



UCA

Universidad
de Cádiz

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

Desarrollo en Symfony de plataforma web para partidas de
rol a distancia

Jaime Jesús Serrano Rodríguez

30 de octubre de 2016



ESCUELA SUPERIOR DE INGENIERÍA
GRADO EN INGENIERÍA EN INFORMÁTICA

Desarrollo en Symfony de plataforma web para partidas de
rol a distancia

Departamento: Ingeniería Informática
Director del proyecto: Pablo de la Torre Moreno
Autor del proyecto: Jaime Jesús Serrano Rodríguez

Puerto Real, 30 de octubre de 2016

Jaime Jesús Serrano Rodríguez

Agradecimientos

Durante toda mi etapa universitaria y en el desarrollo de mi Trabajo Fin de Grado (TFG), ha habido muchos familiares, compañeros, profesores y amigos que me han apoyado y permitido llegar a presentar el TFG. Quiero agradecerles a todos ellos el apoyo que me han ofrecido y que me ha permitido llegar hasta aquí.

Quiero agradecer especialmente a Pablo de la Torre su dirección como director del proyecto, que me ha permitido realizar el proyecto que deseaba, con las ideas que he tenido y siempre me ha ayudado a conseguir este objetivo.

A mi compañero Miguel, que ha decidido que su TFG, junto con el mío, sea una pieza de un proyecto más grande que puede llegar a ser una gran aplicación algún día.

La universidad se puede vivir solo desde el estudio pero también desde el compañerismo, la colaboración y el intentar ayudar a los demás. Por ello como miembro activo de la delegación de alumnos quiero agradecer especialmente a Patricia, Paco, Diego, Eliezer y a Ana por haber estado ahí siempre ha que hecho falta y trabajando por una Escuela Superior de Ingeniería mejor.

A mis amigos y compañeros Álvaro, Kisko, Emilio, Juan Carlos, Edu y Chema, que me han dado buenos consejos durante toda esta travesía y me han ayudado con sus conocimientos a terminar este TFG.

Agradecimientos a los profesores que tanto me han enseñado y que me ha permitido completar esta etapa. Así como a la dirección del centro, los diferentes cargos administrativos y de gestión a los que tanto he acudido para resolver algún trámite y que siempre me han ayudado a resolverlo.

Por su puesto no puedo dejar de agradecer a toda mi familia que me ha apoyado siempre en esta aventura y les estoy muy agradecido por ello, gracias. Sobre todo por la incansable pregunta de “¿Cuando vas a terminar?” que tanto me ha acosado estos últimos meses.

Por último quiero agradecer al tribunal el esfuerzo en la evaluación del TFG.

Resumen

Desde hace muchos años los jugadores de juegos de rol presencial (de mesa) hemos tenido y estamos teniendo dificultades para organizar partidas en lo que respecta a encuentros con nuevos jugadores, intercambio de opiniones sobre los diferentes juegos de rol, agrupar todo el contenido de los juegos de rol actualizado y a la propia ejecución de las partidas. La mayor parte de estas dificultades se deben a que se requiere una comunicación presencial. Existen sistemas y herramientas como foros, páginas web, software de comunicación, lanzadores de datos y creadores de fichas, que intentan dar solución estos problemas; el proyecto actual trata de solventar algunas carencias encontradas en ellas.

Por todo lo mencionado anteriormente el objetivo es crear una plataforma web para suplir estas dificultades, donde jugadores de rol puedan reunirse y practicar su afición con unas herramientas que faciliten las partidas de rol tanto a distancia como presenciales, intentando minimizar las dificultades de realizar partidas a distancia.

La aplicación se ha desarrollado de cara a funcionar en navegadores donde los usuarios, desde el sistema, importan sus fichas a sesiones de juego y mediante el sistema de reglas de cada juego de rol y diferentes herramientas llevan a cabo sus partidas. Además teniendo la figura del director de juego con la capacidad de gestionar los permisos y acciones de los usuarios. Basado en lo anterior, la tecnología que se ha empleado consta principalmente de: PHP, Symfony, JavaScript, jQuery, MySQL, HTML, CSS, Bootstrap y Web Application Messaging Protocol (WAMP).

Palabras clave

Juego de rol, juego de rol de mesa, plataforma web, plataforma de juego, PHP, servidor PHP, Symfony, websocket, JavaScript, Twig, YAML, Web Application Messaging Protocol (WAMP).

Índice general

Prolegómeno	1
1. Introducción.....	3
1.1. Motivación.....	3
1.2. Estado del arte.....	3
1.3. Objetivos y alcance del proyecto.....	5
1.4. Organización del documento.....	5
1.4.1. Introducción.....	5
1.4.2. Planificación.....	5
1.4.3. Análisis de requisitos.....	6
1.4.4. Diseño del sistema.....	6
1.4.5. Implementación del sistema.....	6
1.4.6. Pruebas del sistema.....	6
1.4.7. Conclusiones.....	6
1.4.8. Anexos.....	6
1.5. Glosario de términos.....	6
2. Planificación.....	9
2.1. Metodología de desarrollo.....	9
2.2. Planificación del proyectos.....	9
2.3. Organización.....	11
2.4. Costes.....	11
2.5. Gestión de riesgos.....	12
Desarrollo	16
3. Análisis de requisitos.....	18
3.1. Catálogo de actores.....	18
3.2. Requisitos funcionales.....	19
3.2.1. Subsistema de plataforma web.....	19
3.2.1.1. Subsistema de gestión de usuario.....	20
3.2.1.2. Subsistema de gestión de idioma.....	25
3.2.1.3. Subsistema de gestión del compendio de rol.....	26
3.2.2. Subsistema de plataforma de juego.....	26
3.2.2.1. Subsistema de gestión de juego de rol.....	27
3.2.2.2. Subsistema de gestión de plantilla de ficha de personaje.....	27
3.2.2.3. Subsistema de gestión de ficha de personaje.....	30
3.2.2.4. Subsistema de gestión de sesión de juego.....	34
3.2.3. Subsistema de herramienta de juego.....	39
3.2.3.1. Subsistema de chat.....	40
3.2.3.2. Subsistema de conexión de usuario a la sesión de juego.....	42
3.2.3.3. Subsistema de mapa.....	46
3.2.3.4. Subsistema de lanzamiento de dado.....	50
3.2.3.5. Subsistema de importación de ficha de personaje.....	53
3.2.3.6. Subsistema de funcionalidad de ficha de personaje.....	57
3.2.3.7. Subsistema de turno.....	61
3.2.3.8. Subsistema multimedia.....	61
3.3. Requisitos de información.....	62
3.4. Reglas de negocio.....	65

3.5. Requisitos no funcionales.....	66
3.6. Alternativa y elección tecnológica.....	68
3.7. Análisis GAP.....	70
3.8. Matriz de rastreabilidad.....	71
3.9. Protocolo de comunicación.....	72
3.9.1. Organización.....	72
3.9.2. Ficha de personaje.....	72
3.9.3. Crear ficha de personaje.....	75
3.9.3.1. Pedir campos que derivan.....	75
3.9.3.2. Modificar campos derivados.....	75
3.9.4. Mostrar ficha de personaje.....	76
3.9.5. Editar ficha de personaje.....	76
3.9.6. Importar ficha de personaje.....	76
3.9.6.1. Importar funcionalidad individual.....	76
3.9.6.2. Importar funcionalidad colectiva.....	77
4. Diseño del sistema.....	79
4.1. Diseño de la arquitectura.....	79
4.1.1. Arquitectura física.....	79
4.1.2. Arquitectura lógica.....	80
4.1.3. Arquitectura de diseño.....	81
4.2. Diagrama de clases.....	81
4.3. Diseño de datos.....	83
4.4. Diseño del protocolo a emplear.....	86
4.4.1 Ejemplo de funcionalidad individual con el juego de rol Pathfinder.....	87
4.5. Diseño de la interfaz de usuario.....	90
5. Implementación del sistema.....	100
5.1. Entorno tecnológico.....	100
5.2. Parametrización del software base.....	103
5.2.1. routing.yml.....	103
5.2.2. security.yml.....	103
5.2.3. Generar tablas con doctrine.....	104
5.2.4. Validaciones.....	105
5.2.5. Traducciones.....	106
5.2.6. Enrutamiento.....	107
5.2.7. Dependencias.....	107
5.3. Código fuente.....	107
5.4. Código destacado.....	108
5.4.1. Inyección de dependencias en el servidor de websockets.....	108
5.4.2. Conexión a websocket segura.....	109
5.4.3. Ejemplo importar ficha de personaje con websocket.....	110
5.4.4. Renderización de formularios dinámicos.....	111
6. Pruebas del sistema.....	115
6.1. Pruebas unitarias.....	115
6.2. Pruebas funcionales.....	116
6.3. Pruebas de validación.....	116
6.3.1. Plataforma web.....	116
6.3.2. Plataforma de juego.....	118
6.4. Pruebas de sistema.....	121

Epílogo	124
7. Conclusiones.....	126
7.1. Objetivos.....	126
7.2. Lecciones aprendidas.....	126
7.3. Trabajo futuro.....	127
Anexos	129
Anexo I: Manual de usuario.....	131
1. Requisitos previos.....	131
2. Utilización.....	131
2.1. Usuario no registrado.....	131
2.1.1. Página principal.....	131
2.2. Usuario.....	132
2.2.1. Página principal.....	132
2.2.2. Ver perfil.....	132
2.2.3. Gestionar fichas de personaje.....	133
2.2.4. Crear ficha de personaje.....	133
2.2.5. Crear sesión de juego.....	134
2.2.6. Unirse a sesión de juego.....	135
2.1.7. Sesión de juego.....	136
2.1.8. Importar ficha de personaje en la sesión de juego.....	137
2.1.9. Lanzamiento de dados en la sesión de juego.....	137
2.1.10. Ejecutar funcionalidad de las fichas de personaje.....	137
2.2. Director de juego.....	138
2.2.1. Sesión de juego.....	138
2.2.2. Editar la sesión de juego.....	139
2.2.3. Gestionar usuarios de la sesión de juego.....	139
2.2.4. Mapa de la sesión de juego.....	140
2.3. Administrador.....	140
Anexo II: Manual de instalación y exportación.....	143
1. Requisitos previos.....	143
2. Procedimiento de instalación.....	143
2.1. Configuración previa para crear/importar un proyecto Symfony2.....	143
2.2. Importar un proyecto Symfony2 mediante Eclipse.....	144
2.3. Post configuración para crear/importar un proyecto Symfony2.....	144
3. Buenas prácticas y comandos útiles.....	146
Anexo III: Tormenta de ideas, fase de requisitos.....	152
1. Introducción.....	153
1.1. Participantes.....	153
1.2. Problema a tratar.....	153
1.3. ¿Divisiones del problema necesarias?.....	153
1.4. ¿Informe de antecedentes?.....	153
1.5. Preguntas propuestas.....	153
2. Ejecución de la tormenta de ideas.....	154
2.1. Restricción.....	154
2.2. Inicio.....	154
2.3. Debate.....	155
2.3.1. ¿Qué debería permitir el sistema a los usuarios?.....	155
2.3.2. ¿Qué herramientas debería haber?.....	155
2.4. Conclusión.....	157

Índice de figuras

Logo Roll20.....	3
Logo Fantasy Grounds.....	4
Logo D20PRO.....	4
Ciclo de vida de Agile Unified Process.....	9
Diagrama de Gantt de la planificación del proyecto.....	10
Diagrama de Gantt de los tiempos finalmente empleados.....	11
Subsistemas.....	19
Subsistema de plataforma web.....	20
Subsistema de gestión de usuario.....	20
Subsistema de gestión de idioma.....	26
Subsistema de plataforma de juego.....	27
Subsistema de gestión de juego de rol.....	27
Subsistema de gestión de plantilla de ficha de personaje.....	28
Subsistema de gestión de ficha de personaje.....	31
Subsistema de gestión de sesión de juego.....	35
Subsistema de herramienta de juego.....	40
Subsistema de chat.....	41
Subsistema de conexión de usuario a la sesión de juego.....	43
Subsistema de mapa.....	46
Subsistema de lanzamiento de dado.....	51
Subsistema de importación de ficha de personaje.....	53
Subsistema de funcionalidad de ficha de personaje.....	57
Subsistema de turno.....	61
Diagrama Conceptual.....	65
Ficha de personaje de Pathfinder.....	73
Ficha de personaje de Vampiro la Mascarada.....	74
Arquitectura física del sistema.....	80
Diagrama de Clases.....	82
Diagrama Entidad-Relación.....	86
Ejemplo de funcionalidad individual.....	89
Ejemplo de funcionalidad colectiva, selección de acción.....	89
Boceto "crear sesión de juego".....	90
Boceto "unirse a sesión de juego".....	91
Boceto "gestionar herramientas por el director de juego".....	92
Boceto "gestionar herramientas por el usuario".....	92
Boceto "sesión de juego".....	93
Boceto "subsistema de mapa".....	94
Boceto "subsistema de turno".....	95
Boceto "subsistema de historial".....	95
Boceto "subsistema de chat".....	96
Boceto "subsistema de anotación".....	96
Boceto "subsistema multimedia".....	97
Boceto "subsistema de lanzamiento de dado".....	98

PHP logo.....	101
PHP Unit logo.....	101
Symfony logo.....	101
Twig logo.....	101
WAMP logo.....	101
Composer logo.....	101
MySQL logo.....	101
HTML logo.....	102
JavaScript logo.....	102
jQuery UI logo.....	102
jQuery logo.....	102
CSS logo.....	102
Bootstrap logo.....	102
JSON logo.....	102
Eclipse logo.....	102
Git logo.....	103
Flujo de un formulario en Symfony.....	111
Captura de las pruebas unitarias y funcionales.....	115
Captura "página principal de usuario no registrado".....	132
Captura "página principal de usuario".....	132
Captura "ver perfil".....	133
Captura "gestionar fichas de personaje".....	133
Captura "crear ficha de personaje 1/2".....	133
Captura "crear ficha de personaje 2/2".....	134
Captura "crear sesión de juego".....	134
Captura "acceder a sesiones de juego".....	135
Captura "acceder a sesión de juego".....	136
Captura "sesión de juego del usuario".....	136
Captura "importar ficha de personaje en la sesión de juego".....	137
Captura "lanzamiento de dados en la sesión de juego".....	137
Captura "ejecutar funcionalidad de la ficha de personaje".....	138
Captura "sesión de juego del director de juego".....	139
Captura "editar la sesión de juego".....	139
Captura "gestionar usuarios de la sesión de juego".....	140
Captura "mapa de la sesión de juego".....	140
Captura "administración 1/3".....	141
Captura "administración 2/3".....	141
Captura "administración 3/3".....	141

Índice de tablas

Comparativa del estado del arte.....	4
Objetivo. Crear sistema de plataforma de juego.....	5
Objetivo. Crear sistema de plataforma web.....	5
Costes del proyecto.....	12
Riesgo. Complejidad en el aprendizaje de las tecnologías.....	12
Riesgo. Problemas personales.....	13
Riesgo. Pérdida completa del sistema.....	13
Riesgo. Rotura del hardware de trabajo.....	13
Riesgo. Modificación de los requisitos.....	14
Actor. Usuario anónimo.....	18
Actor. Usuario.....	18
Actor. Administrador.....	18
Actor. Director de juego.....	18
Caso de uso. Registrar.....	21
Caso de uso. Acceder.....	22
Caso de uso. Salir.....	23
Caso de uso. Editar datos básicos.....	24
Caso de uso. Editar contraseña.....	25
Caso de uso. Añadir plantilla de ficha de personaje.....	29
Caso de uso. Editar plantilla de ficha de personaje.....	30
Caso de uso. Añadir ficha de personaje.....	32
Caso de uso. Editar ficha de personaje.....	33
Caso de uso. Eliminar ficha de personaje.....	34
Caso de uso. Añadir sesión de juego.....	36
Caso de uso. Editar sesión de juego.....	37
Caso de uso. Eliminar sesión de juego.....	38
Caso de uso. Buscar sesión de juego.....	39
Caso de uso. Crear chat.....	41
Caso de uso. Añadir entrada de chat.....	42
Caso de uso. Acceder a la sesión de juego.....	44
Caso de uso. Salir de la sesión de juego.....	45
Caso de uso. Crear mapa.....	47
Caso de uso. Añadir ficha de mapa.....	48
Caso de uso. Eliminar ficha de mapa.....	49
Caso de uso. Mover ficha de mapa.....	50
Caso de uso. Realizar lanzamiento de dado.....	52
Caso de uso. Pedir fichas de personaje.....	54
Caso de uso. Importar ficha de personaje.....	55
Caso de uso. Eliminar ficha de personaje.....	56
Caso de uso. Importar funcionalidad de ficha de personaje.....	58
Caso de uso. Ejecutar funcionalidad individual de ficha de personaje.....	59
Caso de uso. Ejecutar funcionalidad colectiva de ficha de personaje.....	61
Requisito de información. Usuario.....	62
Requisito de información. Idioma.....	62

Requisito de información. Juego de rol.....	62
Requisito de información. Sesión de juego.....	63
Requisito de información. Plantilla de ficha de personaje.....	63
Requisito de información. Ficha de personaje.....	63
Requisito de información. Dato grupo de ficha de personaje.....	64
Requisito de información. Dato derivado de ficha de personaje.....	64
Requisito de información. Dato campo de ficha de personaje.....	64
Regla de negocio. Tamaño de la contraseña de usuario.....	65
Regla de negocio. Contraseña necesaria siempre en sesión de juego.....	66
Regla de negocio. Tamaño de la contraseña de sesión de juego.....	66
Regla de negocio. El director de juego solo puede ser el creador de la sesión de juego.....	66
Requisito no funcional. Control de Acceso.....	66
Requisito no funcional. Mantenibilidad.....	66
Requisito no funcional. Disponibilidad.....	67
Requisito no funcional. Multi navegador y diseño responsive.....	67
Requisito no funcional. Reconexión de usuarios.....	67
Requisito no funcional. Sistema multi idioma.....	68
Matriz de rastreabilidad.....	71
Diseño de datos. usuario.....	83
Diseño de datos. idioma.....	83
Diseño de datos. juegoRol.....	83
Diseño de datos. sesionJuego.....	84
Diseño de datos. plantillaFichaPersonaje.....	84
Diseño de datos. fichaPersonaje.....	84
Diseño de datos. datoFichaPersonaje.....	85
Prueba. Conexión de usuario.....	117
Prueba. Modificación de los datos de usuario.....	117
Prueba. Gestión de ficha de personaje.....	118
Prueba. Gestión de sesiones de juego y conexión de usuarios.....	119
Prueba. Funcionalidad de chat.....	119
Prueba. Funcionalidad de mapa.....	120
Prueba. Funcionalidad de lanzamiento de dado.....	120
Prueba. Funcionalidad de importación de fichas de personaje.....	120
Prueba. Funcionalidad de ejecución de funcionalidad de fichas de personaje.....	121

Parte I

Prolegómeno

1. Introducción

1.1. Motivación

Desde hace muchos años los jugadores de juegos de rol presencial (de mesa) hemos tenido y estamos teniendo dificultades para organizar partidas en lo que respecta a encuentros con nuevos jugadores, intercambio de opiniones sobre los diferentes juegos de rol, agrupar todo el contenido de los juegos de rol actualizado y a la propia ejecución de las partidas. La mayor parte de estas dificultades se deben a que se requiere una comunicación presencial.

Existen sistemas y herramientas como foros, páginas web, software de comunicación, lanzadores de dados y creadores de fichas, que intentan dar solución estos problemas; el proyecto actual trata de solventar algunas carencias encontradas en ellas.

Las carencias más importantes de estos sistemas, y que se pretenden resolver, son fundamentalmente:

- Ofrecer herramientas que son insuficientes para realizar las sesiones de juego, es decir, están incompletos.
- Las reglas de los juegos de rol implementadas solo permiten ejecutar la funcionalidad de cada ficha de forma individual por lo que no hay relación entre las fichas y no hay una verdadera integración de las reglas del juego de rol.
- Poseen uno o pocos juegos de rol implementados en sus sistemas.

1.2. Estado del arte

Como ya comentábamos, actualmente existen sistemas que se asemejan a lo que queremos desarrollar y tienen una gran cantidad de funcionalidad interesante de la que basarse. A continuación se realizará una breve comparativa de cada uno de ellos.

Roll20 [1]

Empezaremos por el más completo de todos ellos, Roll20. Esta aplicación es bastante completa pero le falta el elemento más importante por el que nosotros desarrollamos nuestro sistema y es la implementación de las reglas para que las propias fichas puedan interactuar unas con otras. Roll20 permite hacer acciones concretas entre las fichas pero no se encuentran los sistemas implementados.



Dibujo 1: Logo Roll20

Fantasy Grounds [2]

Se trata de una de las aplicaciones más completas en cuanto a funcionalidad. Los sistemas de reglas que tiene implementados los tiene con una calidad bastante buena. Por desgracia la tecnología que usan es bastante anticuada y la visualización es muy mejorable. Una característica que no hemos hablado hasta ahora pero que se hace imprescindible hablar de ella es el precio que tiene que pagar el usuario por estos servicios.

Fantasy Grounds tiene unos precios muy elevados para lo que ofrece, haciéndolo injugable para la mayoría de los usuarios. Otro problema de Fantasy Grounds es que se centra en pocos juegos, los que usan el sistema d20.



*Dibujo 2: Logo
Fantasy Grounds*

D20PRO [3]

Esta aplicación está muy orientada al un mapa interactivo, pero su funcionalidad e interacción entre las fichas es bastante limitada. Su creador de fichas es muy limitado en cuanto a funcionalidad y solo se centra en un tipo de juego de rol, los que usan sistema d20.



Dibujo 3: Logo D20PRO

Otras aplicaciones

El resto de aplicaciones son en su mayoría herramientas individuales que no permiten gestionar la mayor parte de la funcionalidad que proponemos. Aún así cabe mencionarlas debido a su interés y a su utilidad, dejando a un lado la gran cantidad de foros que se utilizan para jugar a rol a través de ellos. Alguna de estas herramientas son: Combattracker [4], Dicelog [5], 3dvtt [6], RPTools [7] y TTToprpg [8].

Aplicación	Mapa interactivo	Creador de fichas	Interacción entre las fichas	Gestión de sesiones de juego	Tecnología	Gestión del director de juego
Roll20	No muy funcional	Sí	Pobre y a través del mapa	Buena	Se ejecuta en navegador (HTML5)	Buena
Fantasy Grounds	Muy funcional	Sí	Buena y a través del mapa	Normal	Se ejecuta en una aplicación de escritorio	Normal
D20PRO	Muy funcional	Limitado	Pobre y a través del mapa	Normal	Se ejecuta con Java	Normal

Tabla 1: Comparativa del estado del arte

1.3. Objetivos y alcance del proyecto

Con este proyecto, de plataforma web, no se pretenden suplir las quedadas presenciales ya que pensamos que éstas siempre son la mejor opción debido a la propia esencia del rol de mesa donde los jugadores debaten, discuten y juegan al fin y al cabo, y qué mejor manera que hacerlo unos frente a otros. Por ello lo que se pretende es crear una aplicación que le permita a esa comunidad de jugadores realizar sus partidas a través de Internet, acercando la experiencia lo máximo posible a las quedadas presenciales. Además de servirles de apoyo con diferentes herramientas tanto presencialmente como a través de Internet, conocer otros jugadores y jugar con ellos. Todo esto supliendo las carencias que los actuales sistemas poseen.

Los objetivos propuestos para la consecución del proyecto son los siguientes:

OBJ-1	Crear sistema de plataforma de juego
Descripción	La plataforma de juego será la encargada de gestionar las sesiones de juego, la conexión a las mismas y todas las herramientas que los usuarios utilizarán en sus partidas.

Tabla 2: Objetivo. Crear sistema de plataforma de juego

OBJ-2	Crear sistema de plataforma web
Descripción	La plataforma web será el sistema encargado de gestionar a los usuarios y a la visualización de los diferentes complementos fuera de la sesión de juego como la creación de fichas de personaje y las diferentes herramientas de gestión para los administradores.

Tabla 3: Objetivo. Crear sistema de plataforma web

1.4. Organización del documento

A continuación se va a realizar un resumen de lo que consiste cada capítulo.

1.4.1. Introducción

En la introducción, el capítulo actual, se explican los conceptos generales del proyecto, sus objetivos y el estado del arte. Con todo esto podemos obtener una idea general de lo que se desea conseguir con la consecución del proyecto actual.

1.4.2. Planificación

En el capítulo de planificación del proyecto se expone la metodología que se va a emplear para el desarrollo, las personas implicadas en el proyecto, la planificación temporal y económica y la gestión de los distintos riesgos que puedan afectar a la consecución del proyecto.

1.4.3. Análisis de requisitos

En este primer capítulo de la etapa de desarrollo se detalla toda la funcionalidad que se desea realizar sin entrar en detalles del lenguaje de programación y tecnologías empleadas. Por ello se definen los actores que intervendrán en el sistema, los requisitos funcionales junto con los subsistemas asociados, los requisitos de información y las reglas de negocio. Además de lo anterior se muestran las alternativas tecnológicas para desarrollar la aplicación. Finalmente se expone un protocolo de comunicación que será el referente de envío de mensajes entre el cliente y servidor para que el desarrollo de la aplicación sea el deseado.

1.4.4. Diseño del sistema

En el diseño del sistema se define la arquitectura física, lógica y de diseño que se ha decidido para la aplicación. Se detallan tanto el diagrama de clases y el diagrama de datos necesario. Además se diseña la interfaz de usuario para hacer una idea clara de la implementación que se desea, por ello también se detalla el uso del protocolo de comunicación expuesto en el capítulo anterior.

1.4.5. Implementación del sistema

Durante el capítulo de la implementación del sistema se detalla la elección del entorno tecnológico que se va a utilizar en la aplicación, desde los lenguajes de programación, herramientas y *frameworks*. También se define la parametrización del software base, la organización del código del proyecto y el código implementado más destacado.

1.4.6. Pruebas del sistema

En el último capítulo del desarrollo, el de pruebas del sistema, se explican las pruebas unitarias, de validación y de sistema que se han empleado en la aplicación. El objetivo es descubrir posibles fallos de implementación y validar que los usuarios pueden utilizar eficazmente la aplicación.

1.4.7. Conclusiones

En este capítulo se explican las conclusiones de todo el desarrollo del proyecto que incluyen tanto las conclusiones del desarrollo como el posible desarrollo futuro que pueda tener el proyecto. Además se identifica todo lo aprendido en el transcurso del desarrollo de este proyecto.

1.4.8. Anexos

Finalmente se adjuntan los anexos de los manuales de importación y explotación y el manual de usuario. Además se incluye la tormenta de ideas que se realizó previo al desarrollo para tener claro qué es lo que los usuarios desearían tener. Y por último la bibliografía de todo el documento.

1.5. Glosario de términos

En esta sección pretendemos mostrar las palabras más usadas y utilizadas en esta aplicación con el fin de hacer entender exactamente el significado de las mismas.

- **Juego de rol:** un juego de rol es un juego en el que, tal como indica su nombre, uno o más jugadores desempeñan un determinado rol, papel o personalidad. [9]
- **Juego de rol de mesa:** es un juego de rol que casi siempre incluye un sistema de reglas para jugarlo y una ambientación sobre la que basar la narración tanto del director de juego como de los jugadores.
- **Partida:** cuando un grupo de personas se reúnen para jugar a un juego, ya sea por Internet o presencialmente. En este caso nos referimos exclusivamente a juegos de rol de mesa.
- **Sesión de juego:** es la partida que se juega en el sistema, donde los jugadores se unen para jugar.
- **Director de juego:** es el usuario creador de la sesión de juego y el que narra la historia de la partida. En el sistema tiene permisos especiales para poder gestionar la partida adecuadamente.
- **Dirigir partida:** es la acción de narración de la partida por parte del director de juego donde comenta lo que sucede y pide que los demás jugadores realicen acciones y tiradas. Es el que tiene la autoridad en la partida.
- **Tirada:** consiste en un lanzamiento de dados, normalmente basándose en una acción que desea realizarse en el juego de rol y que éste está sujeto a unas reglas determinadas. No todos los juegos de rol tienen las mismas tiradas para la misma acción, utilizan diferentes parámetros, fórmulas y dados.
- **Ficha:** cada personaje de un juego de rol está representado normalmente en un papel con una serie de parámetros que lo diferencian de los demás. Esto es a lo que denominamos ficha o ficha de personaje.

