustez 20: Arrastrar pieza de la paleta a la cuadricula

Arrastrar pieza de la paleta a la cuadricula	
Precondiciones	La lista de piezas colocadas en la cuadricula ha de esta vacía.
Postcondiciones	La lista de piezas colocadas en la cuadricula ha se ser mayor a 0.
Actores	Usuario
Descripción	1. Se coloca el ratón sobre una de las piezas de la paleta. 2. Con el botón izquierdo del ratón se arrastra una pieza a la cuadricula. 3. Se suelta el botón del ratón para colocar la pieza. 4. La pieza se añade a la lista de piezas colocadas en la cuadricula.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El usuario arrastra una pieza fuera de la cuadricula. En este caso la pieza desaparece, no se coloca en la cuadricula y no se añade a la lista de piezas colocadas.

5.5.5.2 Caso de Uso 21: Mover pieza por la cuadricula

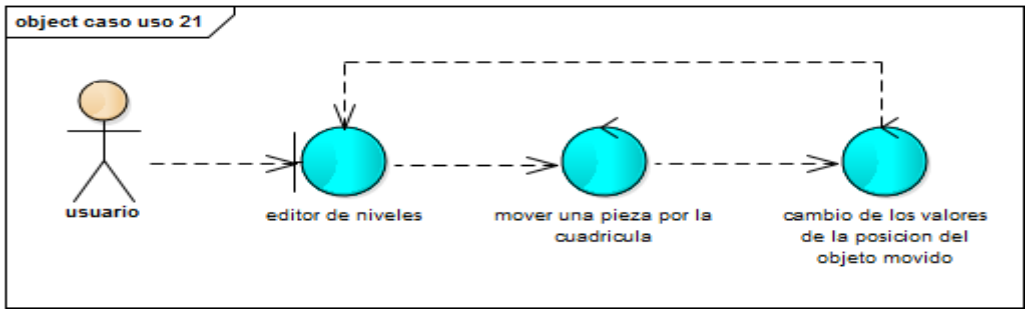


Figura 43 Diagrama de robustez 21: Mover pieza por la cuadricula

Mover pieza por la cuadricula	
Precondiciones	La lista de piezas colocadas en la cuadricula ha se ser mayor a 0.
Postcondiciones	La lista de piezas colocadas en la cuadricula no ha de variar.
Actores	Usuario
Descripción	1. Se coloca el ratón sobre una de las piezas colocadas en la cuadricula. 2. Con el botón izquierdo del ratón se arrastra la pieza a otra posición en la cuadricula. 3. Se suelta el botón del ratón para colocar la pieza.

5.5.5.3 Caso de Uso 22: Mover pieza fuera de la cuadricula

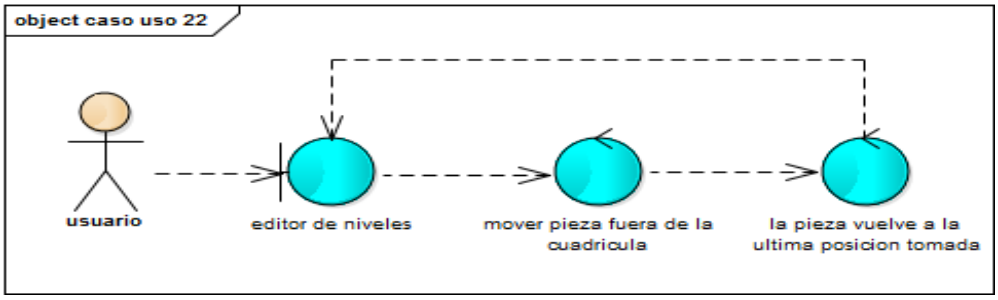


Figura 44 Diagrama de robustez 22: Mover pieza fuera de la cuadricula

Mover pieza fuera de la cuadricula	
Precondiciones	La lista de piezas colocadas en la cuadricula ha se ser mayor a 0.
Postcondiciones	La lista de piezas colocadas en la cuadricula no ha de variar.
Actores	Usuario
Descripción	1. Se coloca el ratón sobre una de las piezas colocadas en la cuadricula. 2. Con el botón izquierdo del ratón se arrastra la pieza fuera de la cuadricula. 3. Se suelta el botón del ratón para colocar la pieza. 4. La pieza vuelve a la última posición en la que se encontraba.

5.5.5.4 Caso de Uso 23: Eliminar pieza

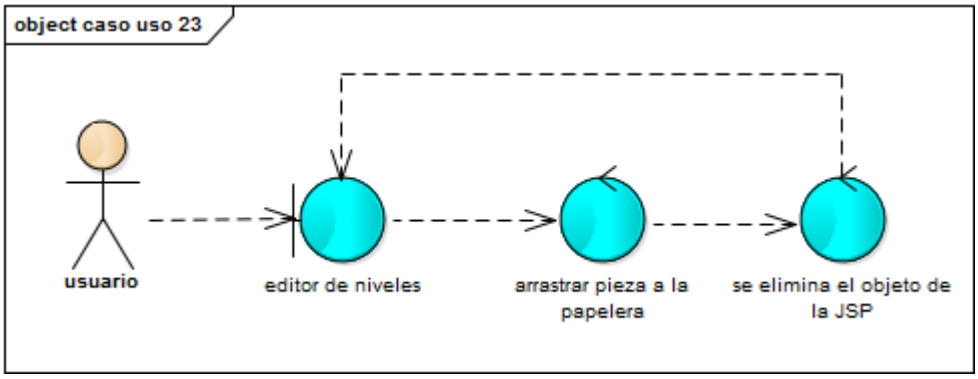


Figura 45 Diagrama de robustez 23: Eliminar pieza

Eliminar pieza	
Precondiciones	La lista de piezas colocadas en la cuadricula ha se ser mayor a 0.
Postcondiciones	La lista de piezas colocadas en la cuadricula ha de contar con una pieza menos.
Actores	Usuario
Descripción	1. Se coloca el ratón sobre una de las piezas colocadas en la cuadricula. 2. Con el botón izquierdo del ratón se arrastra la pieza a la papelera. 3. La pieza es eliminada de la lista de piezas colocadas en la cuadricula. 4. La pieza desaparece de la pantalla.

5.5.5.5 Caso de Uso 24: Escribir el nombre del nivel

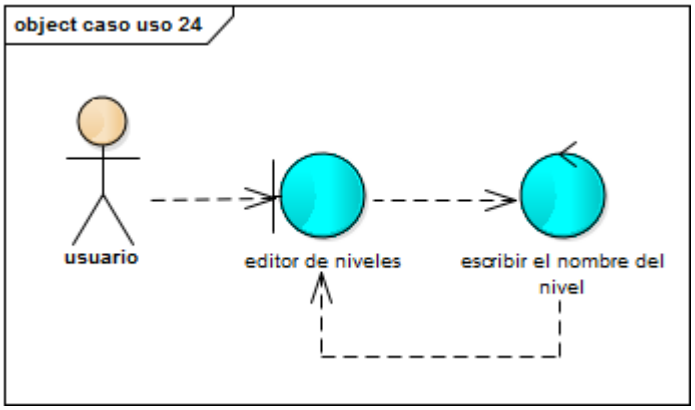


Figura 46 Diagrama de robustez 24: Escribir el nombre del nivel

Escribir el nombre del nivel	
Actores	Usuario
Descripción	1. Se escribe el nombre del nivel en el campo reservado para el nombre.

5.5.5.6 Caso de Uso 25: Escribir la descripción del nivel

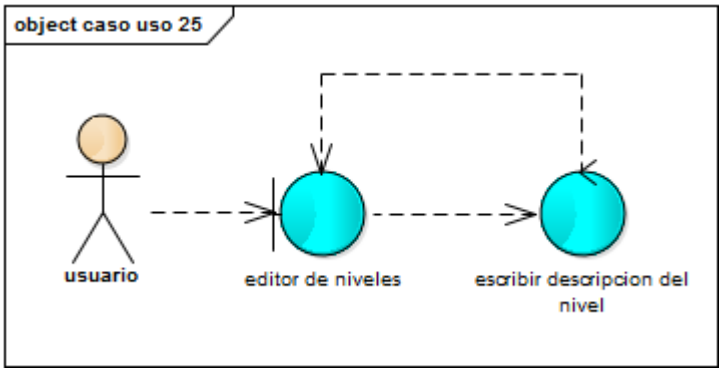


Figura 47 Diagrama de robustez 25: Escribir la descripción del nivel

Escribir la descripción del nivel	
Actores	Usuario
Descripción	1. Se escribe la descripción del nivel en el campo reservado para la descripción.

5.5.5.7 Caso de Uso 26: Guardar nivel

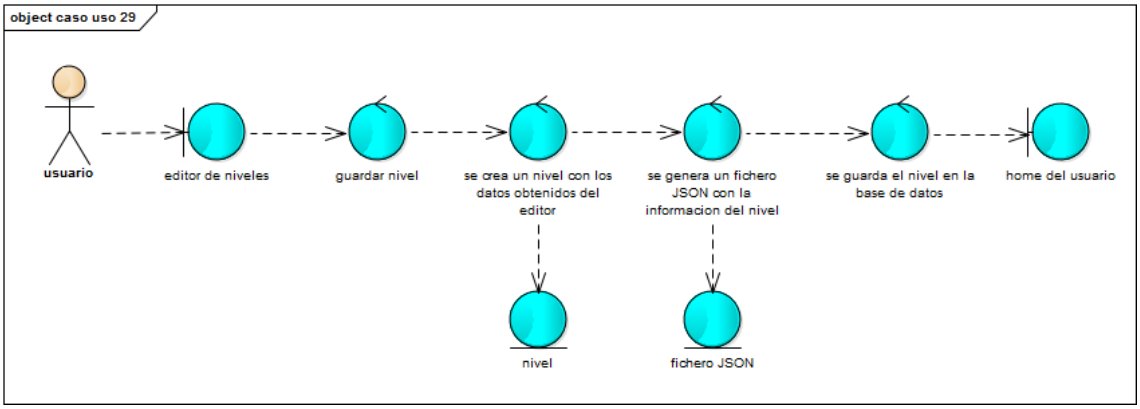


Figura 48 Diagrama de robustez 26: Guardar nivel

Guardar nivel	
Postcondiciones	Ha de existir un fichero JSON con el nivel creado.
Actores	Usuario
Descripción	1. Se pulsa el botón "Save & Share". 2. Se comprueba en la base de datos si el nombre introducido para el nivel es válido. 3. Se crea un objeto nivel con la información obtenida de la herramienta de edición. 4. Se genera un fichero JSON con el contenido den nivel. 5. Se guarda el nivel en la base de datos. 6. Se carga la página "Home" del usuario.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** El usuario introduce un nombre que ya está en la base de datos. Si el nombre ya está en la base de datos, el campo de texto se iluminara en rojo y no permitirá guardar el nivel.

5.5.6 Casos de uso referentes a la herramienta de edición de niveles del juego "Demolition"

5.5.6.1 Caso de Uso 27: Seleccionar posición del cañón

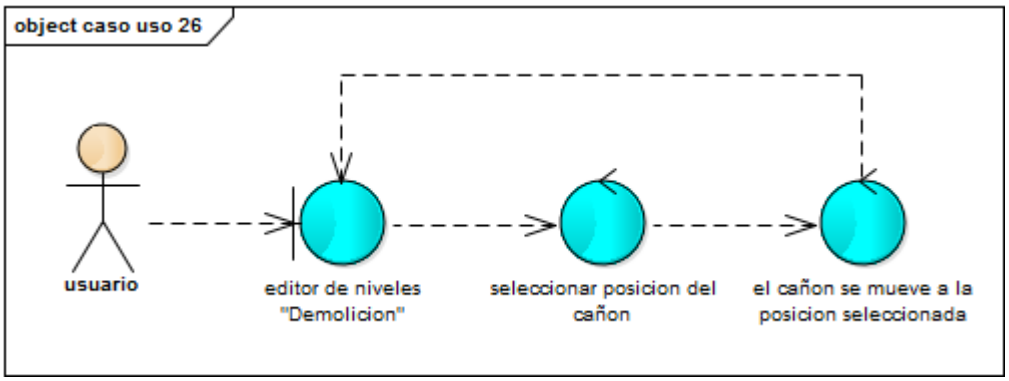


Figura 49 Diagrama de robustez 27: Seleccionar posición del cañón

Seleccionar posición del cañón	
Actores	Usuario
Descripción	1. Se selecciona un valor en el campo de selección de la posición del cañón. 2. El cañón se mueve hasta la posición seleccionada.

5.5.6.2 Caso de Uso 28: Seleccionar posición del la línea limite de altura

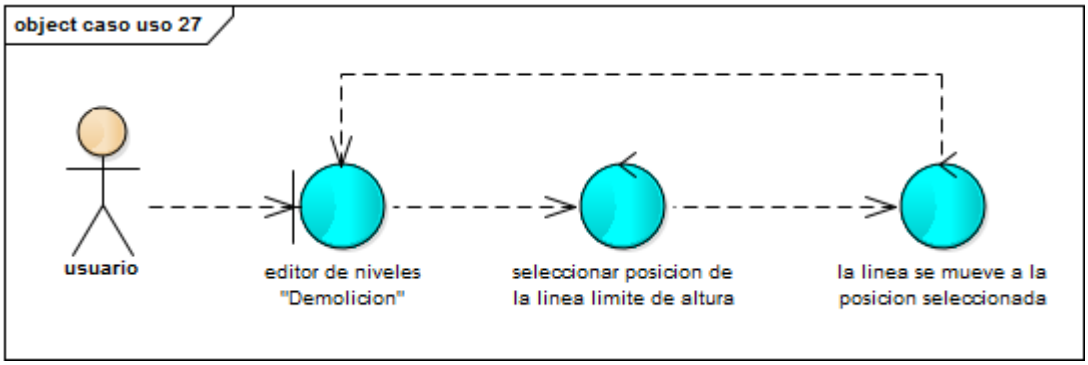


Figura 50 Diagrama de robustez 28: Seleccionar posición de la línea límite de altura

Seleccionar posición de la línea límite de altura	
Actores	Usuario
Descripción	1. Se selecciona un valor en el campo de selección de la posición de la línea límite de altura del nivel. 2. La línea límite se mueve hasta la posición seleccionada.

5.5.6.3 Caso de Uso 29: Seleccionar el número de proyectiles

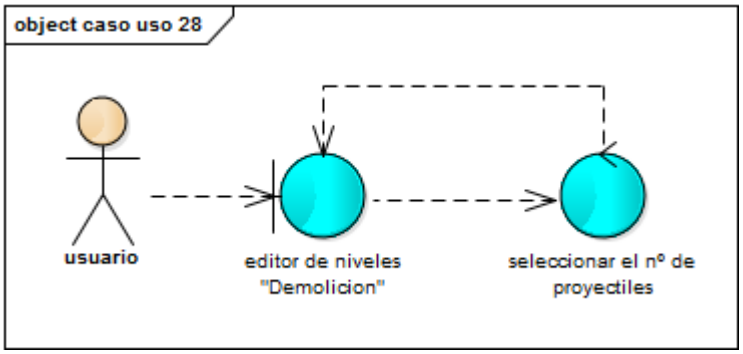


Figura 51 Diagrama de robustez 29: Seleccionar el número de proyectiles

Seleccionar el número de proyectiles	
Actores	Usuario
Descripción	1. Se selecciona el número de proyectiles que se podrán utilizar en el nivel.

5.6 Análisis de Interfaces de Usuario

En esta sección se describirán todas las páginas que el usuario se va a encontrar dentro del portal web de forma detallada.

5.6.1 Descripción de la Interfaz

5.6.1.1 Página Login

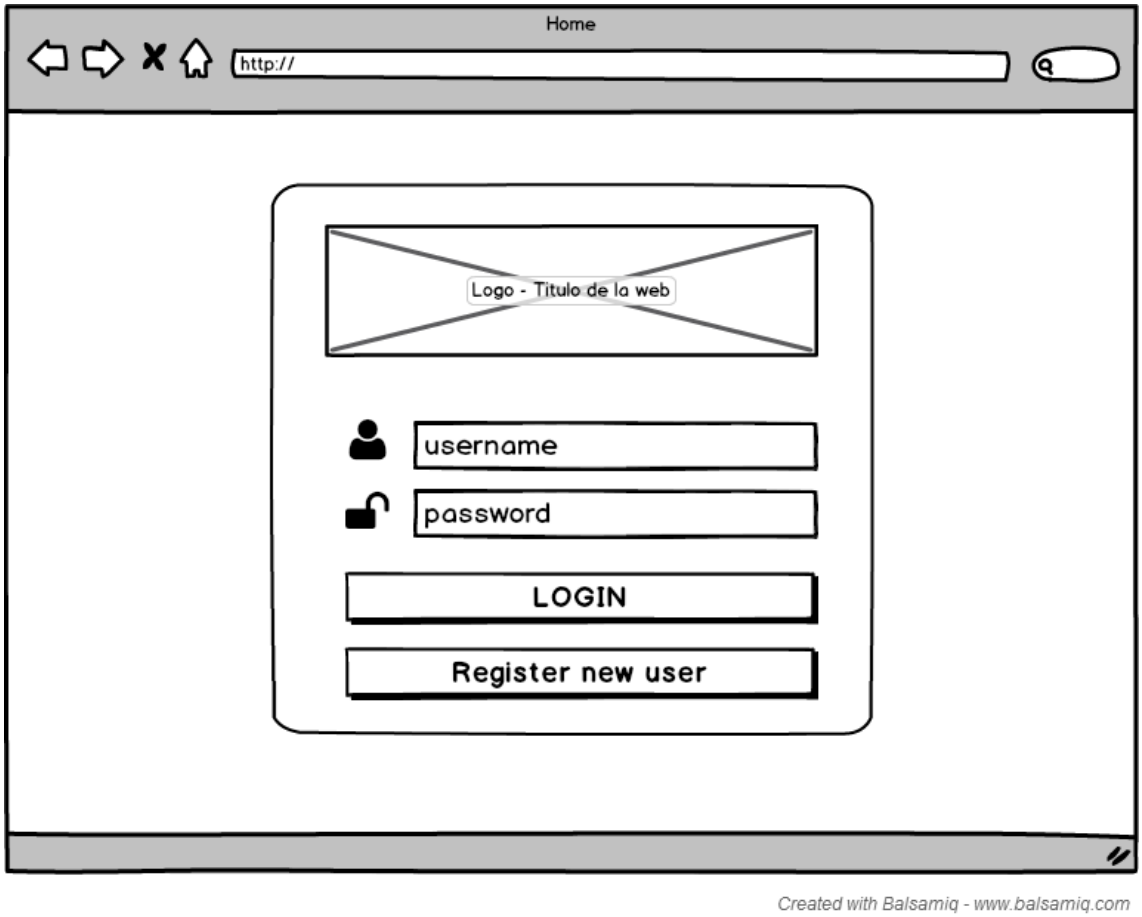


Figura 52 Boceto preliminar de la página "Login"

Esta es la primera página que el usuario vera al entrar en el portal web, por lo que se puede decir que esta página será el **Index** del portal web. Esta pantalla tiene la función de dar acceso al usuario a la aplicación.

Esta página se compone de dos campos de texto: uno para introducir el nombre de usuario y otro para introducir la contraseña del usuario.

También presenta dos botones en la parte inferior del bloque central, el primero de ellos para acceder a la página web y el segundo para acceder a la página de registro de un nuevo usuario.

5.6.1.2 Página Register new user

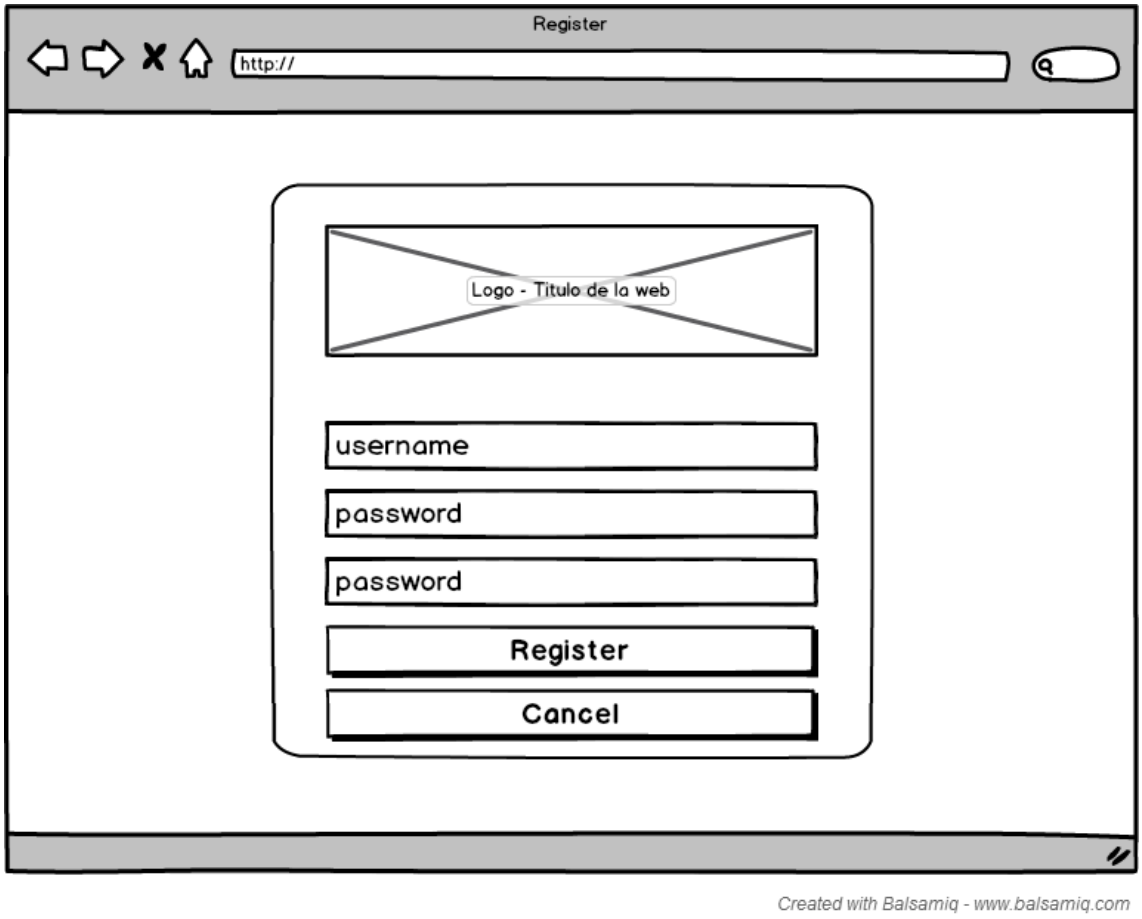


Figura 53 Boceto preliminar de la página "Register"

Esta es la página destinada a crear nuevos usuarios para el portal web. En ella se presentan tres campos de texto, el primero para el nombre de usuario, el siguiente para introducir la contraseña de acceso y el tercero para repetir la contraseña a modo de confirmación.

Por último se encuentra un botón para confirmar la operación y acceder a la página con el nuevo usuario recién creado y otro para cancelar la operación y volver a la pantalla de login.

5.6.1.3 Página User Home

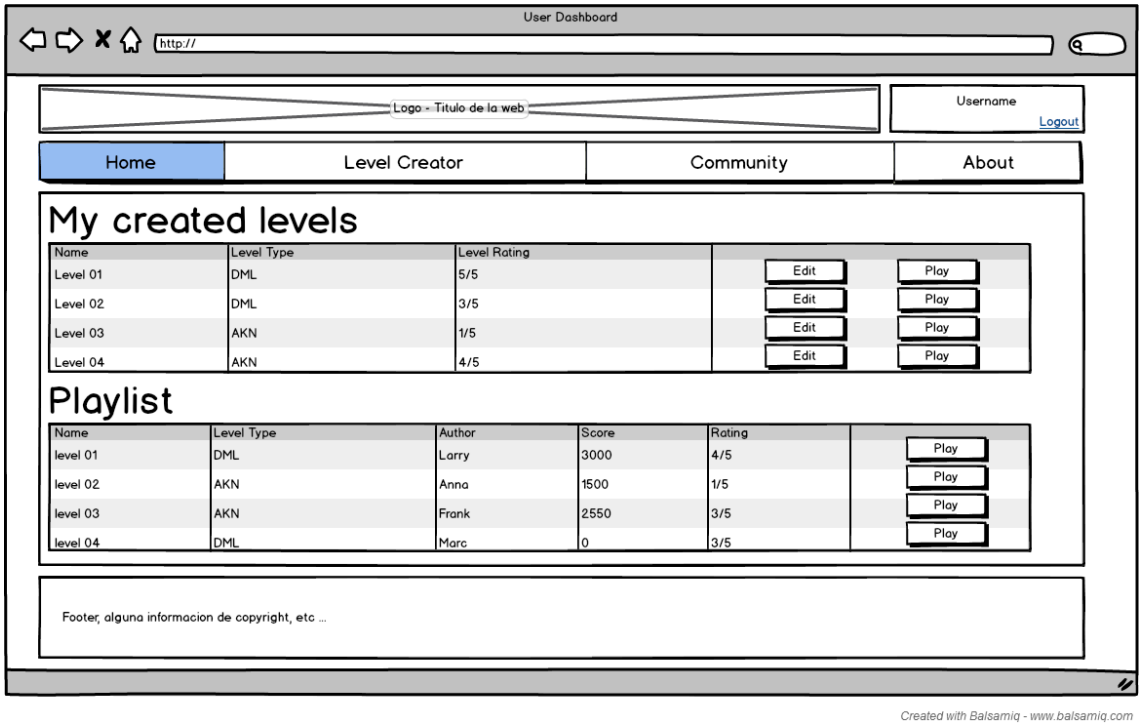


Figura 54 Boceto preliminar de la página "User Home"

Esta es la página personal del usuario, en esta página aparecen todos los contenidos que el usuario ha creado dentro de la página web, más los contenidos seleccionados para jugar que han creado otros jugadores.

Esta página está dividida en tres secciones: el menú de navegación, la lista de contenido creado por el usuario y la lista de niveles creados por otros jugadores y que el usuario ha seleccionado para jugar, también llamada playlist.

En el menú de navegación encontramos todas las secciones de las que se compone el sitio web y que se explicaran cada una de ellas en su sección correspondiente.

En la siguiente parte de la página se encuentra la lista de niveles creados por el usuario. Esta lista está dividida en varias columnas:

- **Nombre:** Nombre del nivel.
- **Tipo de nivel:** A que juego pertenece el nivel creado.
- **Valoración:** Valoración media que los usuarios de la página web le han otorgado al nivel.
- **Editar:** Botón con el que se accede a la herramienta de edición de niveles para modificar el nivel creado.
- **Jugar:** Botón con el que se puede jugar al nivel creado.

En la última parte de la página está situada la playlist del usuario, esta playlist es una lista similar a la anterior pero contiene los niveles creados por otros jugadores y que el usuario ha seleccionado para jugar. Esta lista también está dividida en varias columnas:

- **Nombre:** Nombre del nivel.
- **Tipo de nivel:** A que juego pertenece el nivel creado.
- **Autor:** Usuario creador del nivel.
- **Puntuación:** Puntuación obtenida por el jugador dentro del nivel.
- **Valoración:** Valoración media que los usuarios de la página web le han otorgado al nivel.
- **Jugar:** Botón con el que se puede jugar al nivel seleccionado.

5.6.1.4 Página con la pantalla de juego

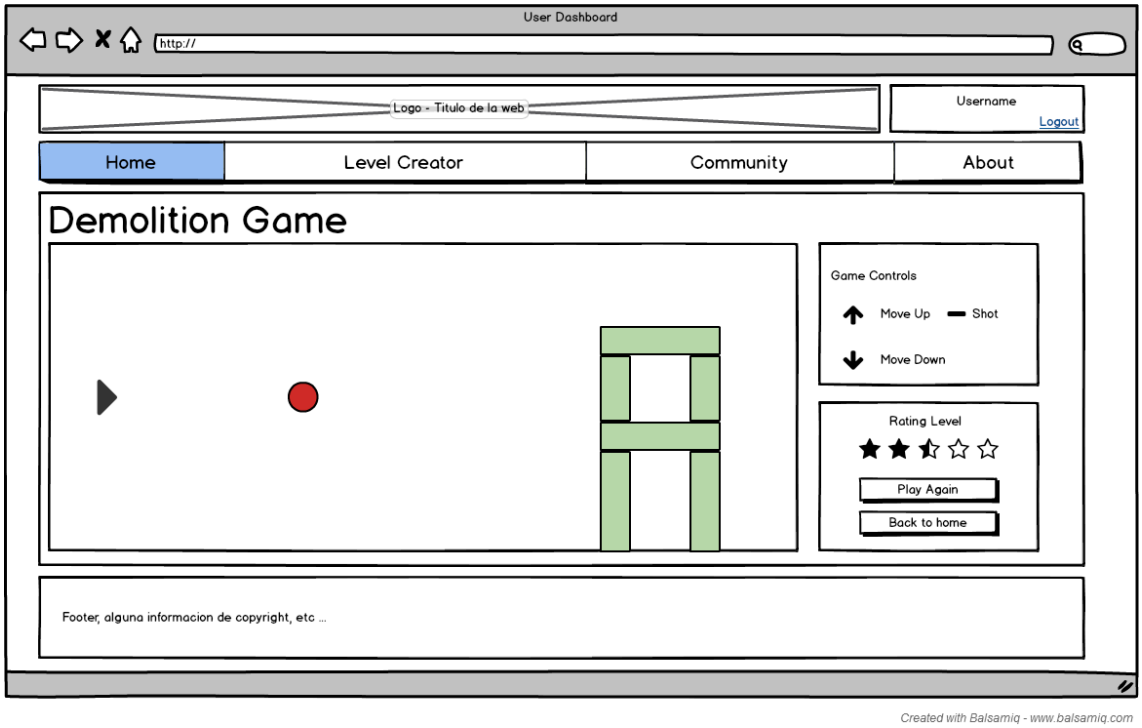


Figura 55 Boceto preliminar de la página "Game"

Esta es la pantalla de juego, los dos juegos implementados en este proyecto presentan una pantalla de juego similar.

Esta pantalla está dividida principalmente en dos zonas, la zona izquierda que es donde se encuentra la zona de juego y la zona derecha donde se ubican los controles del juego y una serie de opciones.

La zona de juego es un gran rectángulo donde se cargan todos los elementos que componen el nivel y es la zona en la que el usuario podrá jugar.

En la zona derecha de la pantalla se encuentran varios iconos que indican cuales son los controles del juego, también se presenta un medidor de estrellas con el que se puede valorar el nivel. Una vez finalizada la partida, en esta misma zona aparecen dos botes más.

- **Volver a jugar:** Botón que reinicia el nivel y permite volver a jugarlo.
- **Volver al home:** Con este botón se regresa a la página home del usuario.

5.6.1.5 Página "Level Editor"

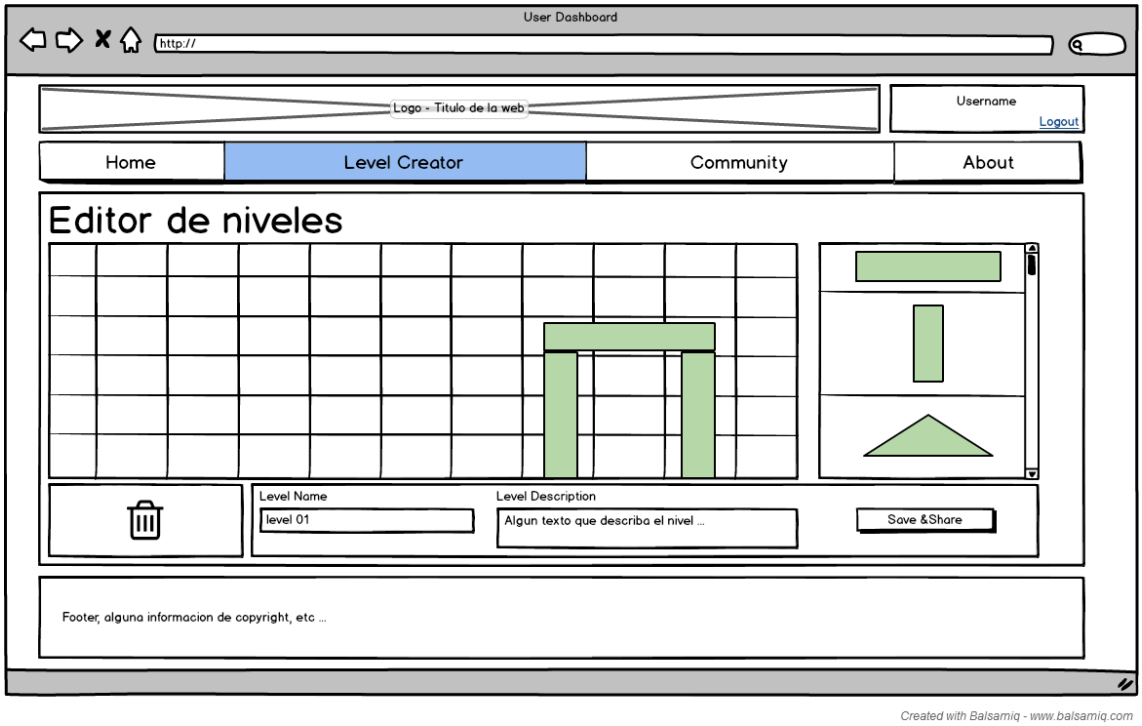


Figura 56 Boceto preliminar de la página "Level Editor"

Esta es la herramienta de edición, es la página donde el usuario puede crear el contenido para los distintos juegos que implementa este proyecto. Esta página está dividida fundamentalmente en dos secciones: la sección superior que es la que recoge todas las funciones de construcción del nivel y la sección inferior que es la encargada de los datos del nivel.

La sección superior a su vez está dividida en dos partes: la parte izquierda que contiene la cuadricula que simula la zona de juego y la parte derecha donde se encuentran los elementos con los que el usuario puede construir el nivel.

Empezando por la parte superior, la zona de la izquierda, como se explica en el párrafo anterior, contiene la cuadricula que simula la zona de juego. Esta zona es donde se deberán colocar las piezas para construir el nivel arrastrándolas desde la paleta.

La zona derecha de la parte superior presenta una lista de elementos con los que construir el nivel, también llamada paleta.

En la zona inferior izquierda de la pantalla encontramos una zona para eliminar las piezas con el icono de una papelera. En la parte derecha se encuentran una serie de campos para editar la información del nivel: Nombre, Descripción, valores de configuración de algunos elementos del juego. También hay un botón que sirve para guardar y publicar el nivel creado.

5.6.1.6 Página "Community"

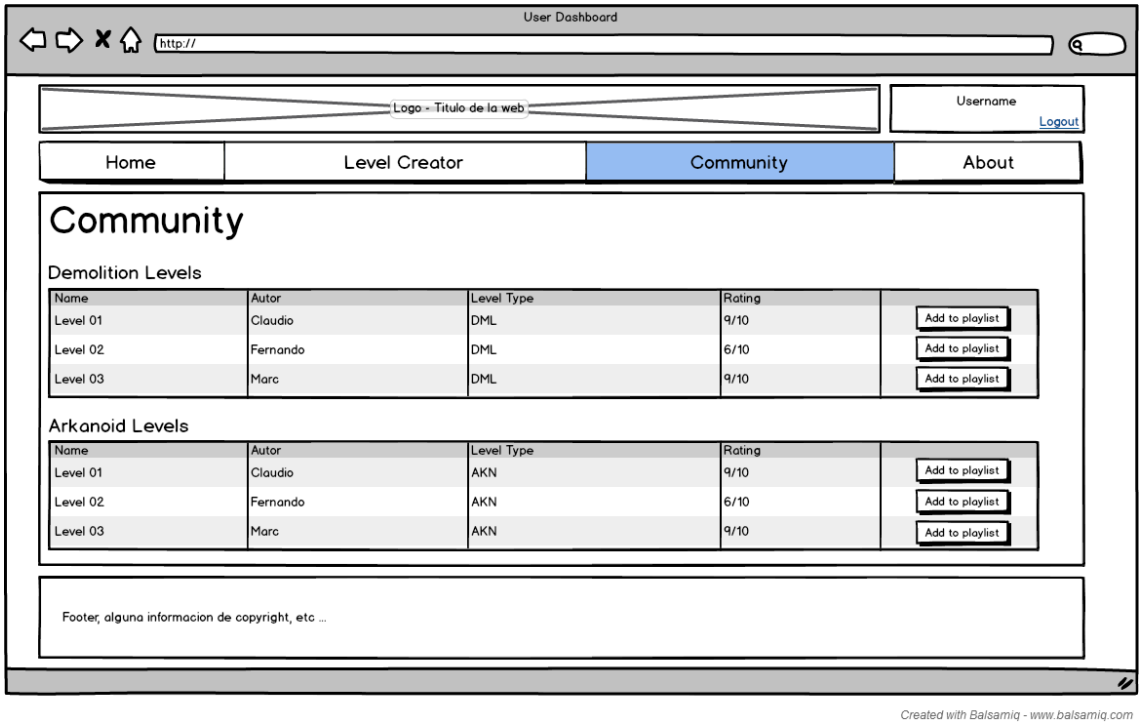


Figura 57 Boceto preliminar de la página "Community"

Esta es la página en la que se muestra el contenido creado por la comunidad de usuario de la página web.

Esta página está dividida en dos parte exactamente iguales, cada parte es una lista con todos los niveles creados por los demás usuarios pero agrupados según el tipo de juego para el que se creó dicho nivel.

Estas listas se dividen en varias columnas:

- **Nombre:** Nombre del nivel.
- **Autor:** Usuario creador del nivel.
- **Tipo de nivel:** A que juego pertenece el nivel creado.
- **Valoración:** Valoración media que los usuarios de la página web le han otorgado al nivel.
- **Añadir a la playlist:** Botón que sirve para añadir el nivel seleccionado a la playlist del jugador.

5.6.1.7 Página "About"

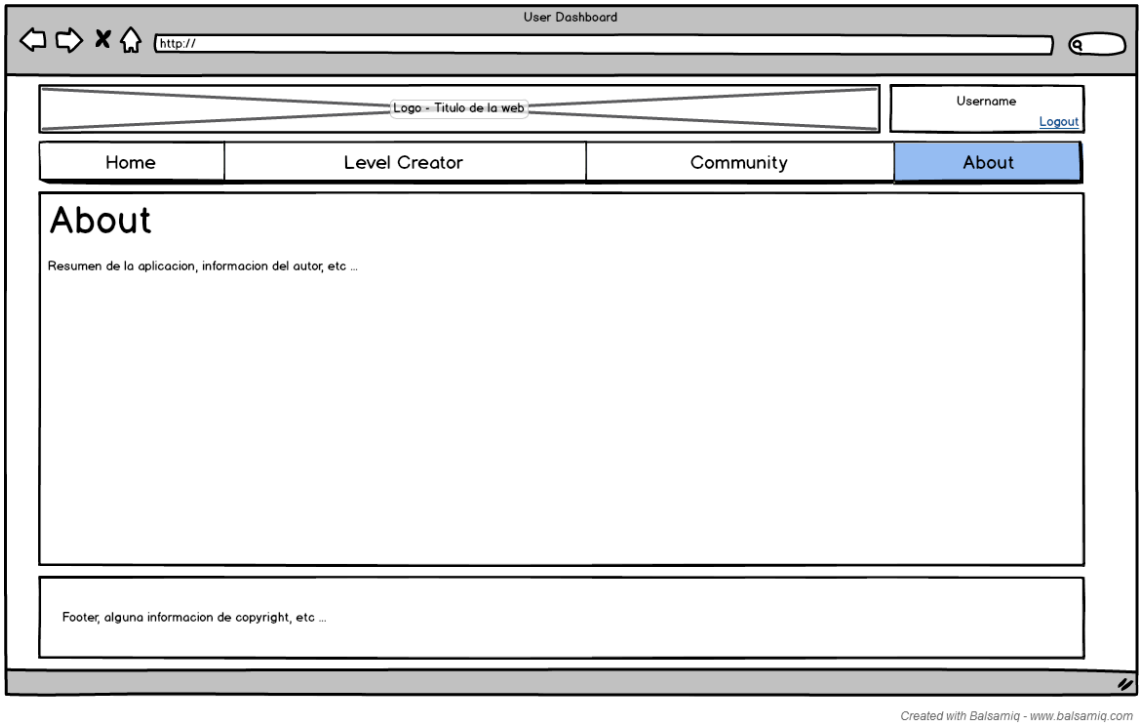


Figura 58 Boceto preliminar de la página "About"

Esta página esta desdichada a mostrar información sobre la aplicación y sobre el autor de la página web. Esta página es muy simple ya que solo contiene un gran bloque de texto informativo.

5.6.2 Diagrama de Navegabilidad

En esta sección se han dividido los diagramas en dos partes para que la legibilidad de los diagramas sea la mejor posible. Están separadas en el acceso a la página web y en la propia página web.

5.6.2.1 Navegabilidad en el acceso a la página web

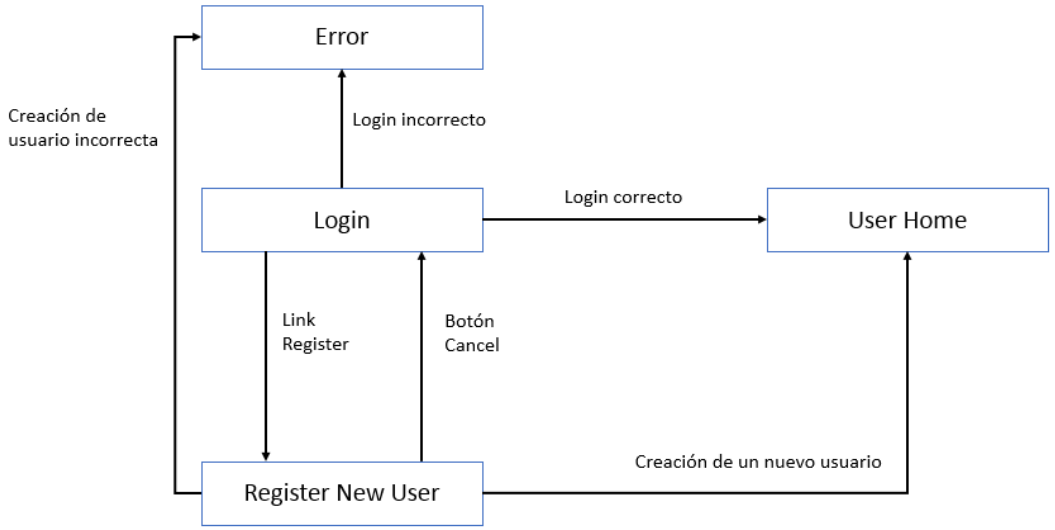


Figura 59 Diagrama de Navegabilidad, acceso a la página web

5.6.2.2 Navegabilidad dentro de la página web

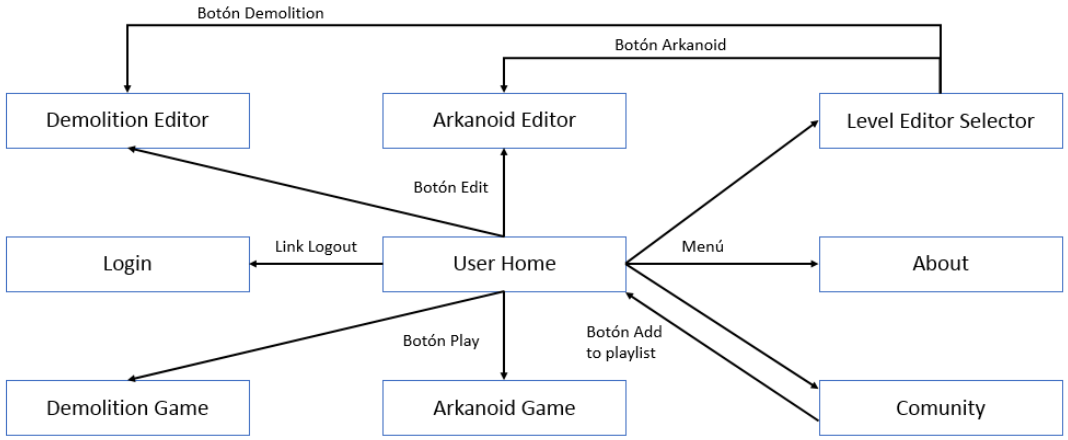


Figura 60 Diagrama de Navegabilidad, dentro de la página web

5.7 Especificación del Plan de Pruebas

A continuación se detallan algunas de las pruebas realizadas al sistema presentado:

5.7.1 Pruebas Unitarias

Prueba 1: Comprobar si existe un nivel determinado.
Entrada
String con el nombre del nivel.
Operación
Se comprueba si existe el nivel dando una respuesta verdadera.
Salida
Booleano con respuesta.

Prueba 2: Obtener la lista de todos los niveles del juego "Arkanoid".
Entrada
No se requiere entrada.
Operación
Se rellena una lista con los niveles encontrados en la base de datos.
Salida
Lista con los niveles del juego "arkanoid" encontrados en la base de datos.

Prueba 3: Leer el fichero JSON del nivel "Arkanoid".
Entrada
String con la ruta del fichero a leer.
Operación
Leer el fichero y decodificar su contenido.
Salida
Objeto ArkanoidLevel, con los datos del nivel: nombre, descripción, piezas colocadas dentro del nivel.

Prueba 4: Obtener la lista de todos los niveles del juego "Demolition".
Entrada
No se requiere entrada.
Operación
Se rellena una lista con los niveles encontrados en la base de datos.
Salida
Lista con los niveles del juego "demolición" encontrados en la base de datos.

Prueba 5: Leer el fichero JSON del nivel "Demolition".
Entrada
String con la ruta del fichero a leer.
Operación
Leer el fichero y decodificar su contenido.
Salida
Objeto DemolitionLevel, con los datos del nivel: nombre, descripción, posición del cañón, numero de proyectiles, posición de la línea límite de altura, piezas colocadas dentro del nivel.

Prueba 6: Obtener la lista de todos los niveles creados por un usuario.
Entrada
String con el nombre del usuario.
Operación
Se rellena una lista con los niveles encontrados en la base de datos.
Salida
Lista con los niveles encontrados en la base de datos.

Prueba 7: Obtener la puntuación de un nivel concreto.
Entrada
String con el nombre del nivel.
Operación
Se busca en la base de datos el rating del nivel.
Salida
Entero con la puntuación media del nivel.

Prueba 8: Obtener un usuario.
Entrada
String con el nombre del usuario.
Operación
Se busca el usuario dentro de la base de datos.
Salida
Objeto User, con los datos del usuario: nombre y password.

Prueba 9: Comprobar si existe un usuario.
Entrada
String con el nombre del usuario.
Operación
Se comprueba si existe el usuario dando una respuesta verdadera.
Salida
Booleano con respuesta.

5.7.2 Pruebas de integración y sistema

Caso de uso: Crear Usuario	
Prueba	**Resultado Esperado**
Crear un usuario que no existe	El sistema posee un usuario más.
Prueba	**Resultado Esperado**
Crear un usuario que ya existe	El sistema no posee un usuario más y muestra una página de error.
Prueba	**Resultado Esperado**
Cancelar la operación	El sistema permanece sin cambios.