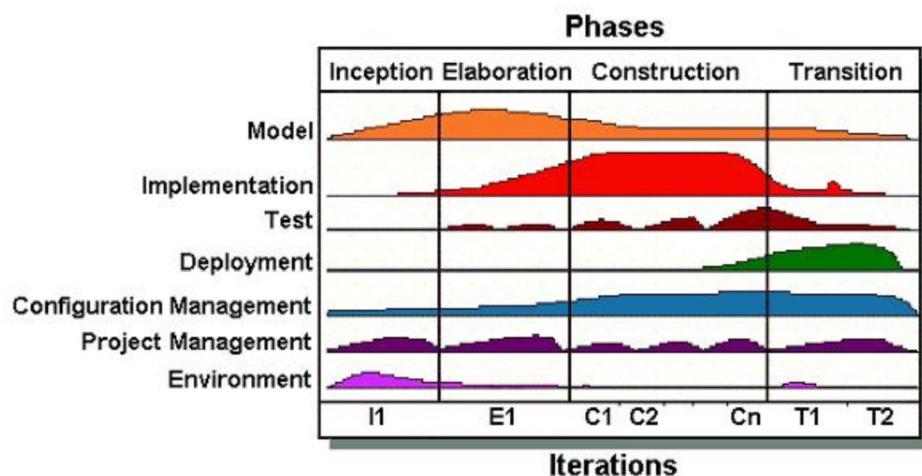
2. Planificación

2.1. Metodología de desarrollo

La metodología empleada para el desarrollo del proyecto es Agile Unified Process (AUP), en español Proceso Unificado Ágil, de Scott Ambler. Es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. También puede entenderse como una versión simplificada del ciclo de vida en espiral. [10] [11]

El ciclo de vida puede verse representado en la siguiente imagen donde consta de cuatro fases:

- Comienzo: identificación del alcance y dimensión del proyecto, propuesta de la arquitectura y del presupuesto del cliente.
- Elaboración: Confirmación de la idoneidad de la arquitectura.
- Construcción: Desarrollo incremental del sistema, siguiendo las prioridades funcionales de los implicados.
- Transición: Validación e implantación del sistema.



Dibujo 4: Ciclo de vida de Agile Unified Process

2.2. Planificación del proyectos

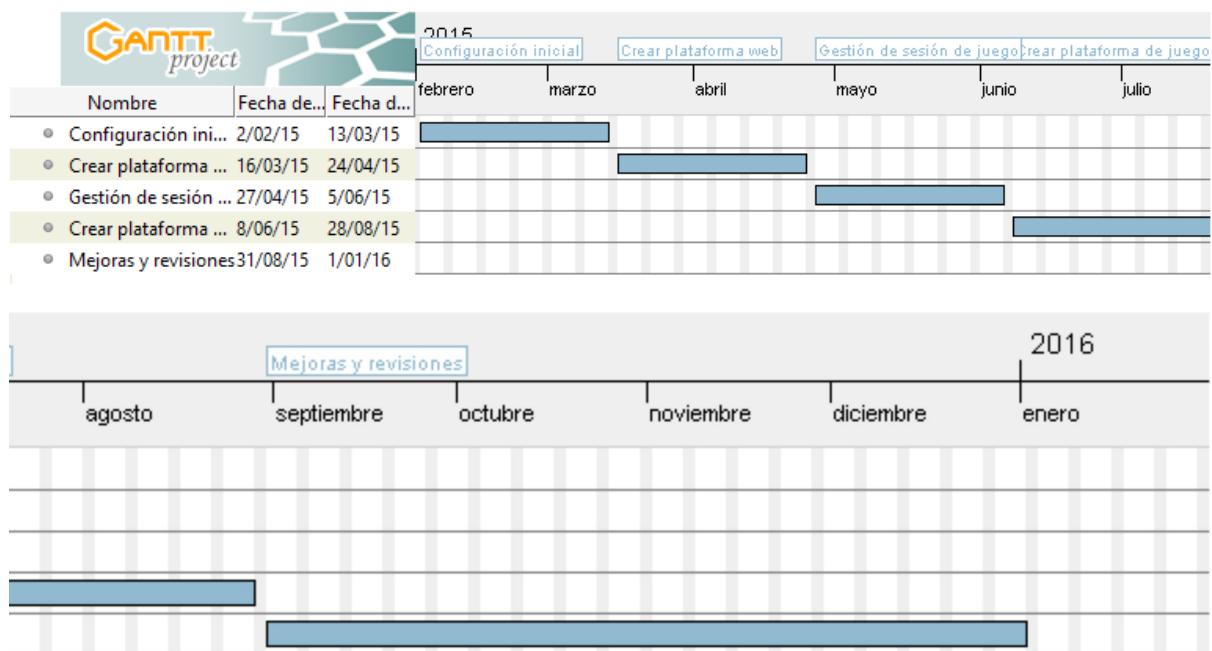
El inicio del proyecto comenzó en Noviembre de 2014 cuando se empezó a concebir el sistema como posible idea a desarrollar como proyecto fin de grado. Se recogieron los requisitos generales del sistema y otros tantos más específicos y se elaboró una estimación temporal.

Durante la estimación se tuvo en cuenta la regla, no escrita, del 30% que se le añade a todo proyecto para mantener un margen por si hubiera algún problema.

La estimación se hizo basándose en cuatro iteraciones, que son:

1. Configuración inicial: montaje del servidor y configuraciones principales.
2. Crear plataforma web.
3. Gestión de sesión de juego.
4. Crear plataforma de juego.
5. Mejoras y revisiones: tanto de la funcionalidad del sistema como de la documentación a entregar.

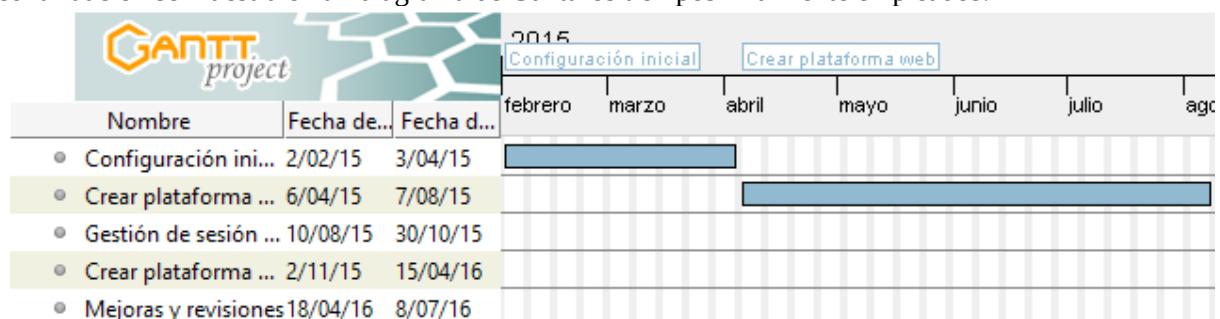
A continuación se muestra en un diagrama de Gantt la previsión inicial:

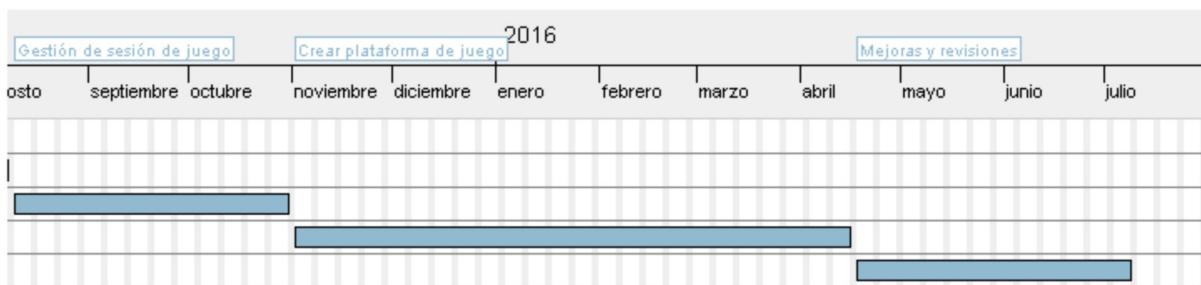


Dibujo 5: Diagrama de Gantt de la planificación del proyecto

Por desgracia no se cumplió debido a un par de riesgos que se cumplieron y por ello aplicaron su impacto en el desarrollo, son: complejidad en el aprendizaje de las tecnologías [RISK-1], problemas personales [RISK-2].

A continuación se muestra en un diagrama de Gantt los tiempos finalmente empleados:





Dibujo 6: Diagrama de Gantt de los tiempos finalmente empleados

Como puede observarse la duración de todos los puntos ha aumentado excepto en el de “Mejoras y revisiones” esto es debido a que se ha conseguido dominar la tecnología y, por ello, se ahorran costes temporales en el desarrollo. Además al haber estado desarrollando de forma mantenible, ha sido mucho más rápido y sencillo hacer modificaciones a la funcionalidad existente. Por otro lado, en los puntos que sí han aumentado, el aumento va desde el 50% hasta el 100% de incremento del coste temporal.

2.3. Organización

En esta sección se definen tanto las personas que han contribuido al desarrollo del proyecto como los recursos software y hardware que han permitido la gestión del mismo.

En el desarrollo del proyecto han participado varias personas con diferentes roles:

- Pablo de la Torre Moreno, como director del proyecto.
- Jaime Jesús Serrano Rodríguez, como alumno desarrollador del proyecto.
- Miguel Castillo Lloret, como compañero del macro proyecto (unión de ambos proyectos para formar la aplicación completa).
- Juan Carlos Trejo Fernández, como cliente experto de juegos de rol de mesa y aplicaciones para jugar a rol de mesa a través de Internet.
- Emilio Ciprés Núñez, como cliente experto de juegos de rol de mesa.

En cuanto a recursos hardware solo se ha hecho uso de un portátil MSI GT70. De software se ha hecho uso de los siguientes:

- Windows 8 y 10 y Ubuntu 14.04 LTS. Como sistemas operativos que soportan el resto de aplicaciones. Así como los navegadores Google Chrome 54 y Firefox 47.
- GanttProject, para desarrollar los diagramas de Gantt.
- Google Drive, para generar documentos.
- LibreOffice, para generar documentos.
- Hangouts, para reuniones virtuales con los colaboradores del proyecto.
- Outlook y Gmail, para mantener un sistema de comunicación por escrito con los colaboradores del proyecto.

2.4. Costes

Aunque este proyecto se trata de un Trabajo Fin de Grado que no ha sido solicitado por un cliente, se va a desglosar el coste que hubiera tenido. Nos vamos a centrar en la previsión inicial y teniendo en cuenta la mayoría de los gastos:

- Alquiler de oficina, incluida luz, agua, aseo, Internet y alarma: 250€ mensuales.

- Dos salarios mínimos mensuales españoles del año 2016 [12], para la nómina del ingeniero informático: 655,20€ * 2 (1310,4€) mensuales.
- Material y transporte: 50€ mensuales.
- Alquiler de servidor VPS de aceptable rendimiento (2 CPU y 4GB de RAM): 10€ mensuales.
- Equipamiento hardware y software inicial: 0€, ya que el desarrollador está provisto de ello.
- No se tienen en cuenta los gastos por impuestos, estos dependerán del tipo de contrato (autónomo o empleado).

Teniendo en cuenta que la previsión para la consecución del proyecto es de 11 meses, ya incluido un 30% extra de tiempo por precaución, quedaría:

Costes	Coste mensual	Coste total en 11 meses
Alquiler	250€	2750€
Nómina	1310,4€	14414,4€
Material y transporte	50€	550€
Alquiler de servidor VPS	10€	110€
Total	1620,4€	17824,4€

Tabla 4: Costes del proyecto

El coste total ascendería a 17824,4€ sin incluir ningún riesgo ni contratiempo.

2.5. Gestión de riesgos

A continuación se van a mostrar todos los riesgos que se han considerado que puedan ocurrir y se comentarán las medidas de actuación oportunas si se da el caso así como las preventivas. El modelo de riesgos se basa en el propuesto en la herramienta REM [13], con una serie de modificaciones. Ninguno de los riesgos tiene en cuenta al cliente debido a la naturaleza del proyecto.

RISK-1	Complejidad en el aprendizaje de las tecnologías
Descripción	Debido a que el desarrollo se basa en tecnologías nunca usadas anteriormente, hay que considerar el riesgo de la dificultad del aprendizaje.
Probabilidad	Alta.
Impacto	Aumento del tiempo de desarrollo, llegando incluso a no poder avanzar sin ayuda.
Plan de actuación	Dedicar más tiempo, pedir ayuda y/o cambiar de proyecto.
Medidas preventivas	Leer varios libros de las tecnologías y aprender mejor inglés ya que la mayoría de la documentación está en este idioma.

Tabla 5: Riesgo. Complejidad en el aprendizaje de las tecnologías

RISK-2	Problemas personales
Descripción	Problemas personales.
Probabilidad	Alta.
Impacto	Aumento del tiempo de desarrollo.
Plan de actuación	Dedicar más tiempo
Medidas preventivas	Ninguna.

Tabla 6: Riesgo. Problemas personales

RISK-3	Pérdida completa del sistema
Descripción	Pérdida de todo el sistema debido a fallos informáticos y eléctricos en múltiples servicios y dispositivos.
Probabilidad	Muy baja.
Impacto	Tener que empezar de cero.
Plan de actuación	Aprovechar lo aprendido para tardar lo mínimo posible. Desarrollando el mismo sistema sin añadir nuevas tecnologías ni nuevas versiones de las actualmente utilizadas.
Medidas preventivas	Guardar en múltiples dispositivos y servicios toda la documentación y el código.

Tabla 7: Riesgo. Pérdida completa del sistema

RISK-4	Rotura del hardware de trabajo
Descripción	Rotura parcial o completa del hardware con el que se está desarrollando el sistema. En este caso en concreto el portátil sería el hardware más importante seguido de otros menos importantes como el router y periféricos.
Probabilidad	Media.
Impacto	Se pierde el contenido no guardado y no se puede continuar desarrollando.
Plan de actuación	Modificar las piezas o equipos afectados lo más pronto posible.
Medidas preventivas	Revisar el estado del hardware periódicamente para evitarlo o prever su rotura además de tener localizado y/o almacenado hardware de sustitución.

Tabla 8: Riesgo. Rotura del hardware de trabajo

RISK-5	Modificación de los requisitos
Descripción	La modificación de alguno de los requisitos durante el desarrollo hay que tenerla en cuenta como riesgo.
Probabilidad	Media.
Impacto	Se puede llegar a perder parte de lo desarrollado, hay que analizar, diseñar e implementar los nuevos requisitos y se aumenta el coste temporal del proyecto.
Plan de actuación	Adaptar el sistema a los nuevos requisitos intentando reutilizar el código lo máximo posible.
Medidas preventivas	Hacer un buen análisis intuyendo las posibles peticiones de cambio. Crear un sistema modular que permita añadir y eliminar componentes del sistema de la forma más sencilla posible, evitando el acoplamiento.

Tabla 9: Riesgo. Modificación de los requisitos

Parte II

Desarrollo

3. Análisis de requisitos

En esta sección se realiza el análisis completo del sistema incluyendo los actores, casos de uso, requisitos de información, requisitos no funcionales, reglas de negocio y finalmente las tecnologías que pueden emplearse para el desarrollo.

3.1. Catálogo de actores

En la aplicación podemos encontrar dos actores principales: el sistema y el usuario. A su vez el actor usuario, registrado, puede ser administrador y/o director de juego aunque la propiedad de director de juego solo se adquiere cuando es el propietario de una partida y la dirige. Por otra parte el administrador tendrá opciones de gestión extra para poder desarrollar, corregir y moderar el sistema con la mayor comodidad posible. Finalmente tenemos al usuario anónimo que es el aquel usuario que ni está registrado ni ha accedido al sistema.

Hay que tener en cuenta que todo administrador puede ejecutar la misma funcionalidad que un usuario además de la añadida por ser administrador. En cambio un usuario no podrá ejecutar la funcionalidad de un administrador.

ACT-1	Usuario anónimo
Descripción	Es el usuario que se encuentran en la aplicación pero no ha accedido al sistema introduciendo sus credenciales.

Tabla 10: Actor. Usuario anónimo

ACT-2	Usuario
Descripción	Un usuario es todo aquel que se encuentra en el sistema porque ha accedido con sus credenciales. Esto implica que se ha registrado.

Tabla 11: Actor. Usuario

ACT-3	Administrador
Descripción	Además de usuario posee permisos extra para gestionar la aplicación.

Tabla 12: Actor. Administrador

ACT-4	Director de juego
Descripción	Este usuario adquiere los permisos de director de juego cuando crear y accede a una sesión de juego. Dentro de ella tiene permisos que el resto de usuarios no tiene, que le permiten gestionar la sesión de juego y a los usuarios que hay en ella.

Tabla 13: Actor. Director de juego

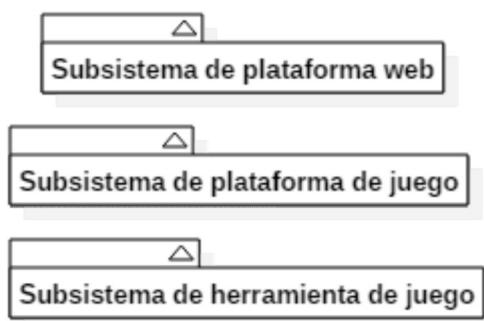
3.2. Requisitos funcionales

Debido a las cuatro iteraciones que se han hecho, la funcionalidad se ha desarrollado en tres grandes subsistemas. La cuarta iteración al haber sido solo de arreglos y mejoras de los tres subsistemas, no tiene el suyo propio.

A continuación se va desglosar cada uno de los subsistemas basado en el orden de las iteraciones.

En los diagramas de los casos de uso se utilizará un fondo gris en los casos de uso para indicar que la funcionalidad no se ha desarrollando finalmente, mientras que una línea roja entre un actor y un caso de uso indicará que ese escenario alternativo no se ha desarrollado.

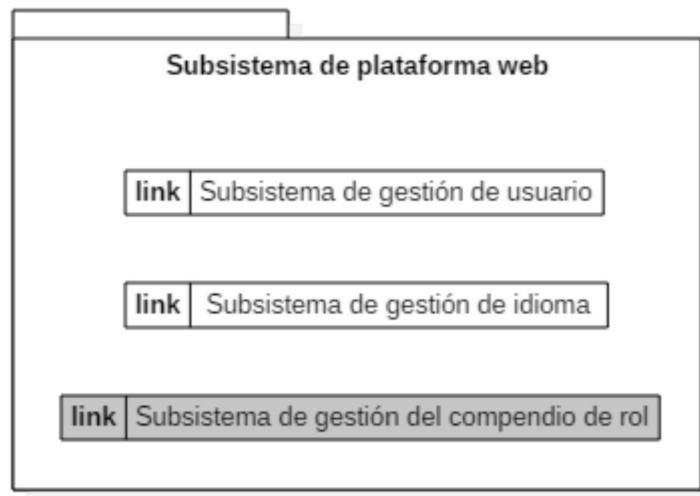
Los subsistemas generales que forman la aplicación son los siguientes:



Dibujo 7: Subsistemas

3.2.1. Subsistema de plataforma web

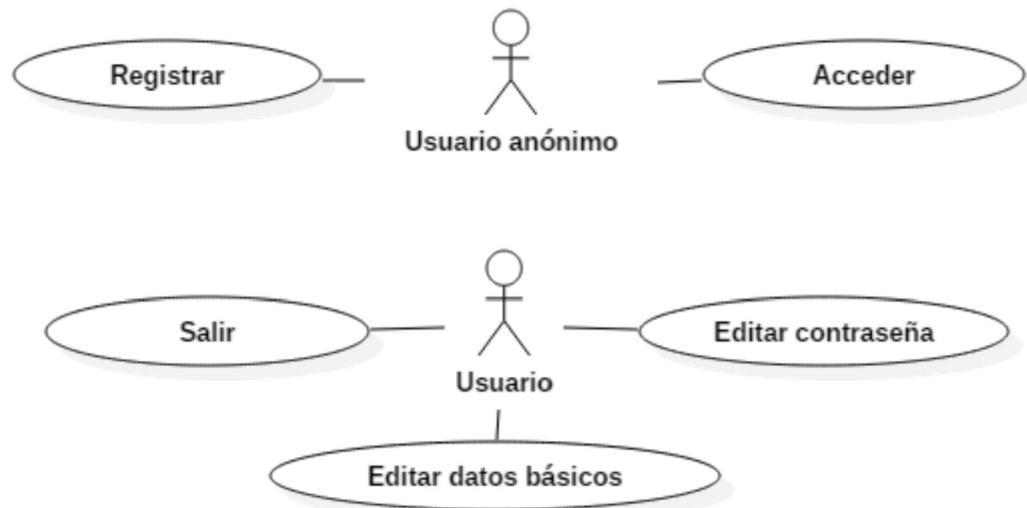
Este subsistema está pensado para incluir toda funcionalidad que sea ajena a las sesiones de juego. Por ello abarca por completo al usuario, desde su registro hasta la edición de sus datos personales, y otros subsistemas como el de gestión de idiomas y el de gestión del compendio de rol.



Dibujo 8: Subsistema de plataforma web

3.2.1.1. Subsistema de gestión de usuario

En este subsistema los casos de uso son exclusivamente para la gestión de los usuarios y el único subsistema que manejará a los usuarios anónimos.



Dibujo 9: Subsistema de gestión de usuario

A continuación se mostrarán los casos de usos de este subsistema:

UC-1	Registrar
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-1] Usuario anónimo accede a la opción de registrarse.
Precondición	El actor [ACT-1] Usuario anónimo no se encuentra accedido en el sistema, es decir es un usuario anónimo y accede a la opción de registrarse.
Secuencia normal	Acción
1	El sistema muestra al usuario el formulario necesario para el registro.
2	El actor [ACT-1] Usuario anónimo introduce todos los datos mínimos requeridos y selecciona la opción de continuar.
3	El sistema valida los datos, los almacena en la base de datos y envía al usuario un mensaje de registro exitoso.
Postcondición	El actor [ACT-1] Usuario anónimo se registra exitosamente en el sistema y queda almacenado en la base de datos.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-1] Usuario anónimo de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-1] Usuario anónimo cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ninguno.

Tabla 14: Caso de uso. Registrar

UC-2	Acceder
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-1] Usuario anónimo se dirige a la opción de acceder al sistema.
Precondición	El actor [ACT-1] Usuario anónimo no ha accedido al sistema, es decir es un usuario anónimo y se dirige a la opción de acceder al sistema.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-1] Usuario anónimo el formulario necesario para el acceso al sistema.
2	El actor [ACT-1] Usuario anónimo introduce todos los datos requeridos y selecciona la opción de continuar.
3	El sistema valida que los datos coinciden con alguno de los usuarios almacenados en la base de datos, almacena el inicio de sesión en la base de datos y envía al actor [ACT-2] Usuario un mensaje de acceso exitoso.
Postcondición	El actor [ACT-1] Usuario anónimo accede al sistema y se almacena en la base de datos y en la memoria del sistema la duración de su sesión, obteniendo los permisos del actor [ACT-2] Usuario .
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-1] Usuario anónimo de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-1] Usuario anónimo cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ninguno.

Tabla 15: Caso de uso. Acceder

UC-3	Salir
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de salir del sistema.
Precondición	El actor [ACT-2] Usuario se encuentra accedido al sistema y accede a la opción de salir del sistema.
Secuencia normal	Acción
1	El sistema elimina la sesión del actor [ACT-2] Usuario en la base de datos y en la memoria del sistema y envía al actor [ACT-1] Usuario anónimo un mensaje de desconexión exitosa.
Postcondición	El actor [ACT-2] Usuario se sale del sistema y se elimina su sesión de la base de datos y la memoria del sistema. Además se le eliminan todos los permisos de usuario y se convierte en el actor [ACT-1] Usuario anónimo .
Comentarios	Ninguno.

Tabla 16: Caso de uso. Salir

UC-4	Editar datos básicos
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de editar los datos básicos de usuario.
Precondición	El actor [ACT-2] Usuario accede a la opción de editar datos básicos del usuario.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-2] Usuario un formulario con los datos actuales del usuario.
2	El actor [ACT-2] Usuario introduce los cambios que estime y selecciona la opción de continuar.
3	El sistema valida que los nuevos datos, los almacena en la base de datos y envía al actor [ACT-2] Usuario un mensaje de modificación exitosa.
Postcondición	Los nuevos datos del actor [ACT-2] Usuario quedan registrados correctamente en la base de datos del sistema.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-2] Usuario de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ninguno.

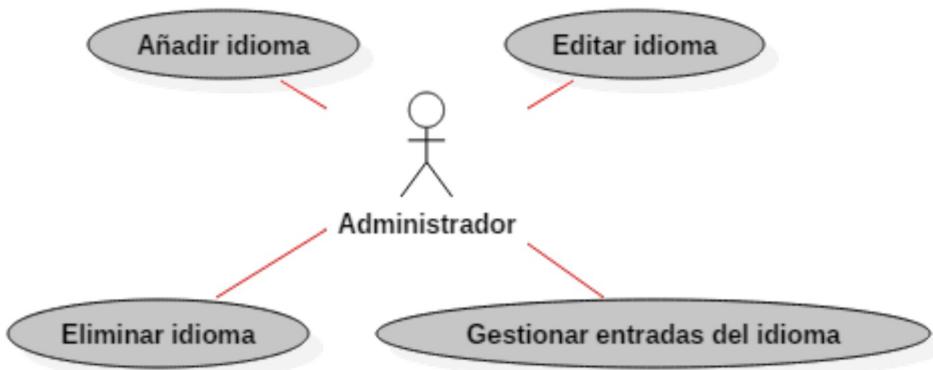
Tabla 17: Caso de uso. Editar datos básicos

UC-5	Editar contraseña
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de editar la contraseña de usuario.
Precondición	El actor [ACT-2] Usuario accede a la opción de editar la contraseña de usuario.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-2] Usuario el formulario necesario para editar la contraseña.
2	El actor [ACT-2] Usuario introduce los datos necesarios y selecciona la opción de continuar.
3	El sistema valida que los nuevos datos, los almacena en la base de datos y envía al actor [ACT-2] Usuario un mensaje de modificación exitosa.
Postcondición	La nueva contraseña del actor [ACT-2] Usuario queda registrada correctamente en la base de datos del sistema.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-2] Usuario de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Aunque cambie la contraseña el actor [ACT-2] Usuario debe continuar en el sistema y no pedirle un nuevo acceso.

Tabla 18: Caso de uso. Editar contraseña

3.2.1.2. Subsistema de gestión de idioma

En este subsistema los casos de uso se centran en la gestión de los idiomas del sistema, donde permiten al administrador gestionar de forma más rápida y segura todos los datos de los mismos así como las entradas de cada una de las traducciones. Esta funcionalidad no ha sido desarrollada posteriormente ya que se introducen directamente a través de una generación inicial de datos para el sistema.



Dibujo 10: Subsistema de gestión de idioma

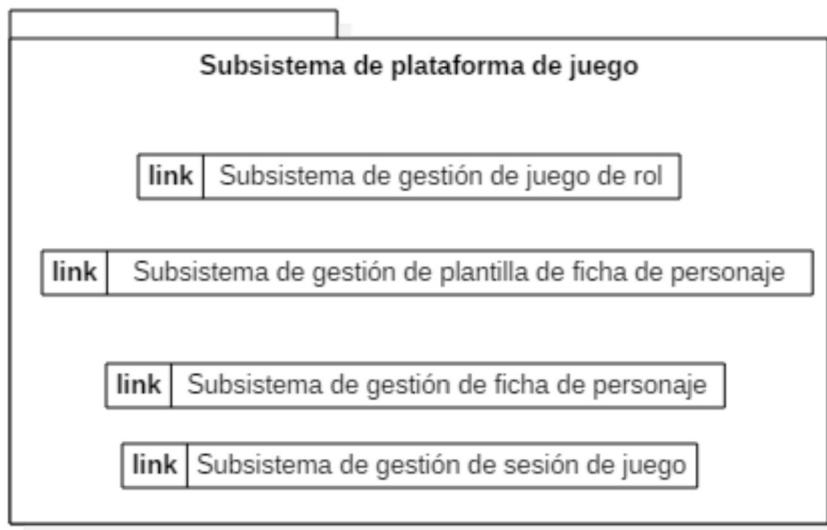
3.2.1.3. Subsistema de gestión del compendio de rol

Este subsistema está planteado para gestionar un compendio con toda la información de los juegos de rol. Donde los administradores puedan añadir información y los usuarios visualizarlas como si de una guía de juegos de rol se tratase. Esta funcionalidad no ha sido finalmente seleccionada para desarrollar.

3.2.2. Subsistema de plataforma de juego

En la segunda iteración se planteó el subsistema de la plataforma de juego que es donde se encuentra toda la lógica que permite a los usuarios gestionar las sesiones de juego y las fichas de personaje. Por otro lado les permite a los administradores gestionar los juegos de rol y las plantillas de fichas de personaje.

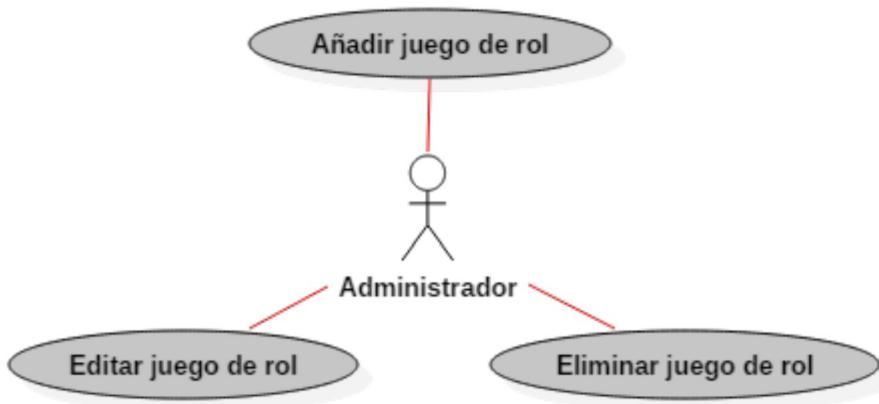
En este subsistema no se encuentra la funcionalidad del juego en sí, si no los sistemas de gestión necesarios para la correcta funcionalidad de una partida.



Dibujo 11: Subsistema de plataforma de juego

3.2.2.1. Subsistema de gestión de juego de rol

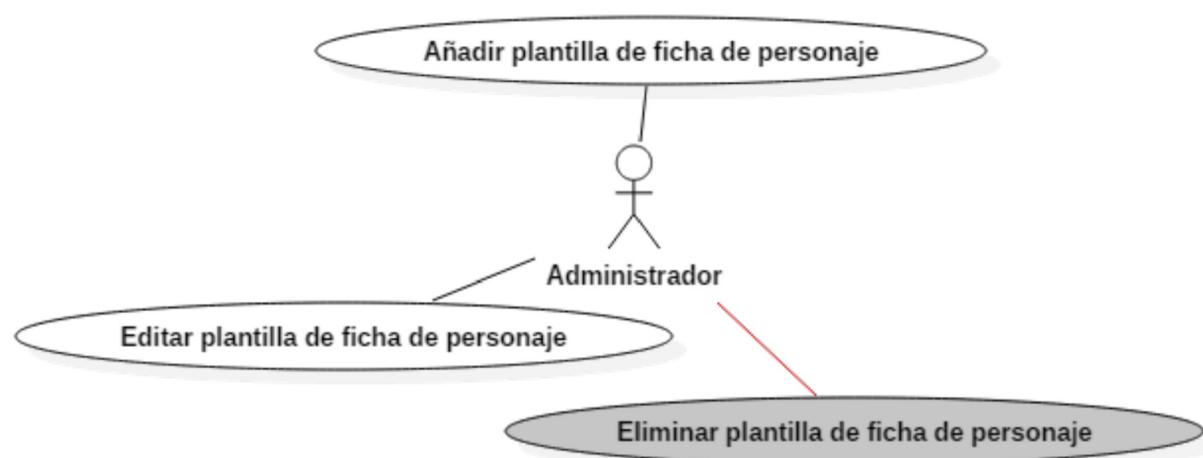
En este subsistema los casos de uso se centran en la gestión de los juegos de rol, permitiendo al administrador gestionar de forma más rápida y segura todos los datos de los mismos. Esta funcionalidad no ha sido desarrollada posteriormente ya que se introducen directamente a través de una generación inicial de datos para el sistema.



Dibujo 12: Subsistema de gestión de juego de rol

3.2.2.2. Subsistema de gestión de plantilla de ficha de personaje

En este subsistema los casos de uso se centran en la gestión de las plantillas de las fichas de personaje, permitiendo al administrador gestionar de forma más rápida y segura los datos de las mismas. De esta funcionalidad ha sido desarrollada el añadido y la edición de las plantillas.



Dibujo 13: Subsistema de gestión de plantilla de ficha de personaje

UC-6	Añadir plantilla de ficha de personaje
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-3] Administrador accede a la opción de añadir plantilla de ficha de personaje.
Precondición	El actor [ACT-3] Administrador accede a la opción de añadir plantilla de ficha de personaje.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-3] Administrador el formulario necesario para añadir una plantilla de ficha de personaje.
2	El actor [ACT-3] Administrador selecciona el juego de rol al que pertenece e introduce los datos necesarios y selecciona la opción de continuar.
3	El sistema valida que los nuevos datos, los almacena en la base de datos y envía al actor [ACT-3] Administrador un mensaje de creación exitosa.
Postcondición	La nueva plantilla de ficha de personaje queda registrada correctamente en la base de datos del sistema y asociada al juego de rol.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-3] Administrador de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-3] Administrador cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ninguno.

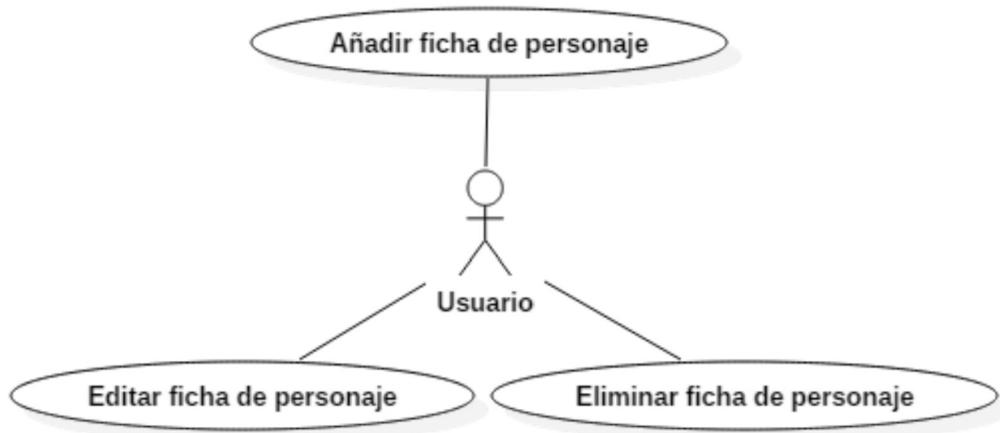
Tabla 19: Caso de uso. Añadir plantilla de ficha de personaje

UC-7	Editar plantilla de ficha de personaje
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-3] Administrador accede a la opción de editar plantilla de ficha de personaje.
Precondición	El actor [ACT-3] Administrador accede a la opción de editar plantilla de ficha de personaje.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-3] Administrador el formulario con los datos almacenados para editar una plantilla de ficha de personaje.
2	El actor [ACT-3] Administrador introduce los cambios oportunos y selecciona la opción de continuar.
3	El sistema valida los nuevos datos, los almacena en la base de datos y envía al actor [ACT-3] Administrador un mensaje de modificación exitosa.
Postcondición	La plantilla de ficha de personaje queda actualizada correctamente en la base de datos del sistema.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-3] Administrador de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-3] Administrador cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ninguno.

Tabla 20: Caso de uso. Editar plantilla de ficha de personaje

3.2.2.3. Subsistema de gestión de ficha de personaje

En este subsistema los casos de uso se centran en la gestión de las fichas de personaje, permitiendo a los usuarios añadir sus fichas de personaje, editarlas y eliminarlas.



Dibujo 14: Subsistema de gestión de ficha de personaje

UC-8	Añadir ficha de personaje
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de añadir ficha de personaje.
Precondición	El actor [ACT-2] Usuario accede a la opción de añadir ficha de personaje.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-2] Usuario el formulario necesario para elegir el juego de rol y la plantilla de ficha de personaje de la que desea crear su ficha de personaje.
2	El actor [ACT-2] Usuario selecciona el juego de rol y la plantilla de ficha de personaje y selecciona la opción de continuar.
3	El sistema valida que los datos y muestra al actor [ACT-2] Usuario el formulario necesario para añadir una ficha de personaje de la plantilla de ficha de personaje seleccionada.
4	El actor [ACT-2] Usuario introduce los datos en el formulario y selecciona la opción de continuar.
5	El sistema valida los datos de la ficha de personaje, los almacena en la base de datos y envía al actor [ACT-2] Usuario un mensaje de creación exitosa.
Postcondición	La nueva ficha de personaje queda registrada correctamente en la base de datos del sistema y asociada a la plantilla de ficha de personaje elegida.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al

	actor [ACT-2] Usuario de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
5b	Al menos uno de los datos no es válido, el sistema vuelve al paso 3 y notifica al actor [ACT-2] Usuario de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-4c	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	<p>Aunque en el paso dos el juego de rol y la plantilla de ficha de personaje sean seleccionables, hay que comprobar que los datos que se le pasen al servidor sean correctos.</p> <p>La ficha de personaje queda asociada a la plantilla de ficha de personaje pero no directamente al juego de rol. A través de la plantilla de ficha de personaje se asocia la ficha de personaje con el juego de rol.</p>

Tabla 21: Caso de uso. Añadir ficha de personaje

UC-9	Editar ficha de personaje
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de editar ficha de personaje.
Precondición	El actor [ACT-2] Usuario accede a la opción de editar ficha de personaje.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-2] Usuario el formulario con los datos de la ficha de personaje elegida.
2	El actor [ACT-2] Usuario modifica los datos que deseé y selecciona la opción de continuar.
3	El sistema valida los nuevos datos de la ficha de personaje, los almacena en la base de datos y envía al actor [ACT-2] Usuario un mensaje de modificación exitosa.
Postcondición	La ficha de personaje queda modificada correctamente en la base de datos del sistema.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-2] Usuario de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Cuando se modifica una ficha de personaje no se puede modificar el juego de rol ni la plantilla de ficha de personaje a la que está asociada.

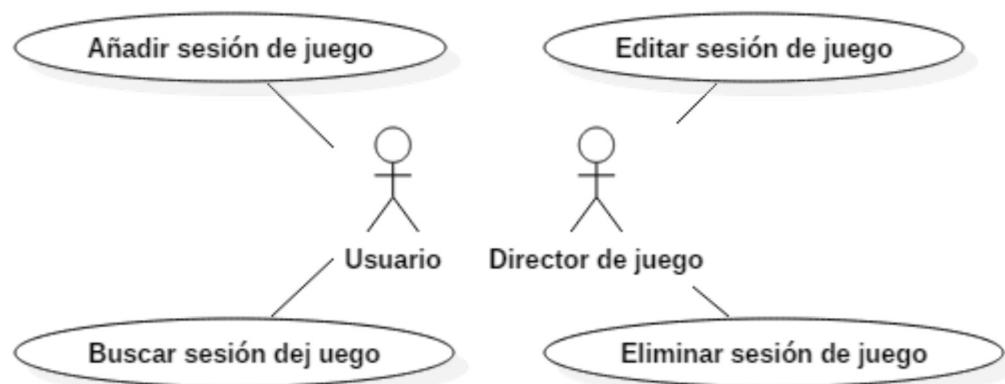
Tabla 22: Caso de uso. Editar ficha de personaje

UC-10	Eliminar ficha de personaje
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de eliminar ficha de personaje.
Precondición	El actor [ACT-2] Usuario accede a la opción de eliminar ficha de personaje y la ficha de personaje está presente en el sistema.
Secuencia normal	Acción
1	El sistema envía un mensaje de confirmación al actor [ACT-2] Usuario para realizar la eliminación de la ficha de personaje seleccionada.
2	El actor [ACT-2] Usuario confirma el mensaje seleccionando la opción de continuar.
3	El sistema valida que el actor [ACT-2] Usuario es propietario o administrador de la ficha en cuestión, la borra de la base de datos y envía al actor [ACT-2] Usuario un mensaje de eliminación exitosa.
Postcondición	La ficha de personaje queda eliminada correctamente en la base de datos del sistema.
Alternativas	Acción
2a	El actor [ACT-2] Usuario selecciona la opción cancelar, es decir, no confirma la eliminación. El caso de uso queda sin efecto y finaliza.
Excepciones	Acción
3b	El sistema verifica que la ficha de personaje que se desea borrar no existe en el sistema, el actor [ACT-2] Usuario no es propietario o administrador. El caso de uso queda sin efecto y finaliza.
1-2c	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ninguno.

Tabla 23: Caso de uso. Eliminar ficha de personaje

3.2.2.4. Subsistema de gestión de sesión de juego

En este subsistema los casos de uso se centran en la gestión de las sesiones de juego ofreciendo a los usuarios toda la funcionalidad necesaria para poder gestionar las sesiones de juego correctamente.



Dibujo 15: Subsistema de gestión de sesión de juego

UC-11	Añadir sesión de juego
Dependencias	[OBJ-1] Crear sistema de plataforma de juego [OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de añadir sesión de juego.
Precondición	El actor [ACT-2] Usuario accede a la opción de añadir sesión de juego.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-2] Usuario el formulario necesario crear la sesión de juego.
2	El actor [ACT-2] Usuario selecciona el juego de rol, el idioma e introduce el resto de los datos en el formulario y selecciona la opción de continuar.
3	El sistema valida los datos de la sesión de juego, los almacena en la base de datos, genera la sesión de juego en el servidor y redirige al actor [ACT-2] Usuario a la ruta de la sesión de juego dándole los permisos del actor [ACT-4] Director de juego .
Postcondición	La nueva sesión de juego queda registrada correctamente en la base de datos del sistema, asociada al actor [ACT-2] Usuario y adquiriendo los permisos del actor [ACT-4] Director de juego para esa sesión de juego. Además la sesión de juego se asocia al idioma y al juego de rol elegidos.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-2] Usuario de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Cuando el actor [ACT-2] Usuario entra en la sesión de juego recién creada, el servidor de sesiones de juego crea la sesión de juego en memoria, permitiendo así persistir los datos que se estén usando durante la partida y mientras el servidor está operativo.

Tabla 24: Caso de uso. Añadir sesión de juego

UC-12	Editar sesión de juego
Dependencias	[OBJ-1] Crear sistema de plataforma de juego [OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-4] Director de juego accede a la opción de editar sesión de juego.
Precondición	El actor [ACT-4] Director de juego accede a la opción de editar sesión de juego.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-4] Director de juego el formulario con los datos de la sesión de juego elegida.
2	El actor [ACT-4] Director de juego modifica los datos que deseé y selecciona la opción de continuar.
3	El sistema valida los nuevos datos de la sesión de juego, los almacena en la base de datos, envía a todos los usuarios que se encuentren en la sesión de juego los cambios pertinentes para actualizarlos y envía al actor [ACT-4] Director de juego un mensaje de modificación exitosa.
Postcondición	La sesión de juego queda modificada correctamente en la base de datos del sistema.
Excepciones	Acción
3a	Al menos uno de los datos no es válido, el sistema vuelve al paso 1 y notifica al actor [ACT-4] Director de juego de los datos incorrectos y el motivo. A continuación este caso de uso continúa.
1-2b	El actor [ACT-4] Director de juego cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Cuando se modifica una sesión de juego no se puede modificar el juego de rol ni el idioma a la que está asociada.

Tabla 25: Caso de uso. Editar sesión de juego

UC-13	Eliminar sesión de juego
Dependencias	[OBJ-1] Crear sistema de plataforma de juego [OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-4] Director de juego accede a la opción de eliminar sesión de juego.
Precondición	El actor [ACT-4] Director de juego accede a la opción de eliminar sesión de juego y la sesión de juego está presente en el sistema.
Secuencia normal	Acción
1	El sistema envía un mensaje de confirmación al actor [ACT-4] Director de juego para realizar la eliminación de la sesión de juego.
2	El usuario confirma el mensaje seleccionando la opción de continuar.
3	El sistema valida que el actor [ACT-4] Director de juego es propietario o administrador de la sesión de juego en cuestión, la borra de la base de datos, expulsa a todos los actores [ACT-2] Usuario que estuvieran en la sesión de juego, incluido el que la elimina, redirigiéndolos a otra ruta para informarles. Finalmente envía al actor [ACT-4] Director de juego un mensaje de eliminación exitosa.
Postcondición	La sesión de juego queda eliminada correctamente en la base de datos y la memoria del sistema.
Alternativas	Acción
2a	El actor [ACT-4] Director de juego selecciona la opción cancelar, es decir, no confirma la eliminación. El caso de uso queda sin efecto y finaliza.
Excepciones	Acción
3b	El sistema verifica que la sesión de juego que se desea borrar no existe en el sistema, el actor [ACT-4] Director de juego no es propietario o administrador. El caso de uso queda sin efecto y finaliza.
1-2c	El actor [ACT-4] Director de juego cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Aunque poder acceder a esta opción hay que ser actor [ACT-4] Director de juego de la sesión de juego y esto implica que tienes que ser el propietario, el sistema deberá comprobar por seguridad que sea el propietario de forma independiente.

Tabla 26: Caso de uso. Eliminar sesión de juego

UC-14	Buscar sesión de juego
Dependencias	[OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de buscar sesión de juego.
Precondición	El actor [ACT-2] Usuario accede a la opción de buscar sesión de juego.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-2] Usuario un formulario de búsqueda para la sesión de juego.
2	El actor [ACT-2] Usuario introduce la búsqueda deseada con los datos oportunos y selecciona la opción de continuar.
3	El sistema buscará toda sesión de juego con un parecido en su nombre, director de juego y comentarios con la búsqueda introducida. Finalmente mostrará al actor [ACT-2] Usuario el resultado de todas las semejanzas encontradas.
Postcondición	El sistema muestra al actor [ACT-2] Usuario todas las sesiones de juego encontradas con la búsqueda realizada.
Alternativas	Acción
3a	No existe ninguna sesión de juego semejante a la búsqueda realizada y el sistema muestra al actor [ACT-2] Usuario que no ha sido posible encontrar ninguna.
Excepciones	Acción
1-2b	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	La búsqueda en cuestión puede realizarse desde una barra de búsqueda que se encuentre en cualquier lugar de la plataforma web.

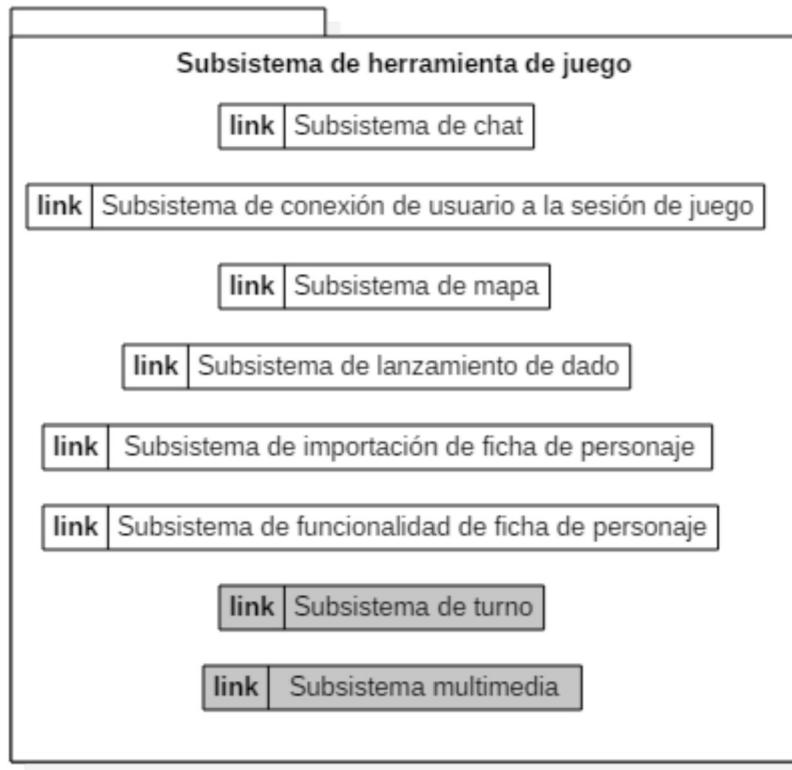
Tabla 27: Caso de uso. Buscar sesión de juego

3.2.3. Subsistema de herramienta de juego

En la tercera iteración se plantearon todas las herramientas que debería tener la sesión de juego para que los usuarios pudieran jugar de la mejor manera posible, asemejándolo lo máximo posible a una partida de rol de mesa.

En este subsistema si se encuentra la funcionalidad del juego por lo que se irán detallando una a una las herramientas de la forma más modularizada posible para poder desarrollarlas con el menor acoplamiento.

Hay que tener en cuenta de que al tratarse de una aplicación web, cada cliente tiene que gestionar su propia interfaz teniendo en cuenta las directrices del servidor.

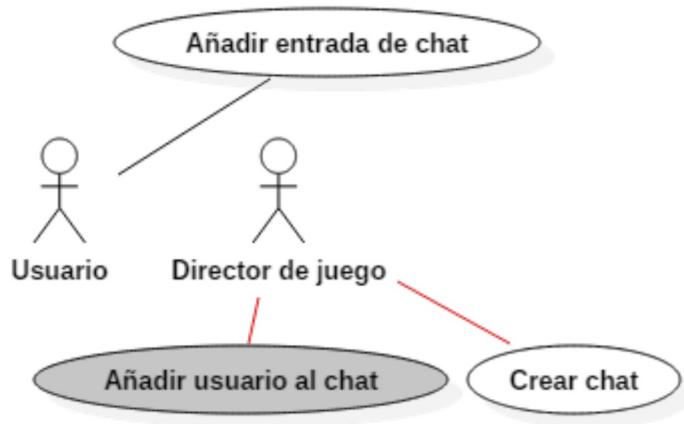


Dibujo 16: Subsistema de herramienta de juego

3.2.3.1. Subsistema de chat

En este subsistema los casos de uso se centran en la gestión de los *chats* que se encuentran en la sesiones de juego, permitiendo a los usuarios comunicarse y ver los resultados de los eventos que activan como la tirada de dados, ejecución de funcionalidad, conexión y desconexión de usuarios e importar fichas como alguna de ellas.

Al principio se barajaba la opción de poder tener *multichats*, es decir, un chat general y otros que pudieran crear los usuarios para chatear de forma privada. Al final se ha optado porque el sistema cree un único chat que usen todos los usuarios y toda la comunicación vaya a través de él.



Dibujo 17: Subsistema de chat

UC-15	Crear chat
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se reciba la petición de crear el chat para el usuario.
Precondición	El actor [ACT-2] Usuario acaba de unirse a una sesión de juego y el actor el sistema solicita la funcionalidad crear chat.
Secuencia normal	Acción
1	El sistema genera toda la interfaz del chat y añade todo el historial almacenado referente a ese chat para que el actor [ACT-2] Usuario lo visualice.
Postcondición	El sistema provee al usuario del chat y todo su historial.
Excepciones	Acción
*	El actor [ACT-2] Usuario se desconecta antes de finalizar el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	El chat se crea en el cliente de cada usuario y el historial mostrado es solo el que se ha almacenado cuando el usuario se encontraba conectado.

Tabla 28: Caso de uso. Crear chat

UC-16	Añadir entrada de chat
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario añade una entrada un chat.
Precondición	El chat está creado y el actor [ACT-2] Usuario puede escribir por él.
Secuencia normal	Acción
1	El actor [ACT-2] Usuario introduce el texto y selecciona la opción de enviar.
2	El sistema valida que el texto enviado es seguro y correcto.
3	El sistema envía al resto de actores [ACT-2] Usuario de la sesión de juego el texto y lo almacena en el historial de cada uno de ellos de forma independiente.
Postcondición	El sistema envía al resto de actores [ACT-2] Usuario de la sesión de juego el texto y lo almacena en el historial de cada uno de ellos de forma independiente.
Alternativas	Acción
1a	El sistema, debido a un desencadenante, es el que envía un texto al resto de actores [ACT-2] Usuario . Continúa en el paso 3.
2b	El texto no es seguro o correcto por lo que el sistema notifica al actor [ACT-2] Usuario en cuestión.
3c	El texto es un comando específico (como el susurro a otro usuario) por lo que el sistema lo interpreta y realiza la gestión pertinente dependiendo del comando.
Excepciones	Acción
*	El actor [ACT-2] Usuario se desconecta del sistema sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Muchas de las entradas del sistema son enviadas por otras herramientas para que los actores [ACT-2] Usuario visualicen los resultados de las mismas.

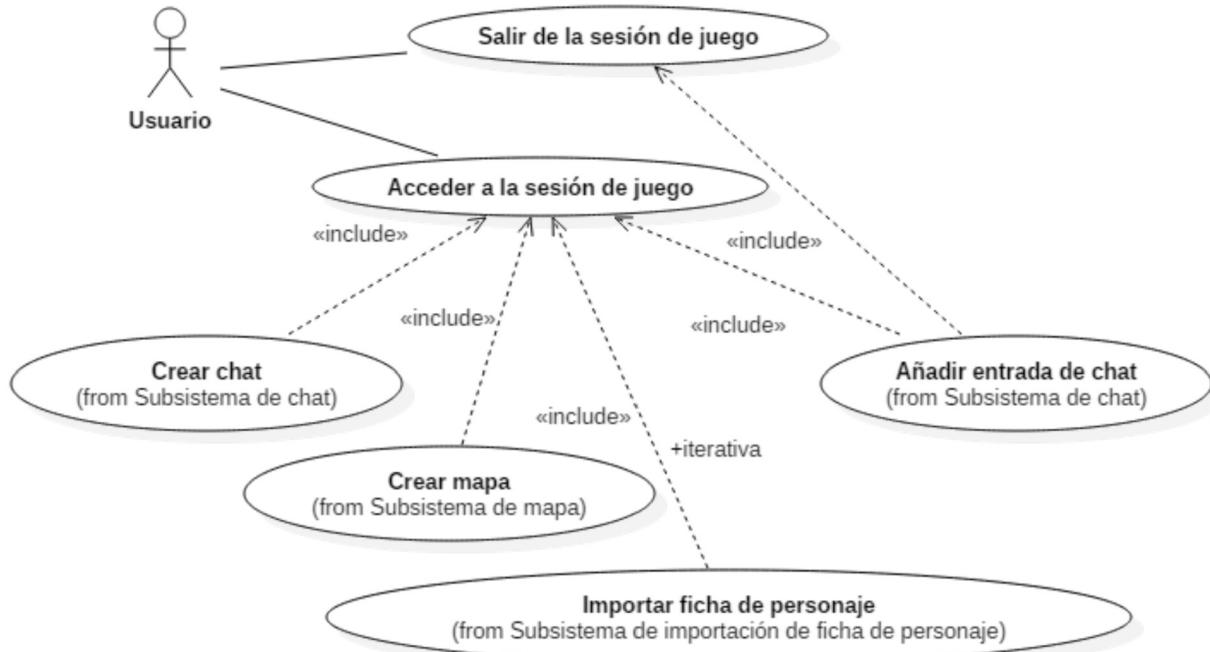
Tabla 29: Caso de uso. Añadir entrada de chat

3.2.3.2. Subsistema de conexión de usuario a la sesión de juego

En este subsistema los casos de uso se centran en la conexión y desconexión de los usuarios en las sesiones de juego. Este proceso tiene que incorporar los permisos para acceder y de la funcionalidad que puede realizar ya que el usuario puede realizar las funciones de director de juego o no.

La conexión y la desconexión a la sesión de juego se debería tramitar de un servidor (el web) al otro (de sesiones de juego).

Hay que tener en cuenta que gran parte de la funcionalidad es almacenada en memoria para que los usuarios se puedan conectar y reconectar y reciban toda la información presente en la sesión de juego de los diferentes subsistemas.



Dibujo 18: Subsistema de conexión de usuario a la sesión de juego

UC-17	Acceder a la sesión de juego						
Dependencias	[OBJ-1] Crear sistema de plataforma de juego [OBJ-2] Crear sistema de plataforma web						
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de acceder a la sesión de juego. Un actor [ACT-2] Usuario no puede acceder más de una vez de forma simultánea a una misma sesión de juego						
Precondición	El actor [ACT-2] Usuario accede a la opción de acceder a la sesión de juego de una sesión de juego concreta.						
Secuencia normal	<p style="text-align: center;">Acción</p> <table> <tr> <td style="vertical-align: top; padding-right: 10px;">1</td><td>El sistema verifica que la sesión de juego existe y que el actor [ACT-2] Usuario no se encuentra conectado a la sesión de juego. Además muestra al actor [ACT-2] Usuario un formulario de acceso.</td></tr> <tr> <td style="vertical-align: top; padding-right: 10px;">2</td><td>El actor [ACT-2] Usuario introduce la contraseña requerida y selecciona la opción de continuar.</td></tr> <tr> <td style="vertical-align: top; padding-right: 10px;">3</td><td>El sistema verifica que la contraseña es correcta y lo redirigirá a la sesión de</td></tr> </table>	1	El sistema verifica que la sesión de juego existe y que el actor [ACT-2] Usuario no se encuentra conectado a la sesión de juego. Además muestra al actor [ACT-2] Usuario un formulario de acceso.	2	El actor [ACT-2] Usuario introduce la contraseña requerida y selecciona la opción de continuar.	3	El sistema verifica que la contraseña es correcta y lo redirigirá a la sesión de
1	El sistema verifica que la sesión de juego existe y que el actor [ACT-2] Usuario no se encuentra conectado a la sesión de juego. Además muestra al actor [ACT-2] Usuario un formulario de acceso.						
2	El actor [ACT-2] Usuario introduce la contraseña requerida y selecciona la opción de continuar.						
3	El sistema verifica que la contraseña es correcta y lo redirigirá a la sesión de						

	juego en cuestión y comunicará a los demás actores [ACT-2] Usuario de la sesión de juego la nueva conexión. Además el actor [ACT-2] Usuario recibirá todos los datos que se encuentran actualmente en la memoria de la sesión de juego incluyendo los casos de uso: [UC-0015] Crear chat , [UC-0019] Crear mapa e [UC-0025] Importar ficha de personaje .
Postcondición	El sistema redirige al actor [ACT-2] Usuario a la sesión de juego, avisará a los demás actores [ACT-2] Usuario de su conexión y se le enviarán los datos en memoria de la sesión de juego.
Alternativas	Acción
1a	El actor [ACT-2] Usuario es el propietario de la sesión de juego y el sistema lo redirige a la sesión de juego avisando a los demás actores [ACT-2] Usuario de su conexión.
1b	El actor [ACT-2] Usuario ya se encuentra conectado en la sesión de juego por lo que se le enviará un mensaje notificándole de que no puede acceder más de una vez de forma simultánea a una misma sesión de juego.
1c	Si un actor [ACT-2] Usuario intenta acceder por otro método (botón atrás o copiar la URL de la sesión de juego) el sistema lo redirigirá al paso 1.
Excepciones	Acción
1d	La sesión de juego no existe en el sistema. El caso de uso queda sin efecto y finaliza.
1-2f	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
*	El actor [ACT-2] Usuario sale de la sesión de juego antes de finalizar el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	A tratarse de dos servidores diferentes es necesario una comunicación a través de la base de datos para verificar las conexiones de los usuarios.