Caso de uso: Añadir nivel a la playlist del usuario	
Prueba	**Resultado Esperado**
Añadir un nivel que el usuario no tiene en su playlist	El sistema posee un nuevo nivel en la playlist del usuario.
Prueba	**Resultado Esperado**
Añadir un nivel que el usuario ya tiene en su playlist	El sistema no posee un nuevo nivel en la playlist del usuario.

Caso de uso: Arrastrar pieza de la paleta a la cuadricula	
Prueba	**Resultado Esperado**
Arrastrar pieza de la paleta a la cuadricula	El nivel actual posee una pieza más.

Caso de uso: Eliminar pieza	
Prueba	**Resultado Esperado**
Arrastrar la pieza a la papelera	El nivel actual ya no posee una pieza más.

Caso de uso: Guardar nivel	
Prueba	**Resultado Esperado**
Guardar nivel que no exista	El sistema actual posee un nuevo nivel más.
Prueba	**Resultado Esperado**
Guardar nivel que ya exista	El sistema actual no posee un nuevo nivel más.

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

Este proyecto se ha implementado bajo la arquitectura de un patrón MVC clásico, con varias capas diferenciando cada uno de los componentes del patrón: Vistas, Modelo, Lógica de Negocio y acceso a datos, más una base de datos externa que se integrara con la capa de acceso a datos.

6.1.1 Diagramas de Paquetes

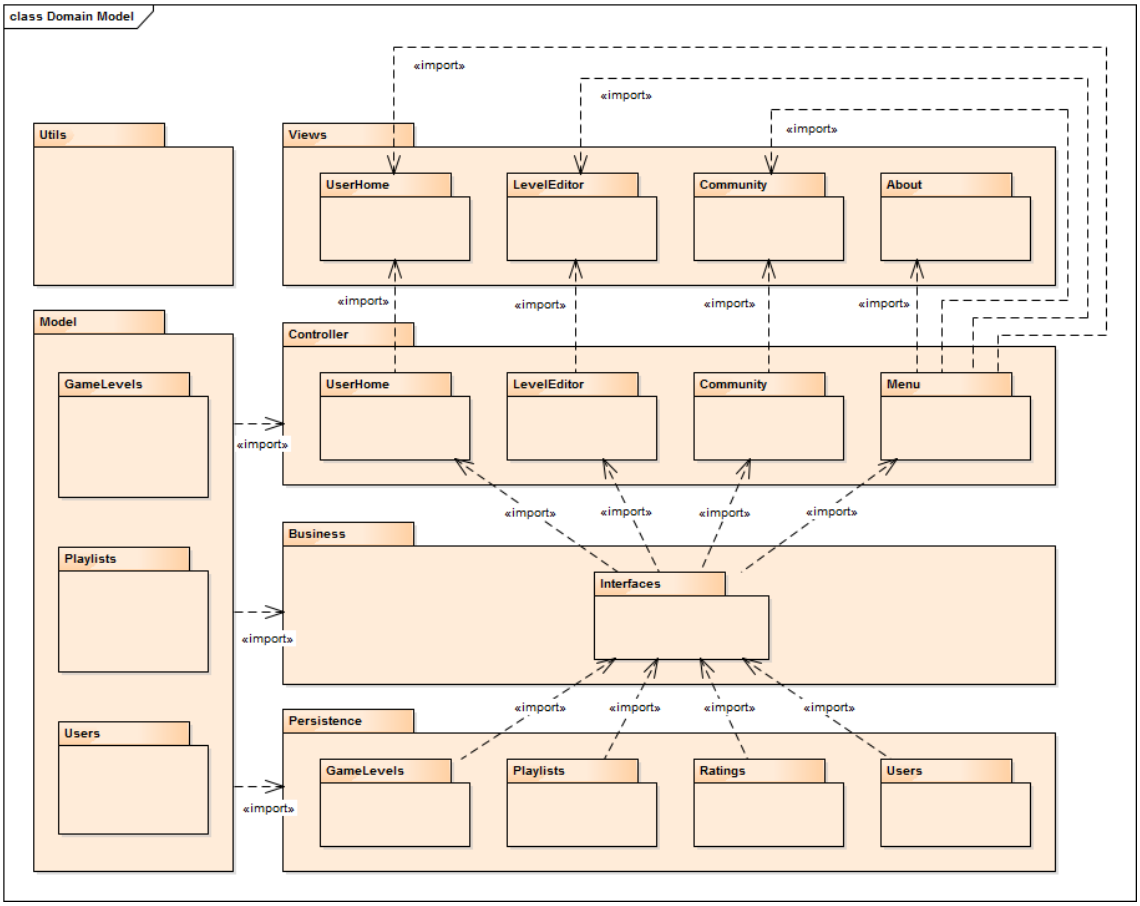


Figura 61 Diagrama de paquetes de la aplicación

A continuación se explican los paquetes que componen la aplicación.

6.1.1.1 Paquete Views

Este paquete contiene todas las vistas de la aplicación, este gran paquete a su vez contiene subpaquetes con cada grupo de vistas:

- **UserHome:** Este paquete contiene la vista de la zona personal del usuario, donde se muestran todos los contenidos de este.
- **LevelEditor:** Este paquete contiene las vistas de la herramienta de edición de niveles.
- **Community:** Este paquete contiene la vista donde se muestran los contenidos creados por todos los usuarios de la página web.
- **About:** Este paquete contiene la vista donde se muestra la información de la página web y del autor de la misma.

6.1.1.2 Paquete Controller

Este paquete contiene todos los controladores, que son clases que conectan la capa de negocio con las vistas. Este paquete contiene subpaquetes con cada grupo de controladores:

- **UserHome:** Este paquete contiene las clases que conectan las vistas del paquete UserHome con la lógica de negocio.
- **LevelEditor:** Este paquete contiene las clases que conectan las vistas del paquete LevelEditor con la lógica de negocio.
- **Community:** Este paquete contiene las clases que conectan las vistas del paquete Community con la lógica de negocio.
- **Menu:** Este paquete contiene todas las clases que conectan los distintos paquetes de vistas con la lógica de negocio para temas de navegabilidad.

6.1.1.3 Paquete Business

Este paquete contiene toda la lógica de negocio de la aplicación, también contiene un paquete con una serie de interfaces para comiscarse con la capa de acceso a datos.

6.1.1.4 Paquete Persistence

Paquete que contiene las clases que se comunican con la base de datos, está dividido en subpaquetes, uno para la comunicación con cada tabla de la base de datos.

- **GameLevels:** Paquete que contiene las clases que se comunican con la base de datos para manejar los datos de los niveles de juego.
- **Playlists:** Paquete que contiene las clases que se comunican con la base de datos para manejar los datos de las playlists.
- **Ratings:** Paquete que contiene las clases que se comunican con la base de datos para manejar los datos de las puntuaciones de los niveles.
- **Users:** Paquete que contiene las clases que se comunican con la base de datos para manejar los datos de los usuario.

6.1.1.5 Paquete Model

Paquete que contiene las clases que modelan los objetos del sistema divididas en subpaquetes.

- **GameLevels:** Paquete que contiene las clases que modelan los niveles del juego.
- **Playlists:** Paquete que contiene las clases que modelan las playlists.
- **Users:** Paquete que contiene las clases que modelan los usuarios de la página.

6.1.1.6 Paquete Utils

Paquete que contiene clases de utilidad como el driver usado para la base de datos o clases para el manejo de ficheros JSON.

6.1.2 Diagramas de Componentes

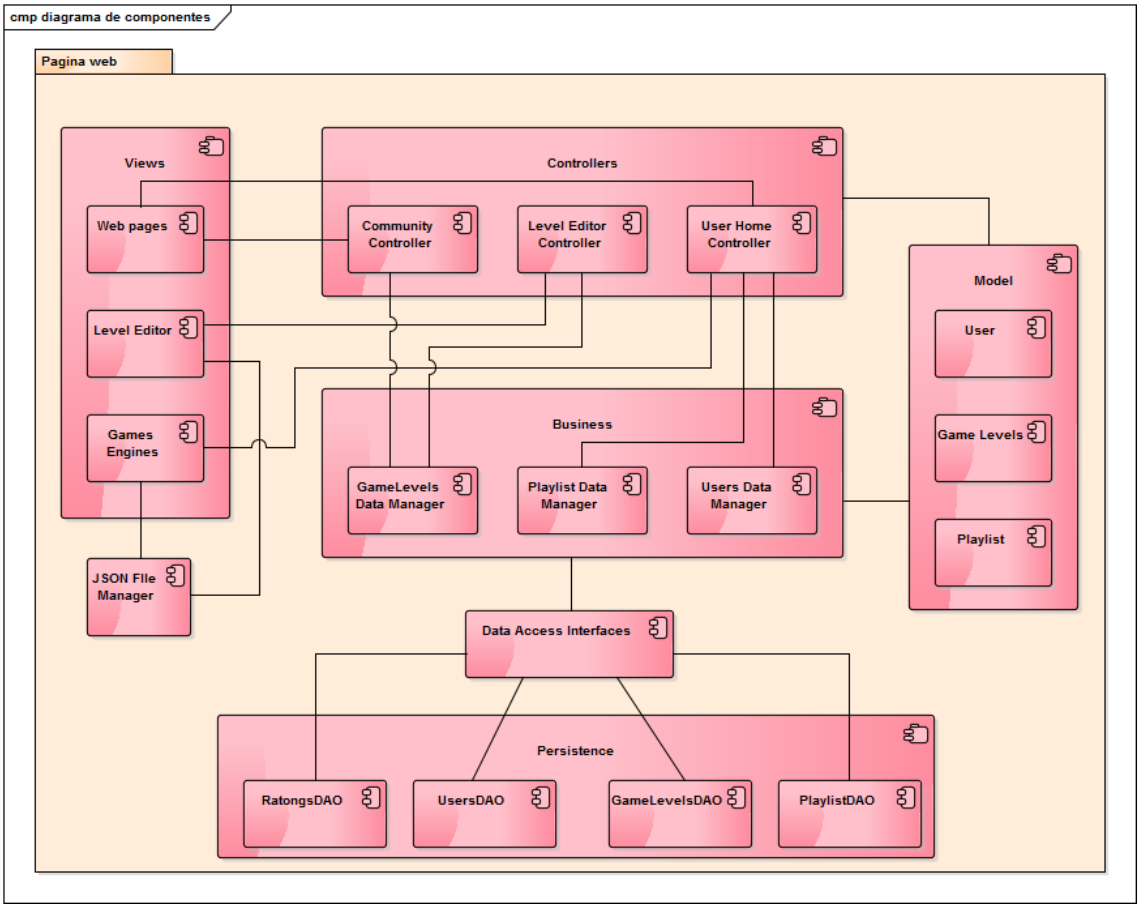


Figura 62 Diagrama de componentes

6.1.2.1 Grupo de componentes: Views

- **Web Pages**

Este componente agrupa todos los ficheros JSP que componen la vista final que verá el usuario. Este componente agrupa las JSP, CSS y JavaScript.

- **Level Editor**

Este componente agrupa todos los ficheros que componen la herramienta de edición de niveles. Incluye ficheros JSP, CSS, JavaScript.

- **Game Engines**

Este componente agrupa todos los ficheros que componen los motores de los juegos. Agrupa ficheros JSP, CSS y JavaScript que mueven los juegos.

6.1.2.2 Grupo de componentes: Controllers

- **User Home Controller**

Componente que contiene los ficheros fuente que manejan el controlador del home del usuario.

- **Level Editor Controller**

Componente que contiene los ficheros fuente que manejan el controlador de la herramienta de edición de niveles.

- **Community Controller**

Componente que contiene los ficheros fuente que manejan el controlador de la sección Community de la página web.

6.1.2.3 Grupo de componentes: Business

- **User Data Manager**

Componente que agrupa los ficheros fuentes que manejan todas las operaciones con los usuarios: Búsquedas, inserciones y comprobaciones.

- **Playlist Data Manager**

Componente que agrupa los ficheros fuentes que manejan todas las operaciones con las playlists: inserción de niveles en la playlist, cargar los niveles de las playlist, etc.

- **Game Levels Data Manager**

Componente que agrupa los ficheros fuentes que manejan todas las operaciones con los niveles de los juegos: creación y carga de niveles, comprobación de la disponibilidad de los niveles, etc.

6.1.2.4 Grupo de componentes: Persistence

- **Rating DAO**

Componente que se encarga de comunicarse con la base de datos, concretamente con la tabla que maneja las puntuaciones de los niveles.

- **User DAO**

Componente que se encarga de comunicarse con la base de datos, concretamente con la tabla que maneja los datos de los usuarios.

- **Game Levels DAO**

Componente que se encarga de comunicarse con la base de datos, concretamente con la tabla que maneja los datos de los niveles creados.

- **Playlist DAO**

Componente que se encarga de comunicarse con la base de datos, concretamente con la tabla que maneja las playlists.

6.1.2.5 Grupo de componentes: Model

- **User**

Componente que contiene los ficheros fuente que modelan los usuarios de la página web.

- **Game Levels**

Componente que contiene los ficheros fuente que modelan los niveles de los juegos de la aplicación.

- **Playlist**

Componente que contiene los ficheros fuente que modelan las playlists.

6.1.2.6 Otros componentes

- **Data Access Interfaces**

Este componente aglutina todas las interfaces que hay entre la capa de Lógica de negocio con la capa de Persistencia. Estas interfaces hacen de separación entre capa.

- **JSON File Manager**

Este componente agrupa los ficheros necesarios para el manejo de los ficheros JSON.

6.1.3 Diagramas de Despliegue

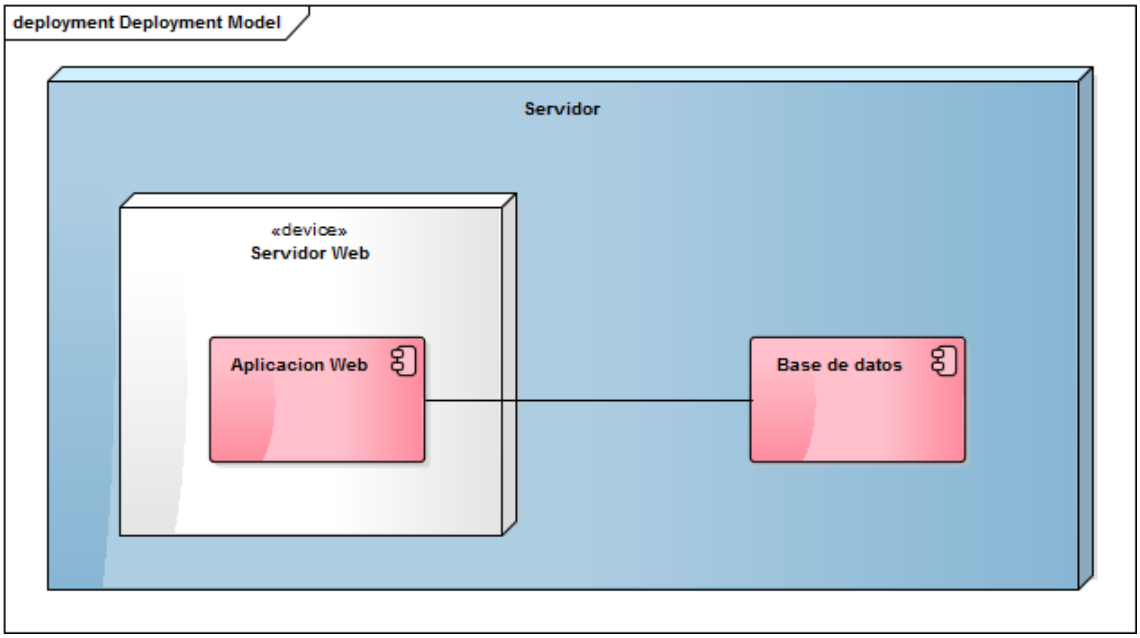


Figura 63 Diagrama de despliegue

El despliegue de esta aplicación requiere de una maquina física o servidor, donde instalar un servidor web de aplicaciones como Tomcat o Apache y una base de datos MYSQL.

Dentro de este servidor web es donde se desplegará la aplicación web que se comunicara directamente con la base de datos.

El sistema operativo del servidor o maquina física puede ser tanto Windows como Linux, ya que el servidor web de aplicaciones es compatible con ambos sistemas.

6.2 Diseño de Clases

6.2.1 Diagrama de Clases

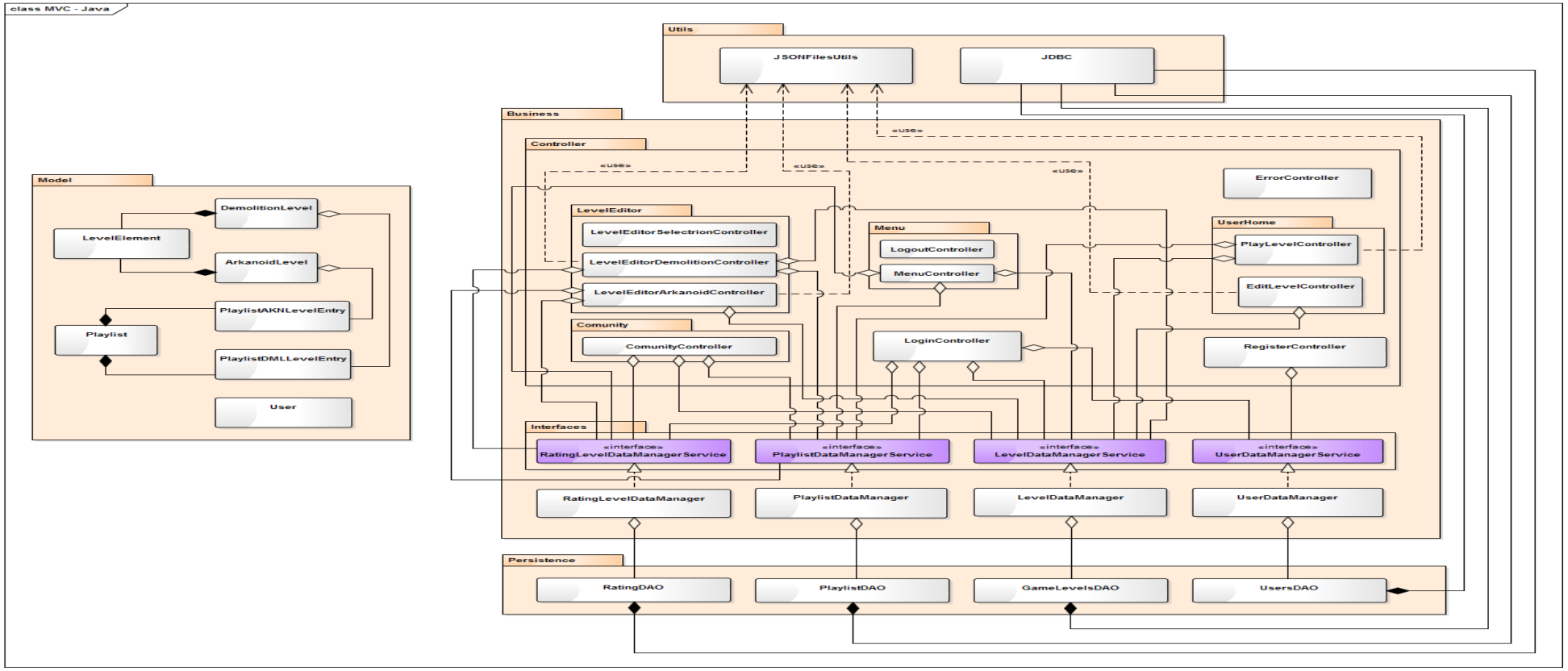


Figura 64 Diagrama de clases, diseño

En la imagen del diagrama se puede apreciar que el paquete "Model" no tiene ninguna conexión con el resto de las clases. Esto se ha querido reflejar así por dos razones. La primera razón es por la legibilidad del diagrama, se ha decidió no incluir las conexiones porque haría aún más ilegible el diagrama y la segunda razón es porque el modelo es una parte transversal que abarca todas las capas del proyecto, por lo que se utiliza en todas las capas del proyecto.

Dado que el diagrama es demasiado grande para que se pueda ver con claridad, se expondrán las clases separadas por paquetes:

6.2.1.1 Paquete Utils:

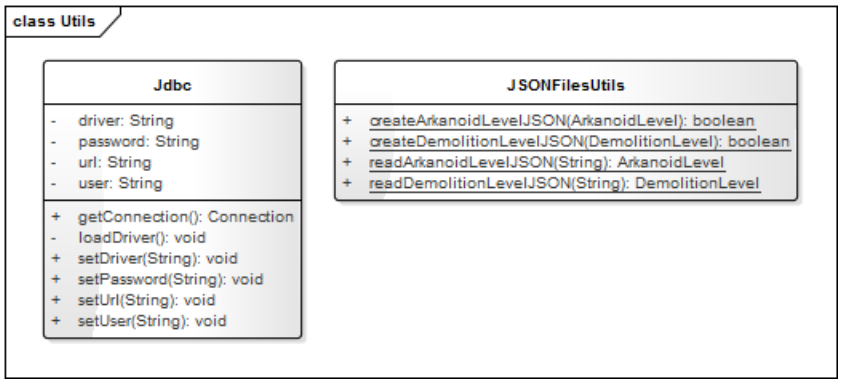


Figura 65 Paquete Utils

6.2.1.2 Paquete Persistence

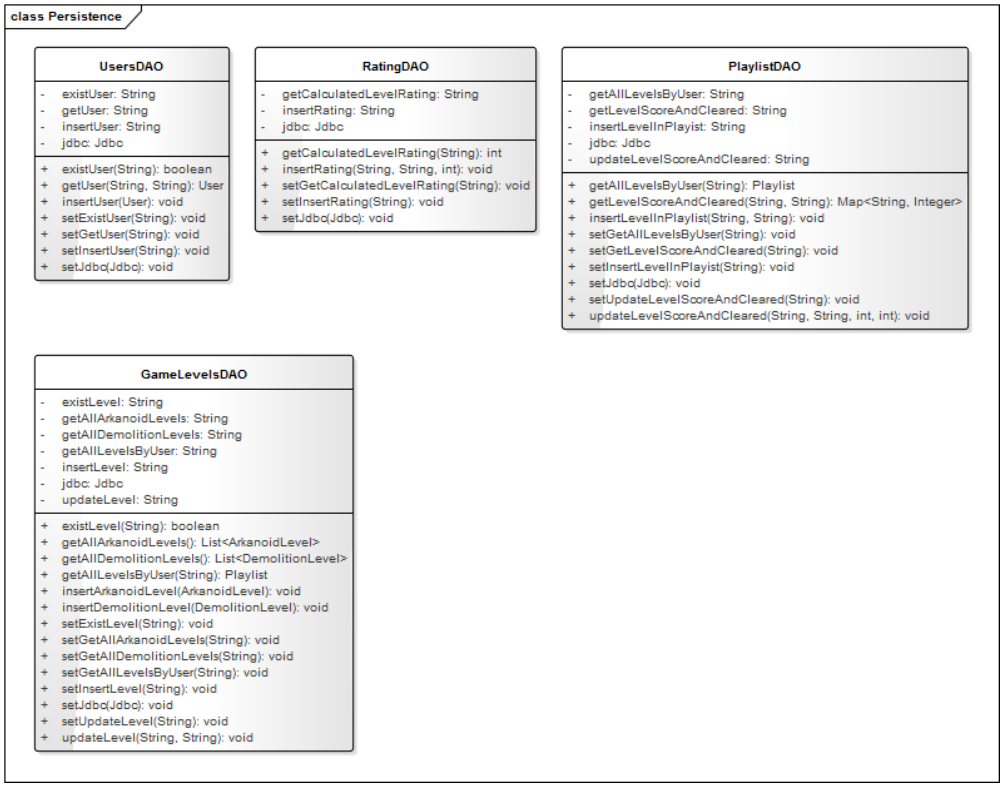


Figura 66 Paquete Persistence

6.2.1.3 Paquete Business

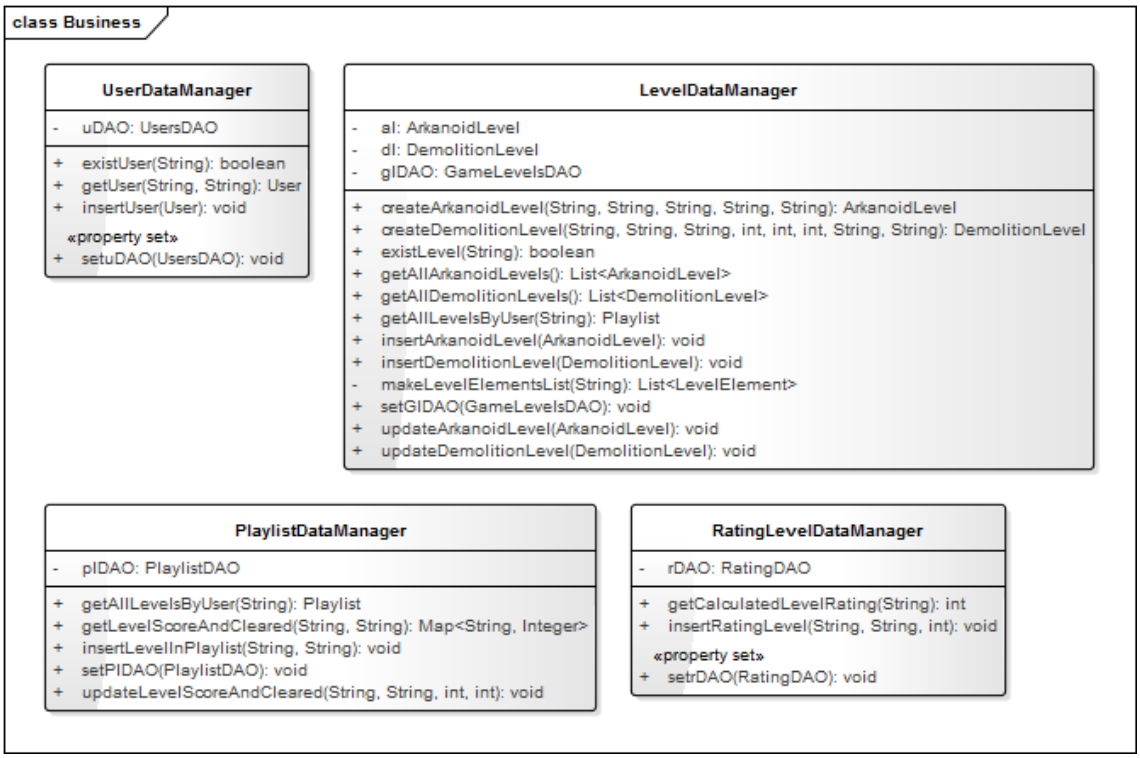


Figura 67 Paquete Business

6.2.1.4 Paquete Controller

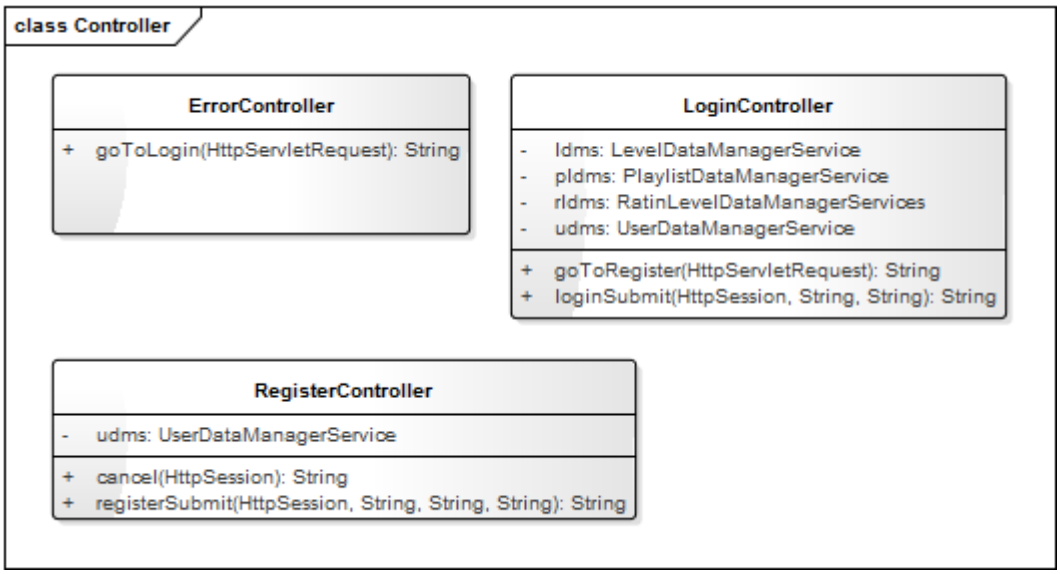


Figura 68 Paquete Controller

6.2.1.5 Paquete Level Editor

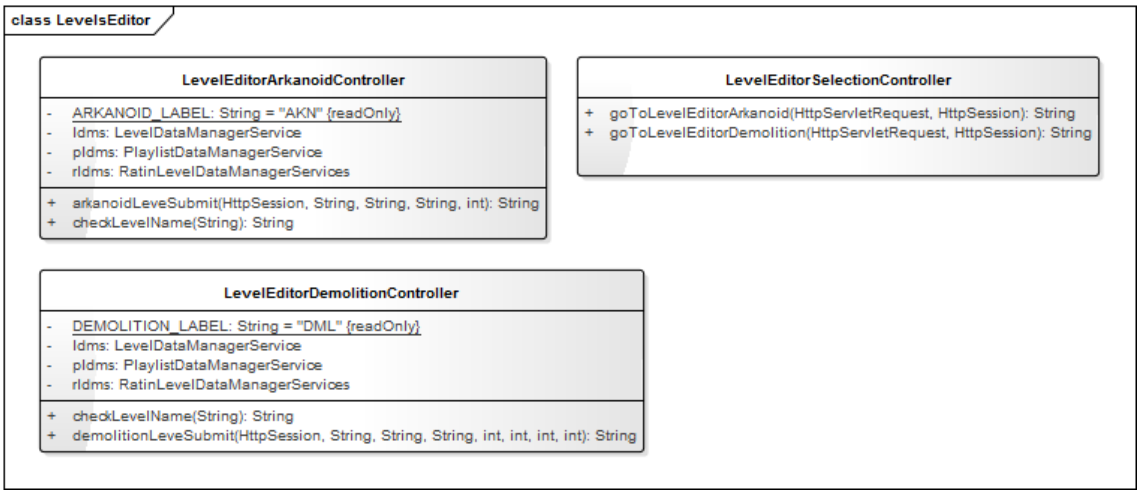


Figura 69 Paquete Level Editor

6.2.1.6 Paquete Community

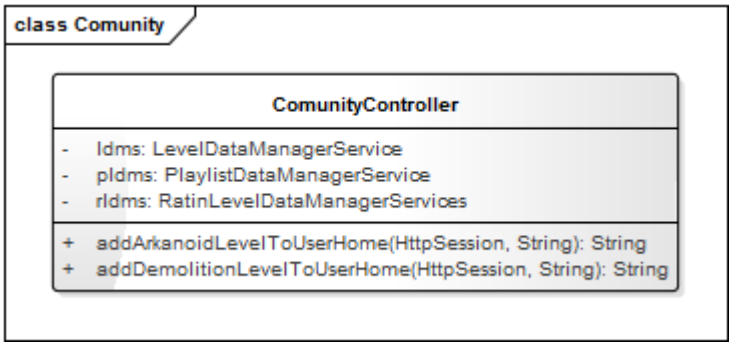


Figura 70 Paquete Community

6.2.1.7 Paquete Menu

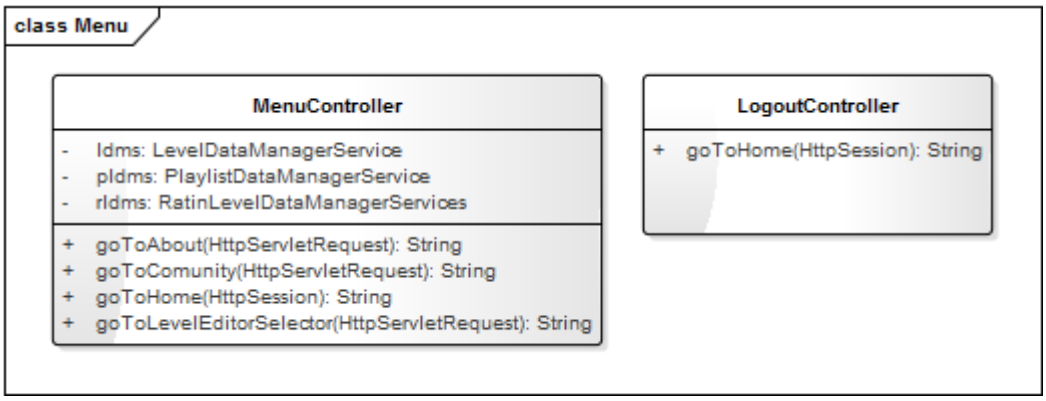


Figura 71 Paquete Menu

6.2.1.8 Paquete User Home

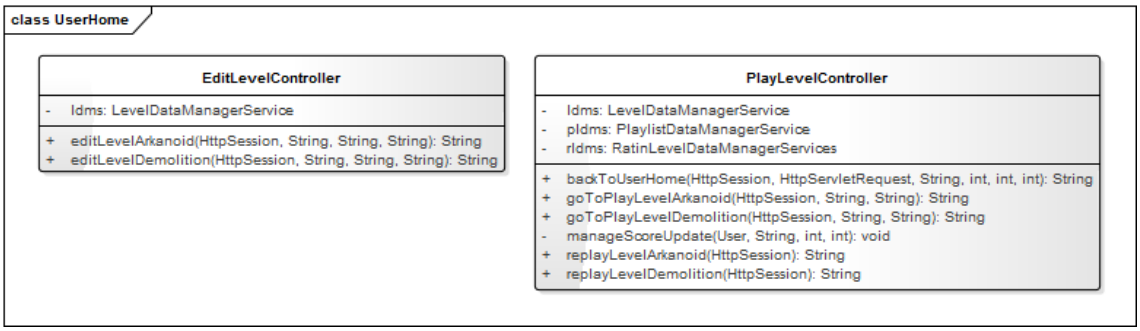


Figura 72 Paquete User Home

6.2.1.9 Paquete Interfaces

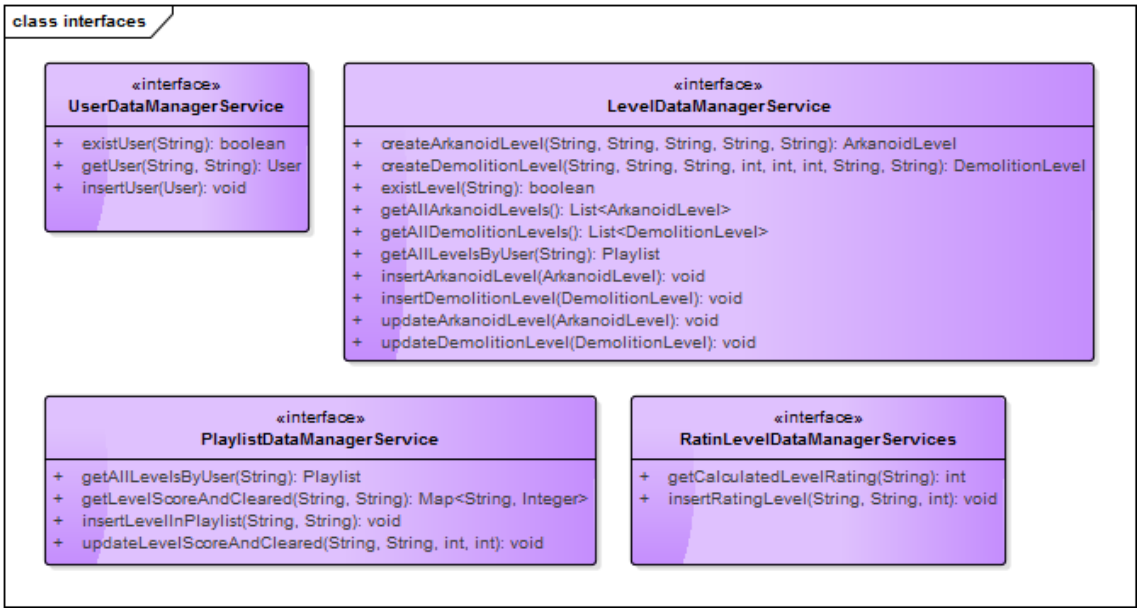


Figura 73 Paquete Interfaces

6.2.1.10 Paquete Model

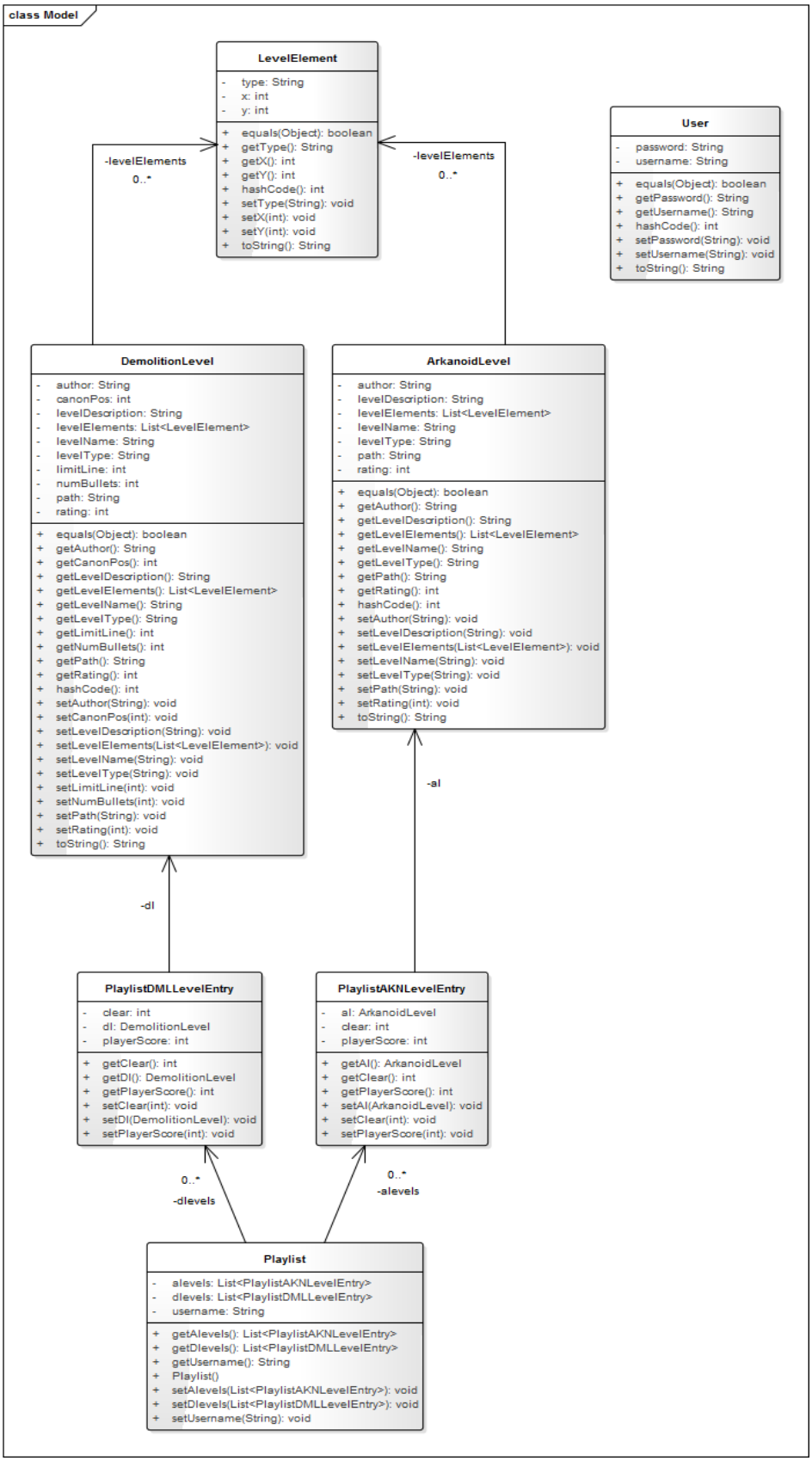


Figura 74 Paquete Model

6.3 Diagramas de Interacción y Estados

Estos son los diagramas de interacción y estados de los casos más importantes que se pueden encontrar en este proyecto, debido a su extensión y complejidad.

6.3.1 Caso de Uso 1 y 4: Login e Ir al home de usuario

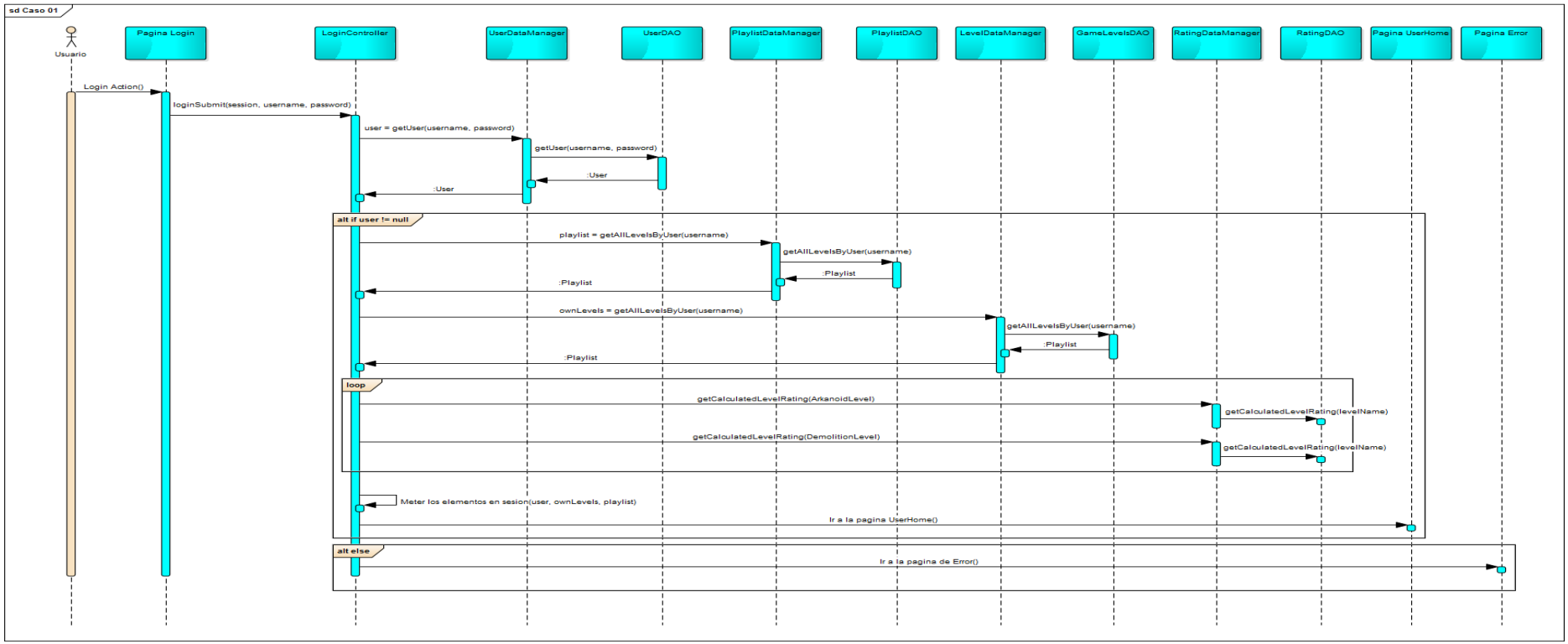


Figura 75 Diagrama de secuencia 1: Casos de uso 1 y 4

Este diagrama agrupa estos dos casos de uso ya que se comportan exactamente igual.