Tabla 30: Caso de uso. Acceder a la sesión de juego

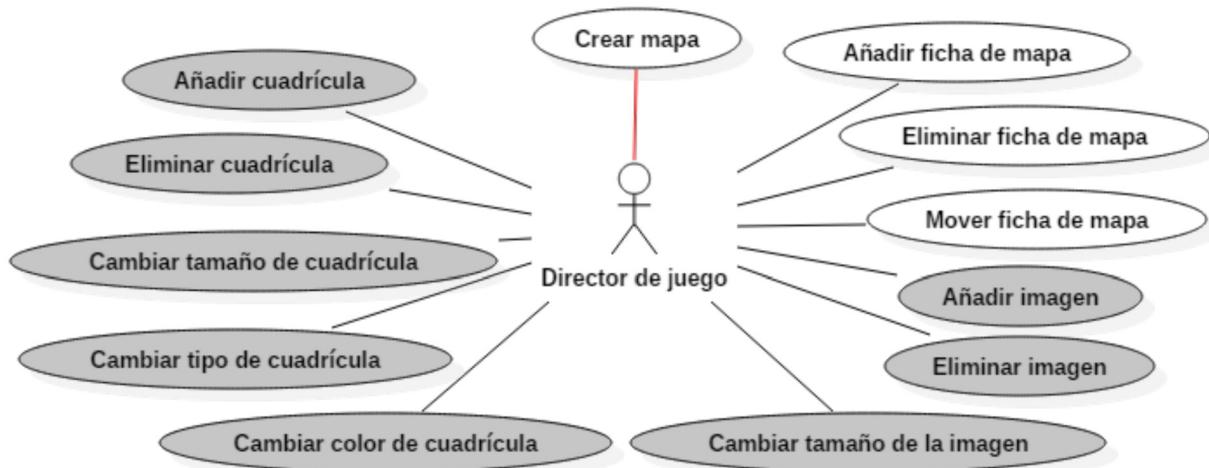
UC-18	Salir de la sesión de juego
Dependencias	[OBJ-1] Crear sistema de plataforma de juego [OBJ-2] Crear sistema de plataforma web
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario accede a la opción de salir de la sesión de juego.
Precondición	El actor [ACT-2] Usuario se encuentra en la sesión de juego.
Secuencia normal	Acción
1	El actor [ACT-2] Usuario accede a la opción de salir de la sesión de juego.
2	El sistema verifica que el actor [ACT-2] Usuario se encuentra conectado en la sesión de juego y le envía una petición de confirmación.
3	El actor [ACT-2] Usuario confirma la desconexión.
4	El sistema redirige fuera al actor [ACT-2] Usuario y le notificará a él y a los actores [ACT-2] Usuario aún conectados en la sesión de juego de su desconexión incluyendo el caso de uso [UC-0015] Crear chat .
Postcondición	El sistema redirige al actor [ACT-2] Usuario fuera de la sesión de juego y lo notificará a él y a los demás actores [ACT-2] Usuario aún conectados en la sesión de juego de su desconexión.
Alternativas	Acción
1a	El actor [ACT-4] Director de juego es el que expulsa al usuario [ACT-2] Usuario . Continúa en el paso 4.
3b	El actor [ACT-2] Usuario no confirma salir de la sesión de juego. El caso de uso queda sin efecto y finaliza.
Excepciones	Acción
1-3c	El actor [ACT-2] Usuario cancela en cualquier momento el caso de uso. El caso de uso queda sin efecto y finaliza.
*	El actor [ACT-2] Usuario sale de la sesión de juego por otros medios como cerrar la pestaña, la ventana, el navegador o apagar el equipo. El sistema va al paso 3 y no redirige ni notifica al actor [ACT-2] Usuario desconectado.
Comentarios	Cuando el actor [ACT-2] Usuario se desconecte sus elementos importados se deberán mantener en la memoria de la sesión de juego para poder ser utilizados por el actor [ACT-4] Director de juego o si se vuelve a conectar el actor [ACT-2] Usuario .

Tabla 31: Caso de uso. Salir de la sesión de juego

3.2.3.3. Subsistema de mapa

En este subsistema los casos de uso se centran en el mapa de la sesión de juego donde los usuarios pueden visualizar de forma eficaz el entorno de la partida que están desarrollando.

Mucha de la funcionalidad que se propuso finalmente se descartó debido a su complejidad gráfica, también la posibilidad de que el director de juego pudiera crear varios mapas e irlos cambiando con forme vaya necesitando. Así que finalmente se optó por lo mínimo e imprescindible para mantener una visualización de la partida con garantía.



Dibujo 19: Subsistema de mapa

UC-19	Crear mapa
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el sistema recibe la petición de crear mapa para el actor [ACT-2] Usuario .
Precondición	El actor [ACT-2] Usuario acaba de unirse a una sesión de juego y se le envía una petición al sistema de crear el chat para el actor [ACT-2] Usuario .
Secuencia normal	Acción
1	El sistema genera toda la interfaz del mapa y añade los elementos asociados a ese mapa que se encuentran actualmente en la sesión de juego para que el actor [ACT-2] Usuario lo visualice.
Postcondición	El sistema provee al actor [ACT-2] Usuario del mapa y todos sus elementos asociados, que se encuentren actualmente en la sesión de juego.
Alternativas	Acción
1a	El actor [ACT-2] Usuario es el actor [ACT-4] Director de juego por lo que el sistema le provee de funcionalidad adicional para poder añadir y eliminar las fichas de mapa. Continúa en el paso 1.
Excepciones	Acción
*	El actor [ACT-2] Usuario sale de la sesión de juego antes de finalizar el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	El mapa se crea en el cliente de cada actor [ACT-2] Usuario y los elementos asociados son los que se encuentran actualmente en la partida en el momento de conectarse.

Tabla 32: Caso de uso. Crear mapa

UC-20	Añadir ficha de mapa
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-4] Director de juego añade una ficha de mapa.
Precondición	El mapa está creado y el actor [ACT-4] Director de juego tiene la funcionalidad necesaria para añadir la ficha de mapa, seleccionando entre otras cosas el color de la ficha.
Secuencia normal	Acción
1	El actor [ACT-4] Director de juego introduce la configuración de la ficha que desea añadir selecciona la opción de continuar.
2	El sistema valida que la configuración es correcta.
3	El sistema envía a todos los actores [ACT-2] Usuario conectados en la sesión de juego la ficha a añadir, la almacena en memoria las coordenadas (0,0) junto con su configuración y provee al actor [ACT-4] Director de juego que la creó de la funcionalidad de moverla y eliminarla.
Postcondición	El sistema envía a todos los actores [ACT-2] Usuario conectados en la sesión de juego la ficha a añadir y provee al actor [ACT-4] Director de juego que la creó la funcionalidad de moverla.
Alternativas	Acción
2a	El sistema detecta que la configuración no es válida por lo que notifica al actor [ACT-4] Director de juego de lo ocurrido. El caso de uso queda sin efecto y finaliza.
Excepciones	Acción
*	El actor [ACT-4] Director de juego sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ningún actor [ACT-2] Usuario que no sea el actor [ACT-4] Director de juego no debe tener ninguna funcionalidad que no sea la mera visualización.

Tabla 33: Caso de uso. Añadir ficha de mapa

UC-21	Eliminar ficha de mapa
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-4] Director de juego elimina una ficha de mapa.
Precondición	El mapa está creado y el actor [ACT-4] Director de juego tiene la funcionalidad necesaria para eliminar la ficha de mapa.
Secuencia normal	Acción
1	El actor [ACT-4] Director de juego selecciona la ficha que desea eliminar y selecciona la opción de eliminar.
2	El sistema pide confirmación de eliminación al actor [ACT-4] Director de juego a través de una ventana de confirmación.
3	El actor [ACT-4] Director de juego confirma la eliminación.
4	El sistema valida que la ficha existe.
5	El sistema elimina la ficha de mapa de todos los actores [ACT-2] Usuario conectados de la sesión de juego y la elimina de la memoria de la sesión de juego.
Postcondición	El sistema envía a todos los actores [ACT-2] Usuario conectados en la sesión de juego la ficha a eliminar y la elimina de la memoria de la sesión de juego.
Alternativas	Acción
3a	El actor [ACT-4] Director de juego cancela la eliminación, es decir, no la confirma. El caso de uso queda sin efecto y finaliza.
Excepciones	Acción
4b	La ficha a eliminar no existe, es decir, no se encuentra en la memoria de la sesión de juego, el sistema avisa al actor [ACT-4] Director de juego . El caso de uso queda sin efecto y finaliza.
*	El actor [ACT-4] Director de juego sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ningún actor [ACT-2] Usuario que no sea el actor [ACT-4] Director de juego no debe tener ninguna funcionalidad que no sea la mera visualización. Hay que seleccionar la ficha previamente antes de eliminarla por lo que habrá que hay que incluir la funcionalidad de seleccionar la ficha.

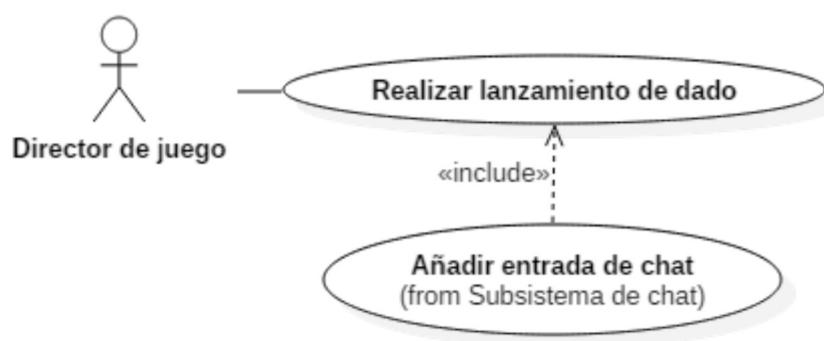
Tabla 34: Caso de uso. Eliminar ficha de mapa

UC-22	Mover ficha de mapa
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-4] Director de juego mueve una ficha de mapa.
Precondición	El mapa está creado y el actor [ACT-4] Director de juego tiene la funcionalidad necesaria para mover la ficha de mapa.
Secuencia normal	Acción
1	El actor [ACT-4] Director de juego arrastra por el mapa la ficha que desea y deja de arrastrarla.
2	Cuando finaliza el desplazamiento, el sistema verifica que el movimiento se encuentra dentro de los márgenes permitidos, actualiza la posición de la ficha de mapa del resto de actores [ACT-2] Usuario conectados a la sesión de juego y actualiza las coordenadas en la memoria de la sesión de juego.
Postcondición	El sistema actualiza la posición de la ficha de mapa del resto de actores [ACT-2] Usuario conectados a la sesión de juego y actualiza las coordenadas en la memoria de la sesión de juego.
Alternativas	Acción
2a	El sistema detecta que las nuevas coordenadas no son válidas por lo que notifica al actor [ACT-4] Director de juego de lo ocurrido. El caso de uso queda sin efecto y finaliza.
Excepciones	Acción
*	El actor [ACT-4] Director de juego sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ningún actor [ACT-2] Usuario que no sea el actor [ACT-4] Director de juego no debe tener ninguna funcionalidad que no sea la mera visualización.

Tabla 35: Caso de uso. Mover ficha de mapa

3.2.3.4. Subsistema de lanzamiento de dado

En este subsistema los casos de uso se centran en lanzamiento de dados. En los juegos de rol suele haber varios tipos de dados a la hora de realizar las tiradas que pide o realiza el director de juego. Los dados más comunes son: 4, 6, 8, 10, 12, 20 y 100 caras. Hay que tener en cuenta que no tiene porqué tirarse un solo dado si no varios del mismo o una combinación de varios tipos dados y diferentes cantidades.



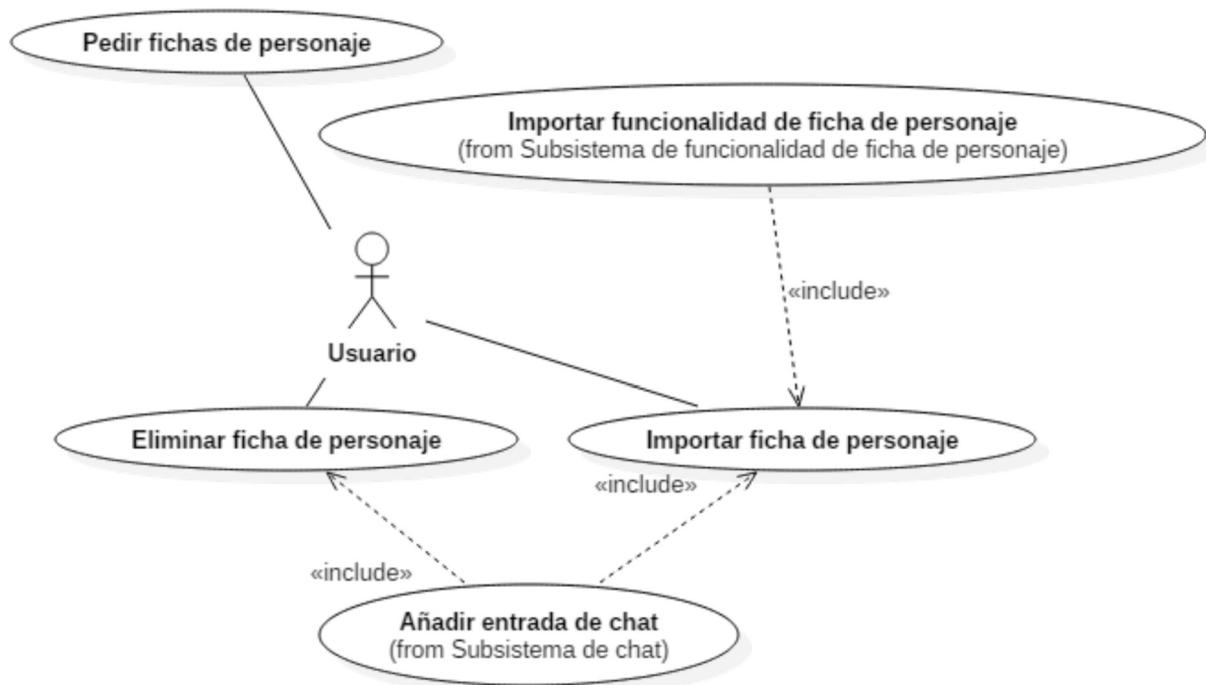
Dibujo 20: Subsistema de lanzamiento de dado

UC-23	Realizar lanzamiento de dado
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario seleccione la opción de realizar lanzamiento de dado.
Precondición	El actor [ACT-2] Usuario se encuentra en la sesión de juego, la herramienta de realizar lanzamiento de dado está habilitada para su uso y el actor [ACT-2] Usuario selecciona la opción de realizar lanzamiento de dado.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-2] Usuario la interfaz con los tipos de datos que puede lanzar.
2	El actor [ACT-2] Usuario selecciona un tipo de dado e introduce el número de dados que desea lanzar.
3	El actor [ACT-2] Usuario selecciona la opción lanzar.
4	El sistema comprueba el lanzamiento, lo opera e incluye el caso de uso [UC-0016] Añadir entrada de chat incluyendo el resultado de la tirada.
Postcondición	El sistema muestra a los actores [ACT-2] Usuario de la sesión de juego y a través del chat, el resultado del lanzamiento de dados realizado.
Alternativas	Acción
3a	El actor [ACT-2] Usuario añade, modifica o elimina un tipo de dado por lo que vuelve al paso 2.
Excepciones	Acción
4b	El lanzamiento de dado no es válido por lo que el sistema informa al actor [ACT-2] Usuario . El caso de uso queda sin efecto y finaliza.
1-3c	El actor [ACT-2] Usuario cancela el lanzamiento de dado. El caso de uso queda sin efecto y finaliza.
*	El actor [ACT-2] Usuario sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ninguno.

Tabla 36: Caso de uso. Realizar lanzamiento de dado

3.2.3.5. Subsistema de importación de ficha de personaje

En este subsistema los casos de uso se centran en gestionar la importación y eliminación de las fichas de personaje de los usuarios en las sesiones de juego. En cada sesión de juego solo puede estar importada una vez una ficha de personaje, así se evita la duplicidad de personajes.



Dibujo 21: Subsistema de importación de ficha de personaje

UC-24	Pedir fichas de personaje
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario seleccione la opción pedir fichas de personaje.
Precondición	El actor [ACT-2] Usuario se encuentra en la sesión de juego y selecciona la opción pedir fichas de personaje.
Secuencia normal	Acción
1	El sistema muestra al actor [ACT-2] Usuario una interfaz donde se visualizan todas las fichas de personaje del juego de rol de la sesión de juego, no importadas en la sesión de juego, con la opción de importarlas.
Postcondición	El sistema muestra al actor [ACT-2] Usuario una interfaz donde se visualizan todas las fichas de personaje del juego de rol de la sesión de juego, no importadas en la sesión de juego, con la opción de importarlas.
Alternativas	Acción
1a	No hay ninguna ficha disponible para mostrar al actor [ACT-2] Usuario , por lo que se le envía un mensaje indicándolo.
Excepciones	Acción
*	El actor [ACT-2] Usuario sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Cuando una ficha ya esté importada en la sesión de juego, no se le dará la opción al actor [ACT-2] Usuario ni de importala ni visualizarla desde la opción de pedir fichas de personaje.

Tabla 37: Caso de uso. Pedir fichas de personaje

UC-25	Importar ficha de personaje
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario seleccione la opción de importar ficha de personaje.
Precondición	El actor [ACT-2] Usuario se encuentra en la sesión de juego y está visualizando las fichas con opción de importarlas.
Secuencia normal	Acción
1	El actor [ACT-2] Usuario selecciona la opción de importar ficha de personaje en una de las fichas.
2	<p>El sistema comprueba que la ficha de personaje pedida para importar existe y pertenece al actor [ACT-2] Usuario. Posteriormente envía a todos los actores [ACT-2] Usuario el contenido de la ficha y les da permisos de manipulación de la ficha (eliminarla por ejemplo) al propietario y al actor [ACT-4] Director de juego. Finalmente, y junto con todo lo anterior, envía la funcionalidad de la ficha incluyendo el caso de uso [UC-0027] Importar funcionalidad de ficha de personaje.</p> <p>Por otro lado envía un mensaje incluyendo el caso de uso [UC-0016] Añadir entrada de chat para avisar a todos los jugadores de la importación de ficha de personaje realizada.</p> <p>Además de esto la ficha se almacena en la memoria de la sesión de juego.</p>
Postcondición	El sistema envía a los actores [ACT-2] Usuario la ficha con los permisos y funcionalidad que corresponda dependiendo del usuario además de notificarlo.
Alternativas	Acción
1a	El sistema solicita la ficha de personaje para importarla, indicando la ficha en cuestión. Por lo que comprueba que existe en la memoria de la sesión de juego. Finalmente envía la ficha junto con su funcionalidad y permisos correspondientes al actor [ACT-2] Usuario indicado.
Excepciones	Acción
2b	La ficha de personaje no puede importarse porque no es válida por lo que se le notifica al actor [ACT-2] Usuario de ello. El caso de uso queda sin efecto y finaliza.
*	El actor [ACT-2] Usuario sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	El paso 1a se realiza cuando un actor [ACT-2] Usuario entra en la sesión de juego. Ninguna ficha es eliminada aunque el usuario la abandone.

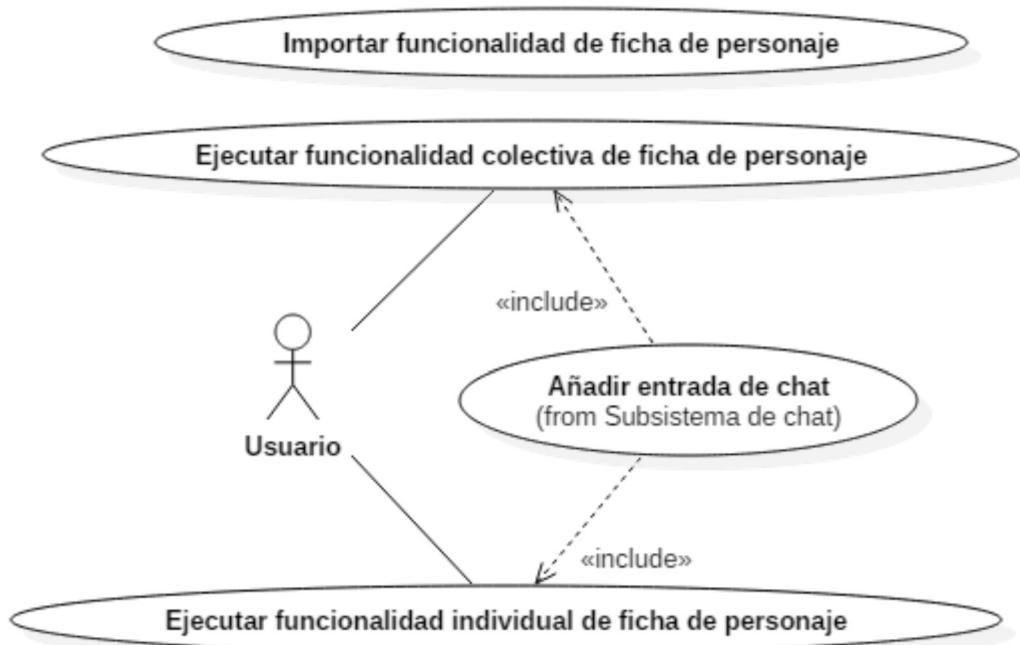
Tabla 38: Caso de uso. Importar ficha de personaje

UC-26	Eliminar ficha de personaje
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario seleccione la opción de eliminar ficha de personaje.
Precondición	El actor [ACT-2] Usuario se encuentra en la sesión de juego.
Secuencia normal	Acción
1	El actor [ACT-2] Usuario selecciona la opción eliminar ficha de personaje de una de las fichas que se encuentren en la sesión de juego.
2	El sistema le pide confirmación al actor [ACT-2] Usuario de la eliminación de la ficha de personaje.
3	El actor [ACT-2] Usuario confirma la eliminación.
4	<p>El sistema verifica que el actor [ACT-2] Usuario es el propietario de la ficha de personaje o es el actor [ACT-4] Director de juego. Además la ficha de personaje existe en la memoria del sistema. Luego elimina la ficha de la sesión de juego para todos los actores [ACT-2] Usuario además de notificar este hecho incluyendo el caso de uso [UC-0016] Añadir entrada de chat.</p> <p>Además de esto elimina la ficha de personaje de la memoria de la sesión de juego.</p>
Postcondición	El sistema elimina la ficha de la sesión de juego para todos los actores [ACT-2] Usuario además de notificarlo.
Alternativas	Acción
3a	El actor [ACT-2] Usuario no confirma la eliminación de la ficha. El caso de uso queda sin efecto y finaliza.
Excepciones	Acción
4b	El actor [ACT-2] Usuario no tiene permisos para eliminar la ficha de personaje o esta no se encuentra en la memoria del sistema. El caso de uso queda sin efecto y finaliza.
*	El actor [ACT-2] Usuario sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Ninguna ficha es eliminada aunque el actor [ACT-2] Usuario abandone la sesión de juego. Quedará en memoria hasta que se elimine manualmente por el actor [ACT-4] Director de juego o el propietario de la ficha de personaje desde la sesión de juego.

Tabla 39: Caso de uso. Eliminar ficha de personaje

3.2.3.6. Subsistema de funcionalidad de ficha de personaje

En este subsistema los casos de uso se centran en gestionar la funcionalidad de las fichas de personaje, permitiendo importala y ejecutarla. Esta funcionalidad es la interacción con las fichas que permite simular las acciones que realizan los jugadores en ellas en los juegos de rol de mesa convencionales.



Dibujo 22: Subsistema de funcionalidad de ficha de personaje

UC-27	Importar funcionalidad de ficha de personaje
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de un actor [ACT-2] Usuario requiera importar funcionalidad de ficha de personaje.
Precondición	Hay una ficha en proceso de importar que requiere de importar la funcionalidad.
Secuencia normal	Acción
1	El sistema obtiene la funcionalidad de la ficha deseada y envía a los actores [ACT-2] Usuario esta funcionalidad haciendo distinción en los permisos de funcionalidad de cada uno, es decir, no todos tendrán la misma funcionalidad, distinguiendo principalmente al usuario, el director de juego y los demás usuarios.
Postcondición	El sistema envía la funcionalidad de la ficha a cada actor [ACT-2] Usuario dependiendo de sus permisos.
Secuencia normal	Acción
*	El actor [ACT-2] Usuario por el cual se está ejecutando el caso de uso sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Aunque este caso de uso es invocado cada vez que se importa una ficha, el componente es lo suficientemente grande como para tener que dividir el caso de uso en importar ficha e importar funcionalidad de la ficha de personaje.

Tabla 40: Caso de uso. Importar funcionalidad de ficha de personaje

UC-28	Ejecutar funcionalidad individual de ficha de personaje
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario seleccione la opción ejecutar funcionalidad individual de ficha de personaje.
Precondición	El actor [ACT-2] Usuario se encuentra en una sesión de juego y posee los permisos necesarios para ejecutar la funcionalidad individual de esa ficha.
Secuencia normal	<p style="text-align: center;">Acción</p> <p>1 El actor [ACT-2] Usuario selecciona una de las funcionalidades individuales que desea ejecutar de la ficha de personaje.</p> <p>2 El sistema le muestra al actor [ACT-2] Usuario mediante una interfaz el resumen de la funcionalidad individual que se va a ejecutar e incluye una confirmación.</p> <p>3 El actor [ACT-2] Usuario confirma la ejecución de la funcionalidad individual.</p> <p>4 El sistema comprueba que la funcionalidad individual a ejecutar es válida, la procesa e incluye el caso de uso [UC-0016] Añadir entrada de chat para mostrar el resultado a los actores [ACT-2] Usuario.</p>
Postcondición	El sistema procesa la funcionalidad individual y muestra el resultado por el chat.
Alternativas	<p style="text-align: center;">Acción</p> <p>3a El actor [ACT-2] Usuario no confirma la ejecución de la funcionalidad individual. El caso de uso queda sin efecto y finaliza.</p> <p>3b El actor [ACT-2] Usuario es el actor [ACT-4] Director de juego y modifica algún parámetro del resumen de la funcionalidad individual que se va a ejecutar y confirma su ejecución. Continúa en el paso 4.</p>
Excepciones	<p style="text-align: center;">Acción</p> <p>4c La funcionalidad individual no es válida o el actor [ACT-2] Usuario no tiene permisos para ejecutarla. El caso de uso queda sin efecto y finaliza.</p>
*	El actor [ACT-2] Usuario sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	<p>Tanto la funcionalidad individual como la colectiva funcionan bajo un mismo protocolo de comunicación pero debido a la gran diferencia de uso se ha decidido dividir en diferentes casos de uso y no en ejecuciones alternativas.</p> <p>La funcionalidad individual es aquella que el actor [ACT-2] Usuario ejecuta en una ficha y el resultado no implica a ninguna, es decir, el resultado es meramente una tirada sin consecuencias (modificación de parámetros e interacciones con las demás principalmente).</p>

Tabla 41: Caso de uso. Ejecutar funcionalidad individual de ficha de personaje

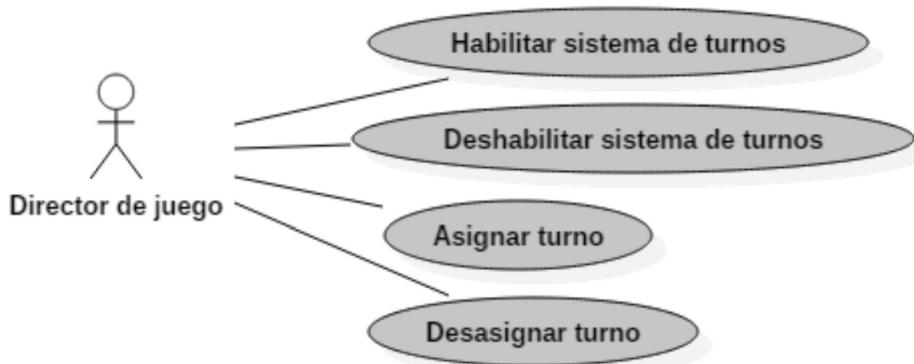
UC-29	Ejecutar funcionalidad colectiva de ficha de personaje
Dependencias	[OBJ-1] Crear sistema de plataforma de juego
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor [ACT-2] Usuario seleccione la opción ejecutar funcionalidad colectiva de ficha de personaje.
Precondición	El actor [ACT-2] Usuario se encuentra en una sesión de juego y posee los permisos necesarios para ejecutar la funcionalidad colectiva de esa ficha.
Secuencia normal	Acción
1	El actor [ACT-2] Usuario selecciona una de las funcionalidades colectivas que desea ejecutar de la ficha de personaje.
2	El sistema le muestra al actor [ACT-2] Usuario mediante una interfaz el resumen de la funcionalidad colectiva que se va a ejecutar y le solicita que seleccione la ficha objetivo a la que se le realizará dicha funcionalidad.
3	El actor [ACT-2] Usuario selecciona la ficha objetivo.
4	El sistema pide confirmación al actor [ACT-2] Usuario para ejecutar la funcionalidad colectiva.
5	El actor [ACT-2] Usuario confirma la ejecución de la funcionalidad colectiva.
6	El sistema comprueba que la funcionalidad colectiva a ejecutar es válida, la procesa y envía a todos los actores [ACT-2] Usuario de la sesión de juego, si fuera necesario, las modificaciones pertinentes a las fichas indicadas que han sido modificadas basándose en la resolución de la funcionalidad colectiva. Además se incluye el caso de uso [UC-0016] Añadir entrada de chat para mostrar el resultado.
Postcondición	El sistema procesa la funcionalidad colectiva, modifica si es necesario alguna de las fichas de la sesión de juego y muestra el resultado por el chat a los actores [ACT-2] Usuario .
Alternativas	Acción
3a	El actor [ACT-2] Usuario no selecciona la ficha objetivo y cancela el caso de uso. El caso de uso queda sin efecto y finaliza.
5b	El actor [ACT-2] Usuario no confirma la ejecución de la funcionalidad colectiva. El caso de uso queda sin efecto y finaliza.
5c	El actor [ACT-2] Usuario tiene los permisos de [ACT-4] Director de juego y modifica algún parámetro del resumen de la funcionalidad colectiva que se va a ejecutar y confirma su ejecución.