6.3.2 Caso de Uso 2: Logout

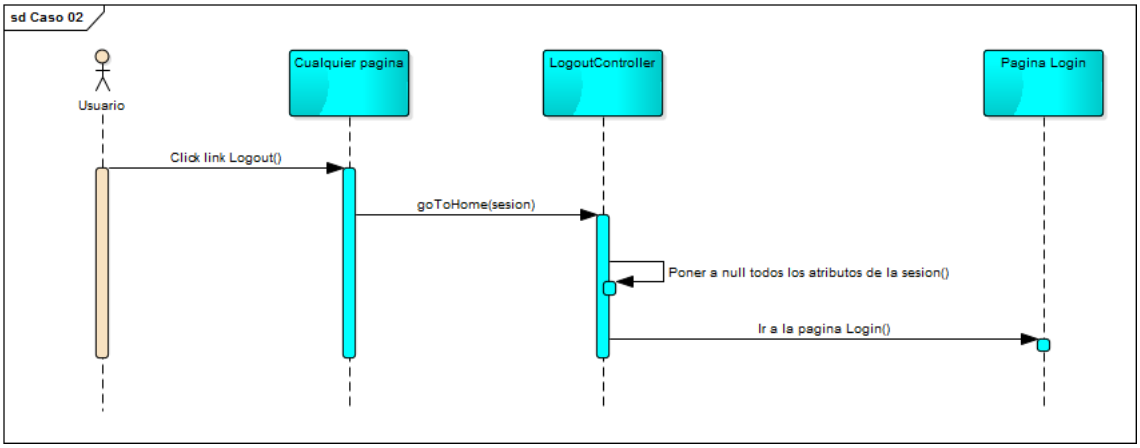


Figura 76 Diagrama de secuencia 2: Casos de uso 2

6.3.3 Caso de Uso 3: Crear usuario

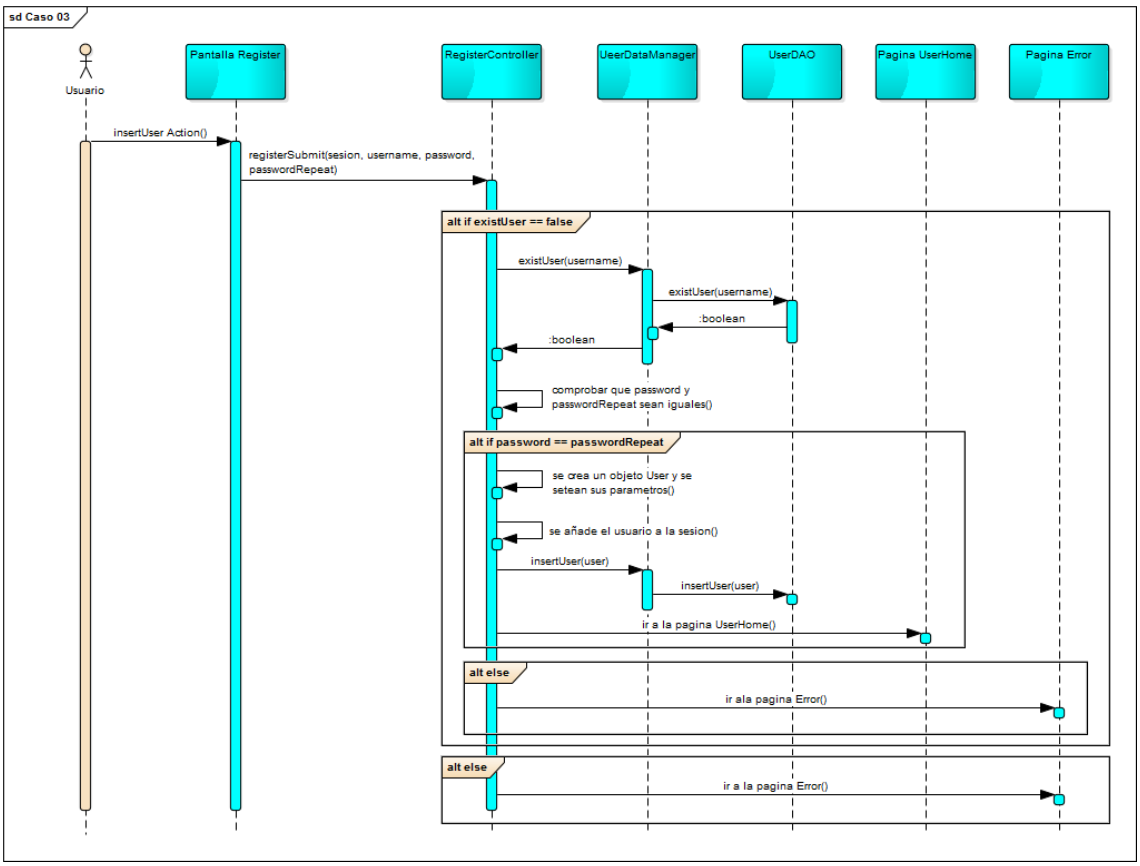


Figura 77 Diagrama de secuencia 3: Casos de uso 3

6.3.4 Caso de uso 5 y 6: Ir al editor de niveles: "Demolition", "Arkanoid"

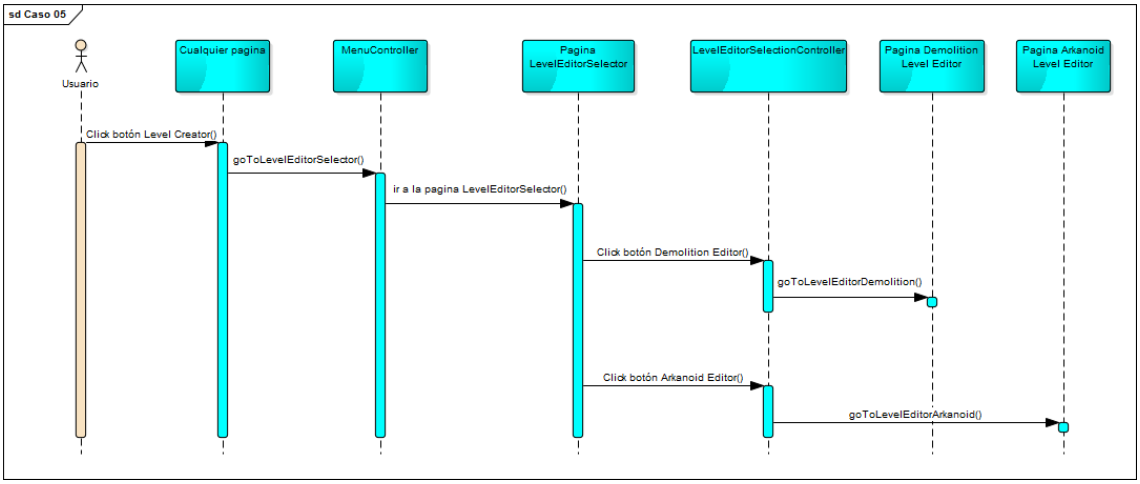


Figura 78 Diagrama de secuencia 4: Casos de uso 5 y 6

6.3.5 Caso de uso 7: Ir a la página "Community"

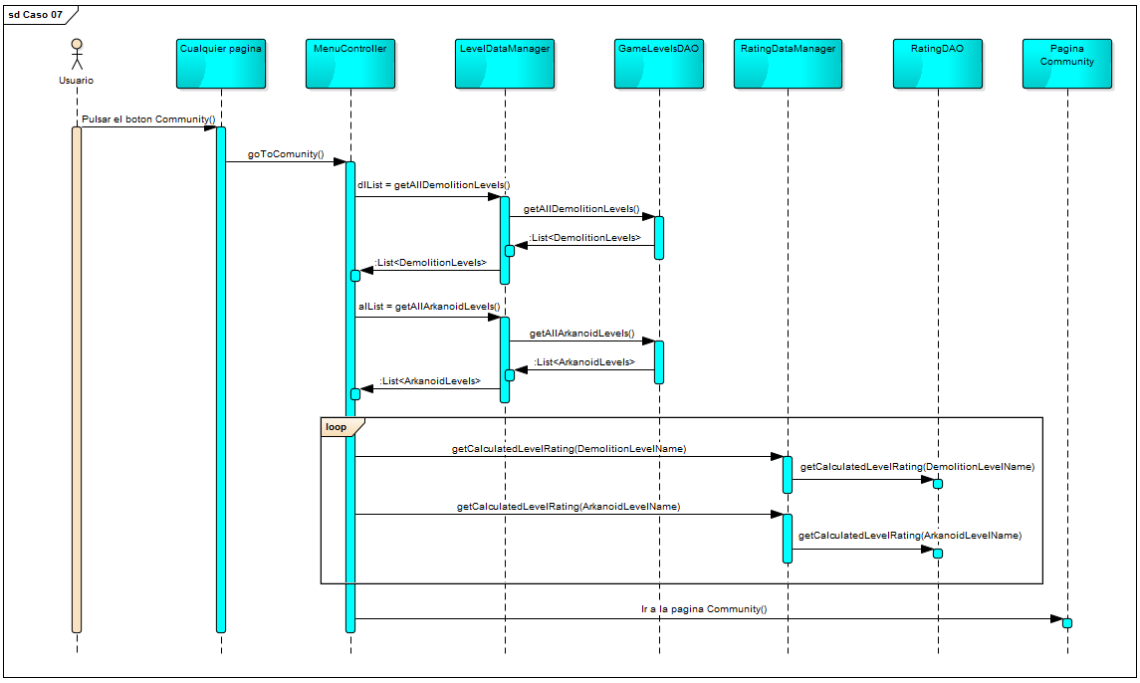


Figura 79 Diagrama de secuencia 5: Casos de uso 7

6.3.6 Caso de Uso 8: Ir a la página "About"

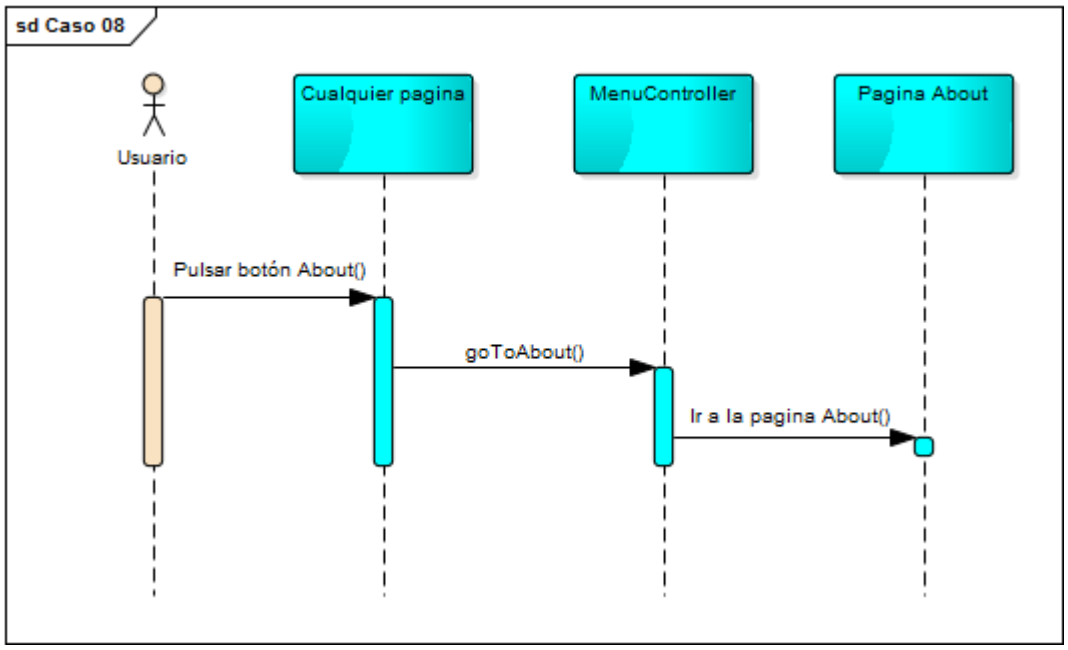


Figura 80 Diagrama de secuencia 6: Casos de uso 8

6.3.7 Caso de Uso 9: Jugar nivel

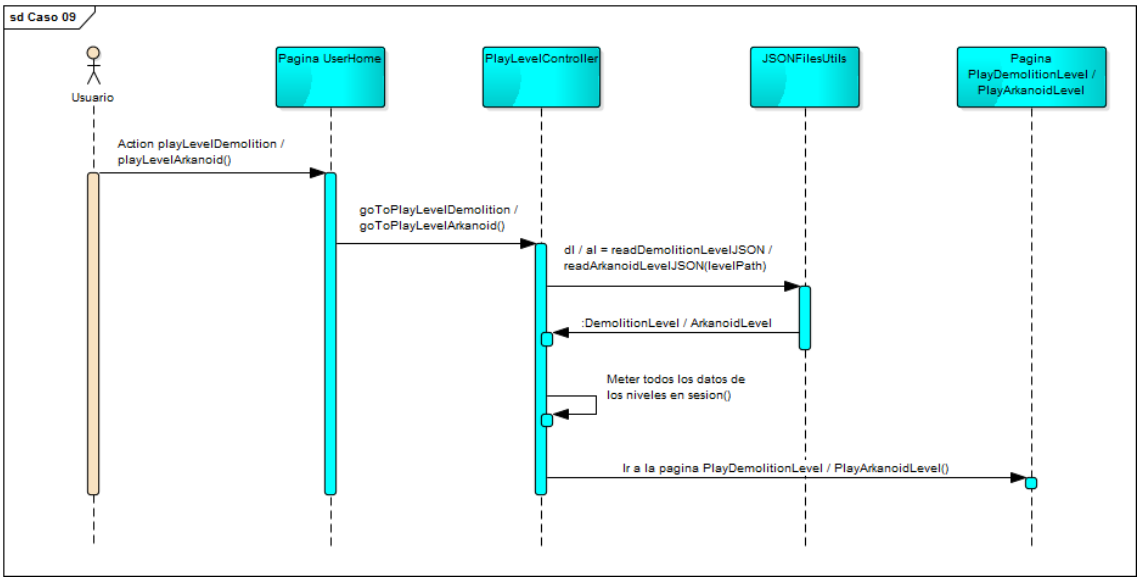


Figura 81 Diagrama de secuencia 7: Casos de uso 9

6.3.8 Caso de Uso 10: Editar nivel

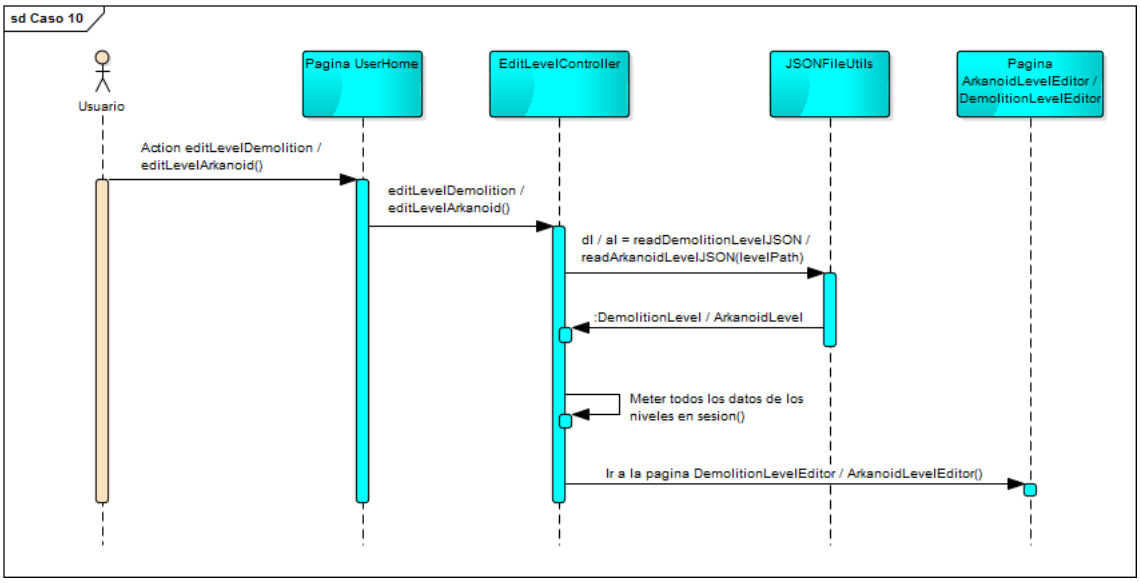


Figura 82 Diagrama de secuencia 8: Casos de uso 10

6.3.9 Caso de Uso 13: Volver a jugar nivel

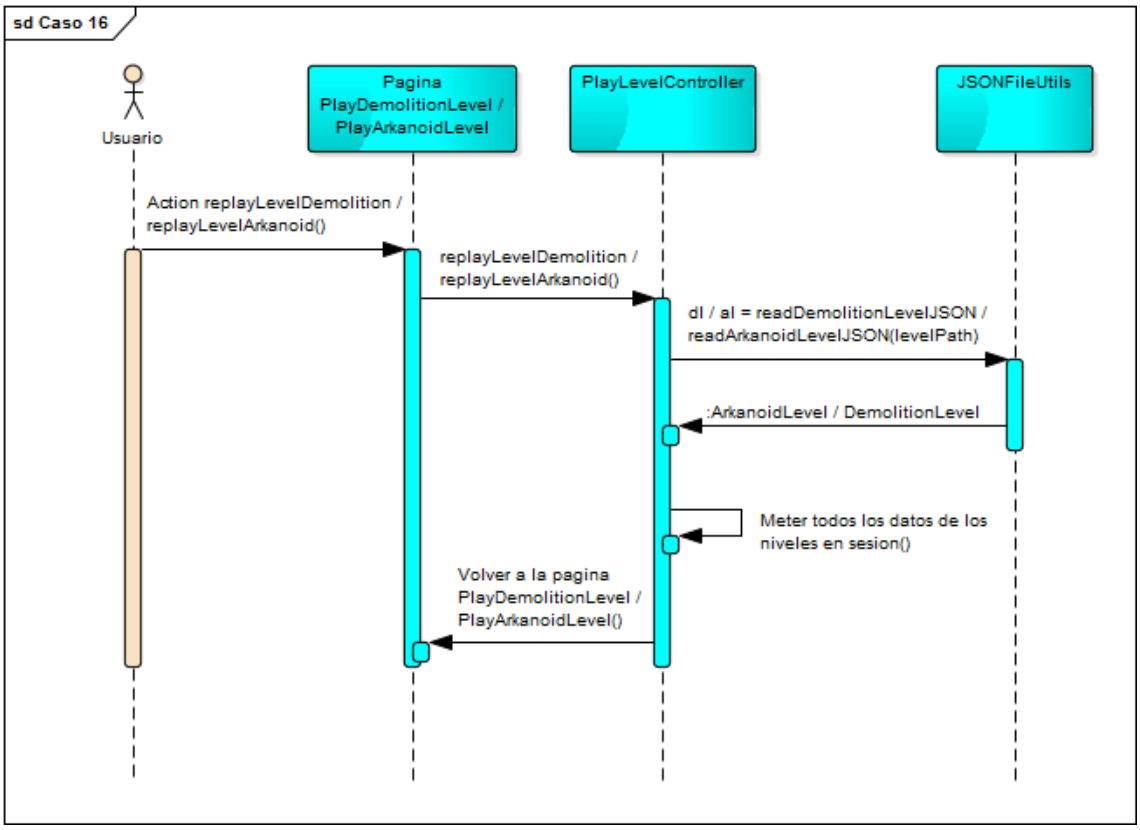


Figura 83 Diagrama de secuencia 9: Casos de uso 13

6.3.10 Caso de Uso 11: Añadir nivel a la playlist del usuario

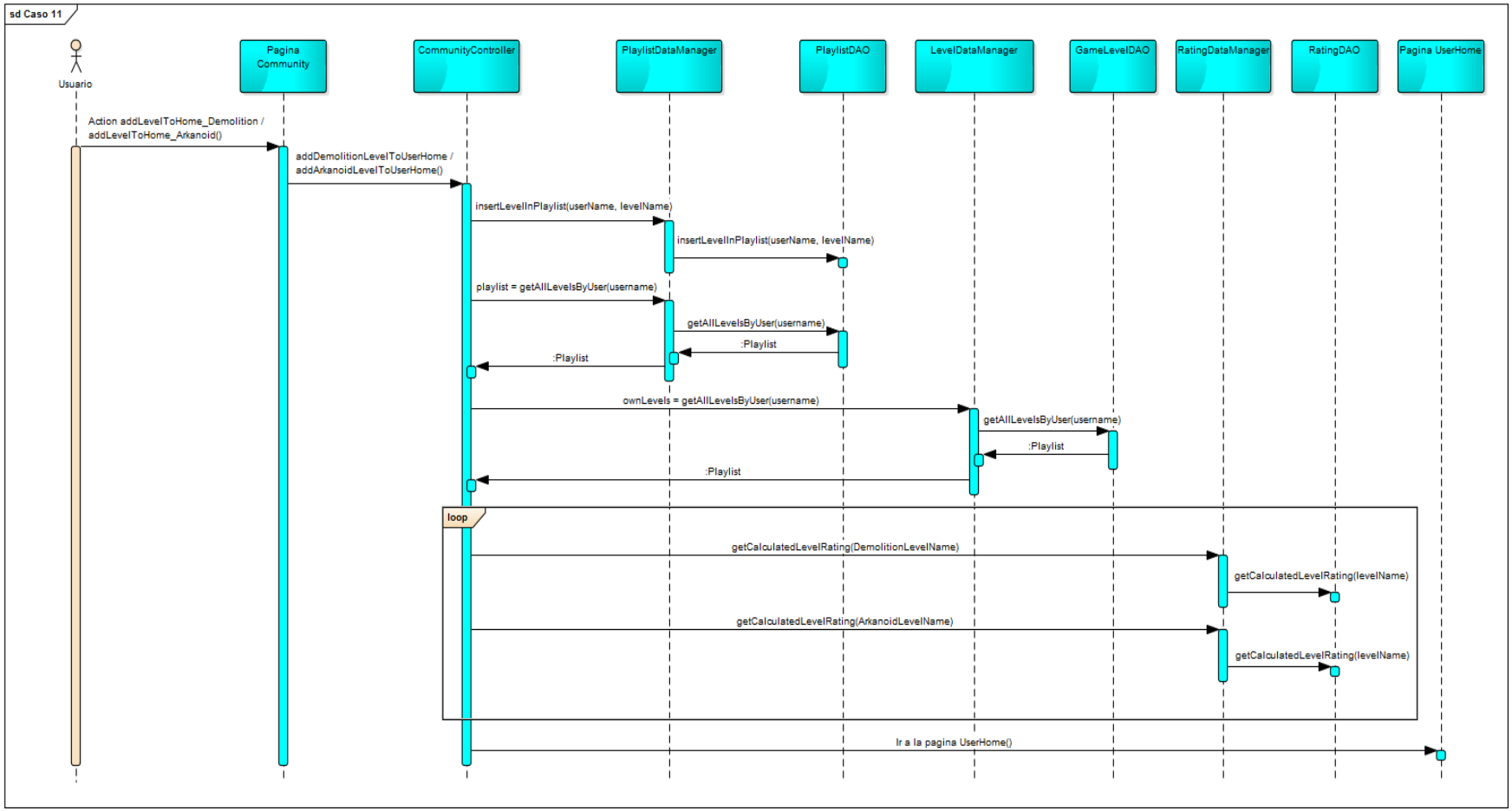


Figura 84 Diagrama de secuencia 10: Casos de uso 11

6.3.11 Caso de Uso 14: Volver al Home del usuario

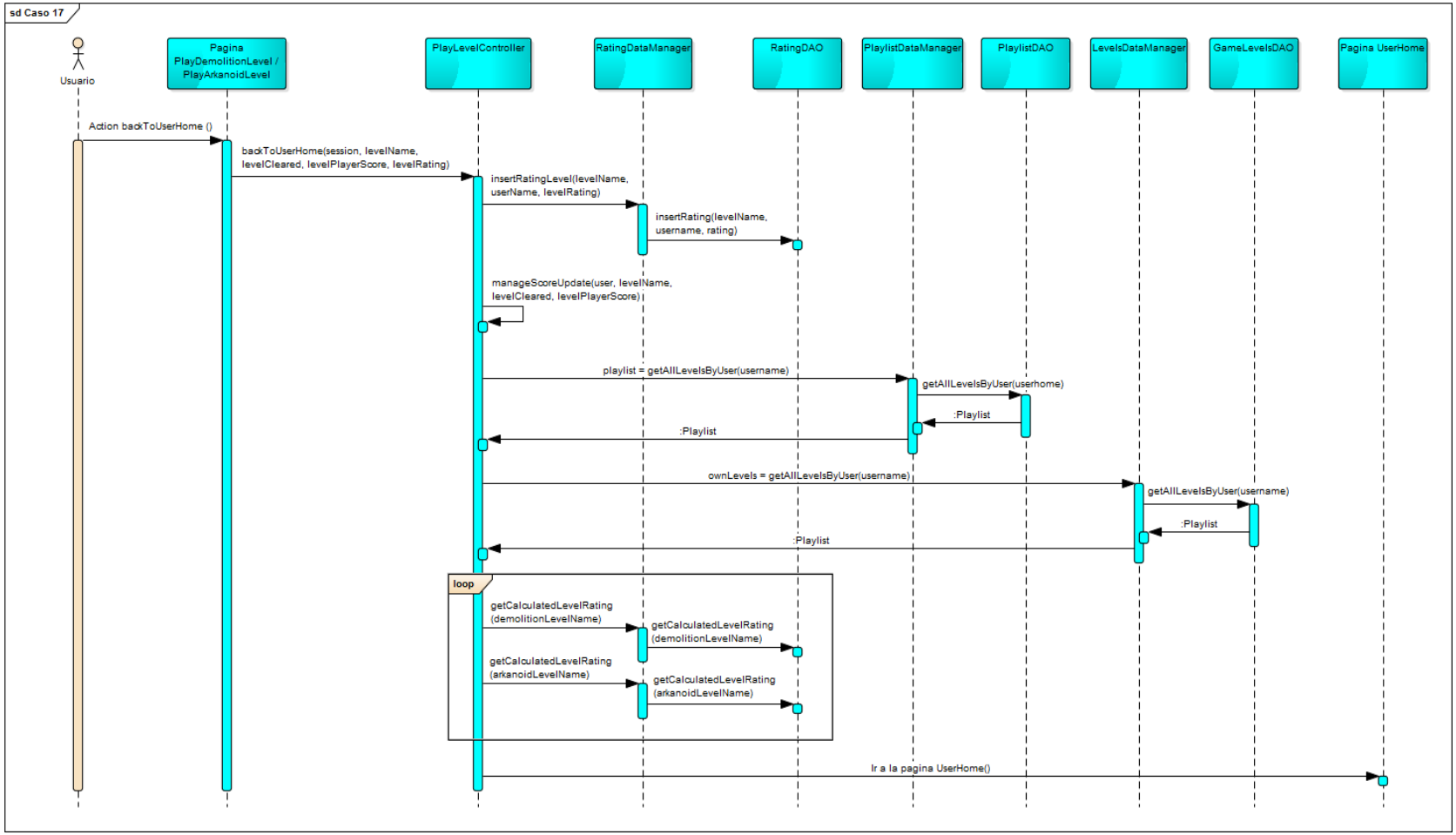


Figura 85 Diagrama de secuencia 12: Casos de uso 14

6.3.12 Caso de Uso 26: Guardar nivel

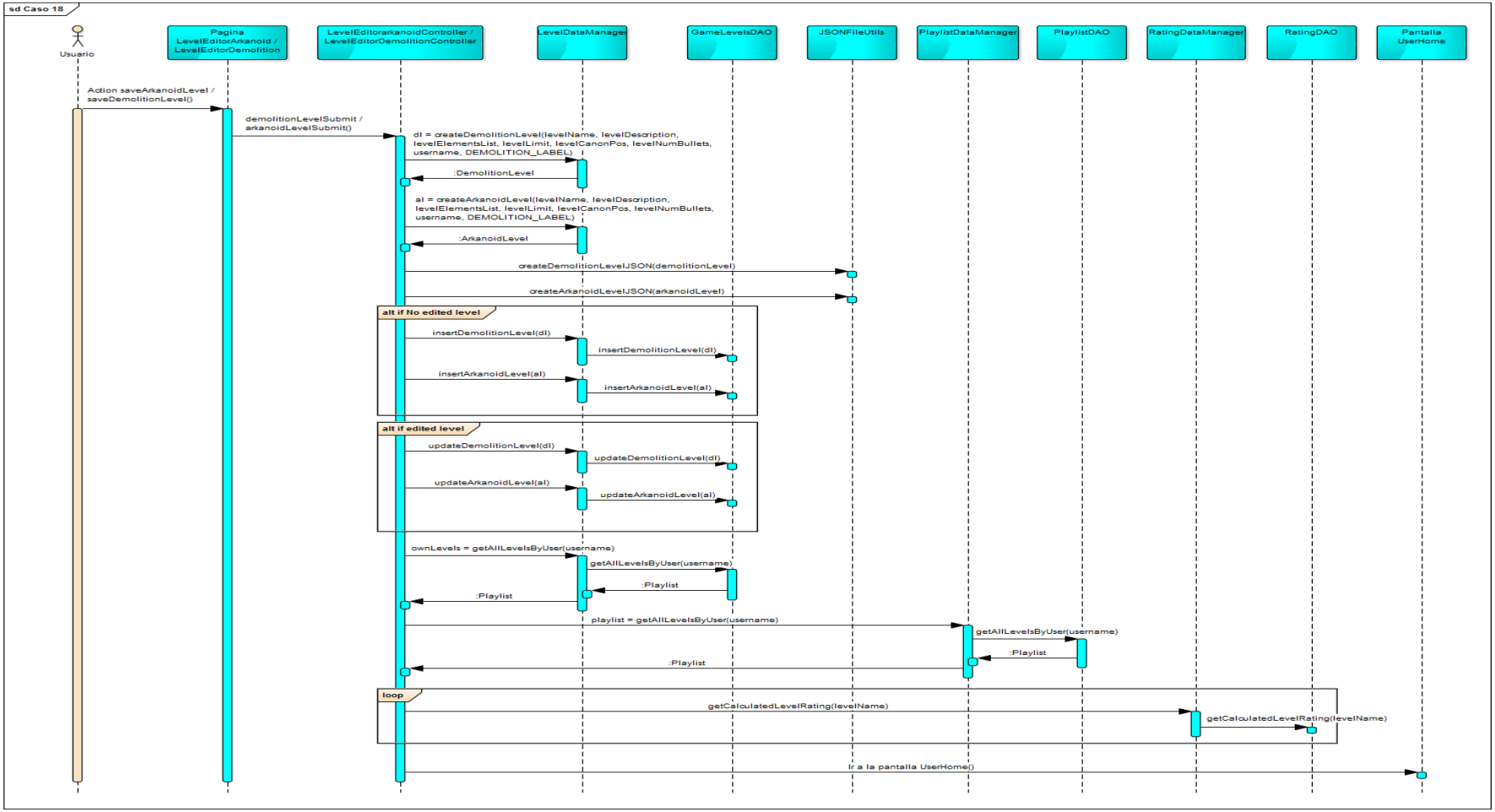


Figura 86 Diagrama de secuencia 13: Casos de uso 26

6.4 Diagramas de Actividades

Los dos aspectos que definen a este proyecto son la creación de contenido y los videojuegos, por lo que resulta interesante detallar el cómo se realizan estas operaciones.

El primer diagrama muestra como es el proceso de creación de contenido para los juegos presentados en la plataforma. En este diagrama se muestran los procesos separados para cada uno de los editores de contenido, el de "Arkanoid" a la izquierda y el "Demolition" a la derecha.

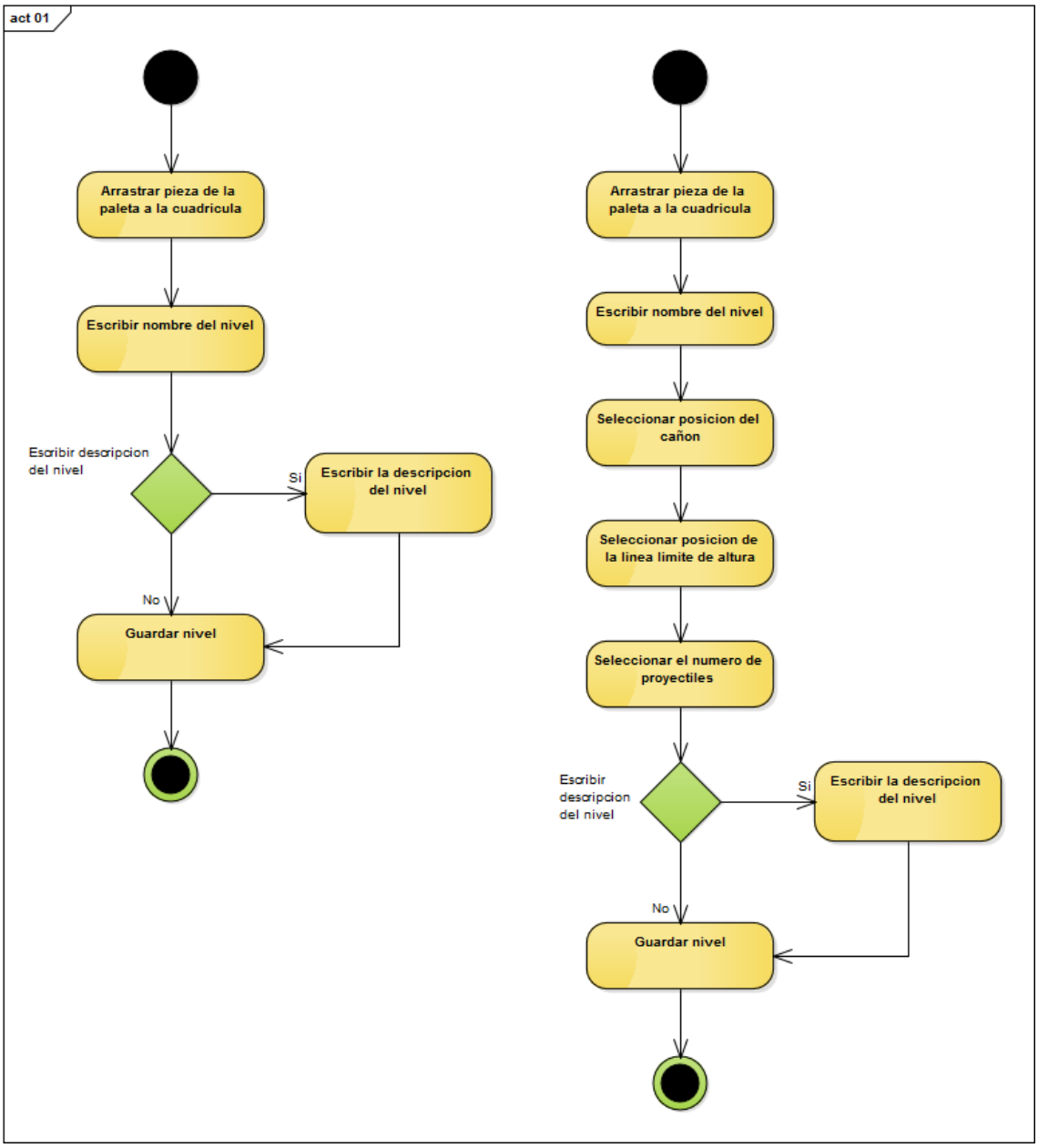


Figura 87 Diagrama de actividad 1

Las operaciones que componen estos procesos pueden realizarse en otro orden para conseguir el mismo resultado, lo mostrado anteriormente seria el proceso lógico de acciones.

El siguiente diagrama muestra como es el proceso de actividad dentro del juego “Demolition”.

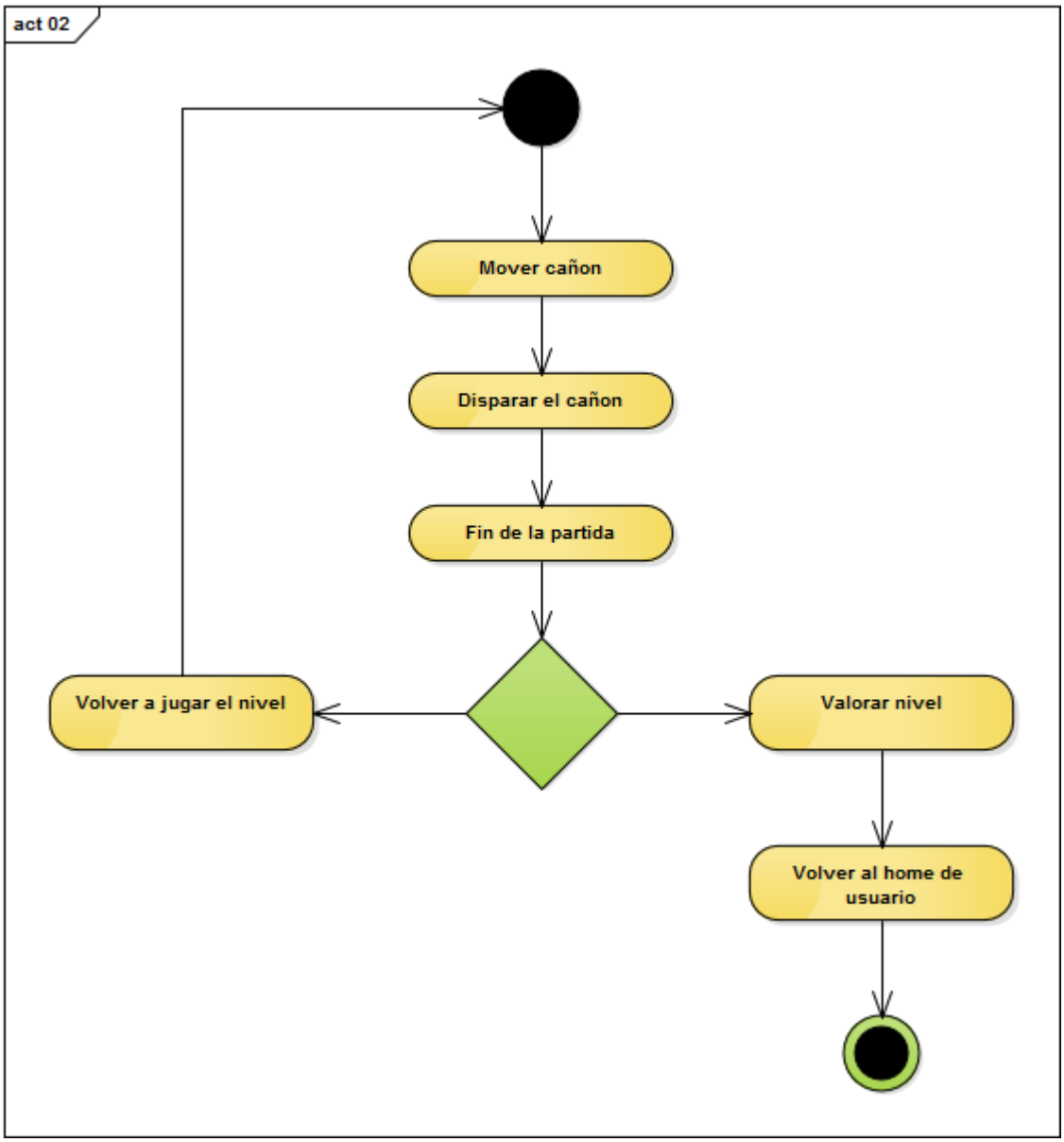


Figura 88 Diagrama de actividad 2

El último diagrama muestra como es el proceso de actividad dentro del juego “Arkanoid”.

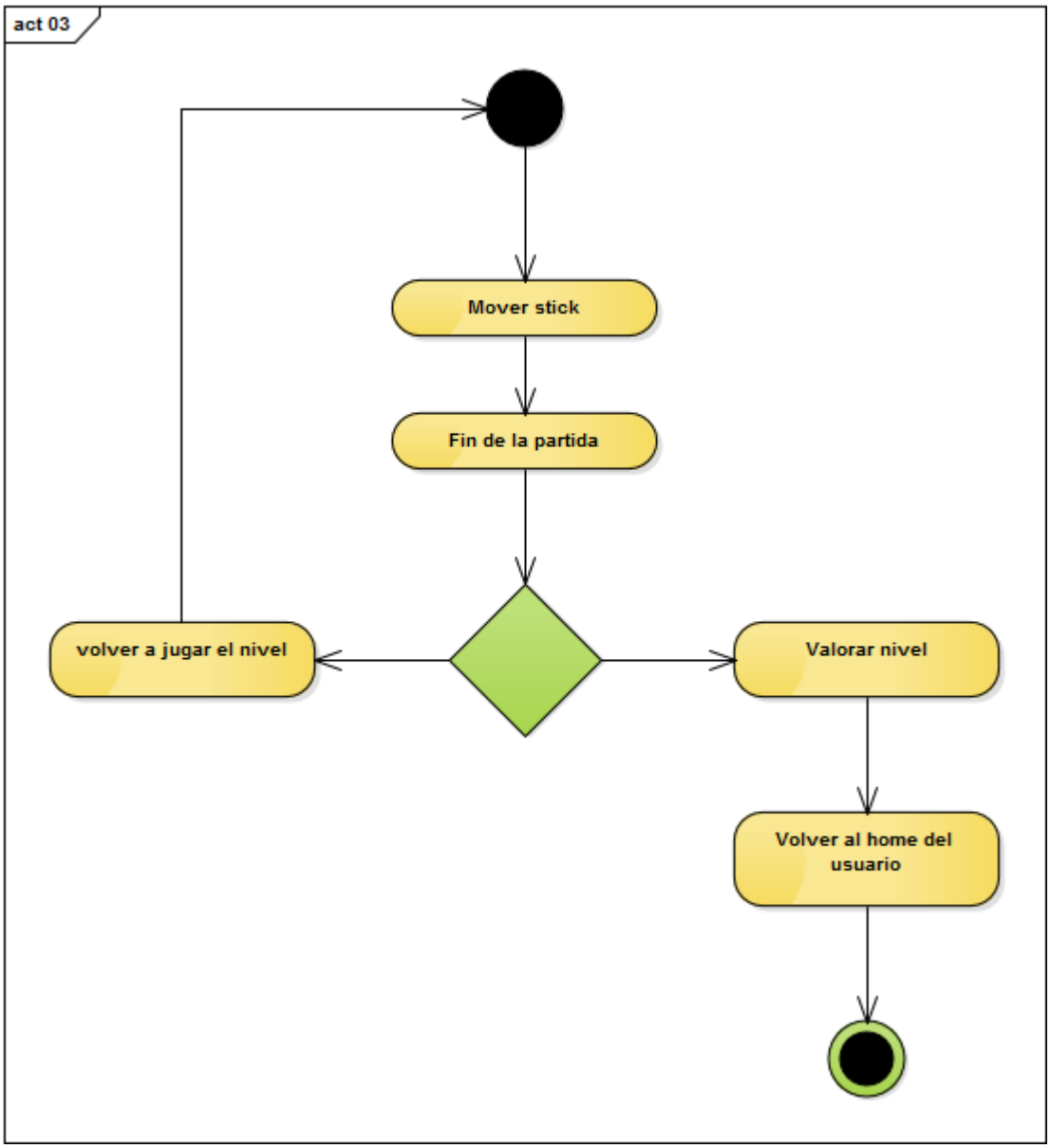


Figura 89 Diagrama de actividad 3

6.5 Diseño de la Base de Datos

6.5.1 Descripción del SGBD Usado

Para el desarrollo de este proyecto se ha utilizado principalmente una base de datos MySQL, concretamente el SGDB MySQL Workbench 6.2.

6.5.2 Integración del SGBD en Nuestro Sistema

La manera en cómo se integra esta base de datos con el proyecto es a través de patrones DAO en las clases Java del proyecto. La aplicación contiene una clase DAO que maneja cada una de las tablas de la base de datos.

La manera en la que la base de datos y la aplicación se comunican entre sí es mediante un driver JDBC implementado en una de las clases del proyecto y con una librería para el manejo de la base de datos integrada también en la aplicación Java.

Para este proyecto se han creado dos bases de datos exactamente iguales, una para desarrollo y otra para realizar las pruebas.

Estas son las tablas utilizadas en el proyecto:

GameLevels	
Campo	**Tipo de dato**
LevelName	VARCHAR(100)
Description	VARCHAR(500)
Type	VARCHAR(3)
Author	VARCHAR(50)
FilePath	VARCHAR(500)

Users	
Campo	**Tipo de dato**
Username	VARCHAR(50)
Password	VARCHAR(50)

Playlist	
Campo	**Tipo de dato**
PlaylistUser	VARCHAR(50)
Level	VARCHAR(100)
PlayerScore	INT(11)
Clear	TINYINT(1)

Rating	
Campo	**Tipo de dato**
User	VARCHAR(50)
Level	VARCHAR(100)
Rating	INT(11)

6.5.3 Diagrama E-R

Este diagrama hace referencia tanto a la base de datos de desarrollo como a la base de datos de pruebas.

Figura 90 Diagrama Entidad-Relación de la base de datos

6.6 Diseño de la Interfaz

En esta sección se muestran los diseños definitivos de todas las pantallas mostradas en el análisis.

6.6.1 Ventana de Login

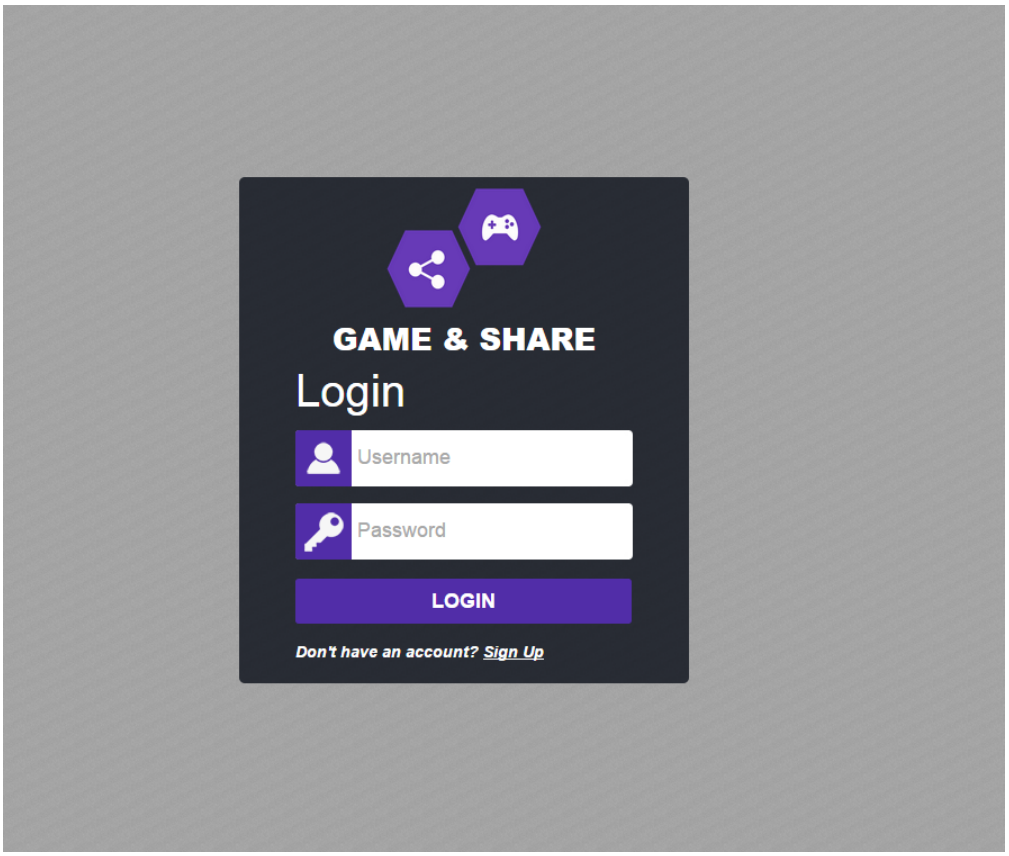


Figura 91 Interfaces: Ventana Login

Este es el diseño final que tendrá la pantalla de login de usuario, que a su vez es la primera pantalla que se encontrar el usuario al entrar en el portal web.

Esta pantalla no ha cambiado apenas en comparación con la mostrada en la fase de análisis. El cambio más significativo es que desaparece el segundo botón que daba acceso a la zona de registro de un nuevo usuario y es sustituido por un enlace que hace la misma función. El resto de elementos que componen la ventana no han sufrido cambios.

La funcionalidad de los elementos presentados en esta pantalla es exactamente la misma que la descrita en el análisis presentado.

Un aspecto que sí que se ha añadido con respecto al análisis son las pantallas de error que derivan de esta. Cuando en esta pantalla introducimos un usuario o una contraseña que es errónea se nos muestra esta ventana de error.

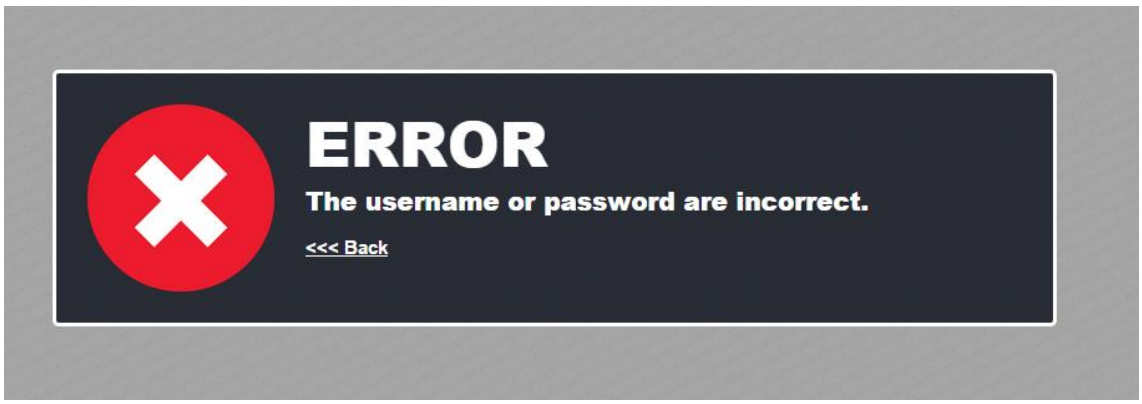


Figura 92 Interfaces: Ventana Error 1

Esta ventana de error simplemente muestra un mensaje al usuario diciéndole que el usuario o la contraseña que ha introducido son incorrectas.

6.6.2 Ventana de Registro de un nuevo usuario

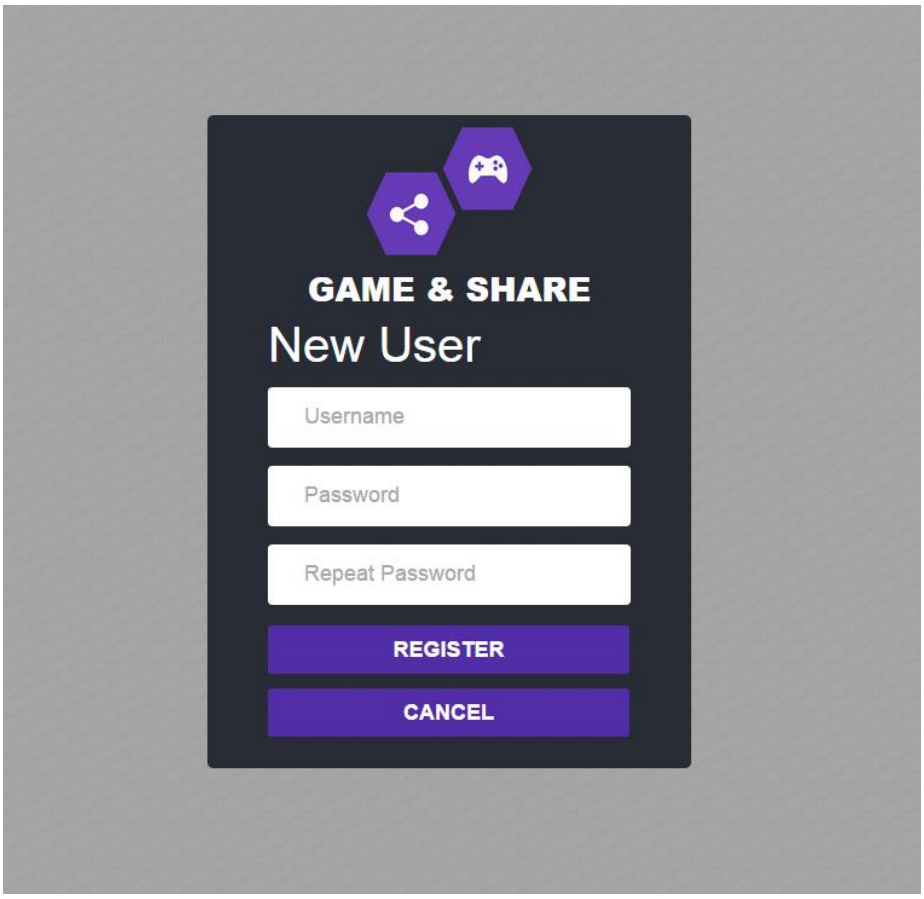


Figura 93 Interfaces: Ventana Register

Esta ventana no ha sufrido ningún cambio con respecto a la mostrada en el análisis, tanto en lo referente a la estructura y a los elementos que presenta en pantalla y también en lo referente a la funcionalidad de los elementos que componen esta ventana.

Esta ventana también tiene un añadido igual que la anterior, añade pantallas de error cuando se dan ciertas condiciones.

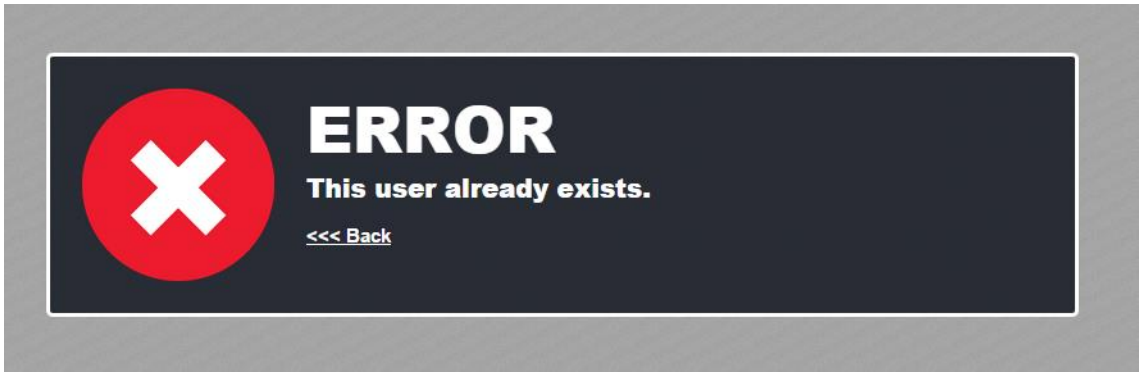


Figura 94 Interfaces: Ventana Error 2

Esta ventana de error se muestra al usuario cuando se introduce un nombre de usuario que ya existe dentro del sistema.

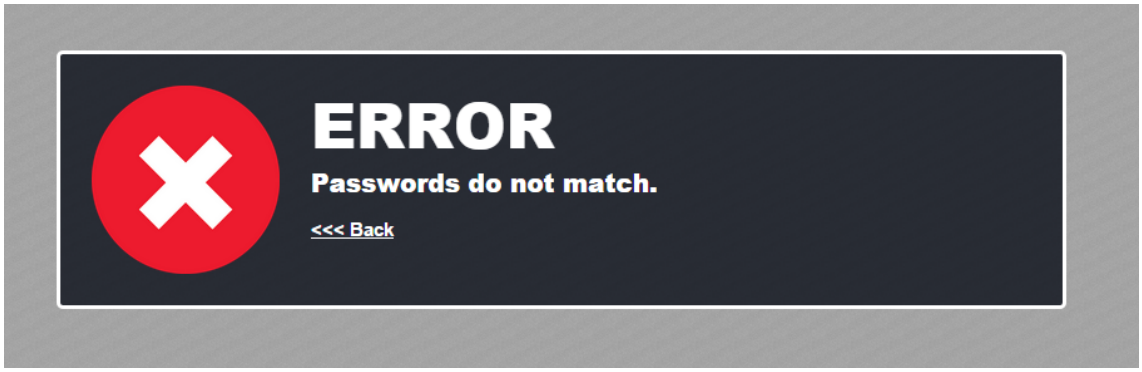


Figura 95 Interfaces: Ventana Error 3

Esta ventana de error se muestra cuando el usuario escribe una contraseña y la repetición no coincide con la contraseña introducida.

6.6.3 Ventana User Home

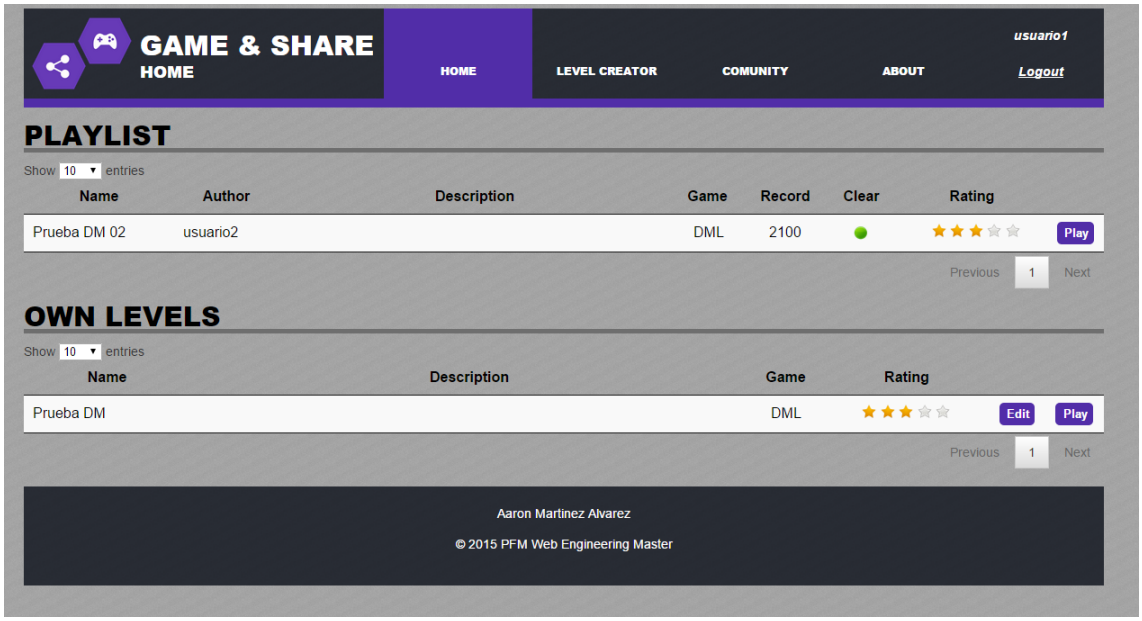


Figura 96 Interfaces: Ventana User Home

La pantalla user home es muy parecida a la presentada en el análisis pero incluye algunas modificaciones.

Lo primero a destacar es la distribución del menú superior, esta redistribución no se ha hecho solo en esta pantalla sino que abarca todas las pantallas de la web, dado que la cabecera es común a todas ellas. Esta redistribución consiste en mover el logo de la página a la parte izquierda de la parte superior para dejar el menú en la parte central. También se ha movido la sección de logout a la parte derecha de la cabecera.

En la parte central se han cambiado el orden en el que aparecen las listas de niveles del usuario, dejando en la parte superior la playlist y en la parte inferior los niveles propios del usuario.

Otro cambio significativo son los controles que aparecen en las listas para manejar el número de entradas que se muestran y la paginación.

Las columnas de dichas listas también se han modificado añadiendo algún campo más. En los niveles de la playlist se han añadido las columnas “Description” y “Clear”.

- **Description:** En este campo aparece la descripción que el usuario ha dado al nivel.
- **Clear:** En este campo se muestra un indicador para ver si el jugador ha superado o no el nivel. Cuando el nivel se ha superado aparece un indicador verde y cuando el nivel esta sin completar aparece un indicador rojo.

Otra modificación que ha sufrido esta pantalla es la manera en la que se muestra la puntuación de cada uno de los niveles, se han sustituido los números por un medidor de estrellas.

6.6.4 Ventana del juego Demolition

Figura 97 Interfaces: Ventana de juego Demolition 1

Esta ventana es exactamente igual a la mostrada en el análisis, tiene la particularidad de que los botones de la parte derecha que se muestran en el prototipo del análisis en esta pantalla se han ocultado para ser mostrados en el momento que termine la partida.

Una cosa que se han añadido en la pantalla de juego son las puntuaciones y el número de proyectiles que se pueden utilizar en el juego.

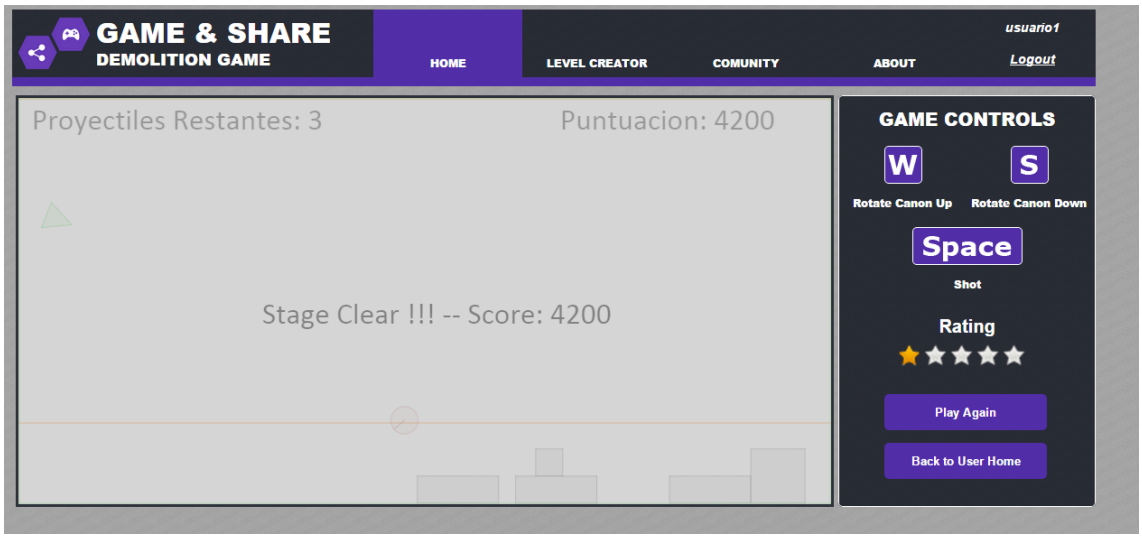


Figura 98 Interfaces: Ventana de juego Demolition 2

Cuando termina la partida vemos cómo cambia la pantalla, la zona de juego se oscurece y se pausa para mostrar un mensaje de fin de partida, se muestra un mensaje u otro depende de si se ha superado o no el nivel.

Al terminar la partida, en la parte derecha de la pantalla, se muestran los botones que se podían ver en el prototipo del análisis.

6.6.5 Ventana del juego Arkanoid

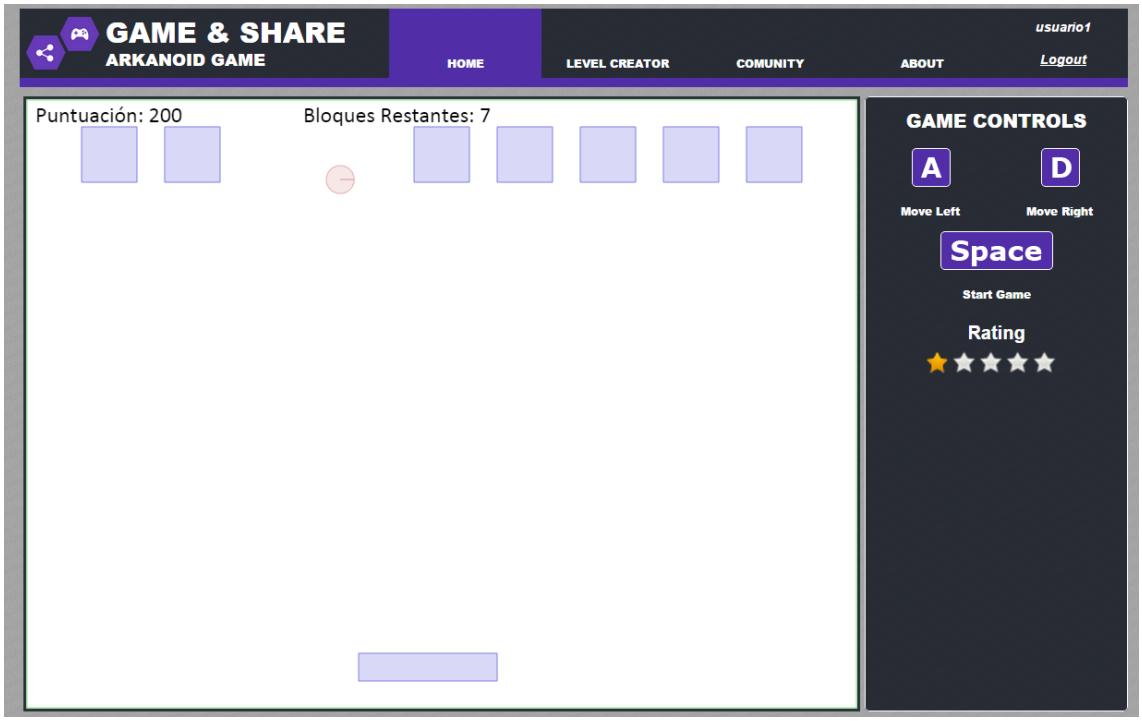


Figura 99 Interfaces: Ventana de juego Arkanoid 1

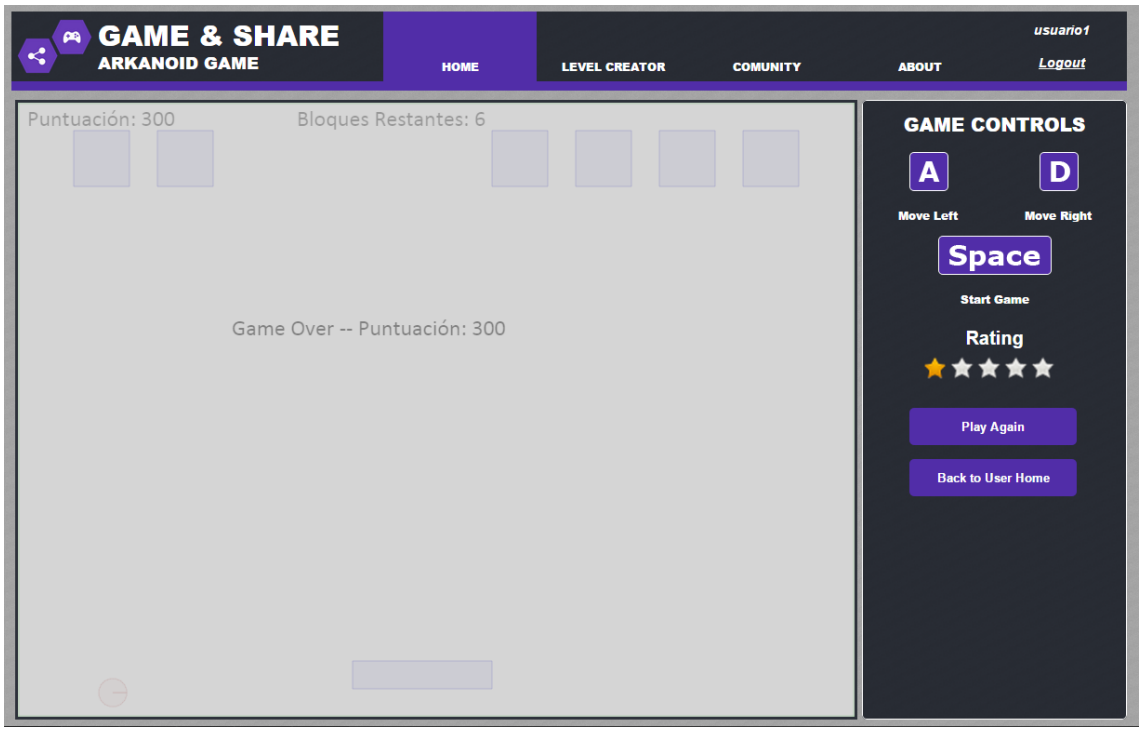


Figura 100 Interfaces: Ventana de juego Arkanoid 2

Las ventanas para este juego son exactamente iguales a las del juego Demolition, los elementos están dispuestos en el mismo sitio y poseen la misma funcionalidad.

6.6.6 Ventana de selección de la herramienta de edición.

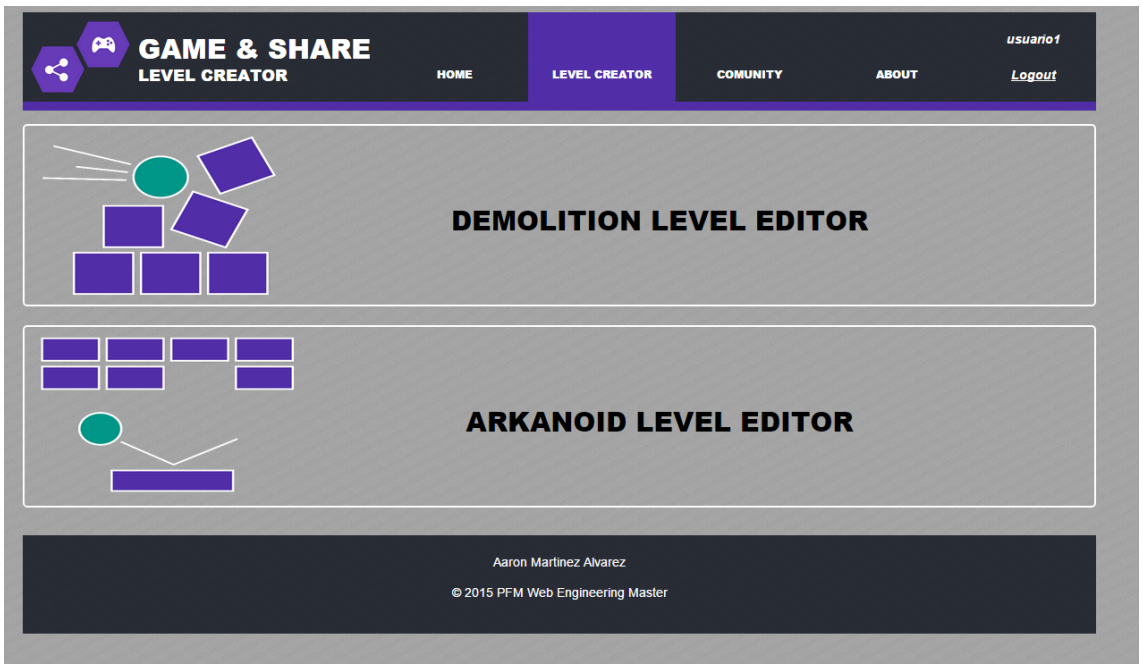


Figura 101 Interfaces: Ventana de selección de la herramienta de edición

Esta pantalla no estaba contemplada en el análisis pero era necesaria añadirla en la versión final de la aplicación. Esta ventana presenta dos grandes botones con los que se pueden seleccionar los distintos editores que implementa la herramienta de edición.

6.6.7 Ventana de edición de contenido para el juego Demolition.

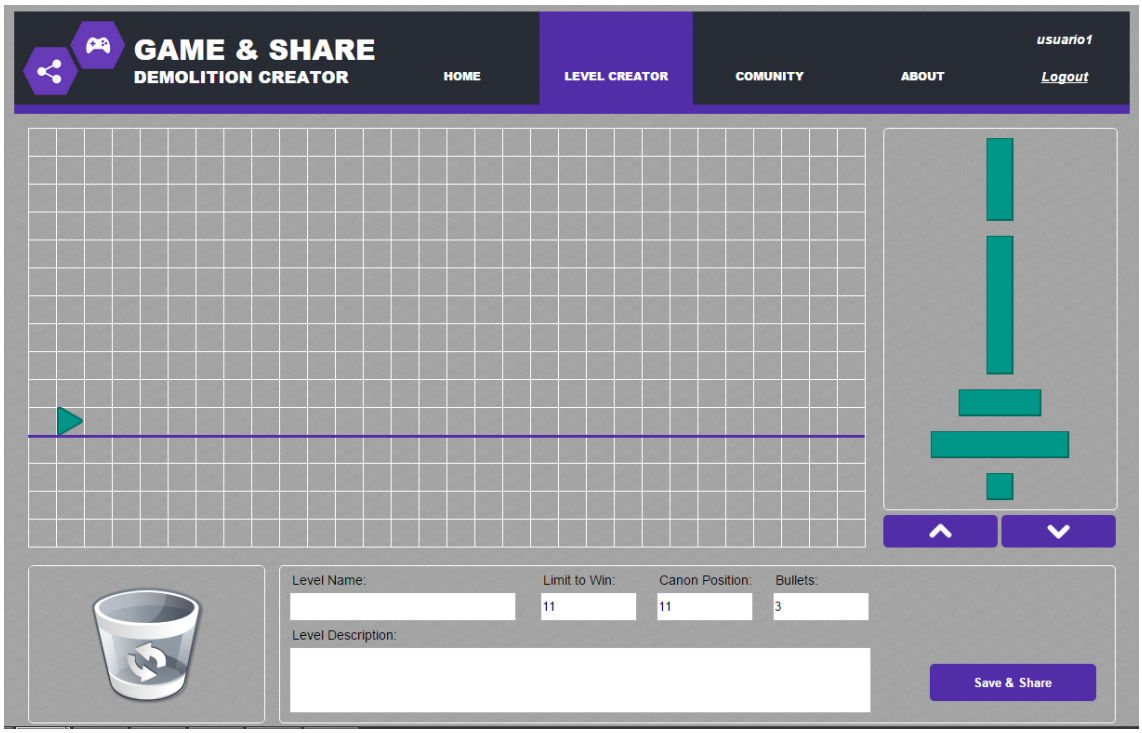


Figura 102 Interfaces: Ventana de edición de contenido para el juego Demolition

Esta ventana es muy similar a la vista en el análisis pero añade muchos detalles porque se requieren necesarios para cubrir todos los aspectos de la edición de este tipo de niveles.

Uno de los cambios introducidos en esta pantalla son los elementos que se pueden ver en la cuadricula: el triángulo que representa el cañón y la línea límite para superar el nivel. Estos elementos tienen asociados unos controles en la zona inferior con los que se puede editar su posición dentro de la cuadricula. Otro de los nuevos elementos introducidos en esta pantalla es el control para seleccionar el número de proyectiles que se podrán usar en el nivel creado.

El resto de elementos permanecen intactos con respecto al prototipo enseñado en el análisis.

6.6.8 Ventana de edición de contenido para el juego Arkanoid.

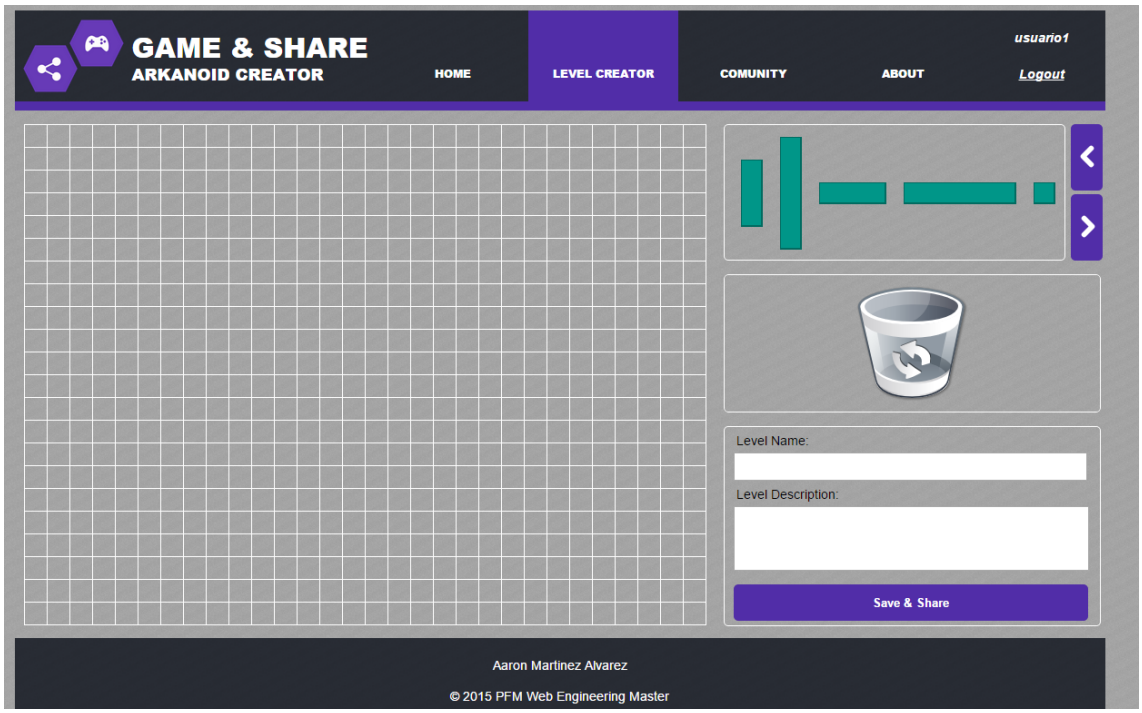


Figura 103 Interfaces: Ventana de edición de contenido para el juego Arkanoid

Esta pantalla es muy similar a la ventana de edición de contenido para el juego "Demolition", la diferencia que hay con dicha pantalla es la colocación de los elementos que la componen.

En esta pantalla se coloca la cuadricula ocupando la totalidad de la parte izquierda de la ventana. En la zona de la derecha vemos como se redistribuyen el resto de elementos, la paleta de elementos se ha reducido en tamaño y se coloca en horizontal. La papelera se coloca bajo la paleta y en la parte inferior de esta zona se colocan los campos para escribir los datos del nivel.

Esta redistribución es necesaria debido a que la pantalla de juego para los niveles Arkanoid es mayor que la de los niveles del juego Demolition, por lo que la cuadricula para editar dichos niveles ha de ser mayor.

6.6.9 Ventana Community

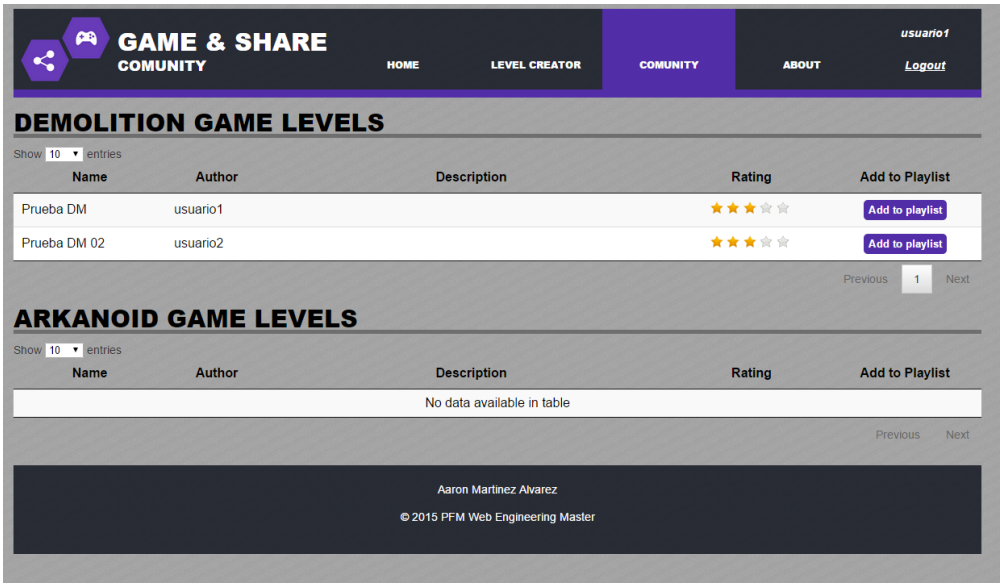


Figura 104 Interfaces: Ventana Community

Esta ventana es prácticamente idéntica a la presentada en el diseño, salvo con el cambio en la visualización de las puntuaciones que pasan de ser numéricos a ser un medidor de estrellas.

6.6.10 Ventana About

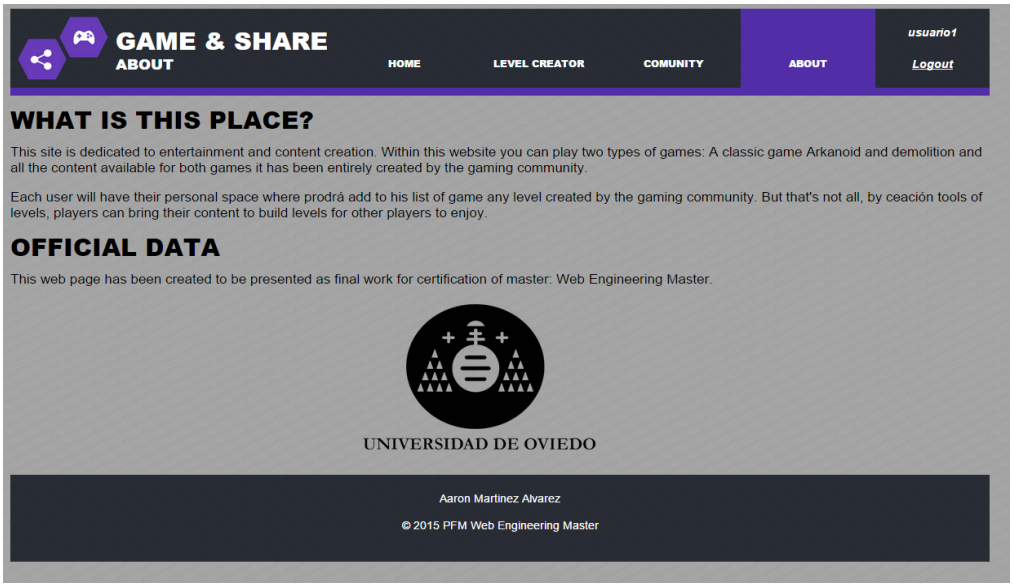


Figura 105 Interfaces: Ventana About

Esta ventana es exactamente la misma pantalla que el prototipo presentado en la sección de análisis.

6.7 Especificación Técnica del Plan de Pruebas

6.7.1 Pruebas Unitarias

Las pruebas unitarias presentadas para este proyecto se han dividido por subsistemas, ya que las pruebas para cada subsistemas son independientes del resto, con esto se consigue verificar y aislar los fallos en cada una de las partes de la aplicación.

6.7.1.1 Batería de pruebas del subsistema de niveles de juego

test_existLevelOk()
Entrada:
La entrada que se ha de proporcionar es un String con el nombre del nivel.
Operación:
Este método consultara en la base de datos si el nombre del nivel ya está en la base de datos.
Salida
La salida que se espera obtener es un booleano a *True*.

test_existLevelFail()
Entrada:
La entrada que se ha de proporcionar es un String con el nombre del nivel.
Operación:
Este método consultara en la base de datos si el nombre del nivel ya está en la base de datos.
Salida
La salida que se espera obtener es un booleano a *False*.

6.7.1.1.1 Pruebas concretas para los niveles Arkanoid

test_getAllArkanoidLevels()
Entrada:
Este método no recibe ninguna entrada.
Operación:
El método buscará en la base de datos todos los niveles del tipo Arkanoid y obtendrá una lista con dichos niveles.
Salida
Se espera obtener una lista de niveles del tipo Arkanoid bien construidos y con todos sus atributos rellenados.

test_readArkanoidLevelJSON_ok()
Entrada:
Este método recibe como entrada un String con el path del fichero JSON que se va a leer.
Operación:
Este método leerá el fichero JSON del nivel y creara un objeto ArkanoidLevel con todos los datos que contiene el fichero JSON.
Salida
Se espera obtener un nivel del tipo Arkanoid creado con los datos que contiene el fichero JSON.

test_readArkanoidLevelJSON_Fail()
Entrada:
Este método recibe como entrada un String con el path del fichero JSON que se va a leer.
Operación:
Este método leerá el fichero JSON del nivel y creara un objeto ArkanoidLevel con todos los datos que contiene el fichero JSON.
Salida
Se espera obtener una excepción IOException, dado que el fichero no existe.

test_readArkanoidLevelJSON_FormatFail()
Entrada:
Este método recibe como entrada un String con el path del fichero JSON que se va a leer.
Operación:
Este método leerá el fichero JSON del nivel y creara un objeto ArkanoidLevel con todos los datos que contiene el fichero JSON.
Salida
Se espera obtener una excepción ParseException, dado que el fichero no tiene el formato requerido.

6.7.1.1.2 Pruebas concretas para los niveles Demolition

test_getAllDemolitionLevels()
Entrada:
Este método no recibe ninguna entrada.
Operación:
El método buscará en la base de datos todos los niveles del tipo Demolition y obtendrá una lista con dichos niveles.
Salida
Se espera obtener una lista de niveles del tipo Demolition bien construidos y con todos sus atributos rellenados.

test_readDemolitionLevelJSON_ok()
Entrada:
Este método recibe como entrada un String con el path del fichero JSON que se va a leer.
Operación:
Este método leerá el fichero JSON del nivel y creara un objeto DemolitionLevel con todos los datos que contiene el fichero JSON.
Salida
Se espera obtener un nivel del tipo Demolition creado con los datos que contiene el fichero JSON.

test_readDemolitionLevelJSON_Fail()
Entrada:
Este método recibe como entrada un String con el path del fichero JSON que se va a leer.
Operación:
Este método leerá el fichero JSON del nivel y creara un objeto DemolitionLevel con todos los datos que contiene el fichero JSON.
Salida
Se espera obtener una excepción IOException, dado que el fichero no existe.

test_readDemolitionLevelJSON_FormatFail()
Entrada:
Este método recibe como entrada un String con el path del fichero JSON que se va a leer.
Operación:
Este método leerá el fichero JSON del nivel y creara un objeto DemolitionLevel con todos los datos que contiene el fichero JSON.
Salida
Se espera obtener una excepción ParseException, dado que el fichero no tiene el formato requerido.

6.7.1.2 Batería de pruebas del subsistema de las playlists

test_getAllLevelsByUser()
Entrada:
Se requiere como entrada un String con el nombre del usuario propietario de la playlist.
Operación:
El método buscara en la base de datos todos los niveles que estén dentro de la playlist del usuario seleccionado.
Salida
Se espera obtener un objeto Playlist con todos los niveles completamente creados, de cualquier tipo que pertenezcan al usuario seleccionado.

test_getLevelScoreAndCleared()
Entrada:
La entrada que se requiere es un String con el nombre de usuario dueño de la playlist y un String con el nombre del nivel.
Operación:
Este método busca en la base de datos el nivel seleccionado y devuelve las puntuaciones que el usuario ha obtenido en ese nivel.
Salida
Se espera obtener un Map<String, Integer> donde el string sea un tag que identifique el tipo de dato y el entero sea o la puntuación obtenida o si se ha superado el nivel o no.

6.7.1.3 Batería de pruebas del subsistema de ratings

test_getCalculatedLevelRating()
Entrada:
Este método requiere como entrada un String con el nombre del nivel seleccionado.
Operación:
Este método buscara en la base de datos la valoración media que los jugadores le han dado al nivel.
Salida
Se espera obtener un entero con la puntuación media del nivel.

6.7.1.4 Batería de pruebas del subsistema de usuarios

test_getUserOK()
Entrada:
Este método requiere como entradas un String con el nombre de usuario y un String con la contraseña del usuario.
Operación:
Este método buscara en la base de datos el usuario y comprobara que la contraseña introducida coincida con la de entrada.
Salida
Se espera obtener un objeto usuario con los datos que se han introducido.

test_getUserFail()
Entrada:
Este método requiere como entradas un String con el nombre de usuario y un String con la contraseña del usuario.
Operación:
Este método buscara en la base de datos el usuario y comprobara que la contraseña introducida coincida con la de entrada.
Salida
Se espera obtener un objeto usuario con sus campos a null.

test_existUserOk()
Entrada:
Este método requiere en la entrada un String con el nombre del usuario.
Operación:
Este método busca en la base de datos el nombre del usuario para ver si existe o no.
Salida
La salida que se espera obtener es un booleano a *True*.

test_existUserFail()
Entrada:
Este método requiere en la entrada un String con el nombre del usuario.
Operación:
Este método busca en la base de datos el nombre del usuario para ver si existe o no.
Salida
La salida que se espera obtener es un booleano a *False*.

6.7.2 Pruebas de Integración y del Sistema

Este tipo de pruebas se basan en colocar una serie de precondiciones o postcondiciones a los distintos módulos de la aplicación para verificar que las acciones que realiza el usuario sobre la misma.

6.7.2.1 Caso de uso: Crear usuario

Prueba	Crear un usuario que aún no existe.
Datos de entrada	String con el nombre de usuario. String con la contraseña del usuario. String con la repetición de la contraseña del usuario.
Resultado positivo	El usuario se crea correctamente en la base de datos.
Resultado negativo	No se puede dar este caso.

Prueba	Crear un usuario que ya existe.
Datos de entrada	String con el nombre de usuario. String con la contraseña del usuario. String con la repetición de la contraseña del usuario.
Resultado positivo	No se puede dar este caso.
Resultado negativo	Se muestra una página informando del error y el sistema permanece sin cambios.

Prueba	Cancelar operación.
Datos de entrada	Ninguna.
Resultado positivo	Se regresa a la página anterior y el sistema permanece sin cambios.
Resultado negativo	No se puede dar este caso.

6.7.2.2 Caso de uso: Añadir nivel a la playlist del usuario

Prueba	Añadir un nivel que el usuario no tiene en su playlist.
Datos de entrada	String con el nombre del nivel. String con el nombre del usuario.
Resultado positivo	El sistema posee una nueva entrada en la playlist del usuario.
Resultado negativo	No se puede dar este caso.

Prueba	Añadir un nivel que el usuario ya tiene en su playlist.
Datos de entrada	String con el nombre del nivel. String con el nombre del usuario.
Resultado positivo	No se puede dar este caso.
Resultado negativo	Se genera una excepción internamente que la aplicación gestiona y el sistema permanece sin cambios.

6.7.2.3 Caso de uso: Arrastrar pieza de la paleta a la cuadricula

Prueba	Arrastrar pieza de la paleta a la cuadricula.
Datos de entrada	String con el tipo de pieza arrastrada. Int con la posición x dentro de la cuadricula. Int con la posición y dentro de la cuadricula.
Resultado positivo	El mapa actual posee una pieza más.
Resultado negativo	No se puede dar este caso.

Prueba	Arrastrar pieza de la paleta fuera de la cuadricula.
Datos de entrada	Ninguno.
Resultado positivo	El mapa actual no posee ninguna pieza más y el sistema permanece sin cambios.
Resultado negativo	No se puede dar este caso.

6.7.2.4 Caso de uso: Eliminar pieza

Prueba	Arrastrar la pieza a la papelera.
Datos de entrada	String con el tipo de pieza arrastrada. Int con la posición x dentro de la cuadricula. Int con la posición y dentro de la cuadricula.
Resultado positivo	El mapa actual posee una pieza menos.
Resultado negativo	No se puede dar este caso.

6.7.2.5 Caso de uso: Guardar nivel

Prueba	Guardar nivel que no exista.
Datos de entrada	String con el nombre del nivel. String con la descripción del nivel.
Resultado positivo	El sistema posee un nivel más guardado en la base de datos.
Resultado negativo	No se puede dar este caso.

Prueba	Guardar nivel que ya exista.
Datos de entrada	String con el nombre del nivel. String con la descripción del nivel.
Resultado positivo	No se puede dar este caso.
Resultado negativo	Se genera una excepción internamente que la aplicación gestiona y el sistema permanece sin cambios.

6.7.3 Pruebas de Usabilidad y Accesibilidad

Estas pruebas irán enfocadas a distintos grupos de usuarios, desde los usuarios familiarizaos con el mundo de los videojuegos o con el manejo de internet a los que no lo estén.

Para realizar estas pruebas se han diseñado una serie de cuestionarios que los usuarios seleccionados para hacer el test han de rellenar y que posteriormente se evaluaran para tomar acciones sobre que modificaciones hay que introducir en el proyecto.

6.7.3.1 6.6.3.1 Diseño de Cuestionarios

6.7.3.1.1 Preguntas de carácter general

¿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
¿Usa internet frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza en internet?
1. Es necesario para el desempeño de mi trabajo o profesión 2. Lo uso para ocio principalmente 3. Lo uso para ver las noticias. 4. Únicamente para acceder a redes sociales y consulta el correo.
¿Ha usado alguna vez alguna aplicación como la de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 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

6.7.3.1.2 Actividades guiadas

Lista de actividades guiadas que se le propondrán a los usuarios que realicen el test.

1. Acceder a la página con un usuario de pruebas.
2. Crear un nuevo usuario.
3. Crear un nivel para el juego Arkanoid.
4. Crear un nivel para el juego Demolition.
5. Jugar a un nivel del juego Arkanoid.
6. Jugar a un nivel del juego Demolition.
7. Añadir un nivel creado por otro usuario a su playlist.

6.7.3.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

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?				
¿La funcionalidad se ajusta a lo esperado?				
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?				

6.7.3.1.4 Cuestionario para el Responsable de las Pruebas

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	
Número de operaciones repetidas por el usuario.	

6.7.3.2 Pruebas de Accesibilidad

Las pruebas de accesibilidad que se van a implementar en este proyecto se basan en una checklist con todos los puntos recogidos en las normas WCAG 2.0 a distintos niveles de conformidad: A, AA, AAA.

Para que la aplicación tenga un grado de aceptación satisfactorio ha de cumplir el nivel AA de esta checklist.

6.7.4 Pruebas de Rendimiento

Las pruebas diseñadas para medir el rendimiento de la aplicación consistirán en medir los tiempos que tarda la aplicación en realizar las tareas más críticas para el perfecto funcionamiento de la misma.

Se han seleccionado estas operaciones para ser medidas:

- Inicio de sesión.
- Cargar un nivel para ser jugado.
- Guardar un nivel.
- Cargar la página “Community”.
- Añadir un nivel a la playlist de un jugador.

Para que se pueda decir que estas pruebas han tenido éxito se espera que las operaciones seleccionadas no tomen tiempos mayores a los 20ms.

Capítulo 7. Implementación del Sistema

7.1 Lenguajes de Programación

Los lenguajes de programación utilizados para implementar este proyecto han sido: Java y JavaScript.

7.1.1 Java

Es un lenguaje de programación de propósito general, concurrente y orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la maquina en la que se ejecuta.

7.1.2 JavaScript

Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java.

7.2 Lenguajes de marcado

Los lenguajes de marcado utilizados para implementar este proyecto han sido: HTML5 y CSS3.

7.2.1 HTML5

Es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML.

7.2.2 CSS3

Es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML. El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

7.3 Librerías

Las librerías utilizadas para implementar este proyecto han sido: JQuery, JQuery-UI, Box2D, JQuery-Raty, JSON-Simple y Datatables.

7.3.1 JQuery

Es una librería de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Enlace a la página del producto: http://jquery.com/

7.3.2 JQuery-UI

Es una biblioteca de componentes para el framework JQuery que le añaden un conjunto de plug-ins, widgets y efectos visuales para la creación de aplicaciones web.

Enlace a la página del producto: https://jqueryui.com/

7.3.3 Box2D

Es una biblioteca libre que implementa un motor físico en dos dimensiones. Está programada en C++ y se distribuye bajo la licencia zlib.

Enlace a la página del producto: http://box2d.org/

7.3.4 JQuery-Raty

Es una librería desarrollada en JavaScript para implementar funciones de Rating con un medidor de estrellas.

Enlace a la página del producto: http://wbotelhos.com/raty

7.3.5 JSON-Simple

Es una librería desarrollada en JavaScript para crear, leer y manejar ficheros JSON.

Enlace a la página del producto: https://code.google.com/p/json-simple/

7.3.6 Datatables

Plug-in para JQuery para la visualización de tablas formateadas.

Enlace a la página del producto: https://www.datatables.net/

7.4 Herramientas y Programas Usados para el Desarrollo

Estas son las herramientas y frameworks usados para desarrollar este proyecto: Eclipse, Spring MVC Framework, Servidor Tomcat, MYSQL Workbench.

7.4.1 Eclipse

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones.

La versión utilizada para desarrollar este proyecto ha sido la versión "Luna", lanzada en Junio de 2014, actualmente es la última versión de este software.

El motivo de utilizar este software para desarrollar el proyecto ha sido su total integración con el servidor Tomcat, que también ha sido utilizado para este proyecto y la integración con el framework Spring que facilita el trabajo con proyectos web.

Enlace a la página del producto: https://eclipse.org/

7.4.2 Spring MVC Framework

Es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Más concretamente el Framework MVC consiste en: un framework basado en HTTP y servlets, que provee herramientas para la extensión y personalización de aplicaciones web y servicios web REST.