Excepciones	Acción
6d	La funcionalidad colectiva no es válida o el actor [ACT-2] Usuario no tiene permisos para ejecutarla. El caso de uso queda sin efecto y finaliza.
*	El actor [ACT-2] Usuario sale de la sesión de juego sin que finalizara el caso de uso. El caso de uso queda sin efecto y finaliza.
Comentarios	Tanto la funcionalidad individual como la colectiva funcionan bajo un mismo protocolo de comunicación pero debido a la gran diferencia de uso se ha decidido dividir en diferentes casos de uso y no en ejecuciones alternativas. La funcionalidad colectiva es aquella que el actor [ACT-2] Usuario ejecuta en una ficha y el resultado implica a al menos una (puede ser ella misma), es decir, el resultado podría modificar, si fuera necesario, parámetros de alguna ficha de personaje.

Tabla 42: Caso de uso. Ejecutar funcionalidad colectiva de ficha de personaje

3.2.3.7. Subsistema de turno

En este subsistema los casos de uso se centran en gestionar un sistema de turnos que le permita al director de juego indicar qué usuario puede realizar acciones y quienes no en un momento determinado. Esta funcionalidad no se ha incluido finalmente.



Dibujo 23: Subsistema de turno

3.2.3.8. Subsistema multimedia

En este subsistema no se han llegado a plantear los casos de uso pero la idea general es permitir al director de juego en la sesión de juego subir un archivo de audio, imagen y/o vídeo y que los demás usuarios escuchen y/o vean este archivo. La ejecución del mismo podría ser libre o solo cuando el director de juego lo reproduzca para así tener un control absoluto sobre lo que está ocurriendo y mostrárselo así a los jugadores.

3.3. Requisitos de información

Los requisitos de información serán detallados en esta sección de forma individual. Además todos se mostrarán en el Diagrama Conceptual y no por separado por iteraciones como en la sección anterior.

IQR-1	Usuario			
Descripción	Es el usuario registrado en la aplicación.			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Nombre de usuario	X	X	Texto
	Nombre real		X	Texto
	Apellido real		X	Texto
	Correo electrónico	X	X	Texto
	Contraseña		X	Contraseña
	Fecha de inscripción		Auto	Fecha
	Fecha de último acceso		Auto	Fecha

Tabla 43: Requisito de información. Usuario

IQR-2	Idioma			
Descripción	Idioma de la aplicación.			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Nombre	X	X	Texto

Tabla 44: Requisito de información. Idioma

IQR-3	Juego de rol			
Descripción	Juego de rol de la aplicación.			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Nombre	X	X	Texto

Tabla 45: Requisito de información. Juego de rol

IQR-4	Sesión de juego			
Descripción	Es la sesión de juego del sistema.			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Nombre		X	Texto
	Contraseña		X	Contraseña
	Comentarios		X	Texto grande
	Idioma		X	[IQR-2] Idioma
	Juego de rol		X	[IQR-3] Juego de rol
	Propietario		X	[IQR-1] Usuario

Tabla 46: Requisito de información. Sesión de juego

IQR-5	Plantilla de ficha de personaje			
Descripción	Es la plantilla de ficha de personaje de los juegos de rol.			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Nombre		X	Texto
	Versión		X	Contraseña
	Juego de rol		X	[IQR-3] Juego de rol

Tabla 47: Requisito de información. Plantilla de ficha de personaje

IQR-6	Ficha de personaje			
Descripción	Es la ficha de personaje de una plantilla de ficha de personaje.			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Nombre		X	Texto
	Plantilla de ficha de personaje		X	[IQR-5] Plantilla de ficha de personaje

Tabla 48: Requisito de información. Ficha de personaje

IQR-7	Dato grupo de ficha de personaje			
Descripción	Son los datos del tipo grupo que componen las fichas de personaje			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Ficha de personaje		X	[IQR-6] Ficha de personaje
	Dato grupo de ficha de personaje			[IQR-7] Dato grupo de ficha de personaje
	Nombre		X	Texto
	Nombre a mostrar			Texto

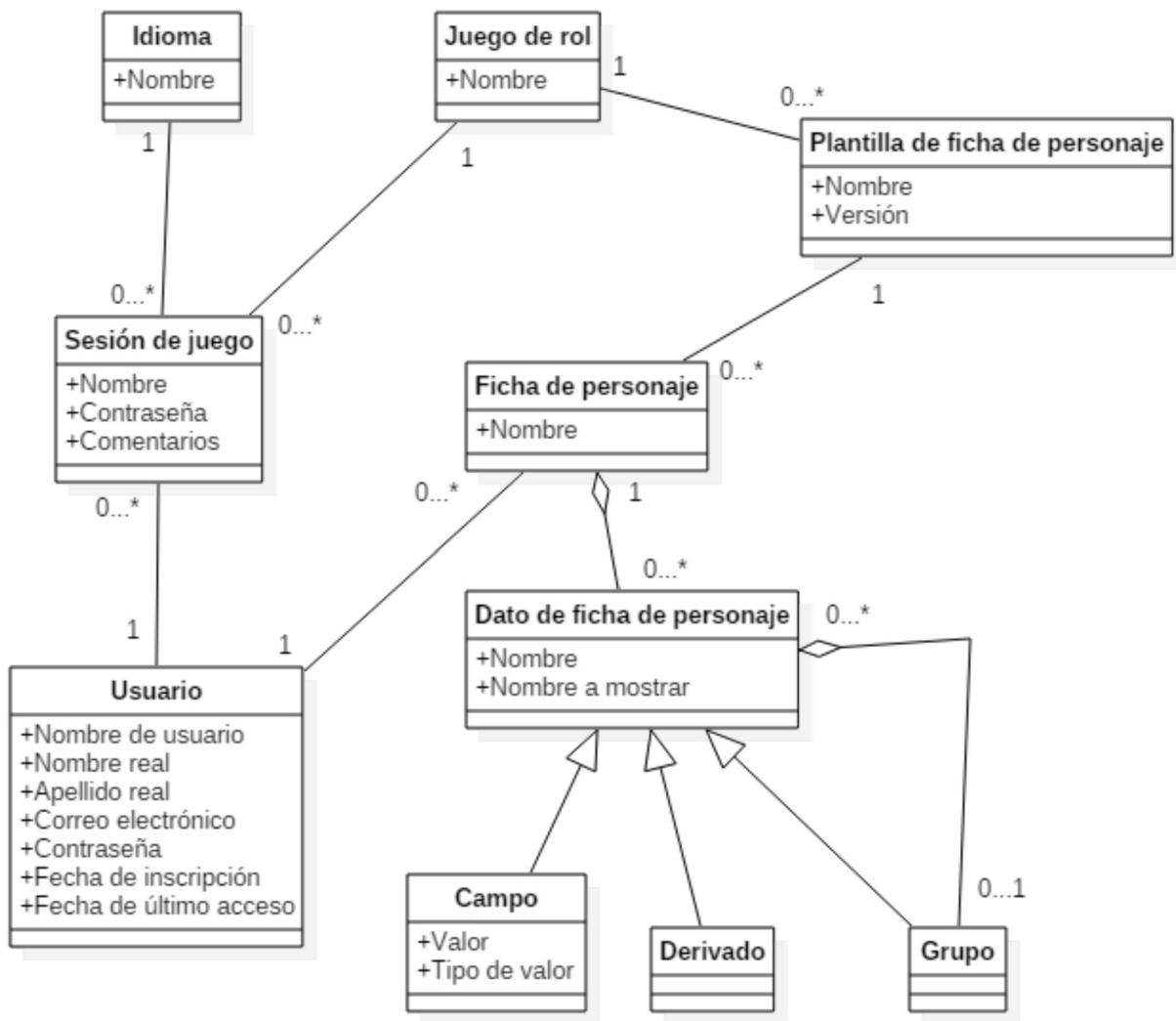
Tabla 49: Requisito de información. Dato grupo de ficha de personaje

IQR-8	Dato derivado de ficha de personaje			
Descripción	Son los datos del tipo derivado que componen las fichas de personaje			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Dato grupo de ficha de personaje		X	[IQR-7] Dato grupo de ficha de personaje
	Nombre	X	X	Texto
	Nombre a mostrar			Texto

Tabla 50: Requisito de información. Dato derivado de ficha de personaje

IQR-9	Dato campo de ficha de personaje			
Descripción	Son los datos del tipo campo que componen las fichas de personaje			
Atributos	Nombre	Único	Obligatorio	Tipo de dato
	Dato grupo de ficha de personaje		X	[IQR-7] Dato grupo de ficha de personaje
	Nombre	X	X	Texto
	Nombre a mostrar			Texto
	Valor		X	Entero
	Tipo de valor		X	Texto

Tabla 51: Requisito de información. Dato campo de ficha de personaje



Dibujo 24: Diagrama Conceptual

3.4. Reglas de negocio

A continuación se van a presentar las reglas de negocio que se creen necesarias para matizar algún aspecto de la aplicación:

BRU-1	Tamaño de la contraseña de usuario
Descripción	Las contraseña de usuario tiene que tener entre 6 y 255 caracteres.

Tabla 52: Regla de negocio. Tamaño de la contraseña de usuario

BRU-2	Contraseña necesaria siempre en sesión de juego
Descripción	Toda sesión de juego tiene que tener una contraseña aunque el director de juego quiera que la partida sea pública, es decir, que entre cualquiera.

Tabla 53: Regla de negocio. Contraseña necesaria siempre en sesión de juego

BRU-3	Tamaño de la contraseña de sesión de juego
Descripción	Las contraseña de sesión de juego tiene que tener entre 6 y 255 caracteres.

Tabla 54: Regla de negocio. Tamaño de la contraseña de sesión de juego

BRU-4	El director de juego solo puede ser el creador de la sesión de juego
Descripción	Un administrador no puede acceder a una sesión de juego como director de juego a no ser que sea el propietario.

Tabla 55: Regla de negocio. El director de juego solo puede ser el creador de la sesión de juego

3.5. Requisitos no funcionales

A continuación vamos a desglosar los requisitos no funcionales que hemos considerado imprescindibles para el proyecto.

NFR-1	Control de Acceso
Descripción	El sistema deberá controlar que un usuario no puede sobrecargar el sistema con cientos de peticiones simultáneas. Debe haber un mecanismo de seguridad para el servidor.
Medidas de actuación	<ul style="list-style-type: none"> • Utilizar un framework o plugin que ya provea de este mecanismo de seguridad.

Tabla 56: Requisito no funcional. Control de Acceso

NFR-2	Mantenibilidad
Descripción	El sistema deberá ser fácilmente mantenible ya que es probable que se añada nueva funcionalidad.
Medidas de actuación	<ul style="list-style-type: none"> • El sistema debe estar desarrollando utilizando normas de codificación. En el caso de PHPel PSR-2. • Organizar el sistema de directorios y ficheros propio del framework a utilizar. • Refactorizar en la mayoría de los casos.

Tabla 57: Requisito no funcional. Mantenibilidad

NFR-3	Disponibilidad
Descripción	El sistema deberá estar disponible y operativo en todo momento ya que los usuarios podrán acceder a él en cualquier hora, minuto y segundo.
Medidas de actuación	<ul style="list-style-type: none"> • Utilizar un framework o plugin que ya provea de este mecanismo de seguridad. • Elegir un buen hosting que permita mantener el sistema el máximo tiempo posible en funcionamiento. • Realizar una buena batería de pruebas para evitar caídas del sistema.

Tabla 58: Requisito no funcional. Disponibilidad

NFR-4	Multi navegador y diseño responsive
Descripción	El sistema deberá ser accesible desde cualquier dispositivo a través del navegador y que este se visualice de forma correcta, adaptándose a sus medidas.
Medidas de actuación	<ul style="list-style-type: none"> • Las tecnologías a utilizar deben permitir un sistema multiplataforma a través del navegador. • La implementación de las interfaces debe ser consecuente con este requisito, haciendo adaptable las distintas resoluciones de dispositivos.

Tabla 59: Requisito no funcional. Multi navegador y diseño responsive

NFR-5	Reconexión de usuarios
Descripción	Los usuarios podrán sufrir alguna caída puntual de su conexión por lo que el sistema deberá permitir a los usuarios reconectarse sin perder datos.
Medidas de actuación	<ul style="list-style-type: none"> • Utilizar un framework o plugin que gestione las reconnexiones. • Guardar en memoria los datos de las sesiones de juego y ofrecerle estos datos a los usuarios que se conecten o reconecten.

Tabla 60: Requisito no funcional. Reconexión de usuarios

NFR-6	Sistema multi idioma
Descripción	Este requisito no funcional recoge toda la configuración necesaria para poner en funcionamiento el multi idioma del proyecto.
Medidas de actuación	<ul style="list-style-type: none"> • Utilizar un framework o plugin que permita gestionar el multi idioma de forma simple y rápida. • Utilizar archivos de configuración para los diferentes idiomas.

Tabla 61: Requisito no funcional. Sistema multi idioma

3.6. Alternativa y elección tecnológica

Como indica el título de la sección se va a realizar un pequeño estudio de las diferentes alternativas tecnológicas. Hay que tener en cuenta que se desea que la aplicación responda peticiones de los clientes en los navegadores web, además de mantener un paso de información constante entre los navegadores de los usuarios para las sesiones de juego.

Teniendo presentes las alternativas tecnológicas se realizará las elecciones de las tecnologías que se van a usar en el desarrollo de este sistema y el motivo de su selección.

Sistema operativo

En cuanto a sistemas operativos nos planteamos dos opciones:

- Servidor en Linux.
- Servidor en Windows.

El sistema operativo elegido es Ubuntu Server 14.04 LTS (Linux). Su elección se debe a que desarrollador tiene experiencia con la tecnología para montar servidores web, al contrario que con Windows Server.

Servidor HTTP

En el servidor HTTP nos planteamos como opciones:

- Apache.
- Tomcat.
- Nginx.

El servidor HTTP elegido es Apache porque el desarrollador tiene experiencia con el mismo en Linux.

Servidor web

Por otro tenemos las tecnologías con las que desarrollar la aplicación en el lado del servidor:

- PHP con o sin framework (Symfony, Laravel, Codeigniter, Zend o CakePHP).
- JavaScript con Node.js.
- Java.
- Unity.
- Adobe Media Server (AMS).

- AJAX.
- Application Messaging Protocol (WAMP).

Las tecnologías seleccionadas para el desarrollo del sistema son PHP y Symfony como framework. La elección es debida a que el director del proyecto lo ha considerado una buena herramienta para el desarrollo del sistema y porque el desarrollador conoce PHP mejor que cualquier otra alternativa.

Además se incluye la tecnología WAMP en vez de AJAX, esto es debido a que Symfony posee un bundle con la tecnología WAMP implementada permitiendo montar un servidor de websockets de forma rápida y sencilla, mientras que con AJAX las llamadas al servidor serían poco estructuradas y más caóticas de mantener.

Base de datos

- MySQL.
- MongoDB.

Se ha elegido MySQL como base de datos ya que el desarrollador solo tiene experiencia con las bases de datos relacionales y concretamente con esta tecnología.

Cliente

- HTML, CSS y JavaScript.
- Java web.
- Unity Web Player.
- Adobe Flash Player.

Debido a las decisiones anteriores la parte del cliente se realizará en HTML, CSS y JavaScript permitiendo que los usuarios interactúen desde el navegador con la aplicación y desde cualquier dispositivo que soporte el navegador.

Además se añaden las tecnologías jQuery y jQuery UI como frameworks de JavaScript que son muy útiles para el manejo de los eventos en la web.

Entorno de desarrollo

- Netbeans.
- Eclipse.
- Sublime Text.

Eclipse ha sido la herramienta seleccionada como entorno de desarrollo porque posee plugins de Symfony y de jQuery que permite autocompletar y facilitar el desarrollo del sistema.

Control de versiones

- Git.
- Subversion (SVN).

En el control de versiones se ha elegido Git con el único motivo, a parte del control de versiones, de aprender su uso ya que es uno de los más demandados del mercado.

3.7. Análisis GAP

Con las tecnologías que hemos elegido no completamos de forma directa ninguno de los requisitos funcionales, no funcionales ni de información que se han propuesto. Aunque sí nos permiten completarlos con una gran facilidad. A continuación se explicarán los puntos principales:

- **[NFR-1] Control de acceso.** Gracias a Symfony, este requisito se puede cumplir sin mucha complejidad. Symfony nos permite configurar el control de acceso de forma sencilla y permite modificarla si fuera necesario.
- **[NFR-3] Disponibilidad.** Para que se cumpla este requisito es necesario que la implementación del sistema sea correcta pero unos buenos servicios ofrecidos por el servidor en Symfony dan robustez al sistema permitiendo cumplir en gran medida este requisito, ya que entre otras cosas permite testar la aplicación para buscar errores.
- **[NFR-4] Multi navegador y diseño responsive.** Este requisito se puede cumplir gracias a las tecnologías HTML, CSS y JavaScript aunque por su puesto hay que usarlas de forma correcta en su implementación. Además al usar Symfony permite gestionar el enrutamiento de la web.
- **[NFR-2] Mantenibilidad.** Parte de este requisito se cumple si se usa Symfony ya que posee su propia organización de directorios y ficheros que permite una fácil gestión.
- **[NFR-5] Reconexión de usuarios.** Al usar Symfony y su sistema de sesiones el usuario podrá reconectarse siempre que su tiempo de sesión no haya expirado. Además el uso de WAMP permite la reconexión de los usuarios a su servidor de websockets aunque es necesario gestionar esa reconexión.
- **[NFR-6] Sistema multi idioma.** Symfony tiene una gran cantidad de complementos y bundles que se pueden configurar y entre ellos está el de traducciones que permite configurar los textos que se muestran para que se adapten al idioma que seleccione el usuario.
- Los requisitos funcionales y de información no se ven afectados directamente por el uso de estas tecnologías ya que hay que implementarlos todos. Aun así las tecnologías seleccionadas facilitan la implementación de los subsistemas gracias, entre otras cosas, al generador de entidades de Symfony.

3.8. Matriz de rastreabilidad

Casos de uso	[OBJ-1]	[OBJ-2]
[UC-1]		X
[UC-2]		X
[UC-3]		X
[UC-4]		X
[UC-5]		X
[UC-6]		X
[UC-7]		X
[UC-8]		X
[UC-9]		X
[UC-10]		X
[UC-11]	X	X
[UC-12]	X	X
[UC-13]	X	X
[UC-14]		X
[UC-15]	X	
[UC-16]	X	
[UC-17]	X	X
[UC-18]	X	X
[UC-19]	X	
[UC-20]	X	
[UC-21]	X	
[UC-22]	X	
[UC-23]	X	
[UC-24]	X	
[UC-25]	X	
[UC-26]	X	
[UC-27]	X	
[UC-28]	X	
[UC-29]	X	

Tabla 62: Matriz de rastreabilidad

3.9. Protocolo de comunicación

Para entender esta sección hay que hablar primero de los tres módulos que forman el sistema completo: Plataforma web, Plataforma de juego y el Núcleo de rol. También hay que tener en cuenta que este protocolo ha sido definido como un análisis previo al desarrollo y hay algunas características que no han sido implementadas, aunque sí en su mayor parte.

En la Plataforma web los usuarios pueden gestionar sus fichas de personaje creándolas, eliminándolas y modificándolas. Para ello, la Plataforma web, tendrá que pedir información al Núcleo de rol para obtener datos concretos de la ficha de personaje de ese juego de rol. También es el módulo donde se gestionan los datos de usuario.

La Plataforma de juego es el módulo donde los usuarios desarrollan su partida por lo que hace uso de las fichas de personaje, propias de la Plataforma web, y de la funcionalidad que estas pueden tener, obtenidas del Núcleo de rol ya que las reglas de los juegos de rol se encuentran en este módulo. Además de ello, la Plataforma de juego, gestionará las sesiones de juego donde se podrán crear las sesiones de juego, visualizarlas, unirse a ellas, editarlas y eliminarlas.

Hay que tener en cuenta que los jugadores no pueden tener acceso directo a las fórmulas y reglas de los juegos de rol por lo que cada acción que se ejecute en el cliente debe ser comprobada y resuelta en el servidor (concretamente en el módulo del Núcleo de rol) pero el cliente si tiene que tener la información necesaria para saber cómo operar y qué opciones darle al usuario.

Por todo lo anterior se crea este protocolo de comunicación donde se define la configuración necesaria de los datos para cada tipo de petición entre los componentes del sistema. Esta petición está orientada a las fichas de personaje que componen los juegos de rol ya que son ellas las que tienen la funcionalidad que puede ser ejecutada.

3.9.1. Organización

Se comenzará por la definición de los datos que conforman una ficha de personaje y posteriormente las funcionalidades que pueden ejecutarse cuando el usuario pida algún dato o acción de la ficha al servidor.

3.9.2. Ficha de personaje

El protocolo de comunicación se basará en los datos que conforman las fichas de personaje.

Para tener una idea de una ficha de personaje, mostraremos a continuación una fidha de personaje de del juego de rol Pathfinder [14] y otra del juego de rol Vampiro la Mascarada [15]:

Dibujo 25: Ficha de personaje de Pathfinder

EDICIÓN 20º ANIVERSARIO

VAMPIRO

LA MASCARADA

Nombre: Jugador: Crónica:	Naturaleza: Conducta: Concepto:	Clan: Generación: Sire:
Atributos		
Fisicos	Sociales	Mentales
Fuerza _____ Destreza _____ Resistencia _____	Carisma _____ Manipulación _____ Apariencia _____	Percepción _____ Inteligencia _____ Astucia _____
Habilidades		
Talentos	Técnicas	Conocimientos
Alerta _____ Atletismo _____ Callejero _____ Consciencia _____ Empatía _____ Expresión _____ Intimidación _____ Liderazgo _____ Pelea _____ Subterfugio _____	Armas C. C. _____ Armas de Fuego _____ Conducir _____ Etiqueta _____ Hurto _____ Interpretación _____ Pericias _____ Sigilo _____ Supervivencia _____ Trato con Animales _____	Academismo _____ Ciencia _____ Finanzas _____ Informática _____ Investigación _____ Leyes _____ Medicina _____ Ocultismo _____ Política _____ Tecnología _____
Ventajas		
Disciplinas	Trasfondos	Virtudes
_____ _____ _____ _____ _____ _____	_____ _____ _____ _____ _____	Conciencia/Convicción _____ Autocontrol/Instinto _____ Coraje _____
Humanidad/Senda		
Porte: _____ ()		
Fuerza de Voluntad		
Sangre Por Turno: _____		
Salud		
Magullado _____ Lastimado - 1 _____ Lesionado - 1 _____ Herido - 2 _____ Malherido - 2 _____ Tullido - 5 _____ Incapacitado _____		
Debilidad		
Experiencia		

Atributos: 7/5/3 • Habilidades: 13/9/5 • Disciplinas: 3 • Trasfondos: 5 • Virtudes: 7 • Puntos Gratuitos: 15 (7/5/2/1)

Dibujo 26: Ficha de personaje de Vampiro la Mascarada

Teniendo en cuenta los tipos de datos y los campos de las fichas de personaje, hemos elegido la siguiente forma para estructurar los datos:

- character_sheet
 - character_sheet_template_id: id en BD de la plantilla en la que se basa la ficha.
 - user_id: id en BD del usuario.
 - name: nombre que le da el usuario a la ficha para gestionarla.
- character_sheet_data: conjunto de datos que conforman la ficha
 - name: id único para acceder al campo desde el cliente.
 - display_name: si existe, será el nombre a visualizar de ese campo.

- character_sheet_id: el id en BD de la ficha en cuestión. Se usará para jerarquizar los campos y solo lo necesitarán los character_sheet_data principales de los que se forme el resto de la ficha.
- datatype: el tipo de campo.
 - field: campo con un valor a añadir por el usuario.
 - group: campo que contiene un subconjunto de campos a nivel de jerarquía. Sin valor.
 - derived: campo cuyo valor es la suma del conjunto de otros character_sheet_data cuyo datatype es field.
- value:
 - field: valor numérico o cadena.
 - group: no tiene.
 - derived: serialización de los name de los character_sheet_data a los que hace referencia para calcular su valor.
- valuetype: indica el tipo de valor (entero, entero con signo, cadena, etcétera).
- character_sheet_data_group_id: el id en DB de otro character_sheet_data. Se usará para jerarquizar los campos.

3.9.3. Crear ficha de personaje

Teniendo en cuenta la plantilla de la ficha de personaje y por ello del juego de rol para crear una ficha de personaje, la Plataforma Web necesitará ir actualizando alguno de sus valores como los campos derivados a medida que se vaya creando la ficha. Será responsabilidad del Núcleo de rol proporcionarle esta información.

3.9.3.1. Pedir campos que derivan

Durante la creación de una ficha de personaje podrá haber campos que requieran ser recalculados y la Plataforma Web deberá conocer qué campos son los que si se modifican tiene que pedirle al Núcleo de rol los campos que derivados de estos con sus nuevos valores. Para hacer esto la Plataforma web debe solicitar al Núcleo de rol estos campos indicando la plantilla personaje que se está usando y este devolverá una lista con todos los identificadores de esos campos.

Plataforma web → Núcleo de rol (id plantilla de personaje)
 Núcleo de rol → Plataforma web (lista con los campos que derivan)

3.9.3.2. Modificar campos derivados

Una vez que uno de los campos que derivan es modificado, la Plataforma web envía al Núcleo de rol la ficha y el campo modificado para que se le devuelva una lista con el par <identificador, valor> de los elementos derivados que han sido modificados.

Plataforma web → Núcleo de rol (identificador de campo que deriva modificado, ficha de personaje con los datos actuales)

Núcleo de rol → Plataforma de juego (lista con el par <identificador, valor> de los elementos modificados)

3.9.4. Mostrar ficha de personaje

La ficha de personaje puede mostrarse, de forma independiente, tanto en la Plataforma web como en la Plataforma de juego y no tendría ninguna comunicación con los demás módulos. El método consiste en extraer de la BD los datos y formatearlos para mostrarlos.

Plataforma web o Plataforma de juego → Base de datos (identificador de la ficha de personaje)
Base de datos → Plataforma web/Plataforma de juego (ficha de personaje)

3.9.5. Editar ficha de personaje

La ficha de personaje podrá ser editada desde la Plataforma web y, por ello, deberá tener la misma funcionalidad que cuando se crea, pidiendo los campos derivados y realizando su correspondiente modificación.

3.9.6. Importar ficha de personaje

Cuando importas una ficha de personaje en la Plataforma de juego además de mostrarla la estás dotando de funcionalidad, la cual la tiene que proveer el Núcleo de rol.

La funcionalidad que puede ejecutar una ficha se divide en dos formas: la individual y la colectiva. Aun así el protocolo para ejecutar ambas es el mismo, cambiando algunos parámetros.

Los campos derivados necesitan conocer, tanto si se importa como si se muestran, el valor de ese campo o la fórmula por la cual se calcula.

Opciones de la funcionalidad individual:

- Cuando se importa la ficha, los campos que derivan tienen un identificador para que cuando estos campos cambien, cambien a su vez el campo al los que derivan. Este identificador puede ser de dos formas:
 - Ser un solo identificador que pida al Núcleo de rol que opere con todo lo demás.
 - Que sea la propia fórmula, que se encuentra en el cliente la que modifique el campo derivado.
- Que las reglas del juego estén implementadas en jQuery y se modifiquen de esta forma desde el cliente.

3.9.6.1. Importar funcionalidad individual

La funcionalidad individual consiste en realizar tiradas/operaciones con al menos un elemento de la ficha. El resultado no influirá en ningún momento al estado del juego, a la propia ficha, ni del resto de fichas.