La versión utilizada para este proyecto ha sido la "Spring IDE 3.6.4".

Enlace a la página del producto: https://spring.io/

7.4.3 Servidor Tomcat

Es un contenedor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede funcionar como servidor web por sí mismo. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

La versión usada para este proyecto es "Tomcat 8", que actualmente es la última versión.

Enlace a la página del producto: http://tomcat.apache.org/

7.4.4 MySQL Workbench

Es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

La versión usada para este proyecto es la 6.2.

Enlace a la página del producto: https://www.mysql.com/products/workbench/

7.5 Creación del Sistema

7.5.1 Problemas Encontrados

Han existido dos principales problemas en el desarrollo de este proyecto: el uso de una librería de físicas por primera vez y el uso de un framework como es Spring MVC.

Uno de los grandes problemas ha sido la utilización del framework Spring MVC, esta no es la primera vez que utilizo el framework, ya que en una de las asignaturas se usado este framework a través de una librerías importadas dentro de un proyecto JEE normal.

Al intentar reproducir ese mismo proyecto me he encontrado con muchos problemas, ya que dichas librerías requieren muchas dependencias y es muy difícil de controlar.

La solución hallada para este problema fue utilizar un plug-in para eclipse que integra el framwork Spring MVC dentro de un proyecto java. Este plug-in permite crear un proyecto Spring y mediante Maven, se automatiza el proceso de obtener las dependencias que requieren todas las librerías de Spring.

Otro de los problemas fue el manejo de librerías de físicas, durante el desarrollo de este proyecto se han utilizado dos librerías distintas: PhysicsJS y Box2D.

La primera de las librerías utilizadas, PhysicsJS, esta librería es muy sencilla de utilizar y muy intuitiva pero el principal problema que presentaban los prototipos creados con esta librería era el framerate. El framerate de estos prototipos era muy bajo para tener una experiencia jugable mínimamente aceptable.

La segunda de las librerías, Box2D, es más compleja de utilizar, pero compensa todas las carencias que presentaba la anterior librería. Con esta librería se logra una experiencia de juego óptima.

Otro problema surgido a posteriori es que la aplicación no funciona correctamente en dos de los navegadores principales: Safari e Internet Explorer. El problema que se ha encontrados solo hace referencia a la herramienta de edición.

El problema se da cuando intentas arrastrar una de las piezas a la cuadricula, el funcionamiento normal de la aplicación consiste en: cuando se arrastra una pieza de la paleta a la cuadricula esta se clona, por lo que la que se deposita en la cuadricula es un clon. Esta operación se repite siempre que se da esta acción.

El problema encontrado en estos navegadores es que dicha pieza se clona una vez pero las sucesivas veces que se intenta arrastrar la misma pieza esta ya no se clona, por lo que no se crea una nueva pieza.

El funcionamiento de esta característica viene íntegramente implementado por la librería JQuery-UI, por lo que da a pensar que estos navegadores no son del todo compatibles con dicha librería. A día de hoy aún no se ha podido solucionar el problema.

7.5.2 Descripción Detallada de las Clases

En esta sección no se van a describir todas las clases, sino que analizaremos las clases más importantes del proyecto, las que contengan las funcionalidades principales y más complejas de la aplicación.

7.5.2.1 LoginController

Nombre	**Tipo**	**Descripción**	
LoginController	Publica	Controlador que se encarga de realizar todas las operaciones de comunicación con la pagina Login de la vista.	
Responsabilidades			
Número	**Descripción**		
1	Cargar todos los datos del usuario y abrir la página Home del usuario.		
2	Cargar la página de registro de usuarios si es necesario.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Publico	String	loginSubmit	HttpSession session String username String password
Publico	String	goToRegister	HttpServletRequest request
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		UserDataManagerService	udms
Privado		LevelDataManagerService	ldms
Privado		PlaylistDataManagerService	pldms
Privado		RatinLevelDataManagerServices	rldms
Observaciones			

7.5.2.1.1 Descripción de los métodos

```
public String loginSubmit(HttpSession session, @RequestParam("username") String
username, @RequestParam("password") String password)
```

Método encargado de cargar los datos del usuario (el usuario en sí, los niveles que ha creado y los niveles que tiene en su playlist) en la sesión y luego acceder a la página home del usuario.

- HttpSession session: Objeto sesión que se usara para guardar los datos del usuario.
- String username: Nombre introducido por el usuario en la JSP.
- String password: Contraseña introducida por el usuario en la JSP.

```
public String goToRegister(HttpServletRequest request)
```

Metodo encargado de cargar la página de registro de usuario.

- HttpServletRequest request: Objeto request de la petición de la página web.

7.5.2.2 MenuController

Nombre	Tipo	Descripción	
MenuController	Publica	Controlador que se encarga de realizar todas las operaciones de comunicación con las distintas páginas de la vista, ya que esta hace de menú de navegación.	
Responsabilidades			
Número	**Descripción**		
1	Cargar la página del home de usuario.		
2	Cargar la página selectora de editor de nivel		
3	Cargar la página Community		
4	Cargar la página About		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Publico	String	goToHome	HttpSession session
Publico	String	goToLevelEditorSelector	HttpServletRequest request
Publico	String	goToComunity	HttpServletRequest request
Publico	String	goToAbout	HttpServletRequest request
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		LevelDataManagerService	Ldms
Privado		PlaylistDataManagerService	Pldms
Privado		RatinLevelDataManagerServices	rldms
Observaciones			

7.5.2.2.1 Descripción de los métodos

```
public String goToHome(HttpSession session)
```

Método encargado de cargar la página home del usuario, cargando todos sus datos.

- HttpSession session: Objeto sesión que contiene todos los datos de la sesión del usuario.

```
public String goToLevelEditorSelector(HttpServletRequest request)
```

Método que carga la página de selección de herramienta de edición.

- HttpServletRequest request: Objeto request de la petición de la página web.

```
public String goToComunity(HttpServletRequest request)
```

Método que se encarga de cargar la página Community.

- HttpServletRequest request: Objeto request de la petición de la página web.

```
public String goToAbout(HttpServletRequest request)
```

Método que se encarga de cargar la página About.

- HttpServletRequest request: Objeto request de la petición de la página web.

7.5.2.3 PlayLevelController

Nombre	Tipo	Descripción	
PlayLevelController	Publica	Controlador que se encarga de realizar todas las operaciones de comunicación con la página que va a cargar uno de los motores de los juegos con el contenido del nivel seleccionado.	
Responsabilidades			
Número	**Descripción**		
1	Carga los datos del nivel seleccionado en el motor del juego.		
2	Calculo de la valoración media del nivel.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Publico	String	goToPlayLevelDemolition	HttpSession session String levelName String levelPath
Publico	String	goToPlayLevelArkanoid	HttpSession session String levelName String levelPath
Publico	String	backToUserHome	HttpSession sesión HttpServletRequest request String levelName int levelCleared int levelPlayerScore int levelRating
Publico	String	replayLevelDemolition	HttpSession session
Publico	String	replayLevelArkanoid	HttpSession session
Privado		manageScoreUpdate	User user String levelName int levelCleared int levelPlayerScore
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		LevelDataManagerService	Ldms
Privado		PlaylistDataManagerService	Pldms
Privado		RatinLevelDataManagerServices	rldms
Observaciones			

7.5.2.3.1 Descripción de los métodos

```
public String goToPlayLevelDemolition(HttpSession session, @RequestParam("levelName")
String levelName, @RequestParam("levelPath") String levelPath)
```

Método que se encarga de cargar la página del motor del juego Demolition con los datos del nivel seleccionado.

- HttpSession session: Objeto sesión que contiene todos los datos de la sesión del usuario.
- String levelName: Nombre del nivel seleccionado.
- String levelPath: Ruta donde se encuentra el fichero JSON del nivel.

```
public String goToPlayLevelArkanoid(HttpSession session, @RequestParam("levelName")
String levelName, @RequestParam("levelPath") String levelPath)
```

Método que se encarga de cargar la página del motor del juego Arkanoid con los datos del nivel seleccionado.

- HttpSession session: Objeto sesión que contiene todos los datos de la sesión del usuario.
- String levelName: Nombre del nivel seleccionado.
- String levelPath: Ruta donde se encuentra el fichero JSON del nivel.

```
public String backToUserHome(HttpSession session, HttpServletRequest request,
@RequestParam("levelName") String levelName, @RequestParam("levelCleared") int
levelCleared, @RequestParam("levelPlayerScore") int levelPlayerScore,
@RequestParam("levelRating") int levelRating)
```

Método que una vez ha terminado la partida en cualquiera de los juegos vuelve a cargar la página home del usuario con las actualizaciones de puntuaciones del nivel conseguidas.

- HttpSession session: Objeto sesión que contiene todos los datos de la sesión del usuario.
- HttpServletRequest request: Objeto request de la petición de la página web.
- String levelName: Nombre del nivel jugado.
- int levelCleared: Valor binario para marcar si se ha superado el nivel o no.
- int levelPlayerScore: Puntuación obtenida en el nivel jugado.
- int levelRating: Valoración otorgada al nivel.

```
public String replayLevelDemolition(HttpSession session)
```

Método que recarga la página con el nivel jugado del juego Demolition para volver a jugarlo.

- HttpSession session: Objeto sesión que contiene todos los datos de la sesión del usuario.

```
public String replayLevelArkanoid(HttpSession session)
```

Método que recarga la página con el nivel jugado del juego Arkanoid para volver a jugarlo.

- HttpSession session: Objeto sesión que contiene todos los datos de la sesión del usuario.

```
private void manageScoreUpdate(User user, String levelName, int levelCleared, int
levelPlayerScore)
```

Método que se encarga de actualizar las puntuaciones obtenidas por el jugador en el nivel jugado.

- User user: Objeto usuario que tiene los datos del usuario que jugó el nivel.
- String levelName: Nombre del nivel jugado.
- int levelCleared: Valor binario para marcar si se ha superado el nivel o no.
- int levelPlayerScore: Puntuación obtenida en el nivel jugado.

7.5.2.4 ComunityController

Nombre	Tipo	Descripción	
ComunityController	Publica	Controlador que se encarga de realizar todas las operaciones de comunicación con la página Community.	
Responsabilidades			
Número	**Descripción**		
1	Cargar la página Community.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Publico	String	addDemolitionLevelToUserHome	HttpSession session String levelName
Publico	String	addArkanoidLevelToUserHome	HttpSession session String levelName
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		LevelDataManagerService	Ldms
Privado		PlaylistDataManagerService	Pldms
Privado		RatinLevelDataManagerServices	rldms
Observaciones			

7.5.2.4.1 Descripción de los métodos

```
public String addDemolitionLevelToUserHome(HttpSession session,
@RequestParam("levelName") String levelName)
```

Método que añade un nivel del juego Demolition seleccionado a la playlist del usuario, una vez añadido este método carga la página home del usuario.

- HttpSession session:
- String levelName:

```
public String addArkanoidLevelToUserHome(HttpSession session, @RequestParam("levelName")
String levelName)
```

Método que añade un nivel del juego Arkanoid seleccionado a la playlist del usuario, una vez añadido este método carga la página home del usuario.

- HttpSession session:
- String levelName:

7.5.2.5 LevelDataManager

Nombre	Tipo	Descripción	
LevelDataManager	Publica	Clase que se encarga de realizar todas las operaciones referentes al manejo de los niveles de juego.	
Responsabilidades			
Número	**Descripción**		
1	Manejo de los objetos GameLevel.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Publico	DemolitionLevel	createDemolitionLevel	String levelName String levelDescription String levelElementsList int levelLimit int levelCanonPos int levelNumBullets String autor String levelType
Publico	ArkanoidLevel	createArkanoidLevel	String levelName String levelDescription String levelElementsList String autor String levelType
Publico		insertDemolitionLevel	DemolitionLevel dl
Publico		insertArkanoidLevel	ArkanoidLevel al
Publico	List<DemolitionLevel>	getAllDemolitionLevels	
Publico	List<ArkanoidLevel>	getAllArkanoidLevels	
Publico	Playlist	getAllLevelsByUser	String username
Publico	boolean	existLevel	String levelName
Publico		updateDemolitionLevel	DemolitionLevel dl
Publico		updateArkanoidLevel	ArkanoidLevel al
Privado	List<LevelElement>	makeLevelElementsList	String levelElementsListString
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		GameLevelsDAO	glDAO
Privado		DemolitionLevel	Dl
Privado		ArkanoidLevel	Al
Observaciones			

7.5.2.5.1 Descripción de los métodos

```
public DemolitionLevel createDemolitionLevel(String levelName, String levelDescription,
String levelElementsList, int levelLimit, int levelCanonPos, int levelNumBullets, String
author, String levelType)
```

Método que se encarga de crear un nivel para el juego Demolition.

- String levelName: Nombre que tendrá el nivel.
- String levelDescription: Descripción del nivel.
- String levelElementsList: Texto con las piezas que componen el nivel.
- **int** levelLimit: Posición de la línea límite de altura.
- **int** levelCanonPos: Posición del cañón.
- **int** levelNumBullets: Número de proyectiles que se podrán usar en el nivel.
- String autor: Nombre del usuario que ha creado el nivel.
- String levelType: Tipo de juego al que pertenece el nivel.

```
public ArkanoidLevel createArkanoidLevel(String levelName, String levelDescription,
String levelElementsList, String author, String levelType)
```

Método que se encarga de crear un nivel para el juego Arkanoid.

- String levelName: Nombre que tendrá el nivel.
- String levelDescription: Descripción del nivel.
- String levelElementsList: Texto con las piezas que componen el nivel.
- String autor: Nombre del usuario que ha creado el nivel.
- String levelType: Tipo de juego al que pertenece el nivel.

```
public void insertDemolitionLevel(DemolitionLevel dl)
```

Método que añade un nivel del juego Demolition en la base de datos.

- DemolitionLevel dl: Objeto que contiene los datos de un nivel para el juego Demolition.

```
public void insertArkanoidLevel(ArkanoidLevel al)
```

Método que añade un nivel del juego Arkanoid en la base de datos.

- ArkanoidLevel al: Objeto que contiene los datos de un nivel para el juego Arkanoid.

```
public List<DemolitionLevel> getAllDemolitionLevels()
```

Método que obtiene una lista de todos los niveles del tipo Demolition.

```
public List<ArkanoidLevel> getAllArkanoidLevels()
```

Método que obtiene una lista de todos los niveles del tipo Arkanoid.

```
public Playlist getAllLevelsByUser(String username)
```

Método que obtiene una playlist con todos los niveles del usuario introducido.

- String username: Nombre del usuario del que se quiere obtener la playlist.

```
public boolean existLevel(String levelName)
```

Método que comprueba si existe un nivel en la base de datos.

- String levelName: Nombre del nivel que se va a buscar.

```
public void updateDemolitionLevel(DemolitionLevel dl)
```

Método que actualiza el estado de un nivel del tipo Demolition.

- DemolitionLevel dl: Objeto que contiene los datos de un nivel para el juego Demolition.

```
public void updateArkanoidLevel(ArkanoidLevel al)
```

Método que actualiza el estado de un nivel del tipo Arkanoid.

- ArkanoidLevel al: Objeto que contiene los datos de un nivel para el juego Arkanoid.

```
private List<LevelElement> makeLevelElementsList(String levelElementsListString)
```

Método que crea una lista de objetos LevelELement a partir de una cadena de texto con las piezas que componen un nivel.

- String levelElementsListString: Cadena de texto con las piezas que componen el nivel.

7.5.2.6 GameLevelsDAO

Nombre	Tipo	Descripción	
GameLevelsDAO	Publica	Clase que se comunica con la base de datos para acceder a las tablas de los niveles de los juegos.	
Responsabilidades			
Número	**Descripción**		
1	Comunicarse con la base de datos.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
		insertDemolitionLevel	DemolitionLevel dl
		insertArkanoidLevel	ArkanoidLevel al
	List<DemolitionLevel>	getAllDemolitionLevels	
	List<ArkanoidLevel>	getAllArkanoidLevels	
	Playlist	getAllLevelsByUser	String username
	boolean	existLevel	String levelName
		updateLevel	String levelDescription String levelName
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		String	insertLevel
Privado		String	getAllDemolitionLevels
Privado		String	getAllArkanoidLevels
Privado		String	getAllLevelsByUser
Privado		String	existLevel
Privado		String	updateLevel
Observaciones			

7.5.2.6.1 Descripción de los métodos

```
public void insertDemolitionLevel(DemolitionLevel dl)
```

Método que añade un nivel del tipo Demolition en la base de datos.

- DemolitionLevel dl: Objeto Demolition Level que se va a añadir a la base de datos.

```
public void insertArkanoidLevel(ArkanoidLevel al)
```

Método que añade un nivel del tipo Arkanoid en la base de datos.

- ArkanoidLevel al: Objeto Arkanoid Level que se va a añadir a la base de datos.

```
public List<DemolitionLevel> getAllDemolitionLevels()
```

Método que obtiene una lista con todos los niveles del tipo Demolition.

```
public List<ArkanoidLevel> getAllArkanoidLevels()
```

Método que obtiene una lista con todos los niveles del tipo Arkanoid.

```
public Playlist getAllLevelsByUser(String username)
```

Método que obtiene un objeto playlist con todos los niveles pertenecientes a un usuario.

- String username: Nombre del usuario.

```
public boolean existLevel(String levelName)
```

Método que comprueba si existe un nivel dentro de la base de datos.

- String levelName: Nombre del nivel.

```
public void updateLevel(String levelDescription, String levelName)
```

Método que actualiza el estado de un nivel de la base de datos.

- String levelDescription: Descripción del nivel.
- String levelName: Nombre del nivel.

7.5.2.7 UsersDAO

Nombre	Tipo	Descripción	
UsersDAO	Publica	Clase que se comunica con la base de datos para acceder a las tablas de los usuarios.	
Responsabilidades			
Número	**Descripción**		
1	Comunicarse con la base de datos.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Publico		insertUser	User user
Publico	Boolean	existUser	String username
Publico	User	getUser	String userName String password
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		String	existUser
Privado		String	getUser
Privado		String	insertUser
Observaciones			

7.5.2.7.1 Descripción de los métodos

```
public void insertUser(User user)
```

Método que añade un usuario en la base de datos.

- User user: Objeto usuario que se va a introducir en la base de datos.

```
public boolean existUser(String username)
```

Método que comprueba si existe el usuario seleccionado en la base de datos.

- String username: Nombre del usuario.

```
public User getUser(String userName, String password)
```

Método que obtiene un usuario de la base de datos.

- String userName: Nombre del usuario.
- String password: Contraseña del usuario.

7.5.2.8 JSONFilesUtils

Nombre	Tipo	Descripción	
JSONFilesUtils	Publica	Clase de utilidades encargada del manejo de los ficheros JSON.	
Responsabilidades			
Número	**Descripción**		
1	Manejo de los ficheros JSON.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public static	Boolean	createDemolitionLevelJSON	DemolitionLevel dl
Public static	boolean	createArkanoidLevelJSON	ArkanoidLevel al
Public static	DemolitionLevel	readDemolitionLevelJSON	String levelPath
Public static	ArkanoidLevel	readArkanoidLevelJSON	String levelPath
Observaciones			

7.5.2.8.1 Descripción de los métodos

```
public static boolean createDemolitionLevelJSON(DemolitionLevel dl)
```

Método que crea un fichero JSON formateado con el estilo para un nivel del tipo Demolition.

- DemolitionLevel dl: Nivel que va a ser guardado en el fichero.

```
public static boolean createArkanoidLevelJSON(ArkanoidLevel al)
```

Método que crea un fichero JSON formateado con el estilo para un nivel del tipo Arkanoid.

- ArkanoidLevel al: Nivel que va a ser guardado en el fichero.

```
public static DemolitionLevel readDemolitionLevelJSON(String levelPath)
```

Método que lee un fichero JSON formateado con un nivel del tipo Demolition.

- String levelPath: Ruta donde se encuentra el fichero.

```
public static ArkanoidLevel readArkanoidLevelJSON(String levelPath)
```

Método que lee un fichero JSON formateado con un nivel del tipo Arkanoid.

- String levelPath: Ruta donde se encuentra el fichero.

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

Las pruebas realizadas para este proyecto se han realizado todas sobre la misma máquina utilizada para el desarrollo. Solo se requiere una máquina para realizar las pruebas pese a ser una aplicación web, ya que debido a que está desarrollada con el framework MVC de Spring y al ser un aplicación Java, todas las pruebas se ejecutan desde el entorno de desarrollo Eclipse.

El entorno de desarrollo utilizado es Eclipse Luna y corre en una maquina con un sistema operativo Windows 8.1. También ha sido necesaria una base de datos, que se ha creado específicamente para la realización de dichas pruebas y el SGDB usado ha sido MySQL Workbench 6.2.

8.1.1 Resultados

Todas las pruebas descritas en el apartado **6.7.1** de esta documentación han dado un resultado positivo, ninguna ha presentado problemas a la hora de pasar las pruebas.

8.2 Pruebas de Integración y del Sistema

Caso de uso: Crear usuario	
Prueba	**Resultado Esperado**
Crear un usuario que aún no existe.	El sistema tiene que tener un usuario más en la base de datos.
	Resultado Obtenido
	Efectivamente, el sistema tiene un usuario más en la base de datos.
Prueba	**Resultado Esperado**
Crear un usuario que ya existe.	El sistema debería permanecer con el mismo número de usuario en la base de datos.
	Resultado Obtenido
	Efectivamente, el sistema se mantiene con el mismo número de usuarios en la base de datos.
Prueba	**Resultado Esperado**
Cancelar operación.	El sistema se mantiene inalterado.
	Resultado Obtenido
	El sistema se mantiene inalterado.

Caso de uso: Añadir nivel a la playlist del usuario	
Prueba	**Resultado Esperado**
Añadir un nivel que el usuario no tiene en su playlist.	El sistema tiene un nivel más asignado a la playlist del usuario dentro de la base de datos.
	Resultado Obtenido
	Efectivamente, el sistema tiene un nivel más asignado a la playlist del usuario dentro de la base de datos.
Prueba	**Resultado Esperado**
Añadir un nivel que el usuario ya tiene en su playlist.	El sistema no debería tener un nivel más asignado a la playlist del usuario dentro de la base de datos.
	Resultado Obtenido
	Efectivamente, el sistema no posee ningún nivel más asignado a la playlist del usuario en la base de datos.

Caso de uso: Arrastrar pieza de la paleta a la cuadricula	
Prueba	**Resultado Esperado**
Arrastrar pieza de la paleta a la cuadricula.	El mapa actual debería contener una pieza más.
	Resultado Obtenido
	El mapa actual tiene una pieza más.
Prueba	**Resultado Esperado**
Arrastrar pieza de la paleta fuera de la cuadricula.	El mapa actual se mantiene con las piezas que tenía.
	Resultado Obtenido
	Efectivamente, el mapa actual se mantiene con el mismo número de piezas que tenía.

Caso de uso: Eliminar pieza	
Prueba	**Resultado Esperado**
Arrastrar la pieza a la papelera.	El mapa actual debería tener una pieza menos.
	Resultado Obtenido
	Efectivamente, el mapa actual tiene una pieza menos.

Caso de uso: Guardar nivel	
Prueba	**Resultado Esperado**
Guardar nivel que no exista.	El sistema tiene que tener un nivel más guardado en la base de datos.
	Resultado Obtenido
	Efectivamente, el sistema tiene un nivel más guardado en la base de datos.
Prueba	**Resultado Esperado**
Guardar nivel que ya exista.	El sistema debería mantenerse con el mismo número de niveles guardados en la base de datos.
	Resultado Obtenido
	Efectivamente, el sistema se mantiene con el mismo número de niveles guardados en la base de datos.

8.3 Pruebas de Usabilidad y Accesibilidad

8.3.1 Pruebas de Usabilidad

Las pruebas que se van a implementar en esta sección consisten en usar serie de cuestionarios de usabilidad hechos a varios perfiles de usuarios. Estos cuestionarios recogen una serie de temas relacionados no solo con el manejo de la aplicación sino con las características y conocimientos de los usuarios que van a ser clientes potenciales de dicha plataforma.

Se ha consultado a un grupo de 5 usuarios de distintos perfiles. Los perfiles varían desde personas que conocen el mundo del videojuego y el mundo del software, usuarios que están habituados a navegar por internet y que conocen el funcionamiento de las páginas web y personas que no poseen muchos conocimientos informáticos ni de videojuegos.

Los distintos perfiles y su distribución dentro del grupo de personas consultadas son:

- **Grupo 1:** 2 Personas conocedoras del mundo del videojuego y de Internet.
- **Grupo 2:** 2 Personas que habitualmente navegan por internet y conocen el funcionamiento de las páginas web.
- **Grupo 3:** 1 Persona que ocasionalmente usan ordenadores o juegan a videojuegos.

A continuación se detallan los resultados obtenido de pasar los distintos formularios diseñados para estas pruebas.

8.3.1.1 Actividades de las Pruebas de Usabilidad: Grupo 1

8.3.1.1.1 Preguntas de carácter general

¿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
¿Usa internet frecuentemente?
1. **Todos los días** 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza en internet?
1. **Es necesario para el desempeño de mi trabajo o profesión** 2. Lo uso para ocio principalmente 3. Lo uso para ver las noticias. 4. Únicamente para acceder a redes sociales y consulta el correo.
¿Ha usado alguna vez alguna aplicación como la de esta prueba?
1. Sí, he empleado software similar 2. **No, aunque si empleo otros programas que me ayudan a realizar tareas similares** 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**

Este grupo es gente trabaja en el ámbito de la informática y es aficionada a los videojuegos, en alguna ocasión han trabajado o utilizado herramientas para crear videojuegos por hobby, por trabajo o simplemente por curiosidad.

8.3.1.1.2 Actividades guiadas

Los usuarios de este grupo no tienen ningún problema en realizar estas tareas ya que al estar familiarizados tanto con el manejo de páginas web y con todos los elementos que componen un videojuego, han realizado las tareas con bastante facilidad sin apenas cometer errores.

8.3.1.1.3 Preguntas Cortas sobre la Aplicación

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
¿La funcionalidad se ajusta a lo esperado?	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		

Las respuestas de los usuarios de este grupo reflejan que la aplicación cumple perfectamente con lo esperado para una herramienta de estas características.

8.3.1.1.4 Observaciones

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	El usuario se adapta con rapidez a la aplicación.
Tiempo en realizar cada tarea	Los usuarios no tardan demasiado en realizar las tareas.
Errores leves cometidos	No se han cometido errores.
Errores graves cometidos	No se han cometido errores.
Número de operaciones repetidas por el usuario.	Ninguna.

8.3.1.2 Actividades de las Pruebas de Usabilidad: Grupo 2

8.3.1.2.1 Preguntas de carácter general

¿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
¿Usa internet frecuentemente?
1. **Todos los días** 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza en internet?
1. Es necesario para el desempeño de mi trabajo o profesión 2. **Lo uso para ocio principalmente** 3. **Lo uso para ver las noticias.** 4. **Únicamente para acceder a redes sociales y consulta el correo.**
¿Ha usado alguna vez alguna aplicación como la de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 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

Este grupo de usuarios está habituado a usar ordenadores y también son usuarios frecuentes de internet. También juegan a videojuegos aunque sea de forma ocasional, especialmente con dispositivos móviles. Este grupo de personas nunca ha manejado software de edición de videojuegos.

8.3.1.2.2 Actividades guiadas

Este grupo de personas han desempeñado estas pruebas con relativa facilidad ya que están familiarizados con el manejo de páginas web. Han tenido alguna dificultad al utilizar la herramienta de edición al notar poco intuitivo algún punto de la interfaz, como la paleta de elementos, que no mostraba ningún tipo de instrucciones de cómo actuar con dicho elemento. Tras varios intentos y repeticiones han sabido adaptarse al manejo de la aplicación y han conseguido terminar las pruebas con éxito.

8.3.1.2.3 Preguntas Cortas sobre la Aplicación

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
¿La funcionalidad se ajusta a lo esperado?	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		

El grupo de usuarios ha detectado, como se comentaba anteriormente, ciertas dificultades con algunos elementos de la pantalla de edición de niveles, por lo que se ha optado por añadir indicaciones que faciliten el manejo de este elemento.

8.3.1.2.4 Observaciones

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	Tras varios intentos el usuario termina adaptándose al manejo de la aplicación.
Tiempo en realizar cada tarea	Los usuarios han tardado un poco en adaptarse a algún tipo de mecánica de funcionamiento de la aplicación.
Errores leves cometidos	Manejo de la paleta de elementos de creación.
Errores graves cometidos	Apenas se han cometido errores.
Número de operaciones repetidas por el usuario.	Se han repetido varias veces algunas operaciones de manejo de la paleta de elementos de la herramienta de edición.

8.3.1.3 Actividades de las Pruebas de Usabilidad: Grupo 3

8.3.1.3.1 Preguntas de carácter general

¿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**

¿Usa internet frecuentemente?

1. Todos los días
2. Varias veces a la semana
3. Ocasionalmente
4. **Nunca o casi nunca**

¿Qué tipo de actividades realiza en internet?

1. Es necesario para el desempeño de mi trabajo o profesión
2. Lo uso para ocio principalmente
3. **Lo uso para ver las noticias.**
4. **Únicamente para acceder a redes sociales y consulta el correo.**

¿Ha usado alguna vez alguna aplicación como la de esta prueba?

1. Sí, he empleado software similar
2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares
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

Este grupo de personas conocen muy poco del mundo de la informática y aunque utilizan internet no dominan el manejo de muchas aplicaciones online. Han jugado a videojuegos pero de forma muy anecdótica ya que es un campo que no les atrae. Nunca han utilizado aplicaciones de edición de videojuegos.

8.3.1.3.2 Actividades guiadas

Este grupo fue el que más complicaciones tuvo para manejar la aplicación ya que no estaban familiarizados con el funcionamiento de aplicaciones de este tipo. Han tenido problemas similares a los encontrados en el grupo dos. Tras varios intentos al final han conseguido realizar las tareas guiadas.

8.3.1.3.3 Preguntas cortas sobre la aplicación

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**		
¿La funcionalidad se ajusta a lo esperado?		**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**

Como se comentaba en las secciones anteriores, los usuarios de este grupo han tenido dificultades para utilizar la aplicación dado que no son usuarios habituales de este tipo de aplicaciones, para solucionar este tema se ha incluido una ayuda más detallada de los distintos pasos a seguir para manejar a la perfección la herramienta.

8.3.1.3.4 Observaciones

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	El usuario tarda un tiempo en aprender a utilizar ciertas funciones de la aplicación.
Tiempo en realizar cada tarea	El usuario tarda bastante tiempo en realizar cada tarea.
Errores leves cometidos	El usuario crea niveles sin ningún elemento.
Errores graves cometidos	Carga de ficheros con formatos erróneos.
Número de operaciones repetidas por el usuario.	El número de repeticiones de las tareas es bastante elevado.

8.3.1.4 Cambios realizados en el proyecto tras evaluar los formularios de usabilidad.

Se ha modificado la herramienta de edición de contenido ligeramente para mostrar más información al usuario cuando interactúa con las piezas de la paleta. Se ha modificado el puntero del ratón para que cuando se pase por encima de una de las piezas de la paleta o de las que están colocadas en la cuadricula, se muestre una mano para identificar que esa pieza se puede arrastrar.

8.3.1.5 Guía de usabilidad

Se incluye también una guía de usabilidad con los resultados obtenidos de los usuarios consultados en los test anteriores. En esta guía se muestra el nivel de usabilidad notado por los usuarios que han probado la aplicación.

Criterios	¿Cumplido?
Generales	
¿Cuáles son los objetivos del sitio web? ¿Son concretos y bien definidos? ¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?	**SI**
¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?	**No**
¿Muestra de forma precisa y completa qué contenidos o servicios ofrece realmente el sitio web? El diseño de la página de inicio debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio.	**No**
¿La estructura general del sitio web está orientada al usuario? Los sitios web deben estructurarse pensando en el usuario, sus objetivos y necesidades. La estructura interna de la empresa u organización, cómo funciona o se organiza no interesan al usuario.	**Si**
¿El *look & feel* general se corresponde con los objetivos, características, contenidos y servicios del sitio web? Ciertas combinaciones de colores ofrecen imágenes más o menos formales, serias o profesionales.	**Si**
¿Es coherente el diseño general del sitio web? Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación.	**Si**
¿Es reconocible el diseño general del sitio web? Cuánto más se parezca el sitio web al resto de sitios web, más fácil será de usar.	**Si**
¿El sitio web se actualiza periódicamente? ¿Indica cuándo se actualiza? Las fechas que se muestren en la página deben corresponderse con actualizaciones, noticias, eventos...no con la fecha del sistema del usuario.	**-**
Identidad e Información	
¿Se muestra claramente la identidad de la empresa-sitio a través de todas las páginas?	**Si**
El Logotipo, **¿es significativo, identificable y suficientemente visible?**	**Si**
El eslogan o *tagline*, **¿expresa realmente qué es la empresa y qué servicios ofrece?**	**-**
¿Se ofrece algún enlace con información sobre la empresa, sitio web, 'webmaster',...?	**Si**
¿Se proporciona mecanismos para ponerse en contacto con la empresa? (email, teléfono, dirección postal, fax...)	**Si**
¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?	**-**
En artículos, noticias, informes... **¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?**	**-**

Lenguaje y Redacción	
¿El sitio web habla el mismo lenguaje que sus usuarios? Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario.	**Si**
¿Emplea un lenguaje claro y conciso?	**Si**
¿Es amigable, familiar y cercano? Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario.	**Si**
¿1 párrafo = 1 idea? Cada párrafo es un objeto informativo. Trasmita ideas, mensajes...Se deben evitar párrafos vacíos o varios mensajes en un mismo párrafo.	**Si**
Rotulado	
Los rótulos, **¿son significativos?** Ejemplo: evitar rótulos del tipo "haga clic aquí".	**Si**
¿Usa rótulos estándar? Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de...".	**Si**
¿Usa un único sistema de organización, bien definido y claro? No se deben mezclar diferentes. Los sistemas de organización son: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas.	**-**
¿Utiliza un sistema de rotulado controlado y preciso? Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de"	**Si**
El título de las páginas, **¿Es correcto? ¿Ha sido planificado?** Relacionado con la capacidad para poder buscar y encontrar el sitio *web*.	**Si**
Estructura y Navegación	
La estructura de organización y navegación, **¿Es la más adecuada?** Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada,...	**Si**
En el caso de estructura jerárquica, **¿Mantiene un equilibrio entre Profundidad y Anchura?**	**Si**
En el caso de ser puramente hipertextual, **¿Están todos los clúster de nodos comunicados?** Aquí se mide la distancia entre nodos.	**Si**
¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos,...)? Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado, y ser reconocidos como una unidad	**No**
En menús de navegación, **¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística?** No se deben superar los 7±2 elementos, ni los 2 o, como mucho, 3 términos por elemento.	**Si**
¿Es predecible la respuesta del sistema antes de hacer clic sobre el enlace? Relacionado con el nivel de significación del rótulo del enlace, aunque también con: el uso de globos de texto, información contextual, la barra de estado del navegador,...	**Si**

¿Se ha controlado que no haya enlaces que no llevan a ningún sitio? Enlaces que no llevan a ningún sitio: Los enlaces rotos, y los que enlazan con la misma página que se está visualizando (por ejemplo enlaces a la "home" desde la misma página de inicio)	**Si**
¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación? ...como *breadcrumbs*, enlaces a la página de inicio,...recuerde que el logo también es recomendable que enlace con la página de inicio.	**No**
Las imágenes enlace, **¿se reconocen como clicables? ¿Incluyen un atributo 'title' describiendo la página de destino?** En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean.	**Si**
¿Se ha evitado la redundancia de enlaces?	**Si**
¿Se ha controlado que no haya páginas "huérfanas"? Páginas huérfanas: que aún siendo enlazadas desde otras páginas, éstas no enlacen con ninguna.	**Si**
Layout de la Página	
¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia? (como por ejemplo la zona central)	**Si**
¿Se ha evitado la sobrecarga informativa? Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Los grupos diferentes de objetos informativos de una página deben ser 7±2.	**Si**
¿Es una interfaz limpia, sin ruido visual?	**Si**
¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?	**Si**
¿Se hace un uso correcto del espacio visual de la página? Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia.	**Si**
¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página? (La jerarquía visual se utiliza para orientar al usuario)	**Si**
¿Se ha controlado la longitud de página? Se debe evitar en la medida de lo posible el *scrolling*. Si la página es muy extensa, se debe fraccionar.	**Si**
Búsqueda (si es necesario, por la extensión del sitio, incorporar un buscador interno)	
¿Se encuentra fácilmente accesible? Es decir: directamente desde la home, y a ser posible desde todas las páginas del sitio, y colocado en la zona superior de la página.	**-**
¿Es fácilmente reconocible como tal?	**-**
¿Permite la búsqueda avanzada? (siempre y cuando, por las características del sitio web, fuera de utilidad que la ofreciera)	**-**
¿Muestra los resultados de la búsqueda de forma comprensible para el usuario?	**-**
¿La caja de texto es lo suficientemente ancha?	**-**
¿Asiste al usuario en caso de no poder ofrecer resultados para una consultada dada?	**-**

Elementos Multimedia	
¿Las fotografías están bien recortadas? ¿Son comprensibles? ¿Se ha cuidado su resolución?	**Si**
¿Las metáforas visuales son reconocibles y comprensibles por cualquier usuario? (prestar especial atención a usuarios de otros países y culturas)	**Si**
¿El uso de imágenes o animaciones proporciona algún tipo de valor añadido?	**Si**
¿Se ha evitado el uso de animaciones cíclicas?	**Si**
Ayuda	
Si posee una sección de Ayuda, **¿Es verdaderamente necesaria?** Siempre que se pueda prescindir de ella simplificando los elementos de navegación e interacción, debe omitirse esta sección.	**No**
En enlace a la sección de Ayuda, **¿Está colocado en una zona visible y "estándar"?** La zona de la página más normal para incluir el enlace a la sección de Ayuda, es la superior derecha.	**-**
¿Se ofrece ayuda contextual en tareas complejas? (transferencias bancarias, formularios de registro...)	**-**
Si posee *FAQs*, **¿Es correcta tanto la elección como la redacción de las preguntas? ¿Y las respuestas?**	**-**
Accesibilidad (debería cubrirse con los test de Accesibilidad posteriores)	
¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?	**Si**
¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleadas facilitan la lectura?	**Si**
¿Existe un alto contraste entre el color de fuente y el fondo?	**Si**
¿Incluyen las imágenes atributos 'alt' que describan su contenido?	**Si**
¿Es compatible el sitio web con los diferentes navegadores? ¿Se visualiza correctamente con diferentes resoluciones de pantalla? Se debe prestar atención a: *JScript*, *CSS*, tablas, fuentes...	**Si**
¿Puede el usuario disfrutar de todos los contenidos del sitio web sin necesidad de tener que descargar e instalar *plugins* adicionales?	**Si**
¿Se ha controlado el peso de la página? Se deben optimizar las imágenes, controlar el tamaño del código *JScript*...	**Si**
¿Se puede imprimir la página sin problemas? Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible.	**No**
Control y Retroalimentación	
¿Tiene el usuario todo el control sobre el interfaz? Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos...	**Si**
¿Se informa constantemente al usuario acerca de lo que está pasando? Si el usuario tiene que esperar hasta que se termine una operación, se debe mostrar un mensaje indicándoselo y que debe esperar, con el tiempo de espera estimado o una barra de progreso.	**Si**

¿Se informa al usuario de lo que ha pasado? Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.	**Si**
Cuando se produce un error, **¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?** Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.	**Si**
¿Posee el usuario libertad para actuar? NO restringir la libertad del usuario: Uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación, no impida al usuario poder usar el botón derecho de su ratón...	**Si**
¿Se ha controlado el tiempo de respuesta? Esto tiene que ver con el peso de cada página (accesibilidad) y tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder. El tiempo máximo que esperará un usuario son 10 segundos	**Si**
Aclaraciones	
¿Se ha evaluado adecuadamente la orientación del usuario? (Donde estoy, como volver, que he visitado, que va a pasar)[1]	**Si**
¿Se ha usado correctamente la publicidad?[2]	**-**

1. En este caso la página no presenta página de inicio como tal, porque la página principal es la página de Login.
2. Los enlaces son fácilmente reconocibles, pero por las características de la página, no muestran ningún cambio de color o señal cuando se han visitado, ya que tal y como está construida la página web, no tiene sentido este matiz.
3. El usuario puede disfrutar de todos los contenidos de la página web, siempre y cuando se mantenga activado JavaScript, ya que al tratarse de una aplicación web este elemento es imprescindible.
4. La página web no dispone de versión imprimible, ya que al tratarse de una aplicación web, esto no tendría sentido.

[1] La orientación del usuario se puede evaluar a través de varios criterios: elementos de navegación orientativos, caracterización de los enlaces e información contextual en elementos de interacción (estructura y navegación); distribución visual de la página (*layout*); coherencia del diseño (generales); nivel de significación de los rótulos (rotulado) y retroalimentación del usuarios (control y retroalimentación)

[2] Respecto a la publicidad (normalmente en forma de banners), se puede evaluar desde varios criterios: lenguaje (lenguaje y redacción), nivel de significación de los rótulos (rotulado), jerarquía informativa y ruido visual (*layout* de la página), pop-ups y banners intrusivos (control y retroalimentación)...

8.3.2 Pruebas de Accesibilidad

El nivel de conformidad que se exige tener en la web presentada en este proyecto es como mínimo un AA bajo el marco de las normas de la WCAG 2.0, para ello se realiza una evaluación manual con una checklist de todos los puntos de las normas mencionadas anteriormente.

También se realizara un test de accesibilidad automático a través de herramientas como "TAW" o "AChecker" para obtener una segunda lectura.

8.3.2.1 Revisión Preliminar

En esta revisión preliminar se va a realizar una evaluación manual de la página web mediante un checklist con las normas WCAG 2.0, se estudiaran los resultados y se realizaran los cambios necesarios para realizar un chequeo con alguna herramienta automatizada con el fin de adquirir un grado de conformidad AA en las normas WCAG 2.0.

Las paginas elegidas para realizar el test son: Login, User Home y Level Editor.

8.3.2.2 Página Login

8.3.2.2.1 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)	No hay elementos de audio o video en la página.			x
1.2.2	Subtítulos (grabados)	No hay elementos de audio o video en la página.			x
1.2.3	Audio-descripción o medio alternativo (grabado)	No hay elementos de audio o video en la página.			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	No hay elementos de audio en la página.			x
2.1.1	Teclado		x		
2.1.2	Sin trampas para el foco del teclado		x		
2.2.1	Tiempo ajustable	No hay operaciones en la página que requieran tiempo.			x
2.2.2	Poner en pausa, detener, ocultar	No hay elementos que se desplacen, parpadeen o similares.			x
2.3.1	Umbral de tres destellos o menos		x		
2.4.1	Evitar bloques	No hay bloques repetidos.	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		
3.3.1	Identificación de errores		x		
3.3.2	Etiquetas o instrucciones		x		
4.1.1	Procesamiento		x		
4.1.2	Nombre, función, valor				x

8.3.2.2.2 Nivel AA

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
1.2.4	Sub títulos (en directo)	No hay elementos de audio o video en la página.			x
1.2.5	Audio-descripción (grabado)	No hay elementos de audio o video en la página.			x
1.4.3	Contraste (mínimo)		x		
1.4.4	Cambio de tamaño del texto		x		
1.4.5	Imágenes de texto	No tiene imágenes de texto.	x		
2.4.5	Múltiples vías	Esta es la página inicial.			x
2.4.6	Encabezados y etiquetas		x		
2.4.7	Foco visible		x		
3.1.2	Idioma de las partes	La página no contiene fragmentos en otro idioma.			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

8.3.2.2.3 Nivel AAA

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
1.2.6	Lengua de señas (grabado)	No hay elementos de audio o video en la página.			x
1.2.7	Audio-descripción ampliada (grabada)	No hay elementos de audio o video en la página.			x
1.2.8	Medio alternativo (grabado)	No hay elementos de audio o video en la página.			x
1.2.9	Solo audio (en directo)	No hay elementos de audio en la página.			x
1.4.6	Contraste (mejorado)		x		
1.4.7	Sonido de fondo bajo o ausente	No hay audio de fondo.	x		
1.4.8	Presentación visual			x	
1.4.9	Imágenes de texto (sin excepciones)	No hay imágenes de texto.	x		

2.1.3	Teclado (sin excepciones)		x		
2.2.3	Sin tiempo	No hay acciones que requieran tiempo.	x		
2.2.4	Interrupciones	No hay interrupciones en la página.	x		
2.2.5	Re-autentificación	La sesión no expira			x
2.3.2	Tres destellos		x		
2.4.8	Ubicación				x
2.4.9	Propósito de los enlaces (sólo enlaces)		x		
2.4.10	Encabezados de sección		x		
3.1.3	Palabras inusuales	No hay palabras inusuales.	x		
3.1.4	Abreviaturas		x		
3.1.5	Nivel de lectura		x		
3.1.6	Pronunciación	No hay palabras que requieran pronunciación específica.			x
3.2.5	Cambios a petición				x
3.3.5	Ayuda			x	
3.3.6	Prevención de errores (todos)				x

8.3.2.3 Página User Home

8.3.2.3.1 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)	No hay elementos de audio o video en la página.			x
1.2.2	Subtítulos (grabados)	No hay elementos de audio o video en la página.			x
1.2.3	Audio-descripción o medio alternativo (grabado)	No hay elementos de audio o video en la página.			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	No hay elementos de audio en la página.			x
2.1.1	Teclado		x		
2.1.2	Sin trampas para el foco del teclado		x		
2.2.1	Tiempo ajustable	No hay operaciones en la página que requieran tiempo.			x
2.2.2	Poner en pausa, detener, ocultar	No hay elementos que se desplacen, parpadeen o similares.			x
2.3.1	Umbral de tres destellos o menos		x		
2.4.1	Evitar bloques	No hay bloques repetidos.	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		
3.3.1	Identificación de errores	Esta página no presenta errores.			x
3.3.2	Etiquetas o instrucciones				x
4.1.1	Procesamiento		x		
4.1.2	Nombre, función, valor				x

8.3.2.3.2 Nivel AA

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
1.2.4	Sub títulos (en directo)	No hay elementos de audio o video en la página.			x
1.2.5	Audio-descripción (grabado)	No hay elementos de audio o video en la página.			x
1.4.3	Contraste (mínimo)		x		
1.4.4	Cambio de tamaño del texto			x	
1.4.5	Imágenes de texto	No hay imágenes de texto.	x		
2.4.5	Múltiples vías		x		
2.4.6	Encabezados y etiquetas		x		
2.4.7	Foco visible	Hay elementos en los que no se distingue el foco.		x	
3.1.2	Idioma de las partes	La página no contiene fragmentos en otro idioma.			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

8.3.2.3.3 Nivel AAA

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
1.2.6	Lengua de señas (grabado)	No hay elementos de audio o video en la página.			x
1.2.7	Audio-descripción ampliada (grabada)	No hay elementos de audio o video en la página.			x
1.2.8	Medio alternativo (grabado)	No hay elementos de audio o video en la página.			x
1.2.9	Solo audio (en directo)	No hay elementos de audio o video en la página.			x
1.4.6	Contraste (mejorado)		x		
1.4.7	Sonido de fondo bajo o ausente	No hay audio de fondo.	x		
1.4.8	Presentación visual			x	

1.4.9	Imágenes de texto (sin excepciones)	No hay imágenes de texto.	x		
2.1.3	Teclado (sin excepciones)		x		
2.2.3	Sin tiempo	No hay acciones que requieran tiempo.	x		
2.2.4	Interrupciones	No hay interrupciones en la página.	x		
2.2.5	Re-autentificación	La sesión no expira			x
2.3.2	Tres destellos		x		
2.4.8	Ubicación			x	
2.4.9	Propósito de los enlaces (sólo enlaces)		x		
2.4.10	Encabezados de sección		x		
3.1.3	Palabras inusuales	No hay palabras inusuales.	x		
3.1.4	Abreviaturas		x		
3.1.5	Nivel de lectura		x		
3.1.6	Pronunciación	No hay palabras que requieran pronunciación específica.			x
3.2.5	Cambios a petición				x
3.3.5	Ayuda			x	
3.3.6	Prevención de errores (todos)				x

8.3.2.4 Página Level Editor Demolition

Se ha elegido esta página porque es igual al Level Editor Arkanoid y además contiene más elementos.