Opciones:

- Cada funcionalidad individual corresponde a un identificador único que es importado. Posteriormente para pedir las operaciones previas a ejecutar la funcionalidad, se le pasa al Núcleo de rol el identificador en cuestión. Después, una vez mostrados y modificados o no los datos, se le envía al Núcleo de rol la operación a realizar y éste lo resuelve devolviendo el resultado a la Plataforma de juego.
- Cada funcionalidad individual corresponde a un identificador único que es importado. Posteriormente todo el proceso de pedir operaciones se encuentra en el propio cliente y solo se le pide al Núcleo que devuelva los resultados finales.

- Cada funcionalidad individual recibe una lista con los datos de las operaciones a realizar, proveídos por el Núcleo de rol por lo que las operaciones a mostrar se encuentran en el cliente. Finalmente se le pide al Núcleo de rol que devuelva los resultados finales.

3.9.6.2. Importar funcionalidad colectiva

La funcionalidad colectiva es toda aquella acción de la ficha que puede tener una consecuencia en el estado del juego, la propia ficha o el resto de las fichas. La forma de operar la funcionalidad colectiva es igual a la forma de operar la funcionalidad individual, añadiendo y modificando alguno de los valores que permitan adaptar la comunicación para cada caso.

4. Diseño del sistema

En este capítulo se recoge la arquitectura general del sistema de información, el diseño de la interfaz de usuario, el diseño físico de datos y el diseño de componentes software.

4.1. Diseño de la arquitectura

A continuación se detallarán tanto la arquitectura física, lógica como de diseño del sistema.

4.1.1. Arquitectura física

En la sección 2.3. Organización ya se comentó el equipo que se ha usado durante el desarrollo y la prueba del servidor. No obstante se va a dar un resumen con las características que se requerirían de un servidor para que la aplicación funcione de forma correcta para un número estimado de 1000 personas conectadas simultáneamente:

- 2 CPU vCores
- 4 GB RAM
- 200 GB SSD
- Conexión a Internet de fibra para servidor.

En cuanto a los clientes debería tener un sistema capaz de soportar un navegador Google Chrome 54 o Firefox 47, por lo que podría ser un dispositivo móvil, tablet, sobremesa y portátil. No se asegura que en las futuras versiones de Google Chrome y Firefox funcionara el sistema, sobre todo si realizan cambios en los estándares de desarrollo.

A continuación mostramos la arquitectura física del sistema, con las principales tecnologías que lo conforman.



Dibujo 27: Arquitectura física del sistema

4.1.2. Arquitectura lógica

En cuanto a la arquitectura lógica vamos a desglosar más detalladamente el software usado en el servidor y posteriormente el que el cliente debe poseer para poder acceder al sistema.

En el servidor:

- Ubuntu 14.04 LTS.
- Apache.
- PHP [16], Symfony [17], Twig, YAML y diferentes componentes de Symfony como el ORM Doctrine y WebSocketBundle (GeniusesOfSymfony).
- MySQL [18].
- HTML, CSS, JavaScript, jQuery y jQuery UI [19].
- Composer.
- Git [20].

En el cliente solo es necesario poseer un navegador Google Chrome 54 o Firefox 47. Además el dispositivo debe tener la potencia necesaria para ejecutar la librería de jQuery y jQuery UI. El sistema operativo que use es indiferente mientras permita ejecutar el navegador.

4.1.3. Arquitectura de diseño

A continuación vamos a desglosar en qué capa actúa cada artefacto software con el fin de tener una visión más clara de los mismos.

La arquitectura que se plantea en este sistema es de Modelo Vista Controlador (MVC) y las tecnologías que hemos elegido están orientadas para este tipo de arquitectura. Además tenemos que tener en cuenta que cada vez que hablamos de Symfony estamos hablando de todos sus componentes como Doctrine o WAMP así como PHP [21], ya que Symfony es un framework de PHP [22].

Capa de presentación

El software encargado de visualizar los contenidos al cliente son provistos por HTML, CSS, JavaScript, jQuery, jQuery UI y los navegadores (Google Chrome y Firefox).

Por otro lado el que hace de controlador para la capa de presentación es Symfony, haciendo uso de Twig principalmente.

Capa de negocio

En esta capa el encargado principalmente de gestionar todo es Symfony haciendo uso de YAML para las configuraciones, composer para gestionar las dependencias, WAMP para gestionar la conexión de sockets y Apache para gestionar el servidor.

Capa de acceso a datos

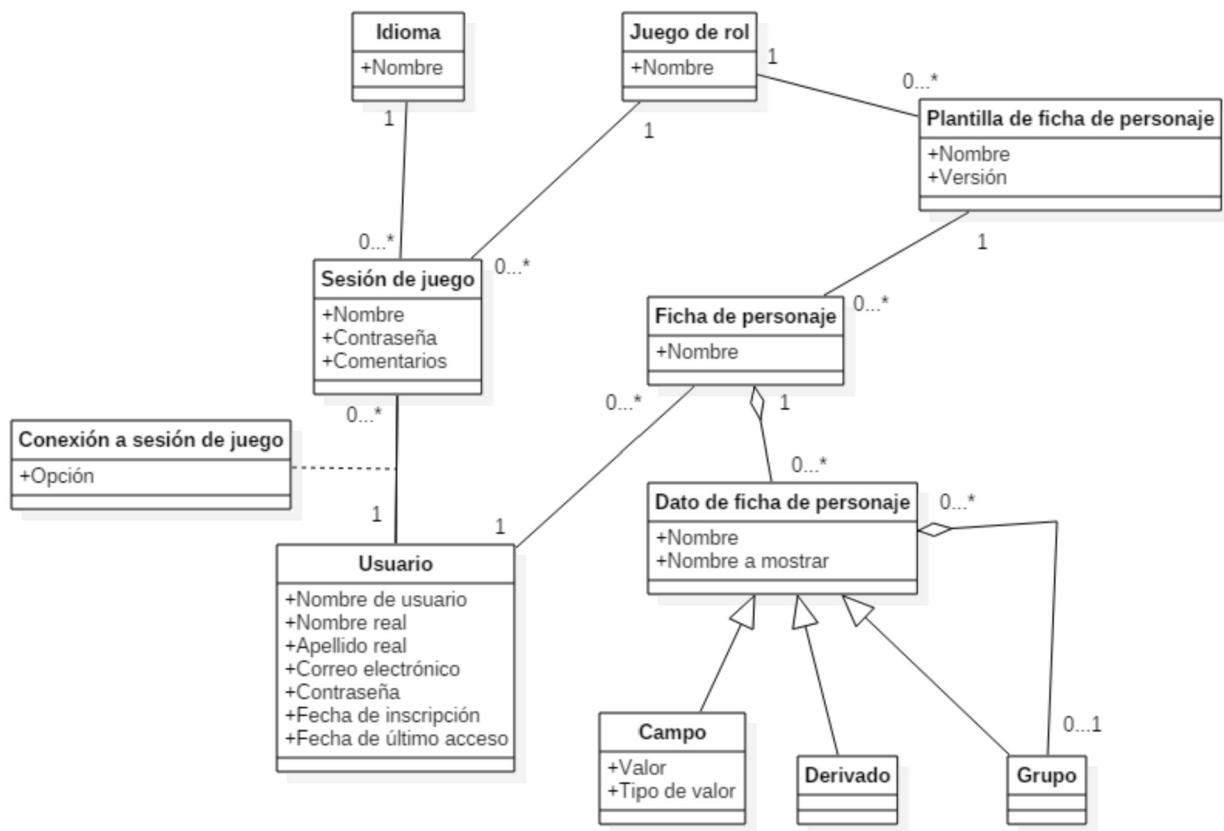
En esta capa actúa principalmente MySQL siendo configurada y tratada en mayor parte por Doctrine y YAML [23].

Servicios transversales

Como servicio transversal tenemos a Git que nos proporciona una herramienta de control de versiones para el código y Symfony que cumple parte de los requisitos no funcionales al permitir gestionarlos con códigos y configuraciones, como es el caso del multi idioma.

4.2. Diagrama de clases

El paso del Diagrama Conceptual al Diagrama de Clases solamente tiene un cambio significativo y es el añadido de una clase que asocia a la clase Usuario y a la clase Sesión de juego. Esta nueva clase es la que mantiene la información de la conexión de los usuarios con las respectivas sesiones de juego, teniendo el atributo opción como indicador del estado actual. Esta clase es la que permitirá conectar los dos servidores para la correcta funcionalidad de las sesiones de juego.



Dibujo 28: Diagrama de Clases

4.3. Diseño de datos

Vamos a detallar las tablas de la base de datos, teniendo como referente el diseño de la sección anterior:

Tabla	usuario			
Descripción	Es el usuario registrado en la aplicación.			
Atributos	Nombre	Único	Obligatorio	Características
	id	X	X	PK INTEGER
	nombreUsuario	X	X	VARCHAR(50)
	nombreReal		X	VARCHAR(50)
	apellidoReal		X	VARCHAR(50)
	correoElectronico	X	X	VARCHAR(255)
	contrasena		X	VARCHAR(255)
	fechaInscripcion		X	DATETIME
	fechaUltimoAcceso		X	DATETIME

Tabla 63: Diseño de datos. usuario

Tabla	idioma			
Descripción	Idioma de la aplicación.			
Atributos	Nombre	Único	Obligatorio	Características
	id	X	X	PK INTEGER
	nombre	X	X	VARCHAR(50)

Tabla 64: Diseño de datos. idioma

Tabla	juegoRol			
Descripción	Juego de rol de la aplicación.			
Atributos	Nombre	Único	Obligatorio	Características
	id	X	X	PK INTEGER
	nombre	X	X	VARCHAR(50)

Tabla 65: Diseño de datos. juegoRol

Tabla	sesionJuego			
Descripción	Es la sesión de juego del sistema.			
Atributos	Nombre	Único	Obligatorio	Características
	id	X	X	PK INTEGER
	nombre		X	VARCHAR(50)
	contrasena		X	VARCHAR(255)
	comentarios		X	LONGTEXT(1023)
	idioma		X	[IQR-2] Idioma
	juegoRol		X	[IQR-3] Juego de rol
	propietario		X	[IQR-1] Usuario

Tabla 66: Diseño de datos. sesionJuego

Tabla	plantillaFichaPersonaje			
Descripción	Es la plantilla de ficha de personaje de los juegos de rol.			
Atributos	Nombre	Único	Obligatorio	Características
	id	X	X	PK INTEGER
	nombre		X	VARCHAR(50)
	version		X	VARCHAR(50)
	juegoRol		X	[IQR-3] Juego de rol

Tabla 67: Diseño de datos. plantillaFichaPersonaje

Tabla	fichaPersonaje			
Descripción	Es la ficha de personaje de una plantilla de ficha de personaje.			
Atributos	Nombre	Único	Obligatorio	Características
	id	X	X	PK INTEGER
	nombre		X	VARCHAR(50)
	plantillaFichaPersonaje		X	[IQR-5] Plantilla de ficha de personaje

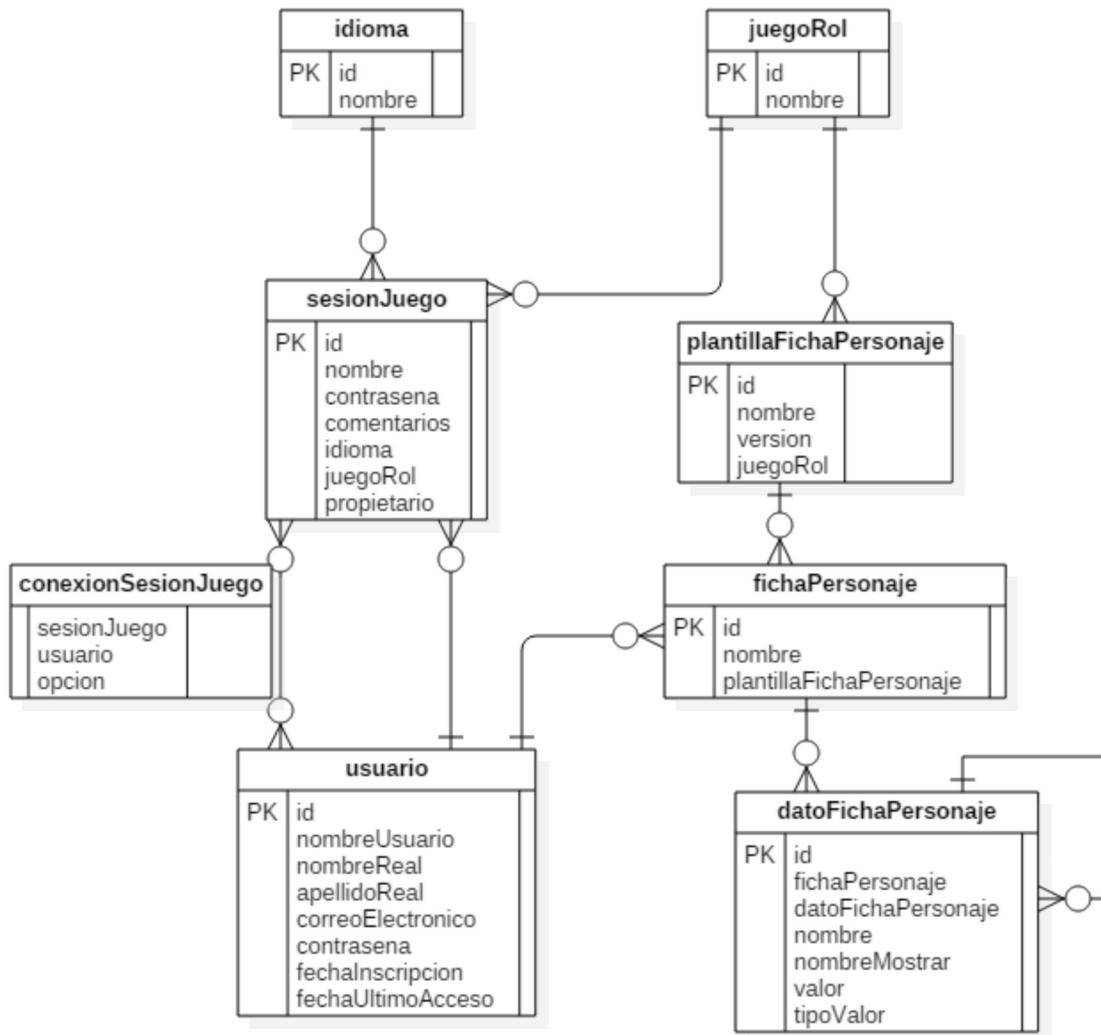
Tabla 68: Diseño de datos. fichaPersonaje

Tabla	datoFichaPersonaje			
Descripción	Son los datos del tipo grupo que componen las fichas de personaje			
Atributos	Nombre	Único	Obligatorio	Características
	id	X	X	PK INTEGER
	fichaPersonaje			[IQR-6] Ficha de personaje
	datoFichaPersonaje			[IQR-7] Dato grupo de ficha de personaje
	nombre		X	VARCHAR(50)
	nombreMostrar			VARCHAR(50)
	valor			INTEGER
	tipoValor			VARCHAR(50)

Tabla 69: Diseño de datos. datoFichaPersonaje

La mayor diferencia respecto al diagrama de clases es que la tabla “Dato de ficha de personaje” engloba todos los tipos de datos de ficha de personaje (grupo, derivado y campo). Esto es así debido a la optimización de la aplicación ya que Doctrine, el ORM que estamos usando en Symfony, opera con mayor facilidad la herencia con una sola tabla.

Finalmente mostramos el Diagrama Entidad-Relación del sistema:



Dibujo 29: Diagrama Entidad-Relación

4.4. Diseño del protocolo a emplear

Los textos tachados representan funcionalidad que no se ha implementado.

Pedir funcionalidad (Plataforma de juego → Núcleo de rol)

Pedir funcionalidad (character_sheet_id, type) //type = owner or gamemaster

Enviar funcionalidad (Núcleo de rol → Plataforma de juego)

Enviar funcionalidad (funcionalidad_lista)

```
array {
  0 {
    functionality_type (individual, collective)
    if functionality_type == individual
      identifier (donde se hace click, puede o no contener un valor)
```

```

    multiple_selection_list (null or list)
endif
launch_system {
    type (indica el tipo de sistema de lanzamiento. Si tiene un factor
dificultad como vampiro, si consiste simplemente en sumar los
resultados, etcétera)
    value {
        other_element1 (si por ejemplo se trata de vampiro, la
                        dificultad está comprendida entre 2 y 10)
        other_element2...
    }
}
list_of_modifiers {
    type (si se trata de una suma, división, multiplicación etc)
    value (una lista con los componentes para la resolución del tipo)
}
if functionality_type == collective
    name (nombre de la funcionalidad)
    access_list (Secuencia de acceso al elemento si es colectiva) {
        name
        name
    }
    function (Dañar salud, curar salud, aplicar estado, mostrar
               solamente en el log...)
    targets_number (Ej: de 1 a 3, 1, sin límite...)
    target_type (Ej: cualquiera, enemigo, aliado, todos menos tú, una
                  lista específica, por ejemplo si se expulsan muertos
vivientes, solamente se podrá hacer uso a muertos
vivientes)
endif
}
1 { (otra funcionalidad)
...
}

```

4.4.1 Ejemplo de funcionalidad individual con el juego de rol Pathfinder

```

array {
    0 {
        functionality_type: individual
        if functionality_type == individual
            identifier: acrobatics
            multiple_selection_list: null
endif
        launch_system {
            type: d20_system
            value: {
                0 {
                    type: difficulty
                    value: 23
                }
            }
        }
    }
}

```

```
list_of_modifiers {
    type: sum
    value: {
        0 {
            type: dice
            name: null
            value: 20
        }
        1 {
            type: derived
            name: acrobatics_total
            value: {
                type: sum
                name: null
                value: {
                    0 {
                        type: field
                        name: temporary_dexterity_modifier
                        value: null
                    }
                    1 {
                        type: field
                        name: acrobatics_rank
                        value: null
                    }
                    2 {
                        type: field
                        name: acrobatics_modifier
                        value: null
                    }
                }
            }
        }
    }
}
```

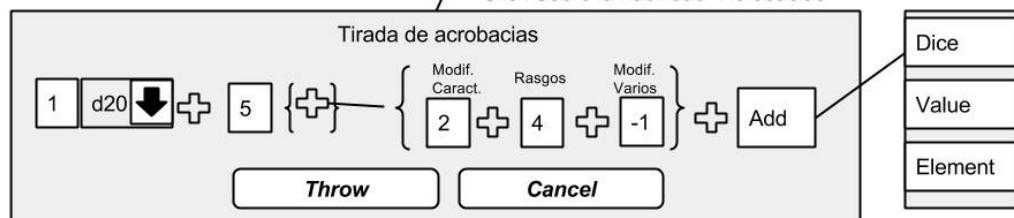
BATFINDER

JUEGO DE ROL

Hoja de personaje

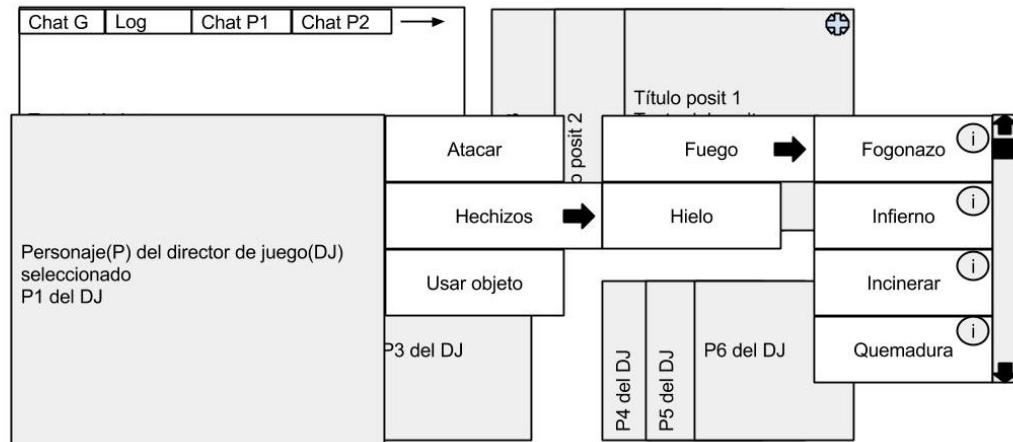
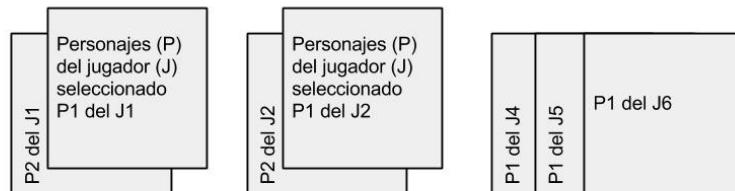
NOMBRE DEL PERSONAJE				ALINEAMIENTO		JUGADOR																																																													
CLASE DE PERSONAJE Y NIVEL				DIOS		PAÍS NATAL																																																													
RAZA		TAMAÑO		SEXO	EDAD	ALTURA	PESO																																																												
							CABELLO OJOS																																																												
PG PUNTO DE GOLPE				VELOCIDAD	TIERRA	PIES/M CAS.	PIES/M CAS.																																																												
HERIDAS/PG ACTUALES					PIES/M VELOCIDAD BASE	PIES/M CON ARMADURA	PIES/M CON ARMADURA																																																												
				VOLAR	PIES/M	PIES/M	PIES/M																																																												
				MANIOBRABILIDAD	NADAR	TRIPAR	EXCAVAR																																																												
DAÑO NO LETAL																																																																			
HABILIDADES																																																																			
<table border="1"> <thead> <tr> <th>NOMBRE DE LA HABILIDAD</th> <th>BONIC TOTAL</th> <th>MODIF CARACT.</th> <th>RANGOS</th> <th>MODIF VARIO</th> </tr> </thead> <tbody> <tr><td><input type="checkbox"/> ACROBACIAS</td><td>5</td><td>=DES</td><td>2 + 4 + -1</td><td></td></tr> <tr><td><input type="checkbox"/> ARTESANIA</td><td></td><td>=INT</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> ARTESANIA</td><td></td><td>=INT</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> ARTESANIA</td><td></td><td>=INT</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> AVERIGUAR INTENCIONES</td><td></td><td>=SAB</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> CONOCIMIENTO DE CONJUROS*</td><td></td><td>=INT</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> CURAR</td><td></td><td>=SAB</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> DIPLOMACIA</td><td></td><td>=CAR</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> DISFRAZARSE</td><td></td><td>=CAR</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> ENGAÑAR</td><td></td><td>=CAR</td><td>+ + +</td><td></td></tr> <tr><td><input type="checkbox"/> ESCAPISMO</td><td></td><td>=DES</td><td>+ + +</td><td></td></tr> </tbody> </table>								NOMBRE DE LA HABILIDAD	BONIC TOTAL	MODIF CARACT.	RANGOS	MODIF VARIO	<input type="checkbox"/> ACROBACIAS	5	=DES	2 + 4 + -1		<input type="checkbox"/> ARTESANIA		=INT	+ + +		<input type="checkbox"/> ARTESANIA		=INT	+ + +		<input type="checkbox"/> ARTESANIA		=INT	+ + +		<input type="checkbox"/> AVERIGUAR INTENCIONES		=SAB	+ + +		<input type="checkbox"/> CONOCIMIENTO DE CONJUROS*		=INT	+ + +		<input type="checkbox"/> CURAR		=SAB	+ + +		<input type="checkbox"/> DIPLOMACIA		=CAR	+ + +		<input type="checkbox"/> DISFRAZARSE		=CAR	+ + +		<input type="checkbox"/> ENGAÑAR		=CAR	+ + +		<input type="checkbox"/> ESCAPISMO		=DES	+ + +	
NOMBRE DE LA HABILIDAD	BONIC TOTAL	MODIF CARACT.	RANGOS	MODIF VARIO																																																															
<input type="checkbox"/> ACROBACIAS	5	=DES	2 + 4 + -1																																																																
<input type="checkbox"/> ARTESANIA		=INT	+ + +																																																																
<input type="checkbox"/> ARTESANIA		=INT	+ + +																																																																
<input type="checkbox"/> ARTESANIA		=INT	+ + +																																																																
<input type="checkbox"/> AVERIGUAR INTENCIONES		=SAB	+ + +																																																																
<input type="checkbox"/> CONOCIMIENTO DE CONJUROS*		=INT	+ + +																																																																
<input type="checkbox"/> CURAR		=SAB	+ + +																																																																
<input type="checkbox"/> DIPLOMACIA		=CAR	+ + +																																																																
<input type="checkbox"/> DISFRAZARSE		=CAR	+ + +																																																																
<input type="checkbox"/> ENGAÑAR		=CAR	+ + +																																																																
<input type="checkbox"/> ESCAPISMO		=DES	+ + +																																																																

Click Sobre la habilidad "Acrobacias"



Dibujo 30: Ejemplo de funcionalidad individual

Herramientas para el Director de Juego



Dibujo 31: Ejemplo de funcionalidad colectiva, selección de acción.

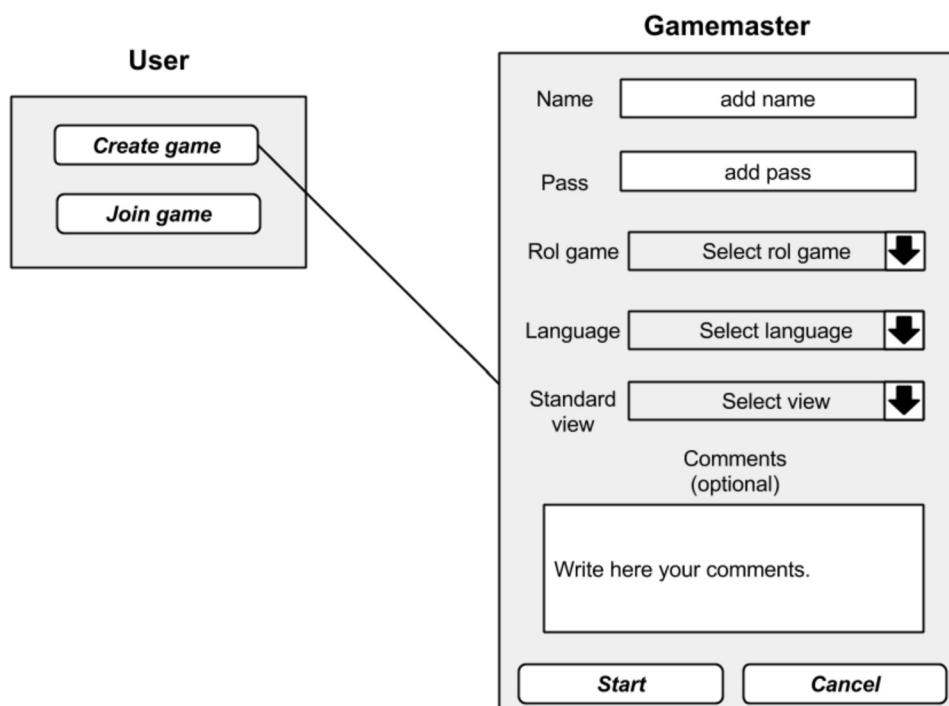
4.5. Diseño de la interfaz de usuario

En esta etapa del proyecto se elaboraron unos cuantos bocetos de interfaz aunque debido a la naturaleza cambiante del proyecto y al aprendizaje de la mayor parte de las tecnologías, estos bocetos solo sirvieron como referente para la implementación y no como producto final.

Cada herramienta que presentemos es, a su vez, un subsistema ya que hemos considerado crear diferentes herramientas en la sesión de juego, como subsistemas independientes.

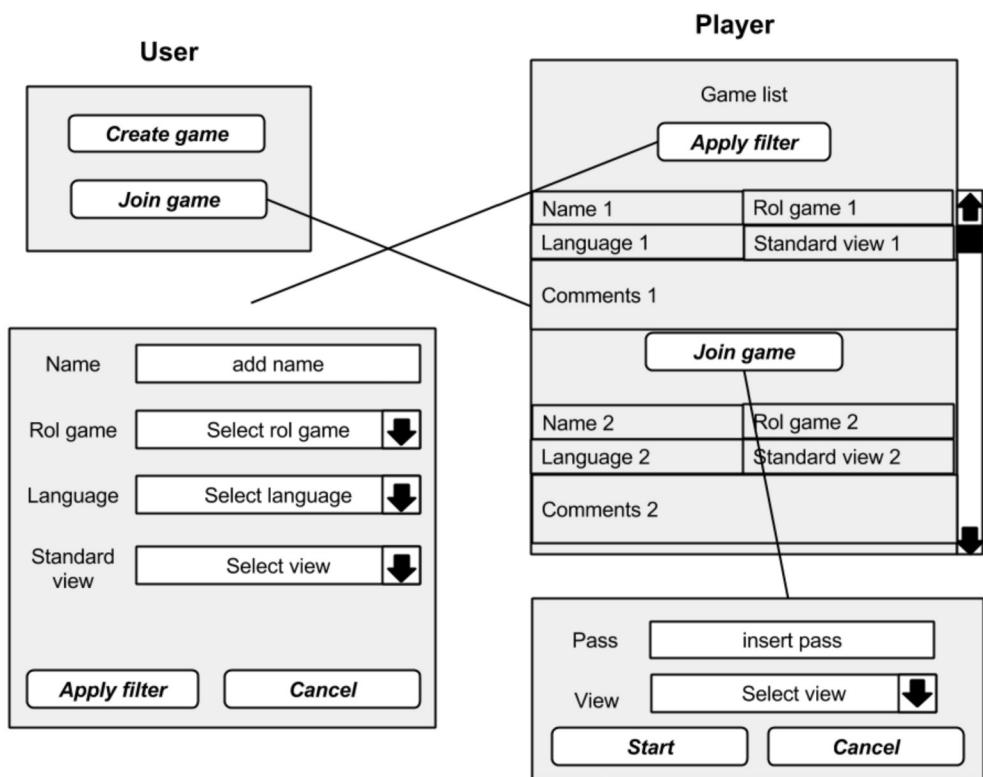
A continuación se mostrarán los bocetos realizados:

En este boceto se plantea la opción de crear la sesión de juego. La vista estándar es una de las opciones que se ha descartado en el posterior desarrollo debido a que pensamos que cualquier usuario debería poder acceder al sistema independientemente del dispositivo del que se conecte, por ello descartamos esta opción.