8.3.2.4.1 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)	No hay elementos de audio o video en la página.			x
1.2.2	Subtítulos (grabados)	No hay elementos de audio o video en la página.			x
1.2.3	Audio-descripción o medio alternativo (grabado)	No hay elementos de audio o video en la página.			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	No hay elementos de audio en la página.			x
2.1.1	Teclado	Hay funciones que requieren del uso del ratón.		x	
2.1.2	Sin trampas para el foco del teclado	Las operaciones que se permiten con teclado si cumplen este punto.	x		

2.2.1	Tiempo ajustable	No hay operaciones en la página que requieran tiempo.			x
2.2.2	Poner en pausa, detener, ocultar	No hay elementos que se desplacen, parpadeen o similares.			x
2.3.1	Umbral de tres destellos o menos		x		
2.4.1	Evitar bloques	No hay bloques repetidos.	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		
3.3.1	Identificación de errores			x	
3.3.2	Etiquetas o instrucciones			x	
4.1.1	Procesamiento		x		
4.1.2	Nombre, función, valor				x

8.3.2.4.2 Nivel AA

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
1.2.4	Sub títulos (en directo)	No hay elementos de audio o video en la página.			x
1.2.5	Audio-descripción (grabado)	No hay elementos de audio o video en la página.			x
1.4.3	Contraste (mínimo)			x	
1.4.4	Cambio de tamaño del texto	Cuando cambia el tamaño del texto, cambia el tamaño de los objetos, lo que provoca que la cuadricula se desajuste y pierda funcionalidad.		x	
1.4.5	Imágenes de texto	No hay imágenes de texto.	x		
2.4.5	Múltiples vías		x		
2.4.6	Encabezados y etiquetas		x		
2.4.7	Foco visible	Hay elementos en los que no se distingue el foco.		x	
3.1.2	Idioma de las partes	La página no contiene fragmentos en otro idioma.			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

8.3.2.4.3 Nivel AAA

WCAG 2.0	Titulo	Comentarios	Si	No	N/A
1.2.6	Lengua de señas (grabado)	No hay elementos de audio o video en la página.			x
1.2.7	Audio-descripción ampliada (grabada)	No hay elementos de audio o video en la página.			x
1.2.8	Medio alternativo (grabado)	No hay elementos de audio o video en la página.			x
1.2.9	Solo audio (en directo)	No hay elementos de audio o video en la página.			x
1.4.6	Contraste (mejorado)			x	
1.4.7	Sonido de fondo bajo o ausente	No hay audio de fondo.	x		
1.4.8	Presentación visual			x	
1.4.9	Imágenes de texto (sin excepciones)	No hay imágenes de texto.	x		
2.1.3	Teclado (sin excepciones)			x	
2.2.3	Sin tiempo	No hay acciones que requieran tiempo.	x		
2.2.4	Interrupciones	No hay interrupciones en la página.	x		
2.2.5	Re-autentificación	La sesión no expira			x
2.3.2	Tres destellos		x		
2.4.8	Ubicación			x	
2.4.9	Propósito de los enlaces (sólo enlaces)		x		
2.4.10	Encabezados de sección		x		
3.1.3	Palabras inusuales	No hay palabras inusuales.	x		
3.1.4	Abreviaturas		x		
3.1.5	Nivel de lectura		x		
3.1.6	Pronunciación	No hay palabras que requieran pronunciación específica.			x
3.2.5	Cambios a petición				x
3.3.5	Ayuda			x	
3.3.6	Prevención de errores (todos)				x

Tras el chequeo manual de la página podemos comprobar que no se cumple del todo el grado de conformidad AA requerido pero por muy pocos puntos. A continuación se hará una evaluación más exhaustiva de dichas páginas mediante herramientas automáticas y se procederá a realizar los cambios necesarios en la web a partir de los resultados obtenidos.

8.3.2.5 Evaluación de Conformidad

Tras la revisión preliminar se van a utilizar herramientas automáticas como "TAW" y "AChecker" para ver el grado final de conformidad de la web.

- **Página Login**

Tras pasar el chequeo por las herramientas automáticas se encontraron los mismos errores en ambas. Estos errores se referían a temas de etiquetado en los distintos Inputs, ya que no había elementos "label" que mostrasen información acerca del Input asociado.

Otro error detectado era con respecto al idioma de la página, no se había establecido el idioma al principio del documento.

- **Página User Home**

Esta página al ser analizada por las herramientas solo mostró errores con respecto al idioma, como ocurría en la página anterior.

- **Página Level Editor Demolition**

Tras el chequeo se han dado los mismos errores que en los dos análisis anteriores, todos los errores detectados hacen referencia al etiquetado de los Inputs y al idioma de la página.

8.3.2.6 Conclusiones

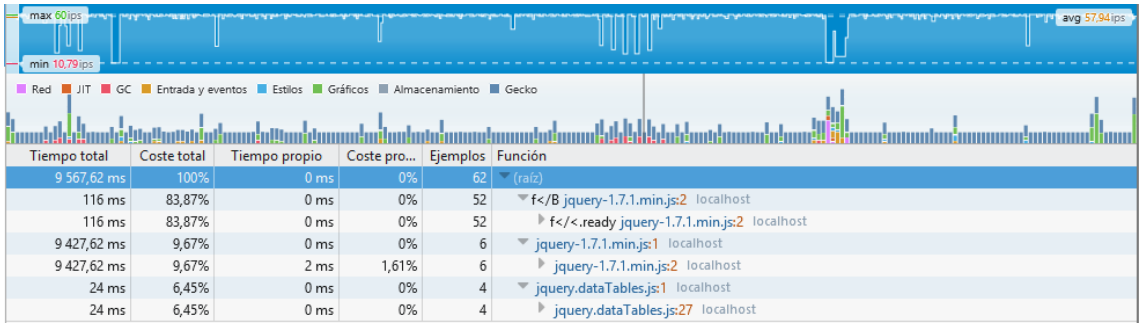
Todos los errores detectados en la página web son referentes al etiquetado y al idioma, por lo que son fácilmente subsanables. Tan solo con añadir a las etiquetas necesarias se arreglarían todos estos errores y la página web pasaría a cumplir el grado de conformidad AA.

8.4 Pruebas de Rendimiento

Las pruebas realizadas son toma de tiempos para ver la respuesta de la aplicación en las partes consideradas más importantes. Estas pruebas de tiempo se realizaran con el navegador Firefox, concretamente con las herramientas de rendimiento que este ofrece.

A continuación se listan las operaciones seleccionadas con sus respectivos resultados:

8.4.1 Inicio de sesión

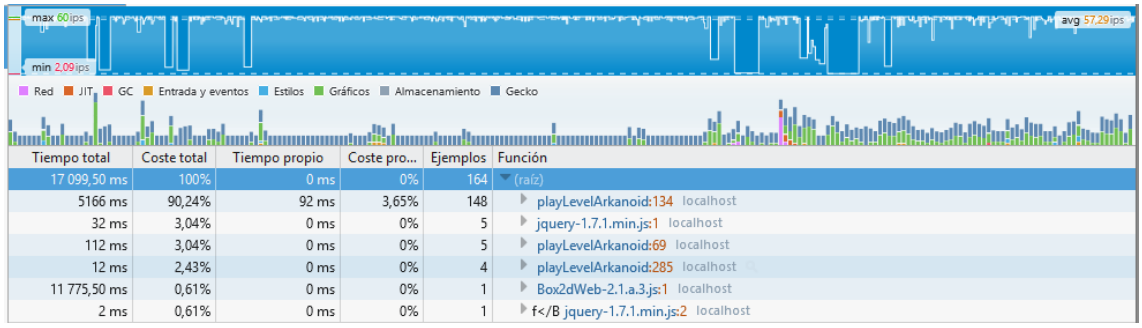


Tiempo total	Coste total	Tiempo propio	Coste pro...	Ejemplos	Función
9 567,62 ms	100%	0 ms	0%	62	(raíz)
116 ms	83,87%	0 ms	0%	52	f</B jquery-1.7.1.min.js:2 localhost
116 ms	83,87%	0 ms	0%	52	f</<.ready jquery-1.7.1.min.js:2 localhost
9 427,62 ms	9,67%	0 ms	0%	6	jquery-1.7.1.min.js:1 localhost
9 427,62 ms	9,67%	2 ms	1,61%	6	jquery-1.7.1.min.js:2 localhost
24 ms	6,45%	0 ms	0%	4	jquery.dataTables.js:1 localhost
24 ms	6,45%	0 ms	0%	4	jquery.dataTables.js:27 localhost

Figura 106 Resultado de las pruebas de rendimiento: Inicio de sesión

El resultado de esta prueba es muy bueno, ya que la operación apenas tarde 9ms, un tiempo aceptable para esta operación.

8.4.2 Cargar la pantalla de juego de un nivel



Tiempo total	Coste total	Tiempo propio	Coste pro...	Ejemplos	Función
17 099,50 ms	100%	0 ms	0%	164	(raíz)
5166 ms	90,24%	92 ms	3,65%	148	playLevelArkanoid:134 localhost
32 ms	3,04%	0 ms	0%	5	jquery-1.7.1.min.js:1 localhost
112 ms	3,04%	0 ms	0%	5	playLevelArkanoid:69 localhost
12 ms	2,43%	0 ms	0%	4	playLevelArkanoid:285 localhost
11 775,50 ms	0,61%	0 ms	0%	1	Box2dWeb-2.1.a.3.js:1 localhost
2 ms	0,61%	0 ms	0%	1	f</B jquery-1.7.1.min.js:2 localhost

Figura 107 Resultado de las pruebas de rendimiento: Cargar la pantalla de juego de un nivel

El resultado de esta prueba es muy bueno, dado que tiene que buscar el fichero en disco y decodificar el contenido de dicho fichero. El resultado marca que esta operación tarda 17ms, un tiempo más que aceptable.

8.4.3 Guardar nivel

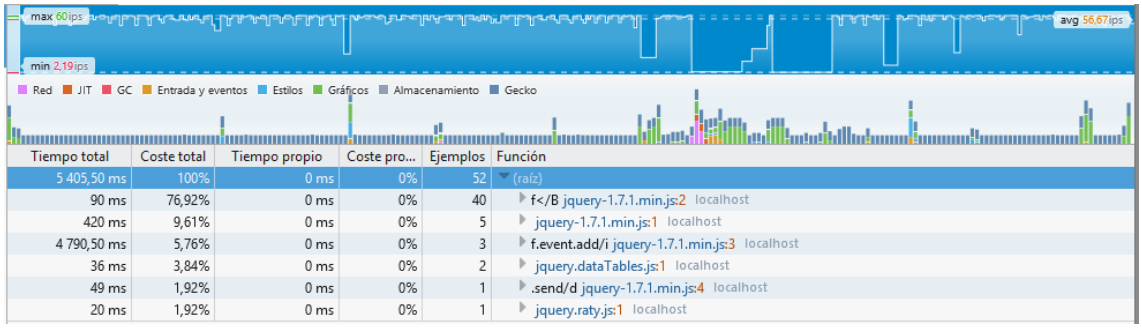


Figura 108 Resultado de las pruebas de rendimiento: Guardar nivel

Esta operación tiene un tiempo de respuesta muy bueno, 5ms para crear un fichero JSON con los datos del nivel.

8.4.4 Cargar la ventana Community

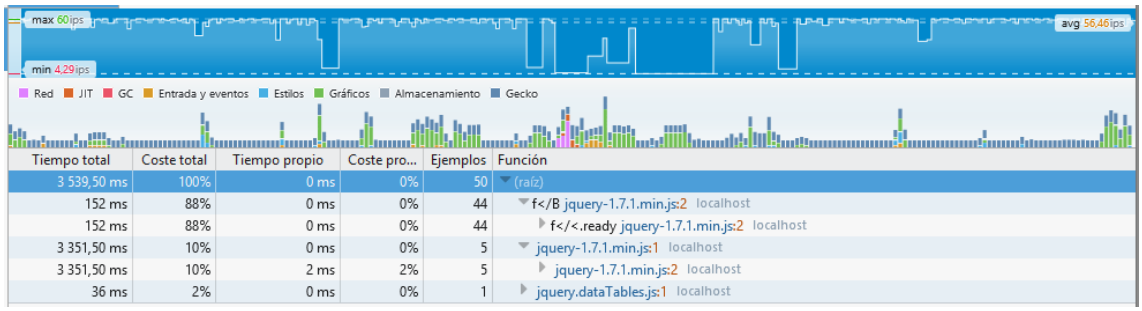


Figura 109 Resultado de las pruebas de rendimiento: Cargar la ventana Community

Esta es una de las operaciones que presuntamente podrían llevar más tiempo, pero visto el resultado obtenido, tan solo de 3ms se puede decir que el resultado obtenido es muy bueno.

8.4.5 Añadir nivel a la playlist de un jugador

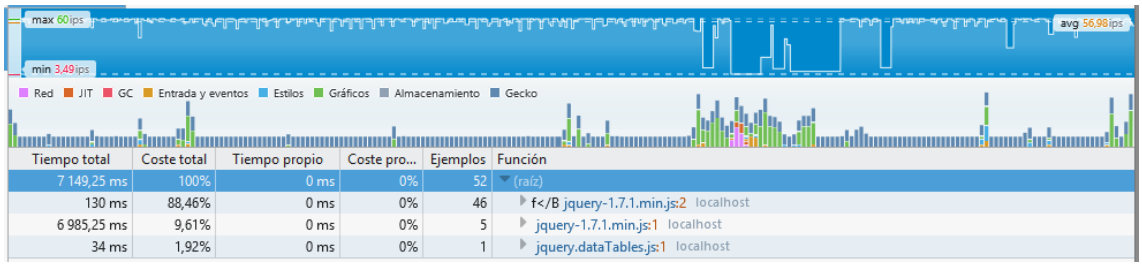


Figura 110 Resultado de las pruebas de rendimiento: Añadir nivel a la playlist de un jugador

Esta operación ha tenido un resultado muy bueno, se ha obtenido un tiempo de 7ms para realizar una operación de inserción en base de datos y carga de una pantalla en la que requiere muchas cargas de lectura en la base de datos.

8.4.6 Conclusión

Estudiando los resultados obtenidos, podemos concluir que se han superado satisfactoriamente las pruebas, dado que todos los tiempos están muy por debajo de los esperados.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

La instalación de la aplicación se va a realizar sobre una máquina virtual, esta máquina virtual utiliza un sistema operativo Linux, más concretamente la distribución Debian.7.8.

A continuación se proporciona un enlace para descargar este sistema operativo:

Debian 7.8: https://www.debian.org/releases/oldstable/debian-installer/

En este manual no se va a explicar cómo instalar el sistema operativo ni como crear una máquina virtual, para ello se facilitan estos enlaces:

Tutorial creación de una máquina virtual:

http://www.ite.educacion.es/formacion/materiales/85/cd/windows/3Instalacion/creacin_de_la_mquina_virtual.html

Tutorial instalación Debian 7.8:

https://www.debian.org/releases/oldstable/amd64/

Para que este proyecto funcione se necesitan un servidor Tomcat 7 y una base de datos MySQL. Estos son los pasos que hay que seguir para instalar estos elementos.

9.1.1 Instalación de MySQL:

- Instalamos MySQL con este comando: `apt-get install mysql-server mysql-client`
- Nos pedirá que le demos una contraseña al usuario root de MySQL:

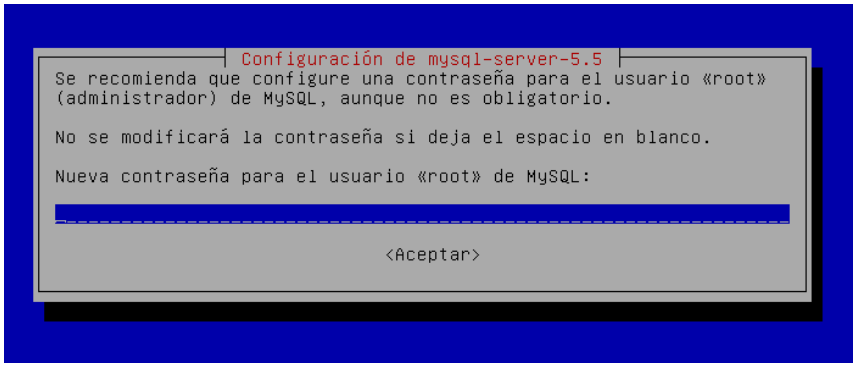


Figura 111 Manual de instalación: Contraseña del usuario root de MySQL

- Escribimos la contraseña y aceptamos.
- En la siguiente pantalla nos pedirá que repitamos la contraseña, la introducimos y aceptamos.
- La instalación terminará de forma automática.

9.1.2 Instalación del servidor Tomcat 7

- Para que el servidor Tomcat funcione necesitamos tener instalado java en el sistema, para instalarlo utilizamos esta instrucción: `apt-get install default-jdk`
- Descargamos el servidor tomcat de su repositorio mediante este comando: `wget http://apache.rediris.es/tomcat/tomcat-7/v7.0.62/bin/apache-tomcat-7.0.62.tar.gz`
- Lo descomprimimos usando este comando: `tar -xvzf apache-tomcat-7.0.27.tar.gz`
- Lo cambiamos de directorio a la ubicación donde necesita estar con este comando: `mv apache-tomcat-7.0.27 /usr/share`
- Creamos un enlace simbólico al directorio con este comando: `ln -s /usr/share/apache-tomcat-7.0.27 /usr/share/tomcat7`
- Nos cambiamos de directorio para acceder al fichero de arranque del servidor: `cd /usr/share/tomcat7/bin`
- Arrancamos el servidor con esta instrucción: `./catalina.sh start`
- Comprobamos que todo funciona correctamente accediendo a la dirección IP de la maquina seguida del puerto 8080: `ip_maquina:8080`

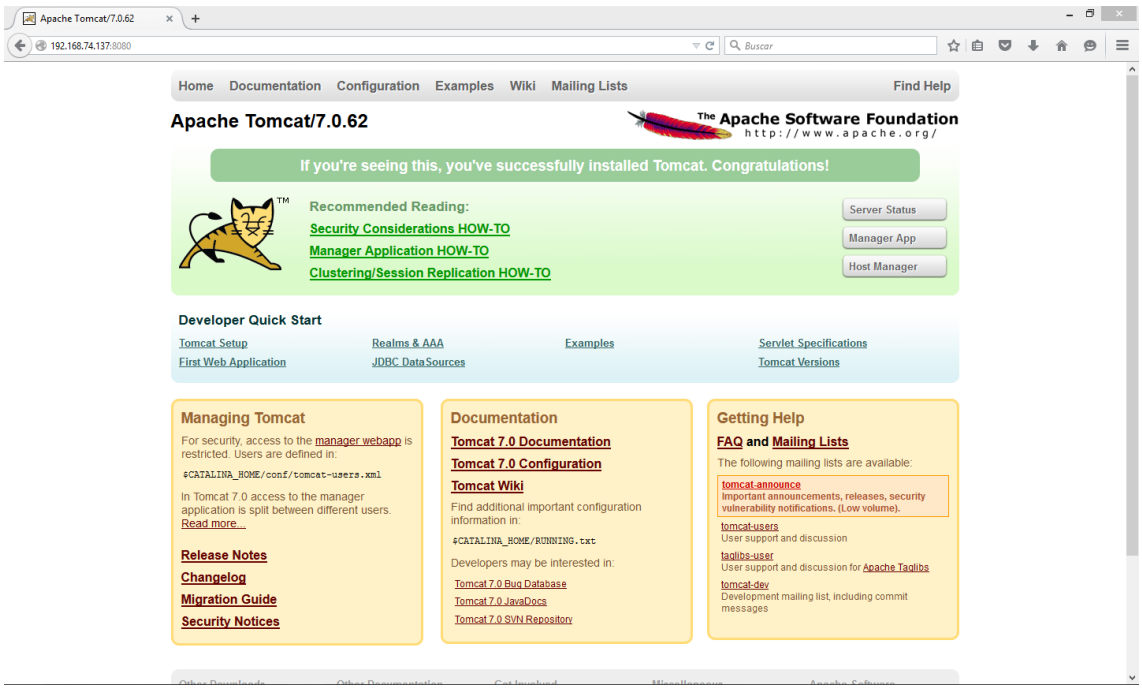


Figura 112 Manual de instalación: Servidor Tomcat

9.1.3 Instalación del PHPMyAdmin

Con esta herramienta podemos gestionar la base de datos MySQL instalada previamente.

- Instalamos phpMyAdmin con este comando: `apt-get install phpmyadmin`
- Nos preguntara el tipo de servidor que se reconfigurará para ejecutar phpMyAdmin. Seleccionamos Apache2.

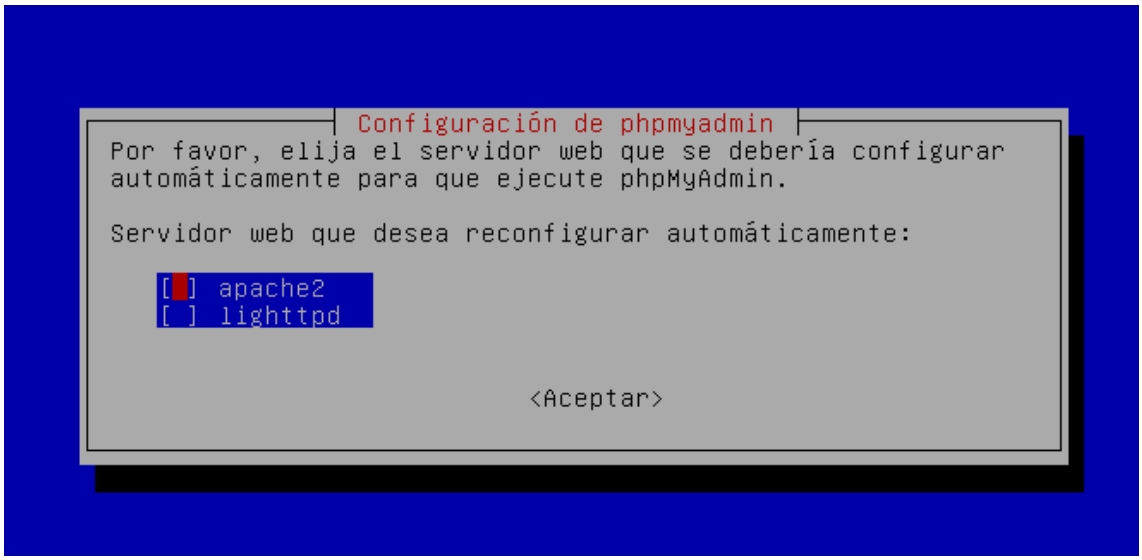


Figura 113 Manual de instalación: phpMyAdmin

- Se nos mostrar una pantalla para seleccionar si queremos configurar la base de datos con "dbconfig-common". Seleccionamos no.

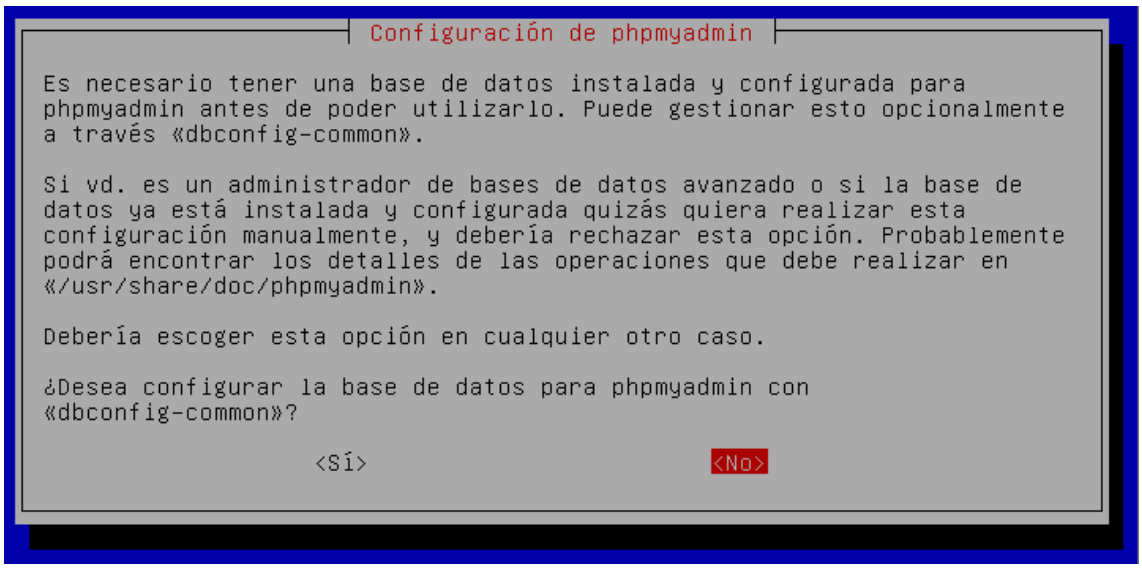


Figura 114 Manual de instalación: phpMyAdmin 2

- La instalación terminará de forma automática.
- Para comprobar que se ha instalado correctamente accedemos con un navegador web a la dirección: `ip_maquina/phpmyadmin/`

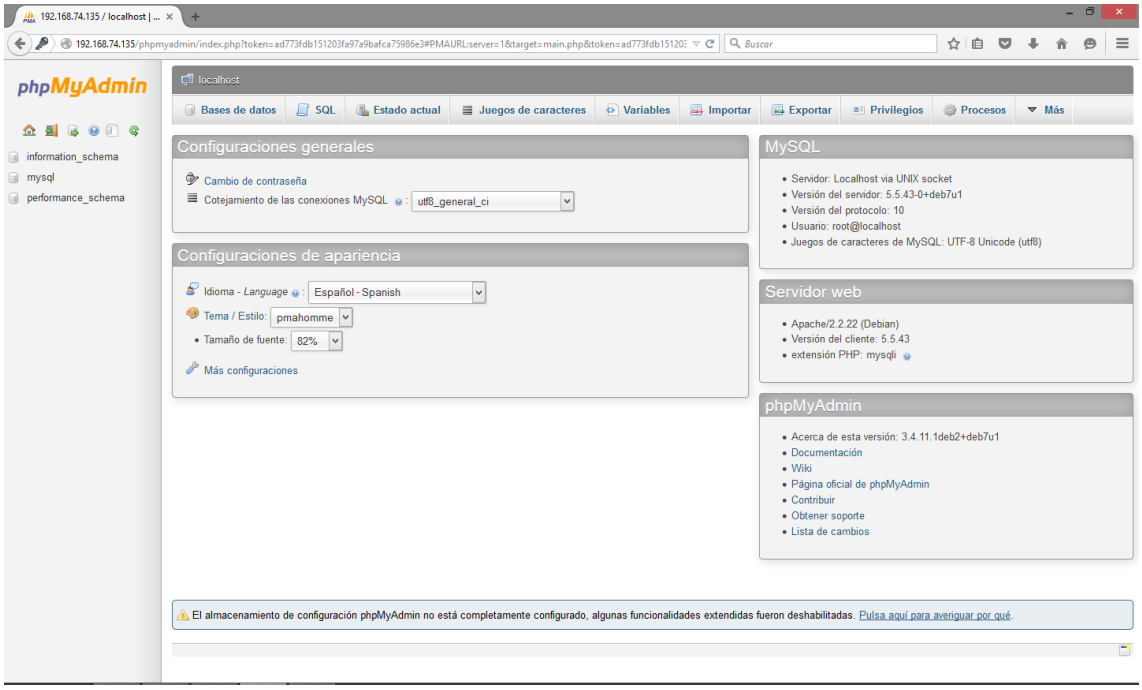


Figura 115 Manual de instalación: phpMyAdmin 3

9.1.4 Creación de las tablas de la base de datos

Se proporciona un fichero llamado pfm.sql que incluye un script para crear las tablas de la base de datos. Estos son los pasos a seguir para crear las tablas.

- Lo primero que se requiere es crear una base de datos, para ello seleccionamos la pestaña "*Base de datos*".

Figura 116 Manual de instalación: Importar tablas de la BBDD

- Escribimos el nombre de la base de datos en el campo "*Crear nueva base de datos*" y la llamamos **pfm.**

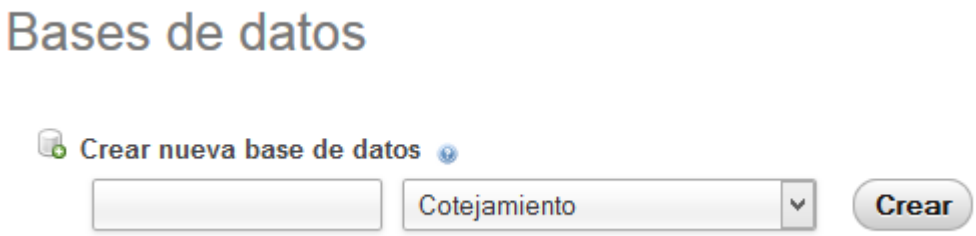


Figura 117 Manual de instalación: Importar tablas de la BBDD 2

Nota: Es muy importante que la base de datos se llame pfm porque si no la aplicación no funcionaria.

- Una vez creada aparecerá en el árbol de la parte izquierda de la pantalla.

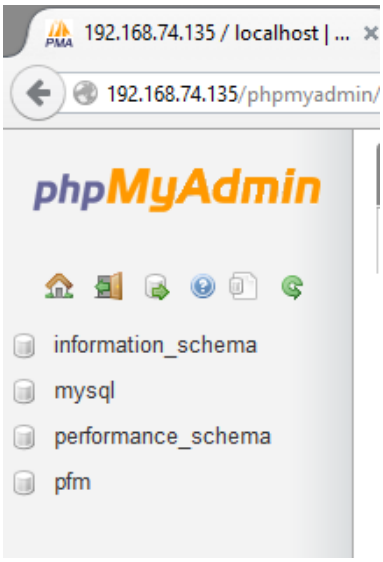


Figura 118 Manual de instalación: Importar tablas de la BBDD 3

- Pulsamos sobre la base de datos pfm en el árbol de la izquierda para seleccionarla.
- Seleccionamos la pestaña "*Importar*".

Figura 119 Manual de instalación: Importar tablas de la BBDD 4

- Seleccionar el archivo pfm.sql con el botón examinar de la sección "*Archivo a importar*".

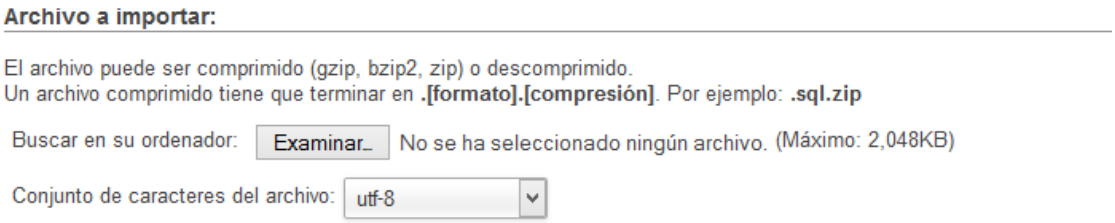


Figura 120 Manual de instalación: Importar tablas de la BBDD 5

- Pulsamos el botón "*Continuar*".
- Se nos mostrar un mensaje confirmándonos que la operación ha sido un éxito.

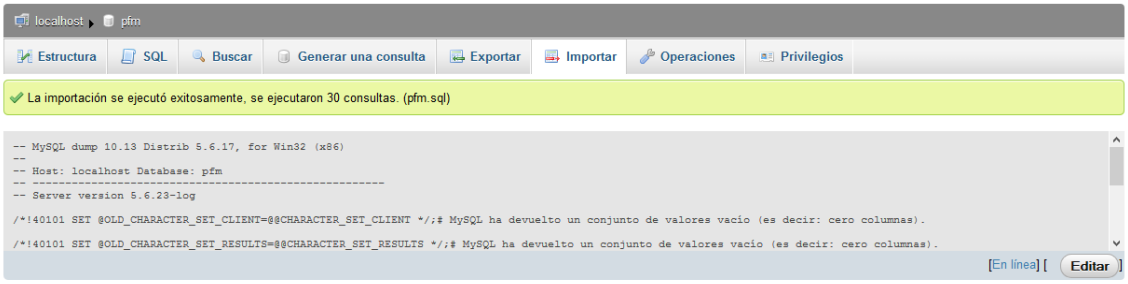


Figura 121 Manual de instalación: Importar tablas de la BBDD 6

- Podemos ver en la parte izquierda de la pantalla las tablas que se han creado.

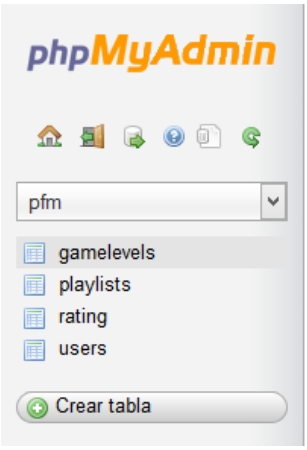


Figura 122 Manual de instalación: Importar tablas de la BBDD 7

9.1.5 Instalar la aplicación web en el servidor tomcat

Se proporciona un fichero war con la aplicación que ha de ser transferido a la máquina virtual.

Para ello hay que instalar un servidor ftp en el servidor y utilizar un programa cliente para transferir el fichero war con la aplicación.

- Instalamos en el servidor un ftp con la instrucción: `apt-get install ftpd`
- Creamos un nuevo usuario para el ft con la instrucción:
  `useradd -m <<nombre del usuario>>`
- Le damos una contraseña con la instrucción: `passwd <<nombre del usuario>>`
- Comprobamos que el ftp funciona conectándonos mediante un navegador web que soporte ftp a la ip del servidor

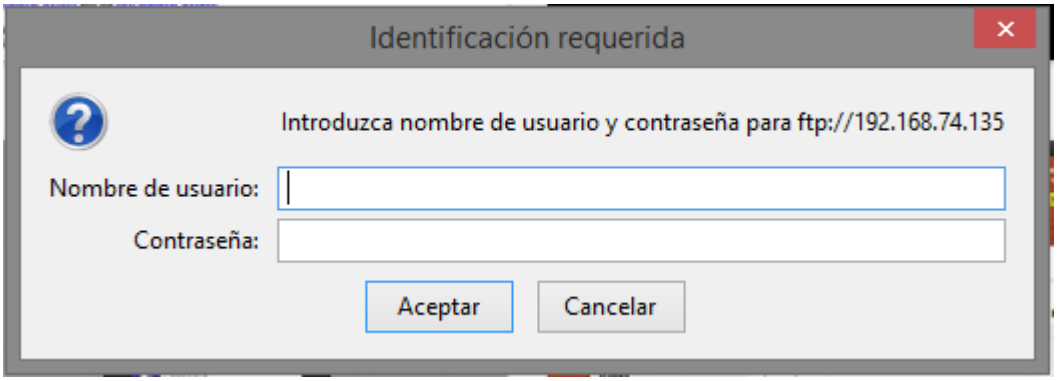


Figura 123 Manual de instalación: FTP

- Nos pedirá un usuario y una contraseña, introducimos el usuario y la contraseña que acabamos de crear. Nos sale el directorio del ftp.

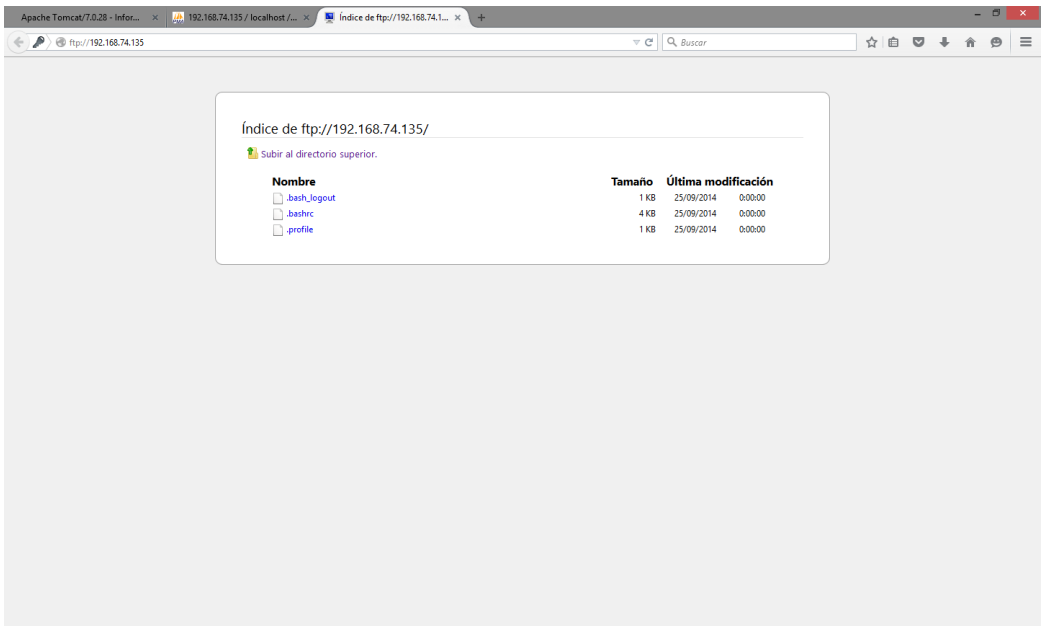


Figura 124 Manual de instalación: FTP 2

- Ahora es el momento de instalar el fichero war dentro del servidor. Mediante un programa cliente ftp como por ejemplo Winscp o Filexilla, nos conectamos con la máquina virtual usando la IP de la máquina, el puerto 22 o 21, depende del caso y la contraseña y usuario que hemos creado para el ftp.

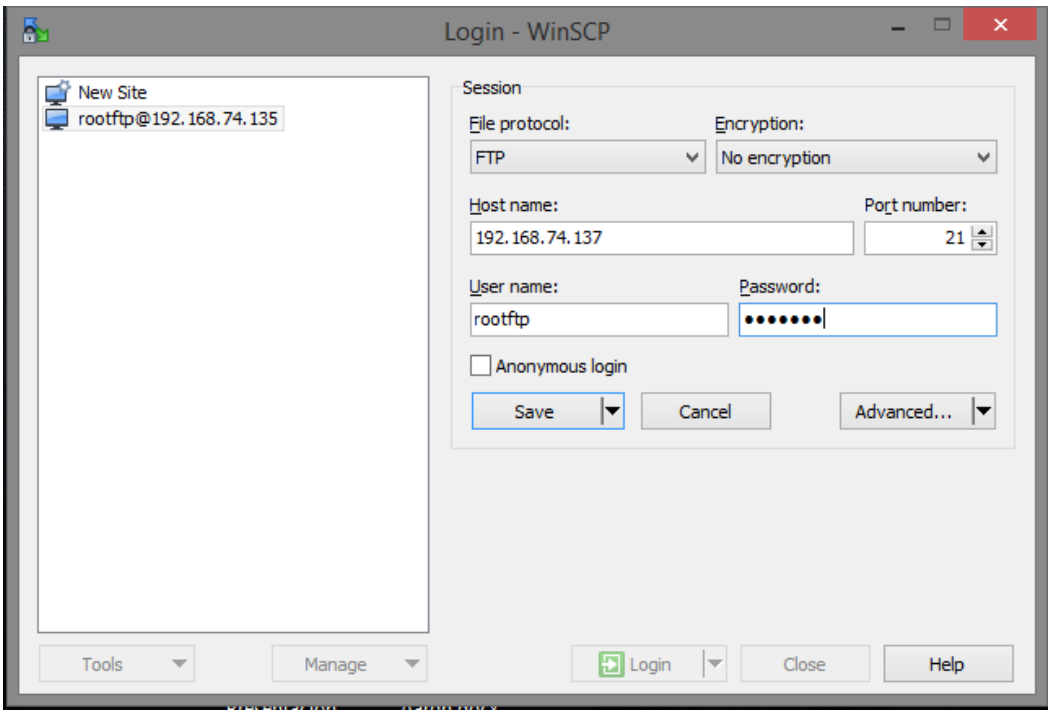


Figura 125 Manual de instalación: FTP 3

- Una vez conectados con la maquina solo hay que copiar el fichero war a la carpeta webapps del servidor tomcat. Esta carpeta se encuentra en la siguiente dirección: `/usr/share/apache-tomcat-7.0.62/webapps`

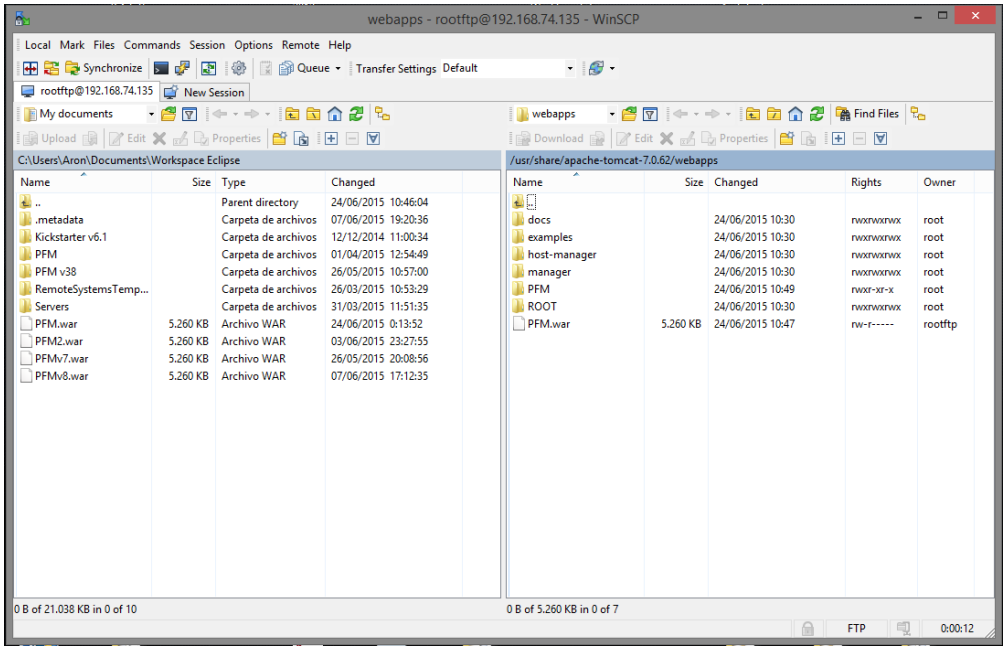


Figura 126 Manual de instalación: FTP 4

***Nota: si al copiar da un error es posible que se necesiten dar permisos a la carpeta webapps para permitir copiar dentro de ella. Esto se puede hacer con la instrucción** `chmod -R 777 webapps`*

- Una vez copiado el fichero podemos acceder a la aplicación web a través del navegador, con la siguiente dirección: `ip_maquina:8080/PFM`

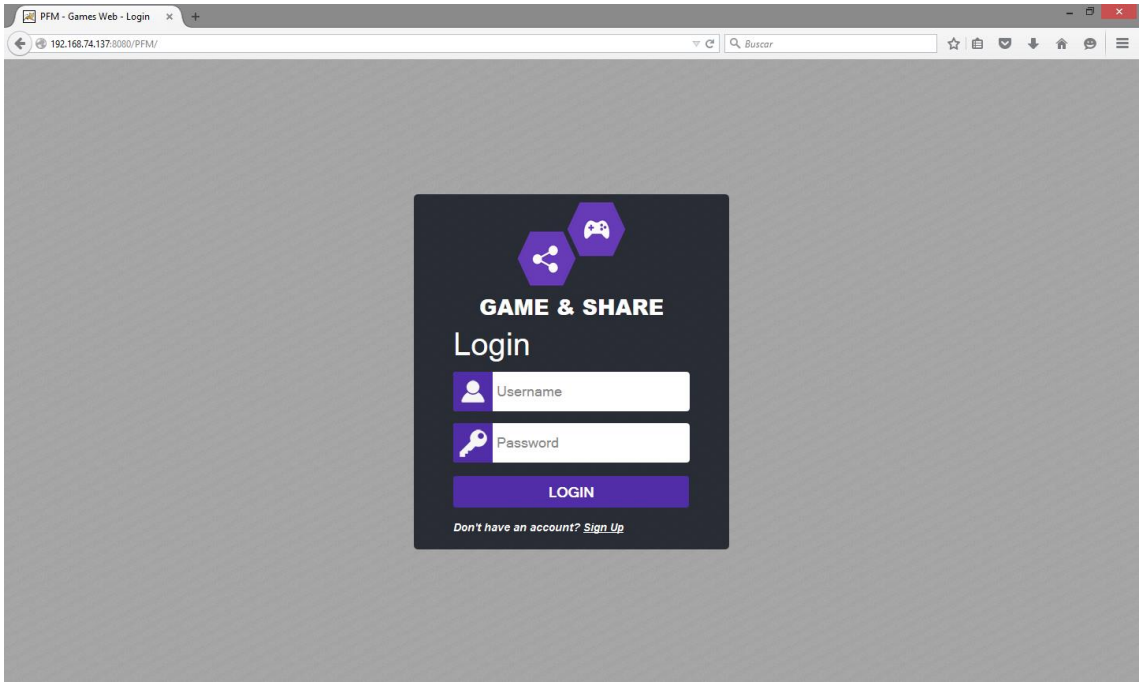


Figura 127 Manual de instalación: Aplicación Web

9.2 Manual de Ejecución

La ejecución de esta aplicación es muy sencilla, solo basta con seguir los siguientes pasos.