Dibujo 32: Boceto "crear sesión de juego"

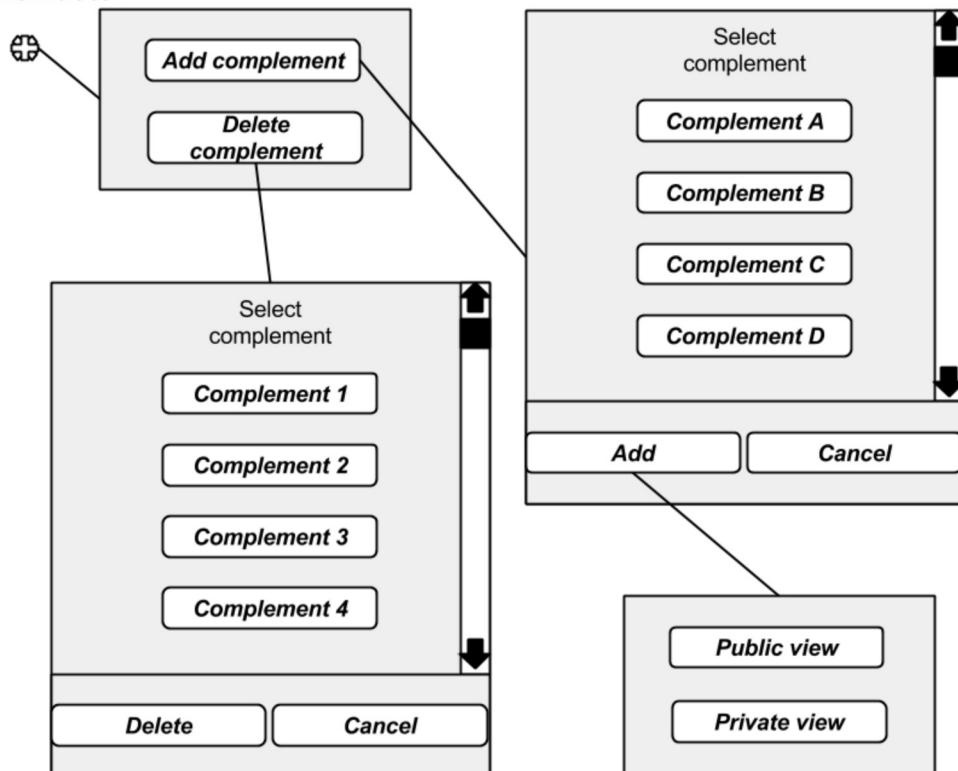
Este boceto representa la interfaz por la cual los jugadores se unen a las sesiones de juego. Al igual que en el anterior boceto en el producto final se ha descartado la opción de vista estándar.



Dibujo 33: Boceto "unirse a sesión de juego"

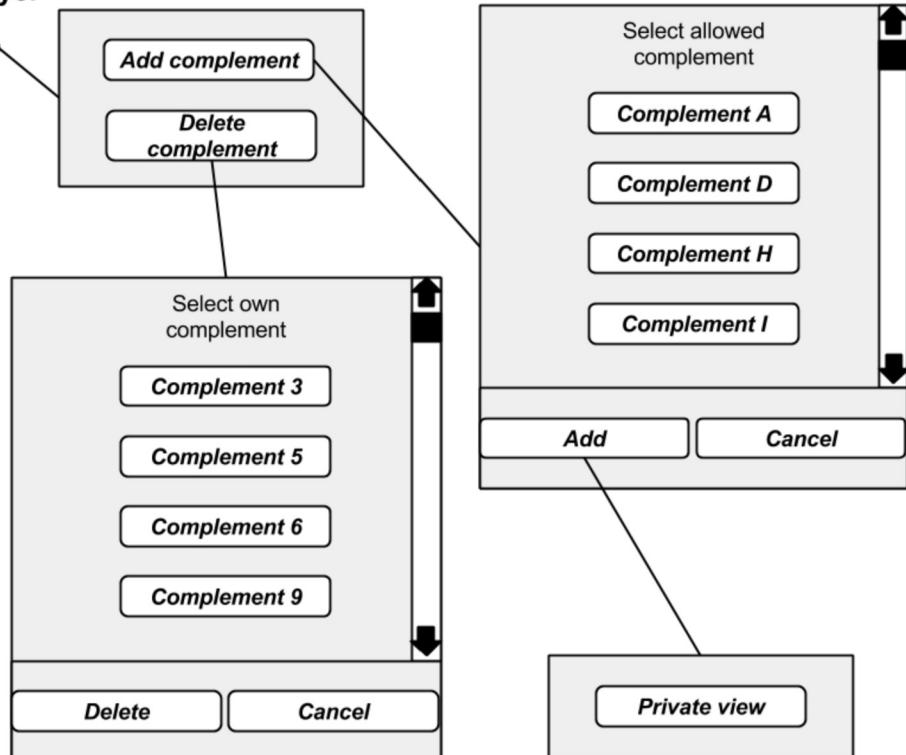
En los siguientes bocetos se presenta una característica que no ha sido implementada finalmente. Se trata de un sistema de opciones que permita visualizar o no visualizar las diferentes herramientas que se encuentren en la sesión de juego. Cada usuario tiene la opción de visualizar de forma privada alguno de los complementos para probar sus características pero el director de juego podría habilitar una visualización global de la herramienta para todos los jugadores.

Gamemaster



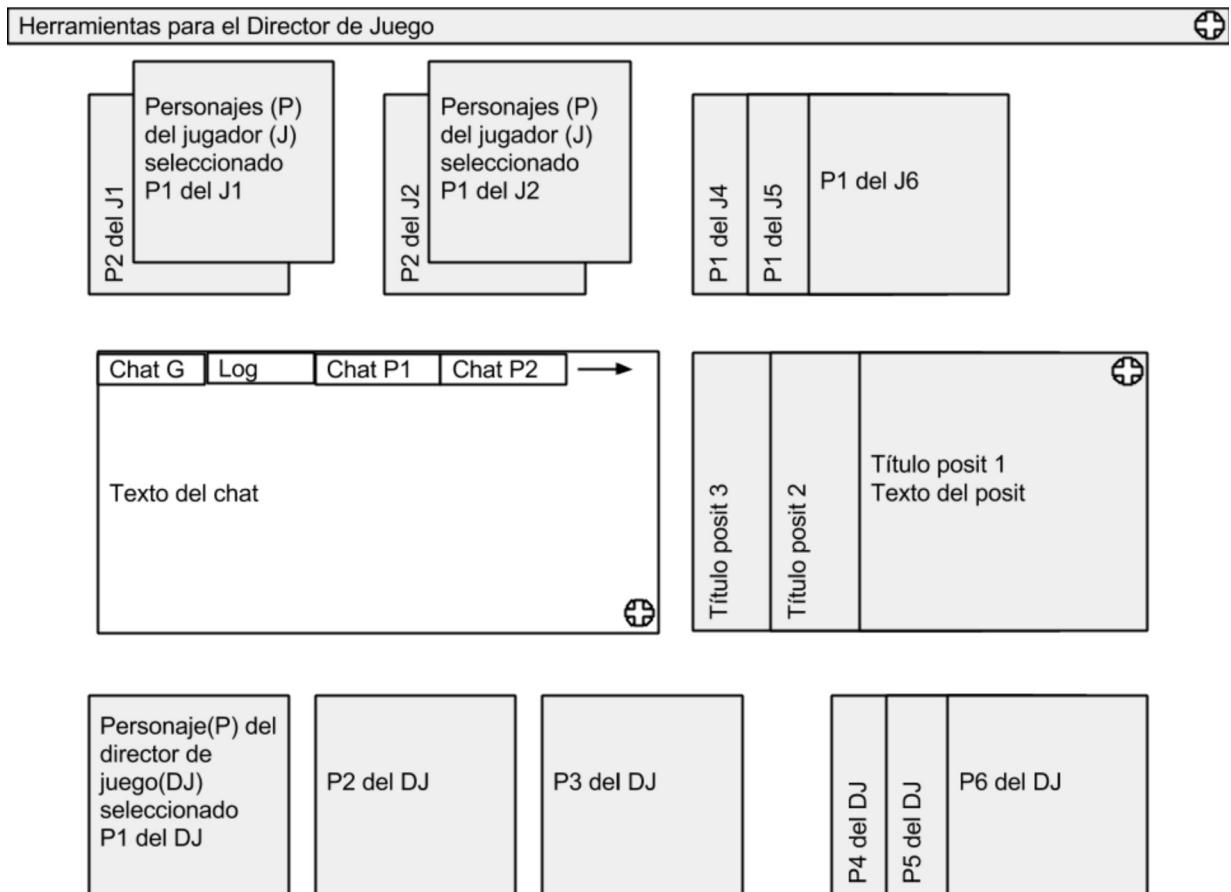
Dibujo 34: Boceto "gestionar herramientas por el director de juego"

Player



Dibujo 35: Boceto "gestionar herramientas por el usuario"

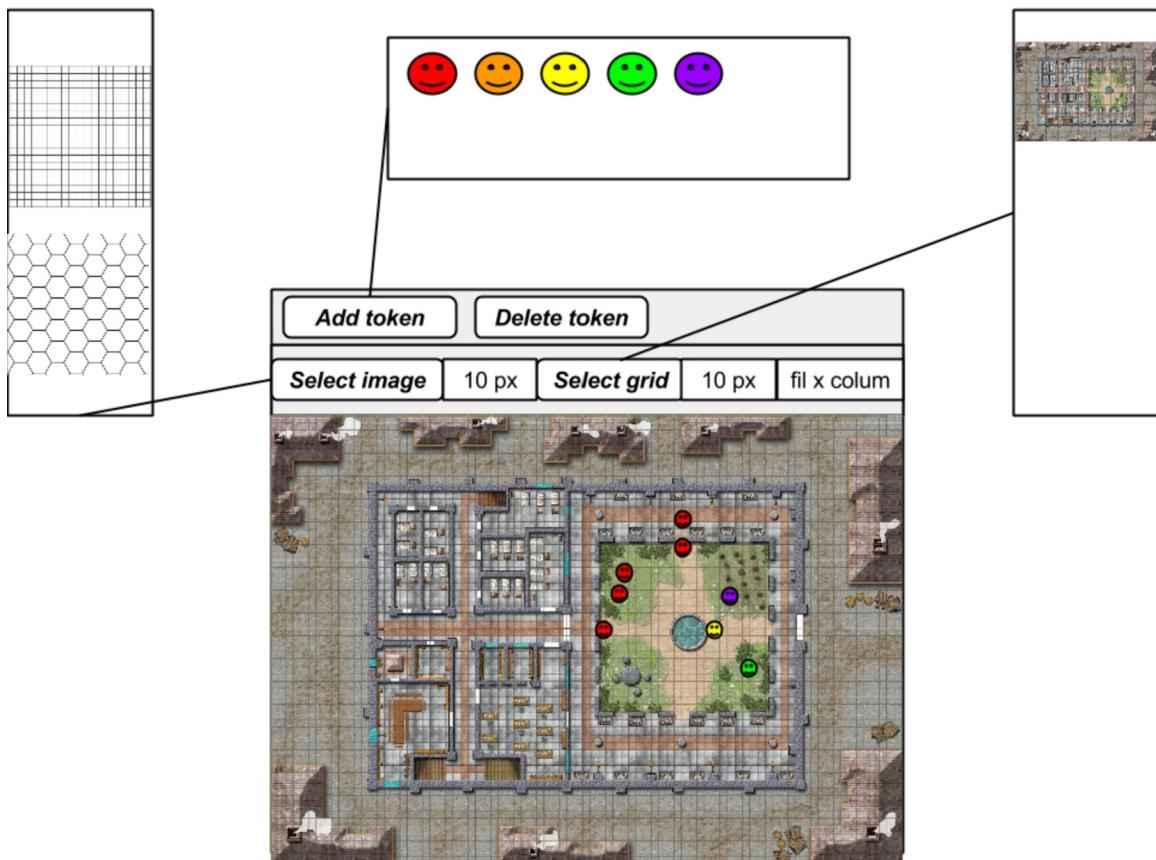
Este boceto representa la interfaz de la sesión de juego. En el diseño inicial se planteaba una visualización de elementos no desplazables, es decir, se importaba una ficha o se habilitaba el chat por ejemplo y estos permanecían siempre en la misma posición de visualización de la sesión de juego. Aunque esto al final no fue así ya que se decidió permitir que el usuario desplazara por la ventana de juego los diferentes elementos para que los situara donde quisiera.



Dibujo 36: Boceto "sesión de juego"

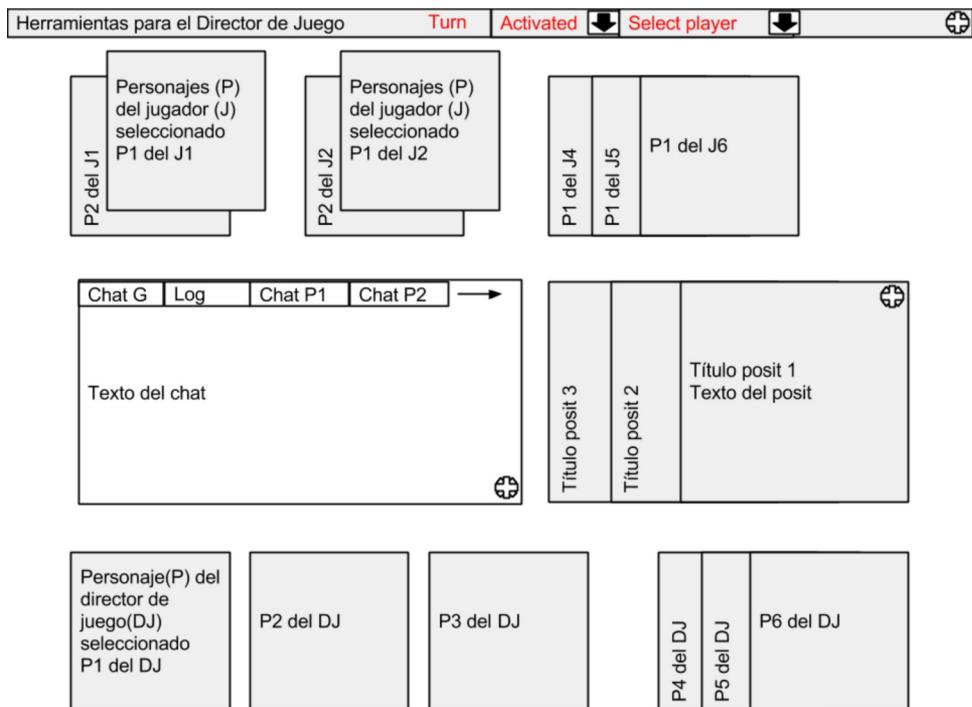
A continuación vamos a mostrar los bocetos de las diferentes herramientas que se pensaron para la sesión de juego.

Esta interfaz muestra la idea del mapa para la sesión de juego, donde se puede añadir un mapa, un tipo de cuadricula y fichas con diferentes configuraciones. Finalmente se optó por un único mapa y con la opción de añadir y eliminar fichas además de moverlos por el mapa.



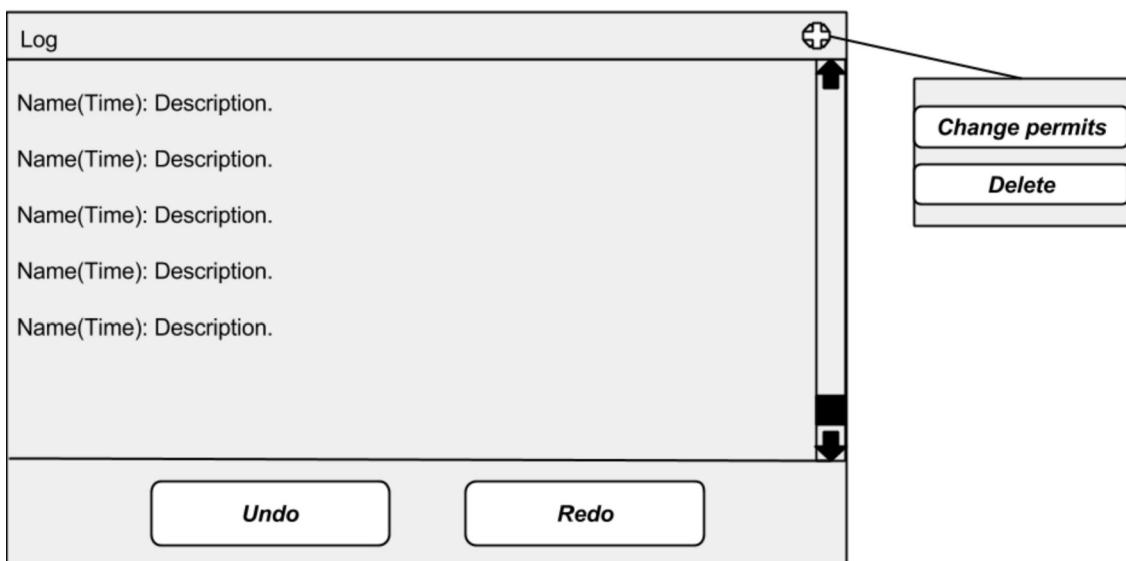
Dibujo 37: Boceto "subsistema de mapa"

La siguiente interfaz muestra la herramienta de turnos que permite al director de juego gestionar qué jugador puede realizar acciones. Por su puesto el director de juego siempre puede actuar y puede habilitar y deshabilitar esta opción cuando desee. Finalmente esta funcionalidad no ha sido implementada en el producto final.

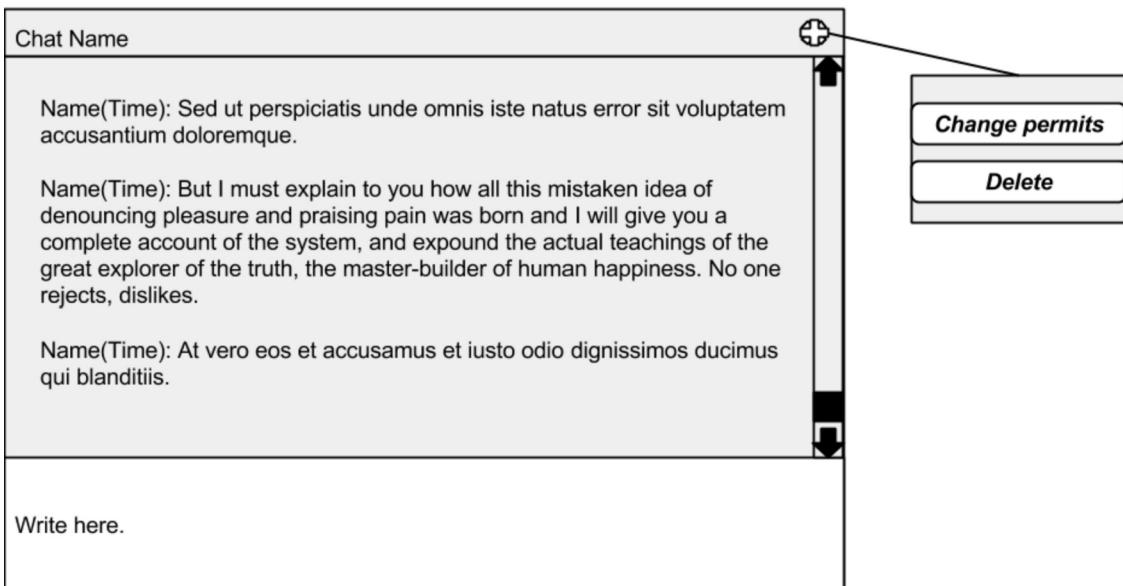


Dibujo 38: Boceto "subsistema de turno"

Las siguientes interfaces representan el chat y el historial de la sesión de juego, aunque finalmente ambas quedaron resumidas en una misma ventana. Cabe destacar la funcionalidad deshacer y rehacer que se planteó en el historial, donde las acciones realizadas por los jugadores que implicaran cambios en los demás elementos de juego pudieran ser rehechas o desecharas, por su puesto por mano del director de juego. Además de esto, el sistema de permisos no se ha desarrollado, ya que implicaría un sistema de chat mucho más elaborado con múltiples ventanas.



Dibujo 39: Boceto "subsistema de historial"



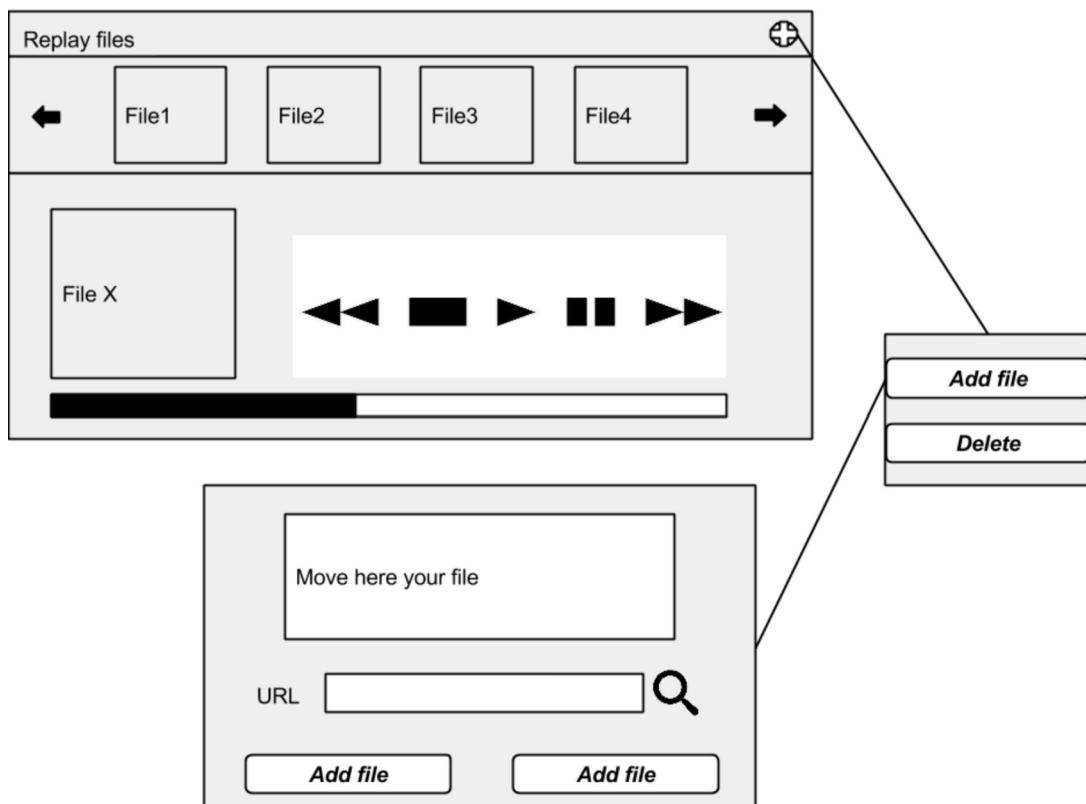
Dibujo 40: Boceto "subsistema de chat"

La siguiente interfaz representa la herramienta de anotaciones, herramienta no desarrollada, cuya función es simple, permitir anotaciones en las sesiones de juego. Debido a su simplicidad y aparente inutilidad fue descartada del desarrollo.



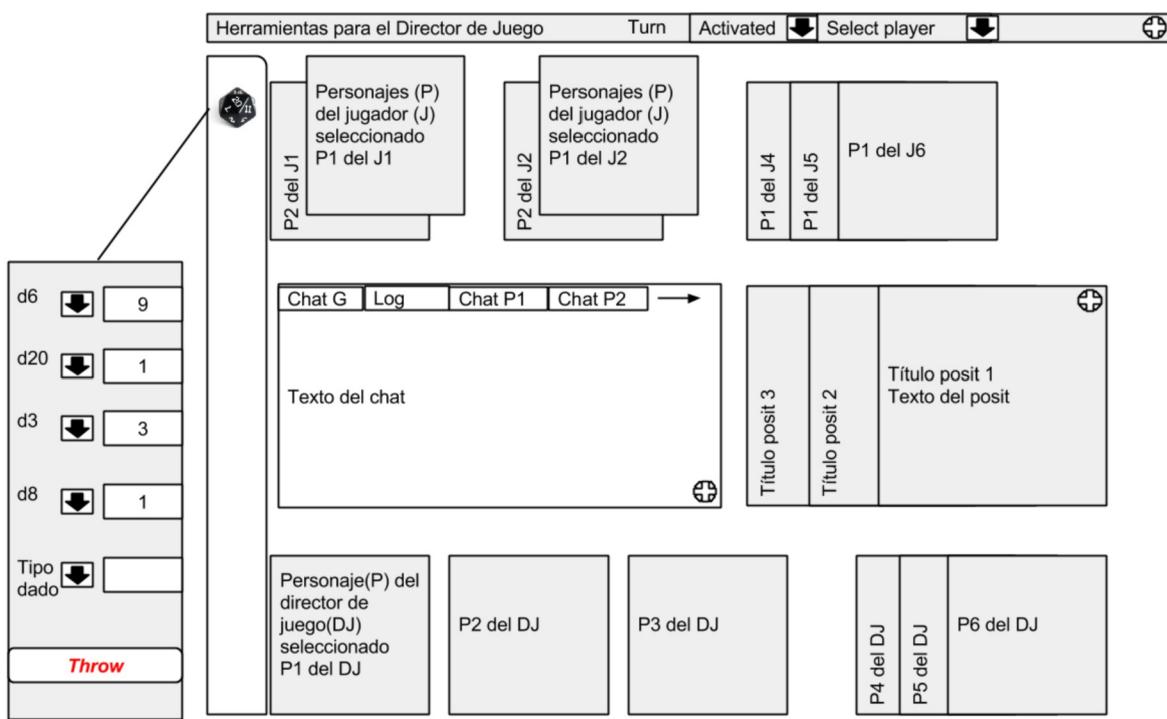
Dibujo 41: Boceto "subsistema de anotación"

La siguiente interfaz representa una herramienta muy potente que al final no ha sido desarrollada debido a su complejidad. Esta herramienta multimedia permitiría al director de juego subir imágenes y vídeos y reproducirlos a tiempo real con el resto de usuarios. Esto permitiría dar una ambiente mucho más cercano a una sesión de rol mesa presencial.



Dibujo 42: Boceto "subsistema multimedia"

Finalmente la última interfaz representa la herramienta de tirada de dato. Esta funcionalidad sí está implementada en el producto final.



Dibujo 43: Boceto "subsistema de lanzamiento de dado"

5. Implementación del sistema

Este capítulo trata sobre todos los aspectos relacionados con la implementación del sistema en código, haciendo uso de un determinado entorno tecnológico.

5.1. Entorno tecnológico

En la sección 3.7. Elección de tecnologías ya definimos las tecnologías y herramientas que íbamos a utilizar pero sin entrar en mucho detalle. Esta sección concretaremos todas y cada una de las herramientas y tecnologías implicadas en el desarrollo del sistema.

Vamos a desglosarlo en diferentes categorías de software para que sea más cómodo:

Sistema operativo

El sistema operativo que debería ser usado para el producto final (producción) es Ubuntu Server 14.04 LTS. Aunque el que se ha usando para el desarrollo y en el que se encuentra ahora mismo el sistema es Ubuntu 14.04 LTS, Ubuntu 14.10, Ubuntu 15.04 y Ubuntu 15.10.



Servidor HTTP

El servidor de HTTP es Apache (~2.4.23) [24] que está instalado y funcionando en Ubuntu.



Servidor web

Esta parte es la más extensa y son las tecnologías con las que se ha desarrollado la aplicación en el lado del servidor:

- PHP (~5.3.9) [25] [26].
- Symfony (~2.8) [27] [28], framework de PHP.
- Composer (1.2.0) [29], para la gestión de dependencias del proyecto.