1. Arrancar la máquina virtual.
2. Iniciar sesión en el sistema como usuario root.
    a. Nombre de usuario: *root*
    b. Contraseña: *root*

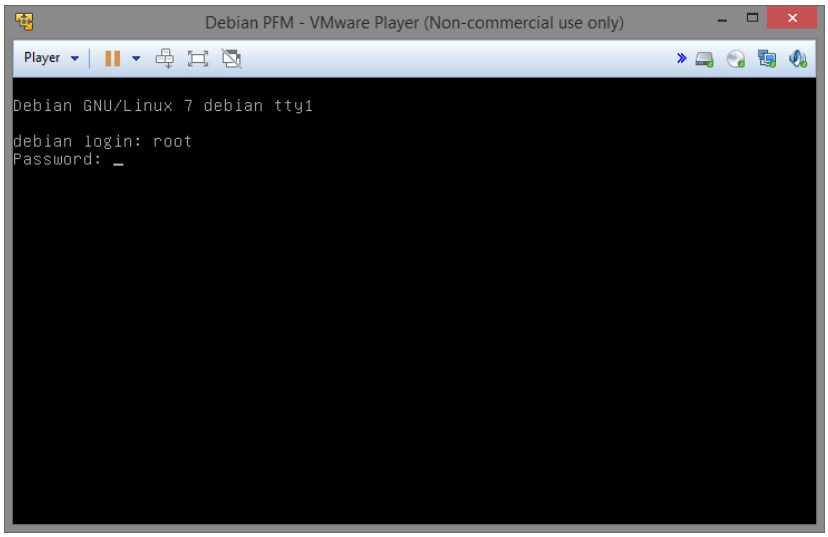


Figura 128 Manual de ejecución: Login en el sistema

3. Nos movemos al directorio donde se encuentra el fichero de arranque del servidor:
   `cd /usr/share/tomcat7/bin`
4. Arrancamos el servidor:
   `./catalina.sh start`

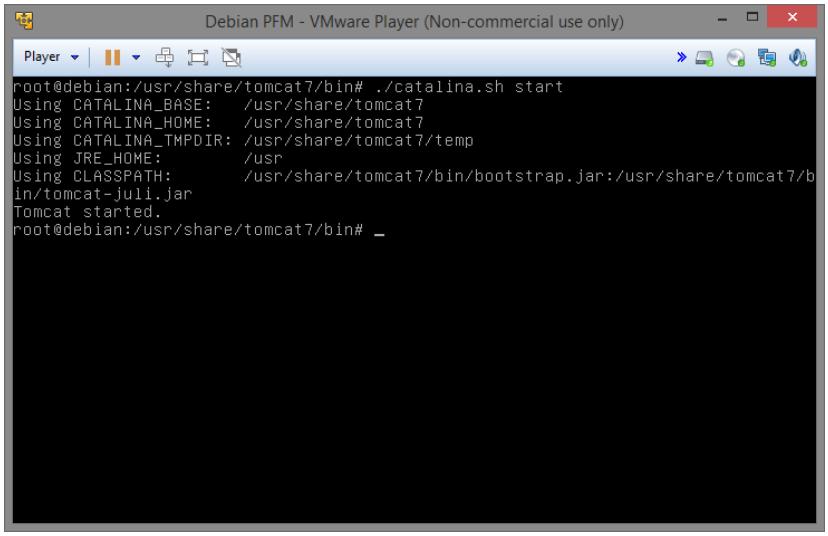


Figura 129 Manual de ejecución: Arranque del servidor

5. Acceder a la página web: `ip_maquina:8080/PFM`

9.3 Manual de Usuario

En este manual se van a describir paso a paso todas las operaciones que se pueden realizar en esta aplicación.

9.3.1 Acceso a la página web

Hay dos maneras de tener acceso a la página web: Logueandose con una cuenta de usuario o creando un usuario nuevo.

9.3.1.1 Login de usuario

Al entrar en la página, la primera pantalla que se muestra es esta, la pantalla de acceso a la página web.

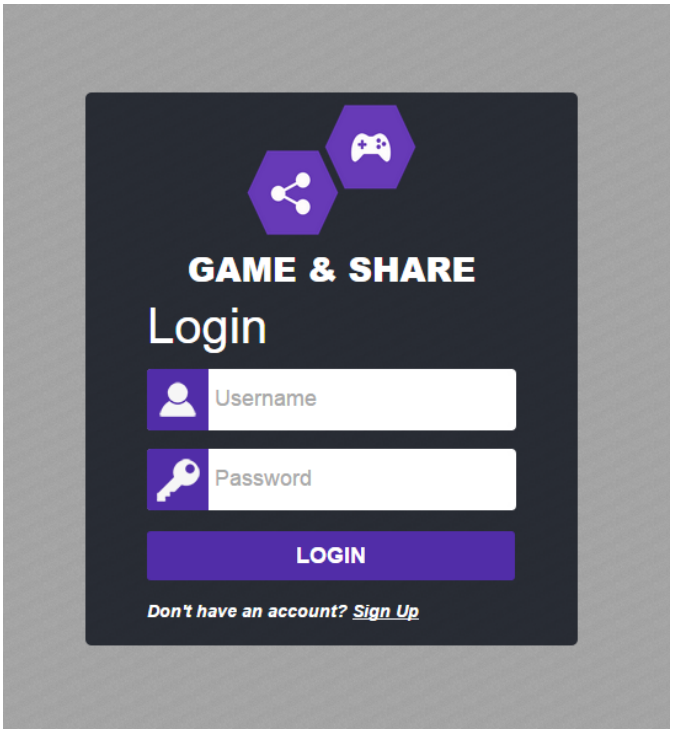


Figura 130 Manual de Usuario: Pantalla de login

Para poder acceder a la página web solo basta con seguir estos pasos:

1. Introducir el nombre de usuario en el campo "*Username*".
2. Introducir la contraseña de acceso en el campo "Password".
3. Pulsar el botón login.

9.3.1.2 Crear nuevo usuario

Si no disponemos de una cuenta de usuario, siempre podremos crear una cuenta nueva para tener acceso a la página.

En la página de "*Login*" podemos pulsar sobre el enlace "*Sign up*" y nos conducirá a la pantalla de registro de un nuevo usuario.

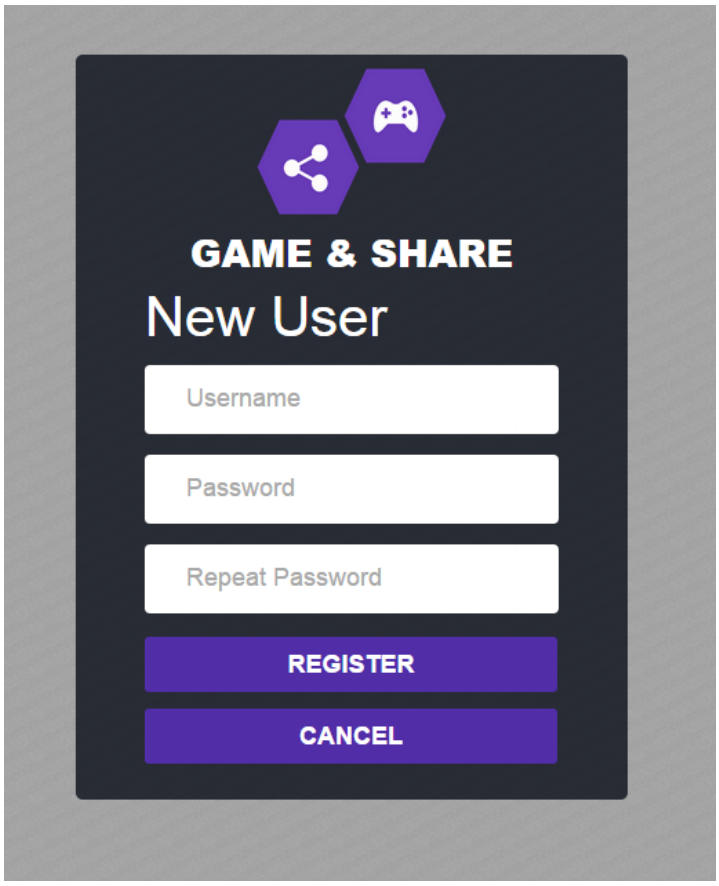


Figura 131 Manual de Usuario: Pantalla de registro

Para crear un nuevo usuario hay que seguir estos pasos:

1. Introducir en el campo "Username" un nombre de usuario.
2. Introducir en el campo "Paasword" una contraseña de acceso.
3. Introducir la misma contraseña de acceso en el campo "Repeat password".
4. Pulsar el botón "Register".

9.3.2 User Home

Una vez dentro de la aplicación, la pantalla que se carga es el home del usuario. En esta pantalla se pueden hacer varias operaciones, como por ejemplo jugar a los niveles que se encuentren dentro de la playlist del jugador o editar los niveles que dicho jugador ha creado.

9.3.2.1 Jugar niveles

Dentro del home del usuario se presentan dos listas de niveles: la playlist del usuario y los niveles que ha creado.

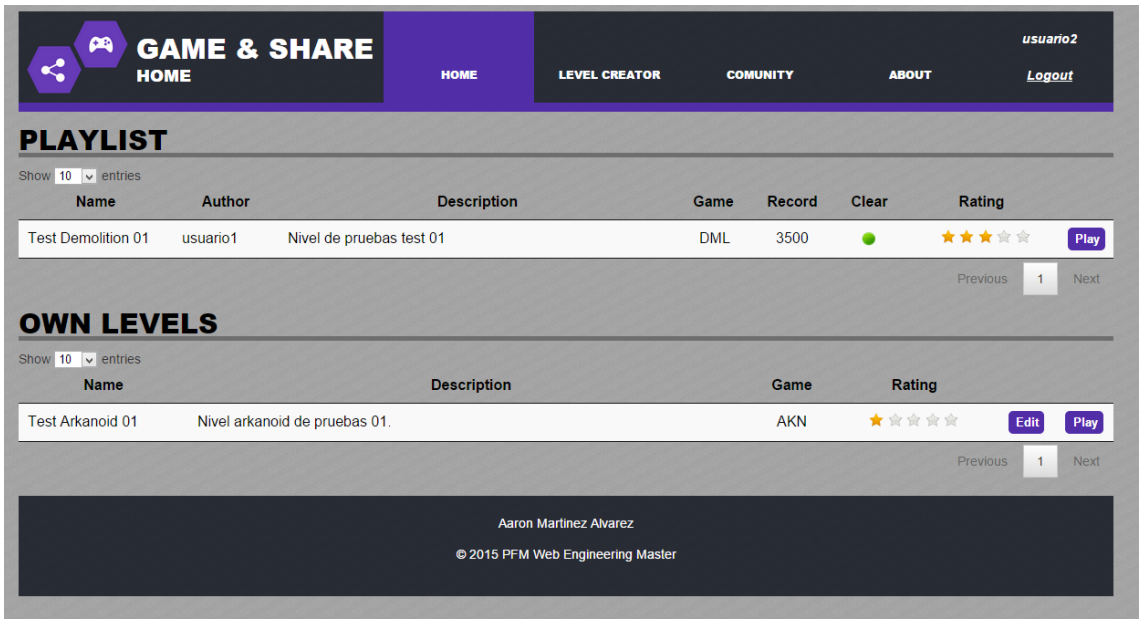


Figura 132 Manual de Usuario: Home del usuario

Para jugar cualquier nivel que el usuario tenga en su home solo basta con pulsar el botón "*Play*" del nivel que desee jugar. Automáticamente se cargará la pantalla de juego con el nivel.

9.3.2.2 Juego Demolition

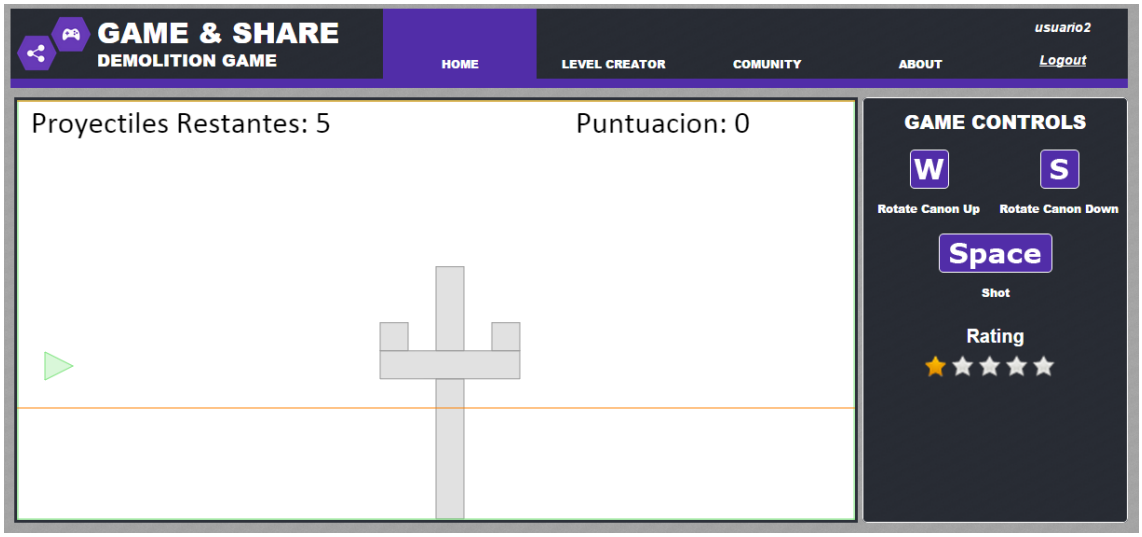


Figura 133 Manual de Usuario: Juego demolition

El juego demolition consiste en tirar todas las piezas de la construcción por debajo de la línea que marca el límite de altura para superar el nivel.

Estos son los controles para jugar a este juego:

- **W**: Rotamos el cañón hacia arriba.
- **S**: Rotamos el cañón hacia abajo.
- **Barra Espacio**: Disparar proyectil.

9.3.2.3 Juego Arkanod

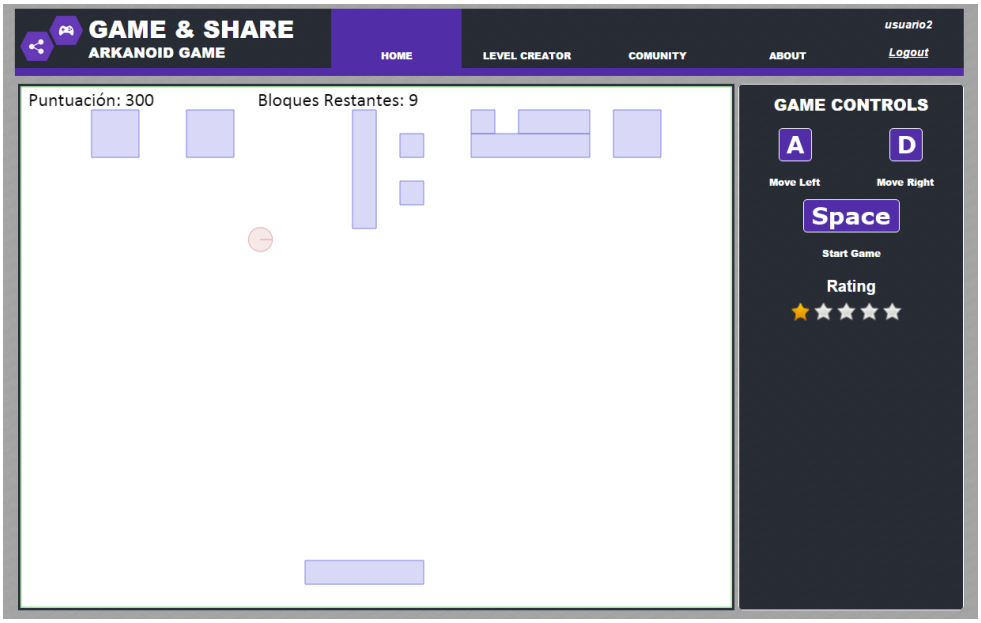


Figura 134 Manual de Usuario: Juego arkanoid

El juego arkanoid consiste en destruir los bloques de la parte superior de la pantalla haciendo rebotar la bola en la barra y sin que la bola toque el borde inferior de la pantalla.

Estos son los controles para jugar a este juego:

- **A**: Mover la barra hacia la izquierda.
- **D**: Mover la barra hacia la derecha.
- **Barra Espacio**: Empezar juego.

9.3.3 Herramienta de edición

Otra de las opciones que presenta esta aplicación es la herramienta para crear contenido para los juegos Demolition y Arkanoid.

9.3.3.1 Editor de niveles Demolition

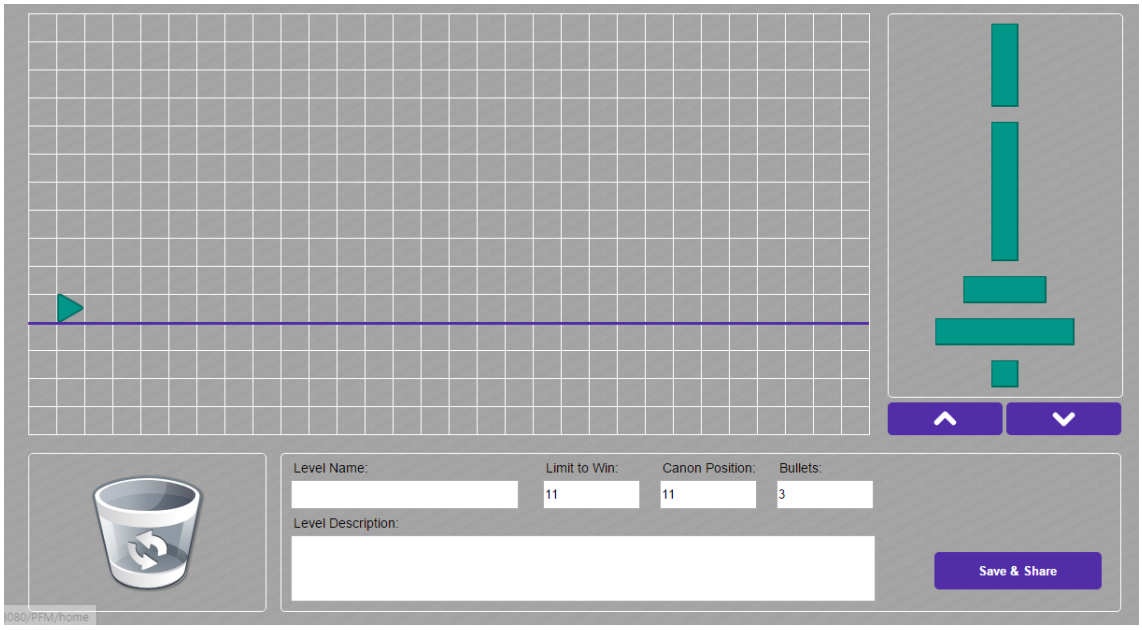


Figura 135 Manual de Usuario: Editor de niveles Demolition

Para crear un nivel para el juego demolitiion hay que seguir estos pasos:

- Arrastrar cualquier pieza de las que se encuentran en la paleta situada a la derecha de la pantalla, a la cuadricula central.
- Si se quiere eliminar una de las piezas colocadas en la cuadricula, solo hay que arrastrarlas a la papelera.
- Una vez compuesta la estructura que compondrá el nivel hay que introducir los datos del nivel.
- Escribir el nombre que se le va a dar al nivel en el campo "*Level Name*".
- Escribir la descripción del nivel en el campo "*Description*".
- Seleccionar la posición de la línea que marca la altura para superar el nivel con el control numérico "*Limit to win*".

- Seleccionar la posición de cañón con el control numérico "Canon Position".
- Seleccionar el número de proyectiles disponibles en el nivel con el control numérico "*Bullets*".
- Una vez terminados todos estos pasos hay que guardar el nivel, para ello basta con pulsar el botón "*Save & Share*".

9.3.3.2 Editor de niveles Arkanoid

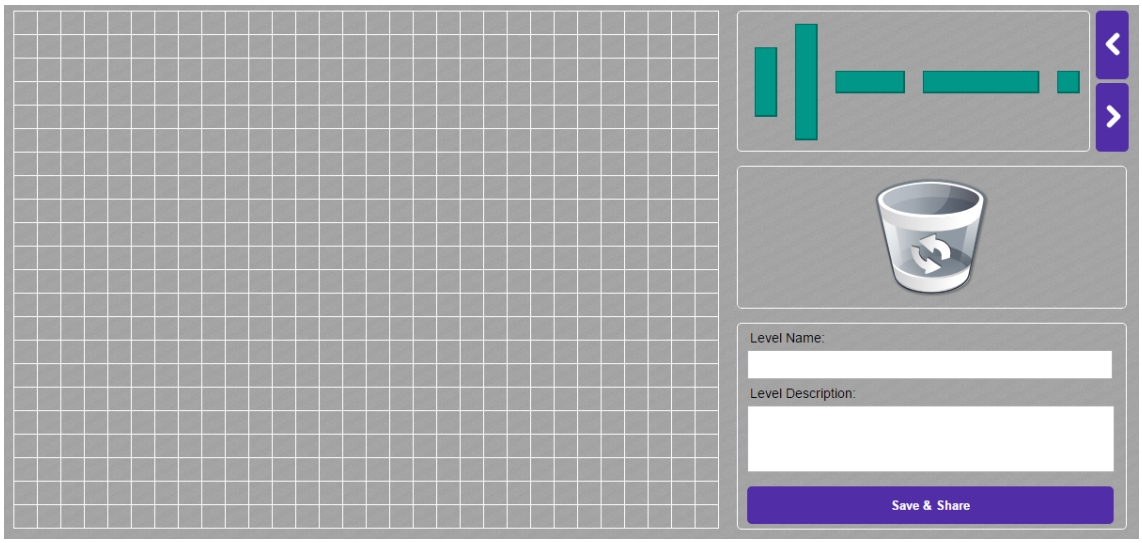


Figura 136 Manual de Usuario: Editor de niveles Arkanoid

Para crear un nivel para el juego arkanoid hay que seguir estos pasos:

- Arrastrar cualquier pieza de las que se encuentran en la paleta situada a la derecha de la pantalla, a la cuadricula central.
- Si se quiere eliminar una de las piezas colocadas en la cuadricula, solo hay que arrastrarlas a la papelera.
- Una vez compuesta la estructura que compondrá el nivel hay que introducir los datos del nivel.
- Escribir el nombre que se le va a dar al nivel en el campo "*Level Name*".
- Escribir la descripción del nivel en el campo "*Description*".
- Una vez terminados todos estos pasos hay que guardar el nivel, para ello basta con pulsar el botón "*Save & Share*".

9.3.4 Community

La última de las opciones más interesantes de la aplicación es la de poder añadir niveles creados por otros usuarios a la playlist para ser jugados posteriormente.

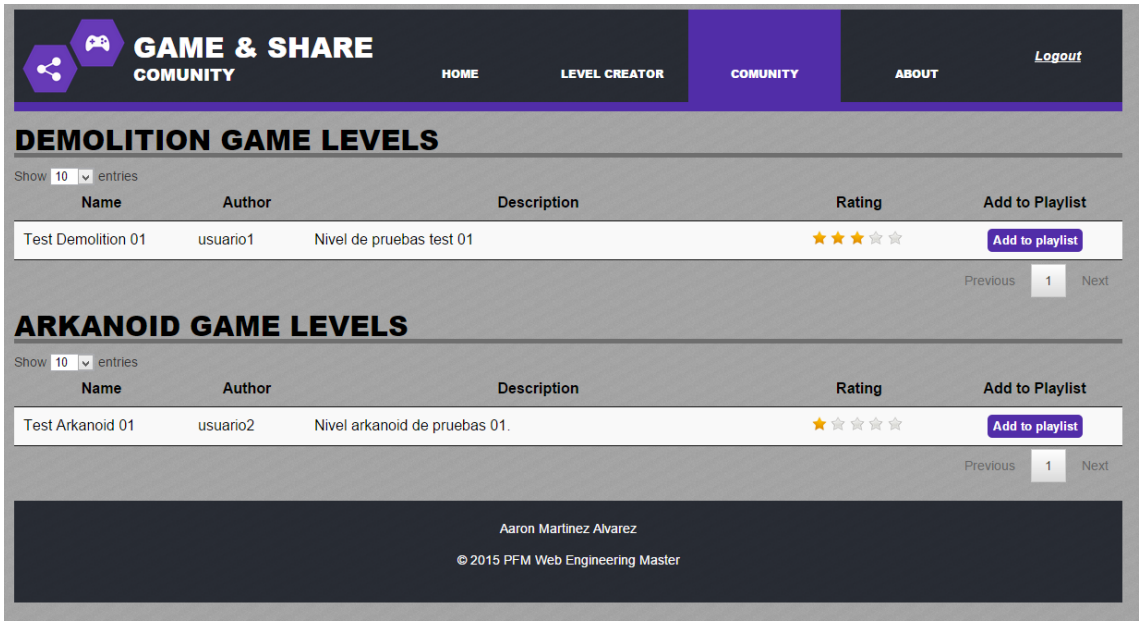


Figura 137 Manual de Usuario: Community

Para añadir un nivel a la playlist basta con pulsar el botón "*Add to playlist*" de cualquiera de los niveles.

9.4 Manual del Programador

En este manual se añaden algunos consejos y recomendaciones para aquellos que deseen ampliar la aplicación.

Uno de los puntos en los que la aplicación podría ser susceptible a sufrir ampliaciones es en los videojuegos ya que por las características de la aplicación puede ser ampliada añadiendo más tipos de juegos y herramientas de edición para dichos juegos.

En este proyecto están separados los juegos y las herramientas del resto de módulos de la aplicación mediante controladores propios para cada uno de estos elementos.

Si se quiere añadir algún tipo de juego nuevo basta con crear el motor del juego en un fichero JSP y una clase Java que haga de controlador. La JSP se comunicaría con su controlador específico y este se integraría con las clases del paquete de negocio gracias a las interfaces que están implementadas.

Con la herramienta de edición para dicho juego pasaría exactamente lo mismo, se desarrollaría un fichero JSP con la vista de la herramienta y un controlador que lo maneje y lo conecte con las clases del negocio.

Estas clases creadas se podrían integrar perfectamente con la clase de utilidades que permite el manejo de ficheros JSON, dado que es independiente o incluso se podría crear otro sistema para el manejo de la información de los nuevos contenidos.

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Este proyecto fue una gran oportunidad para aplicar todos los conocimientos adquiridos en el master, la oportunidad para crear una página web completa y original.

Se ha conseguido crear una plataforma donde poder jugar, poder crear y compartir contenidos. Se han conseguido integrar en una misma plataforma varios videojuegos basados en físicas y unas sencillas herramientas para crear contenidos.

El objetivo ha sido que cada uno de los aspectos de la aplicación resultase atractivo e intuitivo para todo aquel que decida acercase a la plataforma. Se quiere ofrecer una experiencia divertida para que el usuario sea capaz de ampliar junto con la comunidad la vida de la plataforma.

Se ha intentado que cada una de las secciones que componen los pilares principales de la aplicación: la creación de contenido y los juegos, resulten fáciles de manejar al usuario. Se han diseñado estas secciones de tal manera que se asemejen lo más posible a formas de uso con las que el usuario está familiarizado. La manera en la que el usuario puede crear los niveles para los juegos es muy similar a los juguetes de construcción por piezas, se cogen las piezas y se van colocando unas junto a otras para crear algo, pues con la aplicación funciona exactamente igual, se colocan piezas sobre una cuadricula para construir algo.

Con los juegos pasa lo mismo, se han hecho unos juegos muy similares a grandes éxitos que todo el mundo conoce y sabe cómo jugar a ellos solo con un primer vistazo. Los juegos presentados son una versión bastante simplificada de Angry Birds y de Arkanoid respectivamente, ambos grandes éxitos mundiales.

En conclusión, el proyecto realizado cumple con todos los objetivos que se propuso inicialmente y todos los resultados entran dentro de lo esperado y no podrían ser más satisfactorios.

10.2 Ampliaciones

Se han pensado varias ampliaciones para este proyecto:

- Una de las ampliaciones que podría tener este proyecto es añadir más tipos de juegos, esto aportaría más riqueza y variedad a las opciones jugables que presenta el portal web.
- Otra ampliación podría venir de la mano de la anterior y es crear más herramientas de creación de contenido ajustadas a los nuevos juegos que se pudiesen introducir. Con más herramientas se ampliaría la vida de los juegos que están presentes en la plataforma.
- Otra ampliación podría ser añadir rankings de jugadores según las puntuaciones que estos han obtenido jugando a los distintos niveles. Esta ampliación podría acercar más la plataforma al resto de opciones disponibles actualmente en el mercado y que si ofrecen este tipo de características.
- Otra ampliación podría ser incluir mensajes o publicaciones por parte de los jugadores para dar a conocer las creaciones que han hecho, una característica que podría convertir en una especie de red social la plataforma, manteniendo más conectada a la comunidad de usuarios.
- Al igual que la ampliación anterior también se podría añadir algún sistema de seguimiento de usuarios, lista de amigos u otras características similares.

Capítulo 11. Presupuesto

11.1 Costes de personal:

Este proyecto ha sido desarrollado por una sola persona por lo que para ello se ha planificado con un calendario laboral que implica una jornada de ocho horas diarias.

El cálculo del número de horas totales dedicadas al proyecto, asciende en total a 2040 horas con la siguiente distribución de tiempo entre fases.

- **Fase de análisis:** 10 días * 8 horas/día = 80 horas
- **Fase de diseño:** 6 días * 8 horas/día = 48 horas
- **Fase de implementación:** 89 días * 8 horas/día = 712 horas
- **Documentación:** 143 días * 8 hora/día = 1144 horas
- **Pruebas:** 7 días * 8 horas/día = 56 horas

Concepto	Nº Horas	Precio / Hora (€)	Subtotal
Análisis	80	25	2000
Diseño	48	25	1200
Implementación	712	25	17800
Pruebas	56	25	1400
Documentación	1144	25	28600
		Total:	51000

11.2 Costes de Software y Licencias:

Durante la realización del proyecto, se han utilizado diferentes aplicaciones y herramientas para generar tanto el código de la aplicación como documentación y manuales. Por ello se incluyen dichas licencias dentro del presupuesto.

Concepto	Cantidad
Microsoft Windows 8.1	0
MySQL	0
Total:	0

11.3 Otros costes:

Se aplicaran costes de amortización y precio del hardware.

- **Amortización del hardware:** 2400 €

11.4 Beneficio:

El beneficio obtenido por el desarrollo del proyecto se negocia con el cliente.

- **Beneficio:** 3000 €

11.5 Resumen del presupuesto:

Item	SubItem	Concepto	Cantidad	Precio Unitario	Total
1		*Desarrollo de la aplicación*			**51000 €**
	1.1	Análisis	80	25 €	2000 €
	1.2	Diseño	48	25 €	1200 €
	1.3	Implementación	712	25 €	17800 €
	1.4	Documentación	1144	25 €	28600 €
	1.5	Pruebas	56	25 €	1400 €
2		*Software y Licencias*			**0 €**
	2.1	Microsoft Windows 8.1	1		0 €
	2.2	MySQL	1		0 €
3		*Otros*			**5400 €**
	3.1	Amortización del hardware	1	2000 €	2400 €
	3.2	Beneficios	1	3000 €	3000 €
				Subtotal:	56400 €
				IVA (21%)	11844 €
				Total:	**68244 €**

11.6 Presupuesto para el cliente

Item	SubItem	Concepto	Cantidad	Precio Unitario	Total
1		Desarrollo de la aplicación			**51000 €**
	1.1	Análisis y diseño	128 horas	25 €	3200 €
	1.2	Implementación	768 horas	25 €	19200 €
	1.3	Documentación	1144 horas	25 €	28600 €
2		Software y Licencias			**0 €**
	2.1	Microsoft Windows 8.1	1	0 €	0 €
	2.2	MySQL	1	0 €	0 €
3		Otros			**5400 €**
				Subtotal:	56400 €
				IVA (21%)	11844 €
				Total:	**68244 €**

Capítulo 12. Referencias Bibliográficas

12.1 Referencias en Internet

[Hassan08] Hassan Montero, Y. "Guía de Evaluación Heurística de Sitios Web". http://www.nosolousabilidad.com/articulos/heuristica.htm

[Stromberg12] Stromberg, Jonny "JonnyStromber.com". http://www.jonnystromberg.com/box2d-javascript-tutorial

[Bishop10] Bishop, Allan "Allan Bishop Blog". http://blog.allanbishop.com/box2d-2-1a-tutorial-%E2%80%93-part-4-collision-detection/

[Ladd11] Ladd, Seth "Seth Ladd's blog" http://blog.sethladd.com/2011/09/box2d-collision-damage-for-javascript.html

"Box2d". http://box2d.org/

"Box2d GitHub". https://github.com/libgdx/libgdx/wiki/box2d#initialization

"PhysicsJS GitHub". https://github.com/wellcaffeinated/PhysicsJS/wiki/Bodies

"Stackoverflow". http://stackoverflow.com/questions/4926935/find-what-bodies-are-colliding-in-box2d-using-c

"Stackoverflow". http://stackoverflow.com/questions/867469/when-i-make-a-draggable-clone-and-drop-it-in-a-droppable-i-cannot-drag-it-again

"Stackoverflow". http://stackoverflow.com/questions/15622333/creating-svg-through-the-dom-using-jquery

"Stackoverflow". http://stackoverflow.com/questions/3642035/jquerys-append-not-working-with-svg-element

"iforce2d.net". http://www.iforce2d.net/b2dtut/drawing-objects

"iforce2d.net". http://www.iforce2d.net/b2dtut/collision-anatomy

"GotoAndLearn.com". http://www.gotoandlearn.com/play.php?id=176

"GotoAndLearn.com". http://www.gotoandlearn.com/play.php?id=177

"davidmarco.es" http://www.davidmarco.es/spring-mvc

"crunchify.com". http://crunchify.com/simplest-spring-mvc-hello-world-example-tutorial-spring-model-view-controller-tips/

"tutorialesPoint.com". http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

"expertojava.ua.es". http://expertojava.ua.es/j2ee/publico/spring-2012-13/sesion03-apuntes.html

"todosobreprogramacion". http://todosobreprogramacion.blogspot.com.es/2013/11/spring-mvc-formularios-validator.html

"CodeJava.net". http://www.codejava.net/frameworks/spring/reading-properties-files-in-spring-with-propertyplaceholderconfigurer-bean

"JqueryUI". https://jqueryui.com/droppable/

"AuthorCode". http://www.authorcode.com/highlight-a-droppable-area-on-hover-using-jquery-ui/

"jsbin". http://jsbin.com/aseqid/10/edit?html,output

"material palette". http://www.materialpalette.com/light-green/lime

"transparent textures". http://www.transparenttextures.com/

"javainutil". http://javainutil.blogspot.com.es/2013/03/java-escribir-un-json.html

"wdexplorer.com". http://wdexplorer.com/20-examples-beautiful-css-typography-design/

"wbothelos.com". http://wbotelhos.com/raty

"jarroba.com". http://jarroba.com/ajax-con-jsp-y-servelts/

"Ben.Knows.Code()". http://www.benknowscode.com/2012/09/using-svg-elements-with-jquery_6812.html

"cusrosSpring2013". https://cursospring2013.wordpress.com/7-junit-mockito/

"T.A.W". http://www.tawdis.net/

"AChecker".http://achecker.ca/checker/index.php

"howtoforge.com". https://www.howtoforge.com/installing-apache2-with-php5-and-mysql-support-on-debian-wheezy

"ajpdsoft.com". http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=520

"openalfa.com". http://blog.openalfa.com/como-instalar-un-contenedor-tomcat-7-en-debian

Capítulo 13. Apéndices

13.1 Contenido Entregado

Los contenidos presentados en el cd están estructurados en diferentes directorios según el propósito de cada uno de ellos. Habrá un directorio con los ficheros de la aplicación, otro con la documentación y otro con los ficheros fuente usados en el desarrollo del proyecto.

13.1.1 Contenidos

Directorio	Contenido
./ Directorio raíz	Contiene un fichero leeme.txt explicando toda esta estructura.
./ Herramienta web de edición de niveles para videojuegos	Contiene toda la estructura de directorios del proyecto para desarrollo.
./ instalación	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 con la que se han generado los diagramas.

13.1.1.1 Estructura de Directorios de desarrollo

En esta tabla se pondrá un resumen de la estructura de directorios del proyecto, ya que al ser un proyecto generado por una herramienta hay muchos directorios con ficheros de configuración, propiedades, etc. Se mostraran los que realmente tienen contenido importante para el proyecto.

Directorio	Contenido
./ Directorio raíz de desarrollo	Contiene los ficheros de proyecto del IDE utilizado.
./src	Ficheros fuente
./src/main/java	Todos los ficheros Java de la aplicación, agrupados en sus respectivos paquetes.
./src/main/webapp	Ficheros que componen las páginas web de la aplicación.
./src/main/webapp/resources	Ficheros de recursos para las páginas web (imágenes, hojas de estilo, etc).
./src/main/webapp/resources/css	Todos los ficheros CSS de la aplicación.
./src/main/webapp/resources/img	Todas las imágenes utilizadas en las páginas web.

./src/main/webapp/resources/js	Todos los ficheros JavaScript de las páginas web.
./src/main/webapp/resources/lib	Todas las librerías importadas que se usan en las páginas web.
./src/main/webapp/WEB-INF	Contiene los ficheros que componen las vistas, ficheros de propiedades, etc.
./src/main/webapp/WEB-INF/Properties	Todos los ficheros de propiedades con las consultas a la base de datos externalizadas.
./src/main/webapp/WEB-INF/Spring	Ficheros de configuración propios del framework Spring.
./src/main/webapp/WEB-INF/Views	Todos los ficheros JSP de la aplicación.
./src/test	Todos los ficheros de pruebas.
./src/test/java	Ficheros java con las clases utilizadas para las pruebas.

13.1.2 Instalación

Para poder ejecutar la aplicación hay que seguir estos sencillos pasos:

1. Instalar un servidor Tomcat en la maquina en la que se quiera desplegar la aplicación. Se puede descargar desde su página oficial o se puede descargar de algún repositorio oficial. ***Ver manual de instalación*.**
2. Instalar una base de datos MySQL. ***Ver manual de instalación*.**
3. Crear las tablas en la base de datos mediante el script sql facilitado en el directorio de instalación.
4. Introducir el fichero .war de la aplicación facilitado en el directorio de instalación dentro del Tomcat. ***Ver manual de instalación*.**
5. Acceder a la dirección de la maquina donde se ha desplegado la aplicación a través de un navegador web. ***Ver manual de instalación*.**

13.2 Código Fuente

En este apartado se adjuntaran las clases más importantes del proyecto, no se añadirán todas porque el código fuente integro será entregado en documentos aparte.

13.2.1 Paquete Views:

13.2.1.1 User_Home.js

Este es el código JavaScript de la vista del User Home.

```
$(document).ready( function () {
        setTables();
        setStarsRating();
        setClearStage();
} );

function setTables() {
        $('#userHome-playlist-table').DataTable({
            paging: true,
            pagingType: "simple_numbers",
            bInfo: false,
            searching: false,
            ordering:  false,
            "columnDefs": [ { "width": "15%", "targets": 0 },
                            { "width": "10%", "targets": 1 },
                            { "width": "40%", "targets": 2 },
                            { "width": "5%",  "targets": 3 },
                            { "width": "5%",  "targets": 4 },
                            { "width": "5%",  "targets": 5 },
                            { "width": "15%", "targets": 6 },
                            { "width": "5%",  "targets": 7 }]
        });

        $('#userHome-ownLevels-table').DataTable({
            paging: true,
```

```
            pagingType: "simple_numbers",
            bInfo: false,
            searching: false,
            ordering:  false,
            "columnDefs": [ { "width": "15%", "targets": 0 },
                            { "width": "50%", "targets": 1 },
                            { "width": "5%",  "targets": 2 },
                            { "width": "15%", "targets": 3 },
                            { "width": "5%",  "targets": 4 },
                            { "width": "5%",  "targets": 5 }]
        });
}

function setStarsRating() {
        var countOwnLvl = 0;
        var countPlaylist = 0;

        $("#userHome-playlist-table > tbody > tr > .star").each(function () {
                var div = $("<div>", {id: countPlaylist});
                div.raty({
                        score: parseInt($(this).text()),
                        readOnly : true,
                });
                $(this).empty().append(div);
                countPlaylist ++;
        });

        $("#userHome-ownLevels-table > tbody > tr > .star").each(function () {
                var div = $("<div>", {id: countOwnLvl});
                div.raty({
                        score: parseInt($(this).text()),
                        readOnly : true,
                });
                $(this).empty().append(div);
                countOwnLvl ++;
        });
}
```

```
function setClearStage() {
	$("#userHome-playlist-table > tbody > tr > .cleared").each(function () {
		var cleared = parseInt($(this).text());
		if(cleared == 0) {
		 var img = $("<img>", {src: "/PFM/resources/img/User_Home/red.png"});
		 $(this).empty().append(img);
		}
		if(cleared == 1) {
		 var img = $("<img>", {src: "/PFM/resources/img/User_Home/green.png"});
		 $(this).empty().append(img);
		}
	});
}
```

13.2.1.2 Level_Editor_Demolition.js

Este es el código JavaScript de uno de los editores de niveles.

```
var pieces = [];

function init() {
	createGrid(30);

	$("#demolitionLevelEditor-eel-group2").hide();
	showHideEditorElementsList();

	$("#demolitionLevelEditor-deleteElements-box").css("background-image", "url('resources/img/Level_Editor/rubbish-gray.png')");

	setDraggableElements();
	setDeleteElementsBox();

	centerElementEntry();
	drawLimitLine();
	setCanonPos();

	$("#demolitionLevelEditor-ldf-levelLimit").change(function() {
		drawLimitLine();
	});
```

```
        $("#demolitionLevelEditor-ldf-levelCanonPos").change(function() {
                setCanonPos();
        });

        $('#demolitionLevelEditor-ldf-levelName').change(function() {
                $('#demolitionLevelEditor-ldf-submit').attr("disabled", false);
        });

}

function createGrid(size) {

        var gridWidth = 900,
            gridHeight = 450;
        var ratioW = Math.floor(gridWidth / size),
            ratioH = Math.floor(gridHeight / size);
        var parent = $('<div />', {
                class : 'grid',
                width : ratioW * size,
                height : ratioH * size
        }).addClass('grid').appendTo('#demolitionLevelEditor-grid-box');
        for (var i = 0; i < ratioH; i++) {
                for (var p = 0; p < ratioW; p++) {
                        $('<div />', {
                                class : 'cell',
                                width : size - 1,
                                height : size - 1
                        }).addClass('cell').droppable({
                                accept : '.draggable, .editedDraggable'
                        }).appendTo(parent);
                }
        }
}

function showHideEditorElementsList() {
        $("#demolitionLevelEditor-eel-UpButton").click(function() {
                 $("#demolitionLevelEditor-eel-group1").show();
                 $("#demolitionLevelEditor-eel-group2").hide();
        });
        $("#demolitionLevelEditor-eel-DownButton").click(function() {
                 $("#demolitionLevelEditor-eel-group1").hide();
                 $("#demolitionLevelEditor-eel-group2").show();
        });
}
function setDraggableElements() {
        var isRevert = false;
```

```
$(".draggable").draggable({
        revert : function(valid) {
                if (valid) {
                        isRevert = true;
                } else {
                        isRevert = false;
                }
        },
        drag : function() { },
        snap : ".cell",
        helper : 'clone',
        stop : function(event, ui) {
                if (isRevert) {
                        $(ui.helper).clone(true).removeClass('box ui-draggable ui-draggable-dragging')
                        .addClass('clon').draggable({
                                revert : 'invalid',
                                snap : ".cell",
                                stop : function(event, ui) {
                                        $("#demolitionLevelEditor-ldf-listEditionElements").attr("value", getLevelElementsData());
                                }
                        }).appendTo('#demolitionLevelEditor-clonedElementList');

                        $("#demolitionLevelEditor-ldf-listEditionElements").attr("value", getLevelElementsData());
                }
        }
});

$(".editedDraggable").draggable({
        revert : function(valid) {
                if (valid) {
                        isRevert = true;
                } else {
                        isRevert = false;
                }
        },
        snap : ".cell",
        stop : function(event, ui) {
                if (isRevert) {
                        $(ui.helper).removeClass('box ui-draggable ui-draggable-dragging')
                        .draggable({
                                revert : 'invalid',
                                snap : ".cell",
                                stop : function(event, ui) {
                                        $("#demolitionLevelEditor-ldf-listEditionElements").attr("value", getLevelElementsData());
                                }
                        }).appendTo('#demolitionLevelEditor-clonedElementList');
```

```
                                $("#demolitionLevelEditor-ldf-listEditionElements").attr("value", getLevelElementsData());
                        }
                }
        });

}

function setDeleteElementsBox() {
        $("#demolitionLevelEditor-deleteElements-box").droppable({
                accept : '.draggable, .editedDraggable',
                drop: function(event, ui) {
                        var element = ui.draggable.css('position', '');
                        $(this).append(element);
                        $(ui.draggable).fadeOut(0);

                        $("#demolitionLevelEditor-deleteElements-box").css("background-image", "url('resources/img/Level_Editor/rubbish-gray.png')");
                        $("#demolitionLevelEditor-deleteElements-box").css("box-shadow", "0 0 0 0 rgb(81, 45, 168)");

                        $("#demolitionLevelEditor-ldf-listEditionElements").attr("value", getLevelElementsData());
                },
                over: function(event, ui) {
                 $("#demolitionLevelEditor-deleteElements-box").css("background-image", "url('resources/img/Level_Editor/rubbish.png')");
                 $("#demolitionLevelEditor-deleteElements-box").css("box-shadow", "0 0 5px 1px rgb(81, 45, 168)");
                },
                out: function(event, ui) {
                        $("#demolitionLevelEditor-deleteElements-box").css("background-image", "url('resources/img/Level_Editor/rubbish-gray.png')");
                        $("#demolitionLevelEditor-deleteElements-box").css("box-shadow", "0 0 0 0 rgb(81, 45, 168)");
                }
        });
}
```

```
function getLevelElementsData() {
	var levelElements = "";

	var offsetGrid = $("#demolitionLevelEditor-grid-box").offset();
	var clonedElementList = $("#demolitionLevelEditor-clonedElementList");

	clonedElementList.children().each(function() {
		var top = parseInt($(this).css("top").substring(0, $(this).css("top").length - 2));
		var left = parseInt($(this).css("left").substring(0, $(this).css("left").length - 2));
		var data = $(this).data("type");

		top = Math.trunc(((top - offsetGrid.top) / 30) + 1);
		left = Math.trunc(((left - offsetGrid.left) / 30) + 1);

		var levelElement = "@" + left + "," + top + "," + data;
		levelElements += levelElement;
	});

	return levelElements;
}

function centerElementEntry() {
	$("#demolitionLevelEditor-eel-group1").children().each(function() {
		var width = parseInt($(this).children().css("width").substring(0, $(this).css("top").length - 1));
		$(this).attr("style", "margin-left: " + Math.abs(125 - (width/2)) + "px");
	});
	$("#demolitionLevelEditor-eel-group2").children().each(function() {
		var width = parseInt($(this).children().css("width").substring(0, $(this).css("top").length - 1));
		$(this).attr("style", "margin-left: " + Math.abs(125 - (width/2)) + "px");
	});
}

function drawLimitLine() {
	var offsetGrid = $("#demolitionLevelEditor-grid-box").offset();
	var position = parseInt($("#demolitionLevelEditor-ldf-levelLimit").val());

	$("#demolitionLevelEditor-limitLine").css("top", offsetGrid.top + (position * 30));
	$("#demolitionLevelEditor-limitLine").css("left", offsetGrid.left);
}
```

```
function setCanonPos() {
	var offsetGrid = $("#demolitionLevelEditor-grid-box").offset();
	var position = parseInt($("#demolitionLevelEditor-ldf-levelCanonPos").val());

	position = position - 1;
	$("#demolitionLevelEditor-canon").css("top", offsetGrid.top + (position * 30));
	$("#demolitionLevelEditor-canon").css("left", offsetGrid.left + 30);
}
```

13.2.1.3 Play_Demolition_Level.jsp

Fichero que contiene código HTML y JavaScript de uno de los juegos presentados.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<html>

	<head>
		<meta charset = "utf-8">
		<title>PFM - Games Web - Demolition Game</title>

		<script type = "text/javascript" src = "<c:url value="/resources/lib/Game/Box2dWeb-2.1.a.3.js"/>"> </script>
		<script type = "text/javascript" src = "<c:url value="/resources/lib/jquery-1.7.1.min.js"/>"></script>

		<script type = "text/javascript" src = "<c:url value="/resources/lib/jquery.raty.js"/>"></script>
		<link rel = "stylesheet" href = "<c:url value="/resources/lib/jquery.raty.css" />">

		<script type = "text/javascript" src = "<c:url value="/resources/js/Games/Demolition_Game/scenario/CanvasUtils.js"/>"> </script>
		<script type = "text/javascript" src = "<c:url value="/resources/js/Games/Demolition_Game/scenario/Stage.js"/>"> </script>

		<script type = "text/javascript" src = "<c:url value="/resources/js/Games/Demolition_Game/DemolitionFunctions.js"/>"> </script>

		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Canon.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Bullet.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Line_1x3.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Line_1x5.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Line_3x1.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Line_5x1.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Cube_1x1.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Cube_2x2.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/Circle_2x2.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/TriangleA.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/TriangleB.js"/>"> </script>
		<script src = "<c:url value="/resources/js/Games/Demolition_Game/elements/TriangleC.js"/>"> </script>

		<link rel = "stylesheet" href = "<c:url value="/resources/css/global.css" />">
		<link rel = "stylesheet" href = "<c:url value="/resources/css/User_Home/play_demolition_level.css" />">
	</head>
```

```
<body>

        <header>
                <div id = "playGame-logo-box">
                        <img alt = "Logo" src = "<c:url value="/resources/img/logo.png" />">
                        <h1> GAME & SHARE </h1>
                        <h2> DEMOLITION GAME </h2>
                </div>
                <div id = "playGame-menu">
                        <ul>
                                <li> <a class = "menu-selected"> <span class = "menu-text"> HOME </span> </a> </li>
                                <li> <a href = "level_editor_selection"> <span class = "menu-text"> LEVEL CREATOR </span> </a> </li>
                                <li> <a href = "comunity"> <span class = "menu-text"> COMUNITY </span> </a> </li>
                                <li> <a href = "about"> <span class = "menu-text"> ABOUT </span> </a> </li>
                        </ul>
                </div>
                <div id = "playGame-user-box">
                        <p> <c:out value = "${user.getUsername()}"></c:out> </p>
                        <a href = "logout"> Logout </a>
                </div>
        </header>

        <div class = "main-box">

                <section id = "demolitionGame-leftSection">
                        <!-- Tamaño del canvas (30m x 15m) -->
                        <canvas id = "canvas" width = '900' height = '450'> </canvas>
                </section>

                <section id = "demolitionGame-rightSection">
                        <div id = "dg-controls-box">
                                <h3> GAME CONTROLS </h3>
                                <div id = "dg-controls-canonUp">
                                        <img alt = "W Key" src = "<c:url value="/resources/img/User_Home/w.png"/>">
                                        <p> Rotate Canon Up </p>
                                </div>
                                <div id = "dg-controls-canonDown">
                                        <img alt = "S Key" src = "<c:url value="/resources/img/User_Home/s.png"/>">
                                        <p> Rotate Canon Down </p>
                                </div>
                                <div id = "dg-controls-canonShot">
                                        <img alt = "Space Bar Key" src = "<c:url value="/resources/img/User_Home/spaceBar.png"/>">
                                        <p> Shot </p>
                                </div>
                        </div>
                        <p id = "ratingLabel"> Rating </p>
```

```
                <div id = "starsRating"></div>

                <form class = "demolitionLevel-form" action = "replayLevelDemolition" method = "POST">
                        <input id = "dlf-playAgain" type = "submit" value = "Play Again">
                </form>

                <form class = "demolitionLevel-form" action = "backToUserHome" method = "POST">
                        <input id = "dlf-levelName" type = "hidden" name = "levelName" value = "<c:out value="${levelName}"/>">
                        <input id = "dlf-levelCleared" type = "hidden" name = "levelCleared" value = "">
                        <input id = "dlf-levelPlayerScore" type = "hidden" name = "levelPlayerScore" value = "">
                        <input id = "dlf-levelRating" type = "hidden" name = "levelRating" value = "1">
                        <input id = "dlf-submit" type = "submit" value = "Back to User Home">
                </form>
        </section>
</div>

<script>
        /*      Canvas
        ---------------------------------------------------------------------*/
        var canvas = document.getElementById("canvas");
        var ctx = canvas.getContext("2d");
        var     b2Vec2 = Box2D.Common.Math.b2Vec2,
                b2BodyDef = Box2D.Dynamics.b2BodyDef,
                b2Body = Box2D.Dynamics.b2Body,
                b2FixtureDef = Box2D.Dynamics.b2FixtureDef,
                b2Fixture = Box2D.Dynamics.b2Fixture,
                b2World = Box2D.Dynamics.b2World,
                b2MassData = Box2D.Collision.Shapes.b2MassData,
                b2PolygonShape = Box2D.Collision.Shapes.b2PolygonShape,
                b2CircleShape = Box2D.Collision.Shapes.b2CircleShape,
                b2DebugDraw = Box2D.Dynamics.b2DebugDraw;

        /* Variables Globales del juego
        ---------------------------------------------------------------------*/
        var SCALE = 30,
                FPS = 60,
                timestep = 1;
        var world, debugDraw, stage;
        var walls = [],
                canon,
                structureElements = [],
                bullets = [];
```

```
/* Construccion del escenario de juego
-------------------------------------------------------------------*/
// Creacion del mundo con las fisicas
setupWorldPhysics();

// Parametros leidos del JSON del nivel
var totalBullets = bulletCount = ${levelNumBullets};
var winLimit = ${levelLimitLine};
var canonYPos = ${levelCanonPos};
var elementsList = "${levelElements}";

// --- Creacion del espacio de juego ---
// Delimitacion de la pantalla
walls = createStage(canvas.width, canvas.height);

// Cañon
canonPos = {x:(2 * SCALE) , y:(canonYPos * SCALE)};
canon = createCanon(world, canonPos.x, canonPos.y);
var canonCenter = canon.GetPosition();
var canonVector = canon.GetPosition();

// Elementos a derribar
structureElements = buildLevel(elementsList, structureElements, world, SCALE);

/* Pintando en el canvas
-------------------------------------------------------------------*/
setupDebugDraw(ctx);

/* Funcion de actualizacion del juego
-------------------------------------------------------------------*/
setInterval(function()
{
	world.Step(timestep / FPS, 10, 10);  // Avanzar 1 frame la escena
	world.DrawDebugData();
	world.ClearForces();

	// --- Pintamos cosas en el canvas ---
	//drawGrid(ctx, canvasWidth, canvasHeight, SCALE);
	printMessage(ctx, "Proyectiles Restantes: " + bulletCount, 15, 35);
	drawLimitLine(ctx, winLimit * SCALE, canvas.width);

	for(var i = 0; i < bullets.length; i++)
	{
		destroyBulletIsSleep(world, bullets, bullets[i]);
	}
```

```
                for(var i = 0; i < structureElements.length; i++)
                {
                        structureElements[i].isUnderLimitLine = checkPositionUnderLimit(structureElements[i], winLimit);
                }

                var playerScore = calculateScore(structureElements, bullets, totalBullets);
                printMessage(ctx, "Puntuacion: " + playerScore, 600, 35);

                if(checkStageClear(structureElements))
                {
                        timestep = 0;
                        changeCanvasColor(ctx, 900, 600);
                        printMessage(ctx, "Stage Clear !!! -- Score: " + playerScore, 270, 250);

                        $("#dlf-levelCleared").attr("value", 1);
                        $("#dlf-levelPlayerScore").attr("value", playerScore);
                        $(".demolitionLevel-form").show();
                }
                else if(bulletCount == 0)
                {
                        if(bullets[0].GetUserData() == "Death" && bullets[1].GetUserData() == "Death" &&
                           bullets[2].GetUserData() == "Death")
                        {
                                timestep = 0;
                                changeCanvasColor(ctx, 900, 600);
                                printMessage(ctx, "Game Over -- Score: " + playerScore, 270, 250);
                                $("#dlf-levelCleared").attr("value", 0);
                                $("#dlf-levelPlayerScore").attr("value", playerScore);
                                $(".demolitionLevel-form").show();
                        }
                }

        },1000/FPS);

        /* Añadir funciones a los elementos del juego
        ------------------------------------------------------------------*/
        // --- Detectamos la tecla pulsada ---
        function callkeydownhandler(evnt)
        {
                var ev = (evnt) ? evnt : event;
                var code = (ev.which) ? ev.which : event.keyCode;
                canonVector = rotateCanon(canon, code);

                if(code == 32 && bulletCount > 0)
                {
                        // --- Guardamos la posicion de donde sale la bola ---
```

```
                                var radius = 15;
                                var bulletX = canonVector.x,
                                        bulletY = canonVector.y;
                                var bulletForce = 450;

                                var bullet = createBullet(world, bulletX, bulletY, radius);
                                applyForceToBullet(bullet, canon.GetAngle(), bulletForce);
                                bullets.push(bullet);
                                bulletCount--;
                        }
                }

                if (window.document.addEventListener)
                {
                        window.document.addEventListener("keydown", callkeydownhandler, false);
                }
                else
                {
                        window.document.attachEvent("onkeydown", callkeydownhandler);
                }

                $(".demolitionLevel-form").hide();

                $("#starsRating").raty({
                        score: 1,
                        click: function(score, evt) {
                                $("#dlf-levelRating").attr("value", score);
                }
                });
        </script>
    </body>
</html>
```

13.2.2 Paquete Controller

13.2.2.1 LoginController.java

```
@Controller
public class LoginController {

	@Autowired
	private UserDataManagerService udms;

	@Autowired
	private LevelDataManagerService ldms;

	@Autowired
	private PlaylistDataManagerService pldms;

	@Autowired
	private RatinLevelDataManagerServices rldms;

	@RequestMapping(value = "/login", method = RequestMethod.POST)
	public String loginSubmit(HttpSession session,
	                                        @RequestParam("username") String username,
	                                        @RequestParam("password") String password) {

		String nextPage = "";

		try {

			User user = udms.getUser(username, password);
			if(user != null) {
				Playlist playlist = pldms.getAllLevelsByUser(username);
				Playlist ownLevels = ldms.getAllLevelsByUser(username);

				for (PlaylistDMLLevelEntry dle : playlist.getDlevels()) {
					dle.getDl().setRating(rldms.getCalculatedLevelRating(dle.getDl().getLevelName()));
				}

				for (PlaylistAKNLevelEntry ale : playlist.getAlevels()) {
					ale.getAl().setRating(rldms.getCalculatedLevelRating(ale.getAl().getLevelName()));
				}

				for (PlaylistDMLLevelEntry dle : ownLevels.getDlevels()) {
```

```
                                dle.getDl().setRating(rldms.getCalculatedLevelRating(dle.getDl().getLevelName()));
                        }

                        for (PlaylistAKNLevelEntry ale : ownLevels.getAlevels()) {
                                ale.getAl().setRating(rldms.getCalculatedLevelRating(ale.getAl().getLevelName()));
                        }

                        session.setAttribute("user", user);
                        session.setAttribute("playlist", playlist);
                        session.setAttribute("ownLevels", ownLevels);
                        nextPage = "User_Home/user_home";
                } else {
                        session.setAttribute("errorMessage", "The username or password are incorrect.");
                        nextPage = "error";
                }

            } catch (Exception e) {
                e.printStackTrace();
            }

            return nextPage;
        }

        @RequestMapping(value = "/register", method = RequestMethod.GET)
        public String goToRegister(HttpServletRequest request) {
            return "register";
        }

}
```

13.2.2.2 PlayLevelController.java

```
@Controller
public class PlayLevelController {

	@Autowired
	private LevelDataManagerService ldms;

	@Autowired
	private PlaylistDataManagerService pldms;

	@Autowired
	private RatinLevelDataManagerServices rldms;

	@RequestMapping(value = "/playLevelDemolition", method = RequestMethod.POST)
	public String goToPlayLevelDemolition(HttpSession session,
		@RequestParam("levelName") String levelName,
		@RequestParam("levelPath") String levelPath) {

		try {

			DemolitionLevel dl = JSONFilesUtils.readDemolitionLevelJSON(levelPath);

			String levelElementsString = "";
			for (LevelElement le : dl.getLevelElements()) {
				levelElementsString += le.getType() + ";" + le.getX() + ";" + le.getY() + "@";
			}

			levelElementsString = levelElementsString.substring(0, levelElementsString.length() - 1);
			session.setAttribute("levelElements", levelElementsString);
			session.setAttribute("levelCanonPos", dl.getCanonPos());
			session.setAttribute("levelLimitLine", dl.getLimitLine());
			session.setAttribute("levelNumBullets", dl.getNumBullets());
			session.setAttribute("levelName", levelName);
			session.setAttribute("levelPath", levelPath);

		} catch (IOException e) {
			e.printStackTrace();
		} catch (ParseException e) {
			e.printStackTrace();
		}

		return "User_Home/play_Demolition_Level";
	}
```

```
	@RequestMapping(value = "/playLevelArkanoid", method = RequestMethod.POST)
	public String goToPlayLevelArkanoid(HttpSession session,
		@RequestParam("levelName") String levelName,
		@RequestParam("levelPath") String levelPath) {

		try {

			ArkanoidLevel al = JSONFilesUtils.readArkanoidLevelJSON(levelPath);

			String levelElementsString = "";
			for (LevelElement le : al.getLevelElements()) {
				levelElementsString += le.getType() + ";" + le.getX() + ";" + le.getY() + "@";
			}

			levelElementsString = levelElementsString.substring(0, levelElementsString.length() - 1);
			session.setAttribute("levelElements", levelElementsString);
			session.setAttribute("levelName", levelName);
			session.setAttribute("levelPath", levelPath);

		} catch (IOException e) {
			e.printStackTrace();
		} catch (ParseException e) {
			e.printStackTrace();
		}

		return "User_Home/play_Arkanoid_Level";
	}

	@RequestMapping(value = "/backToUserHome", method = RequestMethod.POST)
	public String backToUserHome(HttpSession session,
		HttpServletRequest request,
		@RequestParam("levelName") String levelName,
		@RequestParam("levelCleared") int levelCleared,
		@RequestParam("levelPlayerScore") int levelPlayerScore,
		@RequestParam("levelRating") int levelRating) {

		if(levelCleared == 1) {
			levelCleared = 1;
		}

		User user = (User) session.getAttribute("user");
		try {
			rldms.insertRatingLevel(levelName, user.getUsername(), levelRating);
		} catch (Exception e) {

			manageScoreUpdate(user, levelName, levelCleared, levelPlayerScore);
```

```
        }

        Playlist playlist = pldms.getAllLevelsByUser(user.getUsername());
        Playlist ownLevels = ldms.getAllLevelsByUser(user.getUsername());

        try {

                for (PlaylistDMLLevelEntry dle : playlist.getDlevels()) {
                        dle.getDl().setRating(rldms.getCalculatedLevelRating(dle.getDl().getLevelName()));
                }

                for (PlaylistAKNLevelEntry ale : playlist.getAlevels()) {
                        ale.getAl().setRating(rldms.getCalculatedLevelRating(ale.getAl().getLevelName()));
                }

                for (PlaylistDMLLevelEntry dle : ownLevels.getDlevels()) {
                        dle.getDl().setRating(rldms.getCalculatedLevelRating(dle.getDl().getLevelName()));
                }

                for (PlaylistAKNLevelEntry ale : ownLevels.getAlevels()) {
                        ale.getAl().setRating(rldms.getCalculatedLevelRating(ale.getAl().getLevelName()));
                }

                session.setAttribute("ownLevels", ownLevels);
                session.setAttribute("playlist", playlist);

        } catch (Exception e) {
                e.printStackTrace();
        }

        return "User_Home/user_home";
}

@RequestMapping(value = "/replayLevelDemolition", method = RequestMethod.POST)
public String replayLevelDemolition(HttpSession session) {

        try {

                String path = (String) session.getAttribute("levelPath");
                DemolitionLevel dl = JSONFilesUtils.readDemolitionLevelJSON(path);

                String levelElementsString = "";
                for (LevelElement le : dl.getLevelElements()) {
                        levelElementsString += le.getType() + ";" + le.getX() + ";" + le.getY() + "@";
                }
```

```
                levelElementsString = levelElementsString.substring(0, levelElementsString.length() - 1);
                session.setAttribute("levelElements", levelElementsString);
                session.setAttribute("levelCanonPos", dl.getCanonPos());
                session.setAttribute("levelLimitLine", dl.getLimitLine());
                session.setAttribute("levelName", session.getAttribute("levelName"));

        } catch (IOException e) {
                e.printStackTrace();
        } catch (ParseException e) {
                e.printStackTrace();
        }

        return "User_Home/play_Demolition_Level";
}

@RequestMapping(value = "/replayLevelArkanoid", method = RequestMethod.POST)
public String replayLevelArkanoid(HttpSession session) {

        try {

                String path = (String) session.getAttribute("levelPath");
                ArkanoidLevel al = JSONFilesUtils.readArkanoidLevelJSON(path);

                String levelElementsString = "";
                for (LevelElement le : al.getLevelElements()) {
                        levelElementsString += le.getType() + ";" + le.getX() + ";" + le.getY() + "@";
                }

                levelElementsString = levelElementsString.substring(0, levelElementsString.length() - 1);
                session.setAttribute("levelElements", levelElementsString);
                session.setAttribute("levelName", session.getAttribute("levelName"));

        } catch (IOException e) {
                e.printStackTrace();
        } catch (ParseException e) {
                e.printStackTrace();
        }

        return "User_Home/play_Arkanoid_Level";
}


private void manageScoreUpdate(User user, String levelName, int levelCleared, int levelPlayerScore) {
        Map<String, Integer> scoreClear;
```

```
                try {

                        scoreClear = pldms.getLevelScoreAndCleared(user.getUsername(), levelName);
                        if(scoreClear.get("cleared") == 1 && levelCleared == 1) {
                                if(scoreClear.get("score") < levelPlayerScore) {
                                        pldms.updateLevelScoreAndCleared(user.getUsername(), levelName, levelCleared, levelPlayerScore);
                                }
                        }

                        if(scoreClear.get("cleared") == 0 && levelCleared == 1) {
                                pldms.updateLevelScoreAndCleared(user.getUsername(), levelName, levelCleared, levelPlayerScore);
                        }

                        if(scoreClear.get("cleared") == 0 && levelCleared == 0) {
                                if(scoreClear.get("score") < levelPlayerScore) {
                                        pldms.updateLevelScoreAndCleared(user.getUsername(), levelName, levelCleared, levelPlayerScore);
                                }
                        }

                } catch (Exception e1) {
                        e1.printStackTrace();
                }

        }
}
```

13.2.2.3 CommunityController.java

```
@Controller
public class ComunityController {

	@Autowired
	private LevelDataManagerService ldms;

	@Autowired
	private PlaylistDataManagerService pldms;

	@Autowired
	private RatinLevelDataManagerServices rldms;

	@RequestMapping(value = "/addLevelToHome_Demolition", method = RequestMethod.POST)
	public String addDemolitionLevelToUserHome(HttpSession session,
		@RequestParam("levelName") String levelName) {

		User user = (User) session.getAttribute("user");

		try {

			pldms.insertLevelInPlaylist(user.getUsername(), levelName);

		} catch (SQLException e) {
			e.printStackTrace();
		}

		Playlist playlist = pldms.getAllLevelsByUser(user.getUsername());
		Playlist ownLevels = ldms.getAllLevelsByUser(user.getUsername());

		try {

			for (PlaylistDMLLevelEntry dle : playlist.getDlevels()) {
				dle.getDl().setRating(rldms.getCalculatedLevelRating(dle.getDl().getLevelName()));
			}

			for (PlaylistAKNLevelEntry ale : playlist.getAlevels()) {
				ale.getAl().setRating(rldms.getCalculatedLevelRating(ale.getAl().getLevelName()));
			}

			for (PlaylistDMLLevelEntry dle : ownLevels.getDlevels()) {
				dle.getDl().setRating(rldms.getCalculatedLevelRating(dle.getDl().getLevelName()));
			}
```

```
                for (PlaylistAKNLevelEntry ale : ownLevels.getAlevels()) {
                        ale.getAl().setRating(rldms.getCalculatedLevelRating(ale.getAl().getLevelName()));
                }

                session.setAttribute("playlist", playlist);
                session.setAttribute("ownLevels", ownLevels);

        } catch (Exception e) {
                e.printStackTrace();
        }

        return "User_Home/user_home";
}

@RequestMapping(value = "/addLevelToHome_Arkanoid", method = RequestMethod.POST)
public String addArkanoidLevelToUserHome(HttpSession session,
                @RequestParam("levelName") String levelName) {

        User user = (User) session.getAttribute("user");

        try {
                pldms.insertLevelInPlaylist(user.getUsername(), levelName);
        } catch (SQLException e) {
                e.printStackTrace();
        }

        Playlist playlist = pldms.getAllLevelsByUser(user.getUsername());
        Playlist ownLevels = ldms.getAllLevelsByUser(user.getUsername());

        try {

                for (PlaylistDMLLevelEntry dle : playlist.getDlevels()) {
                        dle.getDl().setRating(rldms.getCalculatedLevelRating(dle.getDl().getLevelName()));
                }

                for (PlaylistAKNLevelEntry ale : playlist.getAlevels()) {
                        ale.getAl().setRating(rldms.getCalculatedLevelRating(ale.getAl().getLevelName()));
                }

                for (PlaylistDMLLevelEntry dle : ownLevels.getDlevels()) {
                        dle.getDl().setRating(rldms.getCalculatedLevelRating(dle.getDl().getLevelName()));
                }

                for (PlaylistAKNLevelEntry ale : ownLevels.getAlevels()) {
                        ale.getAl().setRating(rldms.getCalculatedLevelRating(ale.getAl().getLevelName()));
                }
```

```
                        session.setAttribute("playlist", playlist);
                        session.setAttribute("ownLevels", ownLevels);

                } catch (Exception e) {
                        e.printStackTrace();
                }

                return "User_Home/user_home";
        }
}
```

13.2.3 Paquete Business

13.2.3.1 UserDataManager.java

```
public class UserDataManager implements UserDataManagerService{

        @Autowired
        private UsersDAO uDAO;

        @Override
        public boolean existUser(String username) throws Exception {
                return uDAO.existUser(username);
        }
        @Override
        public User getUser(String username, String password) throws Exception {
                return uDAO.getUser(username, password);
        }
        @Override
        public void insertUser(User user) throws Exception {
                uDAO.insertUser(user);
        }
        public void setuDAO(UsersDAO uDAO) {
                this.uDAO = uDAO;
        }
}
```

13.2.3.2 LevelDataManager.java

```
public class LevelDataManager implements LevelDataManagerService {

	@Autowired
	private GameLevelsDAO glDAO;

	private DemolitionLevel dl;
	private ArkanoidLevel al;

	@Override
	public DemolitionLevel createDemolitionLevel(String levelName,
		String levelDescription, String levelElementsList, int levelLimit,
		int levelCanonPos, int levelNumBullets, String author, String levelType) {

		dl = new DemolitionLevel();
		dl.setLevelName(levelName);
		dl.setLevelDescription(levelDescription);
		dl.setLevelElements(makeLevelElementsList(levelElementsList));
		dl.setLimitLine(levelLimit);
		dl.setCanonPos(levelCanonPos);
		dl.setNumBullets(levelNumBullets);
		dl.setLevelType(levelType);
		dl.setAuthor(author);

		return dl;
	}

	@Override
	public ArkanoidLevel createArkanoidLevel(String levelName,
		String levelDescription, String levelElementsList, String author,
		String levelType) {

		al = new ArkanoidLevel();
		al.setLevelName(levelName);
		al.setLevelDescription(levelDescription);
		al.setLevelElements(makeLevelElementsList(levelElementsList));
		al.setLevelType(levelType);
		al.setAuthor(author);

		return al;
	}
```

```
@Override
public void insertDemolitionLevel(DemolitionLevel dl) {
        glDAO.insertDemolitionLevel(dl);
}

@Override
public void insertArkanoidLevel(ArkanoidLevel al) {
        glDAO.insertArkanoidLevel(al);
}

@Override
public List<DemolitionLevel> getAllDemolitionLevels() {
        return glDAO.getAllDemolitionLevels();
}

@Override
public List<ArkanoidLevel> getAllArkanoidLevels() {
        return glDAO.getAllArkanoidLevels();
}

@Override
public Playlist getAllLevelsByUser(String username) {
        return glDAO.getAllLevelsByUser(username);
}

@Override
public boolean existLevel(String levelName) throws Exception {
        return glDAO.existLevel(levelName);
}

@Override
public void updateDemolitionLevel(DemolitionLevel dl) {
        glDAO.updateLevel(dl.getLevelDescription(), dl.getLevelName());
}

@Override
public void updateArkanoidLevel(ArkanoidLevel al) {
        glDAO.updateLevel(al.getLevelDescription(), al.getLevelName());
}
```

```
	private List<LevelElement> makeLevelElementsList(String levelElementsListString) {
		List<LevelElement> elementsList = new ArrayList<LevelElement>();
		String[] elementEntrys = levelElementsListString.split("@");
		for (int i = 1; i < elementEntrys.length; i++) {
			String[] elementParts = elementEntrys[i].split(",");
			LevelElement le = new LevelElement();
			le.setX(Integer.parseInt(elementParts[0]));
			le.setY(Integer.parseInt(elementParts[1]));
			le.setType(elementParts[2]);
			elementsList.add(le);
		}
		return elementsList;
	}

	public void setGlDAO(GameLevelsDAO glDAO) {
		this.glDAO = glDAO;
	}

}
```

13.2.4 Paquete Persistence

13.2.4.1 UsersDAO.java

```
public class UsersDAO {

	@Autowired
	private Jdbc jdbc;

	private String existUser;
	private String getUser;
	private String insertUser;

	public void insertUser(User user) throws Exception {
		PreparedStatement ps = null;
		Connection con = null;

		try {

			con = jdbc.getConnection();
			ps = con.prepareStatement(insertUser);
			ps.setString(1, user.getUsername());
			ps.setString(2, user.getPassword());
			ps.execute();

		} finally {
			try {
				ps.close();
				con.close();
			}
			catch (Exception e) {
				e.printStackTrace();
			}
		}
	}
```

```
public boolean existUser(String username) throws Exception {
        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        boolean exist = false;

        try {

                con = jdbc.getConnection();
                ps = con.prepareStatement(existUser);
                ps.setString(1, username);
                rs = ps.executeQuery();

                if (!rs.next()) {
                        exist = false;
                } else {
                        do {

                                if(rs.getString("username").equals(username)) {
                                        exist = true;
                                }

                        } while (rs.next());
                }

        } finally {
                try {
                        ps.close();
                        con.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        return exist;
}

public User getUser(String userName, String password) throws Exception {
        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        User user = null;

        try {
```

```
                con = jdbc.getConnection();
                ps = con.prepareStatement(getUser);
                ps.setString(1, userName);
                ps.setString(2, password);
                rs = ps.executeQuery();

                if (!rs.next()) {
                        System.out.println("No records found");
                } else {
                        do {

                                if(rs.getString("username").equals(userName) && rs.getString("password").equals(password)) {
                                        user = new User();
                                        user.setUsername(userName);
                                        user.setPassword(password);
                                }

                        } while (rs.next());
                }

        } finally {
                try {
                        ps.close();
                        con.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        return user;
}

public void setJdbc(Jdbc jdbc) {
        this.jdbc = jdbc;
}

public void setGetUser(String getUser) {
        this.getUser = getUser;
}
```

```
        public void setExistUser(String existUser) {
                this.existUser = existUser;
        }

        public void setInsertUser(String insertUser) {
                this.insertUser = insertUser;
        }

}
```

13.2.4.2 GameLevelsDAO.java

```
public class GameLevelsDAO {

        @Autowired
        private Jdbc jdbc;

        private String insertLevel;
        private String getAllDemolitionLevels;
        private String getAllArkanoidLevels;
        private String getAllLevelsByUser;
        private String existLevel;
        private String updateLevel;

        public void insertDemolitionLevel(DemolitionLevel dl) {
                PreparedStatement ps = null;
                Connection con = null;

                try {

                        con = jdbc.getConnection();
                        ps = con.prepareStatement(insertLevel);
                        ps.setString(1, dl.getLevelName());
                        ps.setString(2, dl.getLevelDescription());
                        ps.setString(3, dl.getLevelType());
                        ps.setString(4, dl.getAuthor());
                        ps.setString(5, dl.getPath());
                        ps.execute();

                } catch (SQLException e) {
                        e.printStackTrace();
                } finally {
                        try {
```

```
                        ps.close();
                        con.close();
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
        }
}

public void insertArkanoidLevel(ArkanoidLevel al) {
        PreparedStatement ps = null;
        Connection con = null;

        try {

                con = jdbc.getConnection();
                ps = con.prepareStatement(insertLevel);
                ps.setString(1, al.getLevelName());
                ps.setString(2, al.getLevelDescription());
                ps.setString(3, al.getLevelType());
                ps.setString(4, al.getAuthor());
                ps.setString(5, al.getPath());
                ps.execute();

        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                try {
                        ps.close();
                        con.close();
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
        }
}

public List<DemolitionLevel> getAllDemolitionLevels() {
        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        List<DemolitionLevel> dlList = new ArrayList<DemolitionLevel>();
        try {

                con = jdbc.getConnection();
```

```
                ps = con.prepareStatement(getAllDemolitionLevels);
                rs = ps.executeQuery();

                while(rs.next()) {
                        DemolitionLevel dl = new DemolitionLevel();
                        dl.setLevelName(rs.getString("levelName"));
                        dl.setLevelDescription(rs.getString("description"));
                        dl.setLevelType(rs.getString("type"));
                        dl.setAuthor(rs.getString("author"));
                        dl.setPath(rs.getString("filePath"));
                        dlList.add(dl);
                }

        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                try {
                        ps.close();
                        con.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        return dlList;
}

public List<ArkanoidLevel> getAllArkanoidLevels() {
        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        List<ArkanoidLevel> alList = new ArrayList<ArkanoidLevel>();
        try {

                con = jdbc.getConnection();
                ps = con.prepareStatement(getAllArkanoidLevels);
                rs = ps.executeQuery();

                while(rs.next()) {
                        ArkanoidLevel al = new ArkanoidLevel();
                        al.setLevelName(rs.getString("levelName"));
                        al.setLevelDescription(rs.getString("description"));
                        al.setLevelType(rs.getString("type"));
                        al.setAuthor(rs.getString("author"));
                        al.setPath(rs.getString("filePath"));
```

```
                        alList.add(al);
                }

        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                try {
                        ps.close();
                        con.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        return alList;
}

public Playlist getAllLevelsByUser(String username) {
        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        Playlist playlist = new Playlist();
        try {

                con = jdbc.getConnection();
                ps = con.prepareStatement(getAllLevelsByUser);
                ps.setString(1, username);
                rs = ps.executeQuery();

                while(rs.next()) {

                        if(rs.getString("type").equals("DML")) {
                                DemolitionLevel dl = new DemolitionLevel();
                                dl.setLevelName(rs.getString("levelName"));
                                dl.setLevelDescription(rs.getString("description"));
                                dl.setLevelType(rs.getString("type"));
                                dl.setAuthor(rs.getString("author"));
                                dl.setPath(rs.getString("filePath"));

                                PlaylistDMLLevelEntry pldle = new PlaylistDMLLevelEntry();
                                pldle.setDl(dl);
                                pldle.setPlayerScore(0);
                                pldle.setClear(0);

                                playlist.getDlevels().add(pldle);
```

```
                        }

                        if(rs.getString("type").equals("AKN")) {
                                ArkanoidLevel al = new ArkanoidLevel();
                                al.setLevelName(rs.getString("levelName"));
                                al.setLevelDescription(rs.getString("description"));
                                al.setLevelType(rs.getString("type"));
                                al.setAuthor(rs.getString("author"));
                                al.setPath(rs.getString("filePath"));

                                PlaylistAKNLevelEntry plale = new PlaylistAKNLevelEntry();
                                plale.setAl(al);
                                plale.setPlayerScore(0);
                                plale.setClear(0);

                                playlist.getAlevels().add(plale);
                        }

                }

        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                try {
                        ps.close();
                        con.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        return playlist;
}

public boolean existLevel(String levelName) throws Exception {
        PreparedStatement ps = null;
        ResultSet rs = null;
        Connection con = null;

        boolean exist = false;

        try {

                con = jdbc.getConnection();
                ps = con.prepareStatement(existLevel);
                ps.setString(1, levelName);
```

```
                rs = ps.executeQuery();

                if (!rs.next()) {
                        exist = false;
                } else {
                        do {

                                if(rs.getString("levelName").equals(levelName)) {
                                        exist = true;
                                }

                        } while (rs.next());
                }

        } finally {
                try {
                        ps.close();
                        con.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        return exist;
}

public void updateLevel(String levelDescription, String levelName) {
        PreparedStatement ps = null;
        Connection con = null;

        try {

                con = jdbc.getConnection();
                ps = con.prepareStatement(updateLevel);
                ps.setString(1, levelDescription);
                ps.setString(2, levelName);
                ps.execute();

        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                try {
                        ps.close();
                        con.close();
                }
                catch (Exception e) {
```

```
                                e.printStackTrace();
                        }
                }
        }

        public void setJdbc(Jdbc jdbc) {
                this.jdbc = jdbc;
        }

        public void setInsertLevel(String insertLevel) {
                this.insertLevel = insertLevel;
        }

        public void setGetAllDemolitionLevels(String getAllDemolitionLevels) {
                this.getAllDemolitionLevels = getAllDemolitionLevels;
        }

        public void setGetAllArkanoidLevels(String getAllArkanoidLevels) {
                this.getAllArkanoidLevels = getAllArkanoidLevels;
        }

        public void setGetAllLevelsByUser(String getAllLevelsByUser) {
                this.getAllLevelsByUser = getAllLevelsByUser;
        }

        public void setExistLevel(String existLevel) {
                this.existLevel = existLevel;
        }

        public void setUpdateLevel(String updateLevel) {
                this.updateLevel = updateLevel;
        }

}
```

13.2.5 Paquete Model

13.2.5.1 DemolitionLevel.java

```
public class DemolitionLevel {

	private String levelName;
	private String levelDescription;

	private List<LevelElement> levelElements;
	private int canonPos;
	private int limitLine;
	private int numBullets;

	private String author;
	private String path;

	private String levelType;
	private int rating;

	public String getLevelName() {
		return levelName;
	}

	public void setLevelName(String levelName) {
		this.levelName = levelName;
	}

	public String getLevelDescription() {
		return levelDescription;
	}

	public void setLevelDescription(String levelDescription) {
		this.levelDescription = levelDescription;
	}

	public List<LevelElement> getLevelElements() {
		return levelElements;
	}
```

```
	public void setLevelElements(List<LevelElement> levelElements) {
		this.levelElements = levelElements;
	}

	public int getCanonPos() {
		return canonPos;
	}

	public void setCanonPos(int canonPos) {
		this.canonPos = canonPos;
	}

	public int getLimitLine() {
		return limitLine;
	}

	public void setLimitLine(int limitLine) {
		this.limitLine = limitLine;
	}

	public int getNumBullets() {
		return numBullets;
	}

	public void setNumBullets(int numBullets) {
		this.numBullets = numBullets;
	}

	public String getAuthor() {
		return author;
	}

	public void setAuthor(String author) {
		this.author = author;
	}

	public String getPath() {
		return path;
	}

	public void setPath(String path) {
		this.path = path;
	}
```

```
	public String getLevelType() {
		return levelType;
	}

	public void setLevelType(String levelType) {
		this.levelType = levelType;
	}

	public int getRating() {
		return rating;
	}

	public void setRating(int rating) {
		this.rating = rating;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((levelName == null) ? 0 : levelName.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		DemolitionLevel other = (DemolitionLevel) obj;
		if (levelName == null) {
			if (other.levelName != null)
				return false;
		} else if (!levelName.equals(other.levelName))
			return false;
		return true;
	}
```

```
        @Override
        public String toString() {
                return "DemolitionLevel [levelName=" + levelName
                                + ", levelDescription=" + levelDescription + ", levelElements="
                                + levelElements + ", canonPos=" + canonPos + ", limitLine="
                                + limitLine + ", numBullets=" + numBullets + ", author="
                                + author + ", path=" + path + ", levelType=" + levelType
                                + ", rating=" + rating + "]";
        }

}
```

13.2.5.2 ArkanoidLevel.java

```
public class ArkanoidLevel {

        private String levelName;
        private String levelDescription;

        private List<LevelElement> levelElements;

        private String author;
        private String path;

        private String levelType;
        private int rating;

        public String getLevelName() {
                return levelName;
        }

        public void setLevelName(String levelName) {
                this.levelName = levelName;
        }

        public String getLevelDescription() {
                return levelDescription;
        }

        public void setLevelDescription(String levelDescription) {
                this.levelDescription = levelDescription;
        }
        public String getPath() {
```

```
        return path;
    }

    public void setPath(String path) {
        this.path = path;
    }

    public List<LevelElement> getLevelElements() {
        return levelElements;
    }

    public void setLevelElements(List<LevelElement> levelElements) {
        this.levelElements = levelElements;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getLevelType() {
        return levelType;
    }

    public void setLevelType(String levelType) {
        this.levelType = levelType;
    }

    public int getRating() {
        return rating;
    }

    public void setRating(int rating) {
        this.rating = rating;
    }
```

```
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((levelName == null) ? 0 : levelName.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		ArkanoidLevel other = (ArkanoidLevel) obj;
		if (levelName == null) {
			if (other.levelName != null)
				return false;
		} else if (!levelName.equals(other.levelName))
			return false;
		return true;
	}

	@Override
	public String toString() {
		return "ArkanoidLevel [levelName=" + levelName + ", levelDescription="
				+ levelDescription + ", levelElements=" + levelElements.toString()
				+ ", author=" + author + ", path=" + path + ", levelType="
				+ levelType + "]";
	}
}
```

13.2.5.3 LevelElement.java

```
public class LevelElement {

	private String type;
	private int x;
	private int y;

	public String getType() {
		return type;
	}

	public void setType(String type) {
		this.type = type;
	}

	public int getX() {
		return x;
	}

	public void setX(int x) {
		this.x = x;
	}

	public int getY() {
		return y;
	}

	public void setY(int y) {
		this.y = y;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((type == null) ? 0 : type.hashCode());
		result = prime * result + x;
		result = prime * result + y;
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
```

```
                        return true;
                if (obj == null)
                        return false;
                if (getClass() != obj.getClass())
                        return false;
                LevelElement other = (LevelElement) obj;
                if (type == null) {
                        if (other.type != null)
                                return false;
                } else if (!type.equals(other.type))
                        return false;
                if (x != other.x)
                        return false;
                if (y != other.y)
                        return false;
                return true;
        }

        @Override
        public String toString() {
                return "LevelElement [type = " + type + ", x: " + x + ", y: " + y + "]";
        }
}
```

13.2.5.4 Playlist.java

```
public class Playlist {

        private String username;
        private List<PlaylistDMLLevelEntry> dlevels;
        private List<PlaylistAKNLevelEntry> alevels;

        public Playlist() {
                dlevels = new ArrayList<PlaylistDMLLevelEntry>();
                alevels = new ArrayList<PlaylistAKNLevelEntry>();
        }

        public String getUsername() {
                return username;
        }
```

```
        public void setUsername(String username) {
                this.username = username;
        }

        public List<PlaylistDMLLevelEntry> getDlevels() {
                return dlevels;
        }

        public void setDlevels(List<PlaylistDMLLevelEntry> dlevels) {
                this.dlevels = dlevels;
        }

        public List<PlaylistAKNLevelEntry> getAlevels() {
                return alevels;
        }

        public void setAlevels(List<PlaylistAKNLevelEntry> alevels) {
                this.alevels = alevels;
        }

}
```

13.2.5.5 PlaylistAKNLevelEntry.java

```
public class PlaylistAKNLevelEntry {

        private ArkanoidLevel al;
        private int playerScore;
        private int clear;

        public ArkanoidLevel getAl() {
                return al;
        }

        public void setAl(ArkanoidLevel al) {
                this.al = al;
        }

        public int getPlayerScore() {
                return playerScore;
        }
```

```
        public void setPlayerScore(int playerScore) {
                this.playerScore = playerScore;
        }

        public int getClear() {
                return clear;
        }

        public void setClear(int clear) {
                this.clear = clear;
        }

}
```

13.2.5.6 PlaylistDMLLevelEntry.java

```
public class PlaylistDMLLevelEntry {

        private DemolitionLevel dl;
        private int playerScore;
        private int clear;

        public DemolitionLevel getDl() {
                return dl;
        }

        public void setDl(DemolitionLevel dl) {
                this.dl = dl;
        }

        public int getPlayerScore() {
                return playerScore;
        }

        public void setPlayerScore(int playerScore) {
                this.playerScore = playerScore;
        }

        public int getClear() {
                return clear;
        }
```

```
        public void setClear(int clear) {
                this.clear = clear;
        }

}
```

13.2.6 Paquete Utils

13.2.6.1 JSONFilesUtils.java

```
public class JSONFilesUtils {

        @SuppressWarnings("unchecked")
        public static boolean createDemolitionLevelJSON(DemolitionLevel dl) {
                boolean complete = false;

                JSONObject obj = new JSONObject();
                obj.put("levelName", dl.getLevelName());
                obj.put("levelDescription", dl.getLevelDescription());

                obj.put("levelLimitLine", dl.getLimitLine());
                obj.put("levelCanonPos", dl.getCanonPos());
                obj.put("levelNumBullets", dl.getNumBullets());

                JSONArray list = new JSONArray();
                for (int i = 0; i < dl.getLevelElements().size(); i++) {
                        JSONObject innerObj = new JSONObject();
                        innerObj.put("piece", dl.getLevelElements().get(i).getType());
                        innerObj.put("x", dl.getLevelElements().get(i).getX());
                        innerObj.put("y", dl.getLevelElements().get(i).getY());
                        list.add(innerObj);
                }
                obj.put("levelElements", list);

                try {

                        String path = "\\home\\pfm\\" + dl.getLevelName() + "_DML.json";
                        FileWriter file = new FileWriter(path);
                        file.write(obj.toJSONString());
                        file.flush();
                        file.close();
```

```
                dl.setPath(path);

                complete = true;

        } catch (IOException e) {
                throw new RuntimeException(
                                "Error, no se ha podido generar el fichero JSON.", e);
        }

        return complete;
}

@SuppressWarnings("unchecked")
public static boolean createArkanoidLevelJSON(ArkanoidLevel al) {
        boolean complete = false;

        JSONObject obj = new JSONObject();
        obj.put("levelName", al.getLevelName());
        obj.put("levelDescription", al.getLevelDescription());

        JSONArray list = new JSONArray();
        for (int i = 0; i < al.getLevelElements().size(); i++) {
                JSONObject innerObj = new JSONObject();
                innerObj.put("piece", al.getLevelElements().get(i).getType());
                innerObj.put("x", al.getLevelElements().get(i).getX());
                innerObj.put("y", al.getLevelElements().get(i).getY());
                list.add(innerObj);
        }
        obj.put("levelElements", list);

        try {

                String path = "\\home\\pfm\\" + al.getLevelName() + "_AKN.json";
                FileWriter file = new FileWriter(path);
                file.write(obj.toJSONString());
                file.flush();
                file.close();

                al.setPath(path);

                complete = true;

        } catch (IOException e) {
                throw new RuntimeException(
                                "Error, no se ha podido generar el fichero JSON.", e);
```

```
            }

            return complete;
        }

        @SuppressWarnings("rawtypes")
        public static DemolitionLevel readDemolitionLevelJSON(String levelPath) throws IOException, ParseException {
            DemolitionLevel dl = new DemolitionLevel();
            List<LevelElement> elementList = new ArrayList<LevelElement>();

            FileReader reader = new FileReader(levelPath);
            JSONParser jsonParser = new JSONParser();
            JSONObject jsonObject = (JSONObject) jsonParser.parse(reader);

            JSONArray levelElements = (JSONArray) jsonObject.get("levelElements");
            Iterator i = levelElements.iterator();
            while (i.hasNext()) {
                LevelElement le = new LevelElement();
                JSONObject innerObj = (JSONObject) i.next();
                le.setType((String) innerObj.get("piece"));
                Long x = (Long) innerObj.get("x");
                le.setX(x.intValue());
                Long y = (Long) innerObj.get("y");
                le.setY(y.intValue());
                elementList.add(le);
            }

            Long canon = (Long) jsonObject.get("levelCanonPos");
            dl.setCanonPos(canon.intValue());
            Long limitLine = (Long) jsonObject.get("levelLimitLine");
            dl.setLimitLine(limitLine.intValue());
            Long numBullets = (Long) jsonObject.get("levelNumBullets");
            dl.setNumBullets(numBullets.intValue());
            dl.setLevelElements(elementList);

            return dl;
        }
```

```
        @SuppressWarnings("rawtypes")
        public static ArkanoidLevel readArkanoidLevelJSON(String levelPath) throws IOException, ParseException {
                ArkanoidLevel al = new ArkanoidLevel();
                List<LevelElement> elementList = new ArrayList<LevelElement>();

                FileReader reader = new FileReader(levelPath);
                JSONParser jsonParser = new JSONParser();
                JSONObject jsonObject = (JSONObject) jsonParser.parse(reader);

                JSONArray levelElements = (JSONArray) jsonObject.get("levelElements");
                Iterator i = levelElements.iterator();
                while (i.hasNext()) {
                        LevelElement le = new LevelElement();
                        JSONObject innerObj = (JSONObject) i.next();
                        le.setType((String) innerObj.get("piece"));
                        Long x = (Long) innerObj.get("x");
                        le.setX(x.intValue());
                        Long y = (Long) innerObj.get("y");
                        le.setY(y.intValue());
                        elementList.add(le);
                }

                al.setLevelElements(elementList);

                return al;
        }
}
```