- Las tecnologías doctrine/orm (~2.5), doctrine/dbal (~2.5) y el bundle doctrine/doctrine-bundle (~1.4) [30] para gestionar como ORM en Symfony.
- Twig [31] como motor de plantillas de Symfony.
- YAML [32] como componente que permite la configuración de Symfony y algunos bundles. Es un componente de Symfony.
- El bundle phpunit/phpunit (5.4) que integra a PHPUnit [33] para las pruebas unitarias de PHP en Symfony.
- La tecnología Application Messaging Protocol (WAMP) [34]. WAMP es un subprotocolo abierto y estándar de WebSocket que ofrece dos patrones de aplicaciones de mensajería en un protocolo unificado: Llamadas a procedimientos remotos además de publicar y suscribirse.
- El bundle GeniusesOfSymfony/WebSocketBundle (~1.8.3) [35] que permite que en Symfony se pueda utilizar la tecnología WAMP. Usa las implementaciones de Autobahn|JS [36] en el lado del cliente y de Ratchet [37] en la parte del servidor.
- El bundle friendsofsymfony/user-bundle (~1.3.6) [38] que provee de una buena base para la gestión de usuarios.
- twbs/bootstrap (3.3.7) [39], es el framework Bootstrap de CSS.
- El bundle braincrafted/bootstrap-bundle (2.2.0) [40] para poder usar Bootstrap en la aplicación.
- El bundle oyejorge/less.php (~1.5) [41] para incorporar Less al sistema, requerido por Bootstrap.
- El bundle knplabs/knp-menu-bundle (2.1.2) [42] para incorporar un sistema de menús para navegación en la web.
- Los componentes de Symfony components/jquery y components/jqueryui para poder usar jQuery (3.1.0) [43] y jQuery UI (1.12.0) [44] en la aplicación.



Dibujo 44: PHP logo



Dibujo 45: PHP Unit logo



Dibujo 46: Symfony logo



Dibujo 47: Twig logo



Dibujo 48: WAMP logo



Dibujo 49: Composer logo

Base de datos

MySQL [45] como base de datos SQL junto con MySQL Workbench [46] que es una aplicación que permite visualizar las bases de datos junto con sus tablas y tuplas, además de permitir realizar operaciones en la base de datos.



Dibujo 50:
MySQL logo

Cliente

En el lado del cliente las tecnologías que se están usando son:

- HTML [47].
- CSS [48] junto con Bootstrap.
- JavaScript [49] junto con jQuery y jQueryUI [50].

Además hay un par de tecnologías que sirven para pasar información entre el servidor y el cliente y son:
JSON [51] y AJAX.



Entorno de desarrollo

El entorno de desarrollo que se utiliza es Eclipse Neon [52] debido a que posee de varios plugins que permiten trabajar fácilmente con algunas de las tecnologías y son: Symfony Plugin, Doctrine Plugin, Twig Plugin y jQuery Plugin.



Dibujo 58:
Eclipse logo

Control de versiones

Para el control de versiones la única herramienta que se está utilizando es Git [53], teniendo el proyecto en un repositorio de GitHub [54].



Dibujo 59: Git logo

5.2. Parametrización del software base

En esta sección se va a presentar diferentes configuraciones de la aplicación.

Los ficheros de configuración de Symfony son en YAML y este es el lenguaje que hemos usado para configurar nuestra aplicación. A continuación explicaremos las configuraciones más importantes.

5.2.1. routing.yml

Symfony utiliza las rutas para configurar el controlador que corresponde a cada una de ellas. Esta es la configuración principal de cada uno de los bundles de nuestra aplicación.

```
web_platform:
    resource: "@WebPlatformBundle/Resources/config/routing.yml"
    prefix:   /

gaming_platform:
    resource: "@GamingPlatformBundle/Resources/config/routing.yml"
    prefix:   /

game_session:
    resource: "@GameSessionBundle/Resources/config/routing.yml"
    prefix:   /
fos_user:
    resource: "@FOSUserBundle/Resources/config/routing/all.xml"
    prefix:   /{_locale}
    defaults:
        _locale: %app.locale_default%
    requirements:
        _locale: %app.locales_string%
```

5.2.2. security.yml

En el fichero de configuración security están configurados los roles que pueden tener los usuarios en el sistema, el firewall principal (solo usamos uno) y las restricciones a la hora de acceder a las diferentes rutas. La restricciones se forman usando las URL y los roles de usuario.

Además de lo anterior se establece cual es la ruta de acceso, la ruta para desconectarse y el tipo de cifrado para la contraseña de los usuarios: bcrypt.

Al estar la contraseña encriptada, un administrador, aún teniendo acceso a la base de datos, no puede ver la contraseña de un usuario. Si el administrador quisiera poner una contraseña de esta manera, debería añadir al campo de la base de datos la contraseña encriptada.

```

encoders:
    FOS\UserBundle\Model\UserInterface: bcrypt

role_hierarchy:
    ROLE_ADMIN:       ROLE_USER
    ROLE_SUPER_ADMIN: ROLE_ADMIN

firewalls:
    main:
        pattern:      ^
        form_login:
            provider: fos_userbundle
            csrf_token_generator: security.csrf.token_manager
            login_path: fos_user_security_login
            check_path: fos_user_security_check
            default_target_path: web_platform_homepage

        logout:
            path: fos_user_security_logout
            target: web_platform_homepage
        anonymous: true

access_control:
    - { path: ^/[^/]+/login$, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/[^/]+/register, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/[^/]+/resetting, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/[^/]+/administration, role: ROLE_SUPER_ADMIN }
    - { path: ^/[^/]+/game, roles: ROLE_USER }

```

5.2.3. Generar tablas con doctrine

A la hora de generar el modelo de datos, hemos optado por configurar doctrine (con YAML) para que nos genere las tablas con los campos, restricciones y relaciones con las demás tablas. Aunque para hacer esto es necesario tener las clases implementadas, así doctrine referencia el código PHP con las tablas y campos de la base de datos.

En este ejemplo vemos como configuramos el modelo de las sesiones de juego:

```

GameSessionBundle\Entity\GameSession:
    type: entity
    repositoryClass: GameSessionBundle\Entity\GameSessionRepository
    table: game_session
    id:
        id:
            type: integer
            generator: { strategy: AUTO }
    fields:
        name:
            type: string
            length: 255
        password:
            type: string
            length: 4096
        start_date:
            type: datetime
        comments:

```

```

        type: text
        length: 200
        nullable: TRUE
manyToOne:
    owner:
        targetEntity: UserBundle\Entity\User
        joinColumn:
            name: owner_id
            referencedColumnName: id
    rol_game:
        targetEntity: GameSessionBundle\Entity\RolGame
        joinColumn:
            name: rol_game_id
            referencedColumnName: id
language:
    targetEntity: GameSessionBundle\Entity\Language
    joinColumn:
        name: language_id
        referencedColumnName: id

```

5.2.4. Validaciones

A la hora de crear y modificar podemos necesitar diferentes tipos de validaciones. A través de una sencilla configuración podemos definir estas validaciones.

En el siguiente ejemplo podemos ver como tenemos dos validaciones para la sesión de juego. En uno de ellos lo utilizamos para crearla y en el otro para acceder. Además nos permite personalizar los mensajes que se le muestran al usuario indicando el problema y con varios idiomas si lo tenemos configurado de esta manera.

```

GameSessionBundle\Entity\GameSession:
    properties:
        id:
            - IsNull:
                groups: [Create]
        name:
            - NotBlank:
                message: name.not_blank
                groups: [Create]
            - Length:
                min: 5
                max: 30
                minMessage: name.minimum_characters
                maxMessage: name.maximum_characters
                groups: [Create]
        owner:
            - IsNull:
                groups: [Create]
        password:
            - NotBlank:
                message: plainPassword.not_blank
                groups: [Create, Login]
            - Length:
                min: 6
                max: 30
                minMessage: password.minimum_characters
                maxMessage: password.maximum_characters
                groups: [Create, Login]

```

```

start_date:
    - IsNull:
        groups: [Create]
rol_game:
    - NotBlank:
        message: rol_game.not_blank
        groups: [Create]
language:
    - NotBlank:
        message: language.not_blank
        groups: [Create]
comments:
    - Length:
        max: 200
        maxMessage: comments.maximum_characters
        groups: [Create]

```

5.2.5. Traducciones

Como hemos mencionado anteriormente, en Symfony podemos configurar un sistema multi idioma que nos permita proveer de forma sencilla diferentes traducciones a los usuarios. Para hacer esto basta con configurar un fichero de traducción que se presenta con el formato siguiente:

```

game_session:
    name: Name
    password: Password
    rol_game: Rol game
    language: Language
    comments: Comments
    gamemaster: Gamemaster

game_session:
    name: Nombre
    password: Contraseña
    rol_game: Juego de rol
    language: Idioma
    comments: Comentarios
    gamemaster: Director de juego

```

En este caso tenemos las traducciones de la sesión de juego tanto en inglés como en español y podríamos añadir fácilmente un nuevo idioma al sistema. Para poder acceder a un valor concreto basta con escribir con el siguiente formato: game_session.password y así accederíamos a esta traducción de forma automática.

Para configurar nuevos idiomas basta con añadir a la configuración principal. Por su puesto hay que modificar la forma de acceder a ellos en el menú de selección de idiomas y añadir a la base de datos el nuevo idioma si la aplicación lo necesita, como es nuestro caso.

El fichero se vería de la siguiente manera:

```

app.locales_string: en|es
app.locales:
    - en
    - es
app.locale_default: en

```

5.2.6. Enrutamiento

Como hemos comentado antes, Symfony maneja las URL mediante el enrutamiento. Este es un ejemplo cuya URL es la indicada para crear una sesión de juego:

```
create_game_session:  
    path:      /{_locale}/game/create-game-session  
    defaults:  
        _controller: GameSessionBundle:GameSession:createGameSession  
        _locale: %app.locale_default%  
    requirements:  
        _locale: %app.locales_string%
```

Como podemos ver es fácil pasar parámetros a las URL incluso si los tenemos configurados en nuestros ficheros, como es el caso de %app.locale_*%

5.2.7. Dependencias

Estamos usando Composer para la gestión de dependencias en el fichero composer.json. Nuestra aplicación tiene la siguiente configuración de dependencias:

```
"require": {  
    "php": ">=5.3.9",  
    "symfony/symfony": "~2.8",  
    "doctrine/orm": "~2.2,>=2.2.3,<2.5",  
    "doctrine/dbal": "<2.5",  
    "doctrine/doctrine-bundle": "~1.4",  
    "symfony/assetic-bundle": "~2.3",  
    "symfony/swiftmailer-bundle": "~2.3",  
    "symfony/monolog-bundle": "~2.4",  
    "sensio/distribution-bundle": "~4.0",  
    "sensio/framework-extra-bundle": "~3.0,>=3.0.2",  
    "incenteev/composer-parameter-handler": "~2.0",  
    "friendsofsymfony/user-bundle": "~2.0@dev",  
    "gos/web-socket-bundle": "^1.6",  
    "braincrafted/bootstrap-bundle": "^2.1",  
    "oyejorge/less.php": "~1.5",  
    "components/jquery": "^2.1",  
    "twbs/bootstrap": "^3.3",  
    "knplabs/knp-menu-bundle": "^2.1",  
    "components/jqueryui": "^1.11",  
    "drmonty/smartymenus": "^1.0",  
    "phpunit/phpunit": "^5.4"  
},
```

5.3. Código fuente

En esta sección se va a comentar la organización del código del proyecto.

Como se ha especificado desde un principio, el proyecto está desarrollando en Symfony2 y éste tiene su propia estructura de carpetas con su respectiva utilidad. A continuación mostramos los directorios y archivos de la raíz del proyecto:

- PHP Language Library. Incluye la librería PHP con toda su API.

- **src/** Código PHP del proyecto.
- **app/** La configuración de la aplicación.
- **bin/** Ficheros ejecutables.
- **vendor/** Las dependencias de terceros.
- **web/** El directorio raíz web donde se encuentran los ficheros CSS e imágenes.
- **composer.json** Es el archivo donde se encuentran gestionadas las dependencias de la aplicación.

En Symfony2 las agrupaciones funcionales suelen estar agrupadas en un bundle. Para Symfony “Un bundle es un concepto similar al de los plugins en otras aplicaciones, pero todavía mejor. La diferencia clave es que en Symfony2 todo es un bundle, incluyendo tanto la funcionalidad básica de la plataforma como el código escrito para tu aplicación”.

La carpeta **src/** se divide en bundles:

- **UserBundle.** Es donde se gestionan a todos los usuarios de la aplicación.
- **GameSessionBundle.** En este bundle recogemos todas las entidades junto con sus métodos para la gestión de la sesión de juego, plantilla de ficha de personaje, ficha de personaje, juego de rol e idioma.
- **GamingPlatformBundle.** Como su nombre indica en este bundle es donde está toda la lógica de la plataforma de juego, donde los usuarios se conectan y desarrollan sus partidas.
- **WebPlatformBundle.** Aquí es donde se tiene la funcionalidad de toda la visualización, excepto la de la plataforma de juego.

5.4. Código destacado

En esta sección vamos a comentar las partes de código que consideramos más interesantes de ver.

5.4.1. Inyección de dependencias en el servidor de websockets

`GameSessionTopic` es la clase que usamos para gestionar la ejecución de las sesiones de juego, es decir, es el servidor de websockets. Para utilizar diferentes componentes de Symfony, utilizamos la inyección de dependencias. Se ve de la siguiente manera:

```
services:
    game_session.topic:
        class: JJSR\Bundle\GamingPlatformBundle\Topic\GameSessionTopic
        tags:
            - { name: gos_web_socket.topic }
        arguments:
            - @gos_web_socket.websocket.client_manipulator
            - @doctrine.orm.entity_manager
            - @form.factory
            - @validator.builder
            - @translator.default

class GameSessionTopic extends Controller implements TopicInterface
{
    protected $clientManipulator;
    protected $em;
    protected $formFactory;
    protected $validation;
```

```

protected $translator;

public function __construct(
    ClientManipulatorInterface $clientManipulator,
    EntityManager $em,
    FormFactory $formFactory,
    ValidatorBuilder $validation,
    Translator $translator)
{
    $this->clientManipulator = $clientManipulator;
    $this->em = $em;
    $this->formFactory = $formFactory;
    $this->validation = $validation;
    $this->translator = $translator;
}
//...
}

```

5.4.2. Conexión a websocket segura

Para modularizar al máximo la aplicación, las diferentes funcionalidades de la sesión de juego que requieren una subscripción se han dividido.

En el siguiente ejemplo vemos como está por un lado la seguridad del sistema, que verifica si la sesión de juego existe y si el usuario existe y tiene permisos para acceder. Por el otro lado vemos diferentes funcionalidades que se utilizan en la sesión de juego como la conexión en sí, el chat, importar fichas de personaje y el mapa.

Hay que tener en cuenta que solo se utiliza una única clase para englobar a todas las demás por lo que en el momento en el que se abra la conexión, puede hacerse uso de cualquier otra funcionalidad aunque no esté en el método `onSubscribe(...)`, ya que este solo se ejecuta en el momento en el que el usuario accede a la sesión de juego, después es `onPublish(...)` el que se encarga de gestionar las acciones de los usuarios en la sesión de juego.

```

/**
 * This will receive any Subscription requests for this topic.
 *
 * @param ConnectionInterface $connection
 * @param Topic $topic
 * @param WampRequest $request
 * @return void
 */
public function onSubscribe(ConnectionInterface $connection, Topic $topic,
WampRequest $request)
{
    if (self::onSubscribeConectionSecurity($connection, $topic, $request) == true)
    {
        self::onSubscribeConnection($connection, $topic, $request);
        self::onSubscribeChat($connection, $topic, $request);
        self::onSubscribeImportCharacterSheet($connection, $topic, $request);
        self::onSubscribeMap($connection, $topic, $request);
    }
}

```

A continuación veremos la configuración desde la parte del cliente, donde se hace la conexión al servidor de websockets:

```

<script type="text/javascript">
// GLOBAL VARIABLES FROM TWIG
var room = "{{ game_session.getId() }}";
var rol_game_name = "{{ game_session.getRolGame.getName() }}";
var user_username = "{{ app.user.username }}";
var game_session_owner_username =
"{{ game_session.getOwner().getUsername() }}";
var _WS_URI = "ws://{{ gos_web_socket_server_host }}:
{{ gos_web_socket_server_port }}";
var websocket = WS.connect(_WS_URI);
// GLOBAL VARIABLES FROM TWIG
// ||||||| WEB SOCKET |||||||
websocket.on("socket/connect", function(session) {
...
})

```

5.4.3. Ejemplo importar ficha de personaje con websocket

Cuando un usuario está viendo las fichas de personaje que puede importar y selecciona una, desde el cliente esto es lo que se ejecuta:

```

$( "#myModal" ).on("click", "#character_sheet_import_button", function() {
  var character_sheet_id =
    getCharacterSheetProperty(this, 'character_sheet_id');
  closePopup();
  session.publish(route_game_session, {
    section:'import_character_sheet',
    option: 'import',
    character_sheet_id: character_sheet_id});
});

```

Como podemos ver se está enviando al websocket el id de una ficha de personaje que se desea importar. Ahora desde el servidor se recoge el id y se obtiene la ficha formateada en JSON con la funcionalidad ejecutable. Todo esto para devolverla al cliente y que éste la formatee a HTML y CSS para que el usuario la visualice.

```

$character_sheet = self::getCharacterSheet($event['character_sheet_id']);

$formatted_character_sheet =
self::getFormattedCharacterSheet($character_sheet, $connection);

$character_sheet_functionality =
self::getFunctionalityFromCharacterSheet($character_sheet->getCharacterSheetTemplate());

$formatted_character_sheet['character_sheet_functionality'] =
$character_sheet_functionality;

```

```

$character_sheet_json = json_encode($formatted_character_sheet);

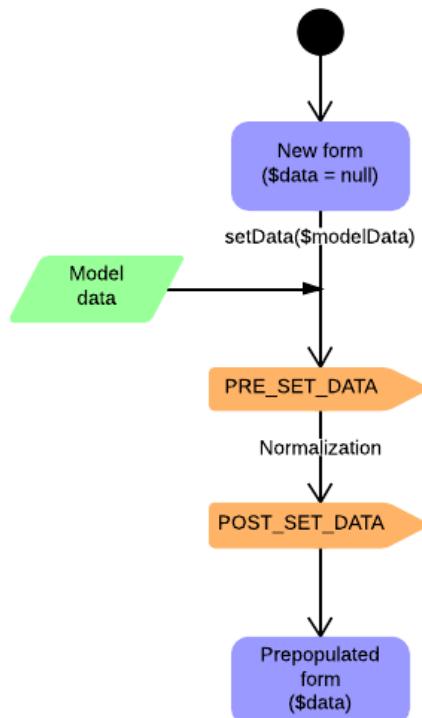
$topic->broadcast([
    'section' => 'import_character_sheet',
    'option' => 'import',
    'character_sheet_json' => $character_sheet_json
]);

```

5.4.4. Renderización de formularios dinámicos

Cuando un usuario va a crear una ficha de personaje, ésta está compuesta de muchas entidades del tipo “CharacterSheetData” pero el formulario no está creado de forma estática si no que se construye en base a una configuración que hemos definido previamente de forma manual. La configuración previa no forma parte de este proyecto ya que sería parte de la creación de plantillas de fichas de personaje.

Los formularios de Symfony poseen un conjunto de eventos que se ejecutan de forma automática cuando el formulario llega al estado de representación. En nuestro caso solo hacemos uso del evento “onPreSetData” que se ejecuta justo después de que los datos iniciales de construcción del formulario se hayan añadido. Esto nos permite generar o modificar nuevas entidades en base a las que ya existen en el formulario.



Dibujo 60: Flujo de un formulario en Symfony

En nuestro código siguiente estamos generando los “CharacterSheetData” mediante la herencia que tienen entre ellos.

```

class CharacterSheetDataEditableType extends AbstractType
{

```

```

public function buildForm(FormBuilderInterface $builder, array
    $options)
{
    $builder->add('name', HiddenType::class);
    $builder->addEventSubscriber(FormEvents::PRE_SET_DATA, array($this,
        'onPreSetData'));
}

public function onPreSetData(FormEvent $event)
{
    $character_sheet_data = $event->getData();
    $form = $event->getForm();
    self::configureElement($character_sheet_data, $form);
}

```

buildForm se ejecuta cuando se genera el formulario por primera vez y añade los eventos que se van a utilizar, en nuestro caso “onPreSetData”. Luego definimos nuestro evento donde por cada elemento que nos llegue vamos a realizarle una configuración que mostramos a continuación:

```

private function configureElement(CharacterSheetData,
    $character_sheet_data, $form)
{
    if ($character_sheet_data != null) {
        $form->add('datatype', HiddenType::class, array(
            'attr' => array('readonly' => true)
        ));
        if ($character_sheet_data->getDisplayName()) {
            $form->add('display_name', TextType::class, array(
                'attr' => array('label_col' => 2, 'widget_col' => 5,
                    'readonly' => true)
            ));
        }
        switch ($character_sheet_data->getDatatype()) {
            case 'group':
                $form->add('character_sheet_data', CollectionType::class,
                    array(
                        'attr' => array('id-form' => $character_sheet_data
                            ->getName()),
                        'entry_type' =>
                            CharacterSheetDataEditableType::class
                    ));
                break;

            case 'field':
                $pattern = array(
                    'pattern' => '/^[\wñáéíóú .\w-]+$/i',
                    'match' => true,
                    'message' => 'constraints.alphanumeric',
                );
                if ($character_sheet_data->getValidationType()) {
                    switch ($character_sheet_data->getValidationType()) {
                        case 'only_integer_numbers':
                            $pattern = array(
                                'pattern' => '/^[\wñáéíóú .\w-]+$/i',
                                'match' => true,
                                'message' =>
                                    'constraints.only_integer_numbers',
                            );
                
```

```

        break;

    case 'alphanumeric':
        $pattern = array(
            'pattern' => '/^[\a-z0-9 ñáéíóú .\-\-]+$/i',
            'match'   => true,
            'message' => 'constraints.alphanumeric',
        );
        break;
    }

$_form->add('value', TextType::class, array(
    'cascade_validation' => true,
    'constraints' => array(
        new NotBlank(),
        new Length(array('min' => 1, 'max' => 50)),
        new Regex($pattern),
    ),
    'attr' => array('id-form' => $character_sheet_data
                    ->getName(), 'label_col' => 2, 'widget_col' => 3)
));
break;
case 'derived':
$_form->add('value', TextType::class, array(
    'attr' => array('id-form' => $character_sheet_data
                    ->getName(), 'label_col' => 2, 'widget_col' => 3,
                    'readonly' => true)
));
break;
}
}
}

```

Como podemos ver, nuestra forma de construir el formulario se basa en una serie de pasos:

1. Comprobamos si existe la entidad.
2. Comprobamos si tiene un nombre que mostrar y lo añadimos si se da el caso.
3. Dependiendo del tipo de dato lo añadimos de forma diferente.
 1. group: añade a entry_type “CharacterSheetDataEditableType” que indica que a su vez tiene que generar los datos de esa entidad por lo que envía a sus hijos al ciclo de renderización del formulario.
 2. field: añade una entrada de texto con las validaciones pertinentes según el valor configurado.
 3. derived: este campo solo indica que se puede leer ya que al ser derivado, el usuario no tiene que poder modificarlo.

6. Pruebas del sistema

En este capítulo se han documentado los diferentes tipos de pruebas que se han llevado a cabo. Algunas mediante listas de comprobación (manuales) y otras mediante PHPUnit para las pruebas funcionales.

6.1. Pruebas unitarias

Esta batería de pruebas se encuentra integrada en la aplicación, en unas clases específicas para realizar las pruebas.

Este conjunto de pruebas se enfoca en verificar que las entidades se construyen, modifican y eliminan correctamente y que las asociaciones entre las clases son correctas.

Para probar estos tests unitarios hay que ir desde la consola hasta la raíz del proyecto y ejecutar el siguiente comando:

\$ phpunit -c app/

- -c indica que se lea el archivo de configuración.

Ejecución de las pruebas:

```
jj@jj-ubuntu-1510:~/projects/Palantir$ phpunit -c app/
PHPUnit 5.6.2 by Sebastian Bergmann and contributors.

..... 65 / 78 ( 83%)
.....
Time: 3.55 seconds, Memory: 51.75MB

OK (78 tests, 124 assertions)
jj@jj-ubuntu-1510:~/projects/Palantir$
```

Dibujo 61: Captura de las pruebas unitarias y funcionales

En su conjunto, las pruebas unitarias y las pruebas funcionales, cuentan con 78 tests y 124 assertions.

Nuestros casos de prueba se encuentran en cada Bundle, en la carpeta Test y son ejecutados desde el controlador. Por buenas prácticas y para que se auto carguen los tests, se generan directorios similares la carpeta Test. Por ejemplo si tenemos una entidad en GameSessionBundle/Entity/Language.php habrá que crear: GameSessionBundle/Test/Entity/LanguageTest.php y así Symfony ejecutará automáticamente el test cuando se le indique.

Un ejemplo sencillo de una de las pruebas unitarias que tenemos es el siguiente:

```
public function testCreateLanguage()
{
    $language = new Language();
    $language->setName('lt');
    $language->setDisplayName('Language test');
    $this->assertEquals('lt', $language->getName());
    $this->assertEquals('Language test', $language->getDisplayName());
}
```

6.2. Pruebas funcionales

Además de las pruebas unitarias se han añadido pruebas funcionales, enfocadas principalmente a los roles de usuarios, para verificar quién puede y quién no puede acceder a las diferentes partes de la aplicación.

La batería de pruebas, mencionada anteriormente en la sección anterior de las pruebas unitarias, ejecuta de forma automática tanto las pruebas funcionales como las pruebas unitarias.

Ejemplo de prueba funcional:

```
/**  
 * @dataProvider urlRoleUserProvider  
 */  
public function testRoleAnonymousIsSuccessfulWithRoleUser($url)  
{  
    $this->client = self::createClient();  
    $this->client->request('GET', $url);  
  
    $this->assertTrue(!$this->client->getResponse()->isSuccessful());  
    $this->assertTrue($this->client->getResponse()->isRedirection());  
}
```

En esta prueba funcional estamos comprobando que los usuarios anónimos no pueden acceder a las URL que le suministramos y que, como consecuencia de ello, se le redireccione.

6.3. Pruebas de validación

Para el desarrollo de esta prueba se ha optado por unas listas de comprobación para las diferentes partes de la aplicación y estas son: navegación en la plataforma web y navegación en la plataforma de juego.

Estas pruebas de validación las ha realizado únicamente el desarrollador principal en varias ocasiones para verificar su correcto funcionamiento. La tasa de error es del 5% debido a que durante las primeras pruebas ha ido saliendo algunos errores. Todos ellos ya han sido arreglados.

Las listas de comprobación siguen siempre un mismo orden y se especifica cuando actúa el usuario y cuando el sistema debe dar una respuesta. A continuación se pasará a describirlas:

6.3.1. Plataforma web

Esta prueba recoge las acciones de los usuarios en la plataforma web. Todos estos pasos han de realizarse de forma consecutiva y es necesario realizar los test en orden.

Conexión de usuario:

Paso	Acción
1	El usuario accede a la página principal.
2	El usuario accede a la página de registro.
3	El usuario se registra con datos incorrectos -> El sistema indica los datos que son incorrectos

	en la página de registro.
4	El usuario se registra con datos correctos -> El sistema registra al usuario y lo redirige a la página principal.
5	El usuario se desconecta desde la funcionalidad de salir.
6	El usuario accede a la página de acceso e introduce incorrectamente sus credenciales -> El sistema indica que las credenciales no son válidas en la página de acceso.
7	El usuario introduce correctamente sus credenciales -> El sistema redirige al usuario a la página principal.

Tabla 70: Prueba. Conexión de usuario

Modificación de los datos personales del usuario:

Paso	Acción
1	El usuario accede a la página de modificar datos personales y los modifica incorrectamente -> El sistema indica las modificaciones incorrectas en la página de modificación de datos personales.
2	El usuario modifica correctamente sus datos personales -> El sistema redirige al usuario a la página de modificación de datos personales con los cambios efectuados e indicándole que los cambios han sido efectuados.
3	El usuario accede a la página de modificar la contraseña y la modifica incorrectamente -> El sistema indica que la modificación es incorrecta en la página de modificación de contraseña.
4	El usuario modifica correctamente su contraseña -> El sistema redirige al usuario a la página de modificación de contraseña y le indica que el cambio ha sido efectuado.

Tabla 71: Prueba. Modificación de los datos de usuario

Gestión de ficha de personaje:

Paso	Acción
1	El usuario accede a la página de crear ficha de personaje y crea una ficha de personaje con datos incorrectos -> El sistema indica los datos que son incorrectos en la página de creación de ficha de personaje.
2	El usuario crea una ficha de personaje de un juego de rol con datos correctos -> El sistema redirige al usuario a su lista personal de fichas de personaje.
3	El usuario crea otra ficha de personaje del mismo juego de rol que el punto anterior -> El sistema redirige al usuario a su lista personal de fichas de personaje.
4	El usuario crea otra ficha de personaje de un juego de rol distinto -> El sistema redirige al usuario a su lista personal de fichas de personaje.
5	El usuario crea otra ficha de personaje del mismo juego de rol que el punto anterior -> El sistema redirige al usuario a su lista personal de fichas de personaje.
6	El usuario crea otra ficha de personaje del mismo juego de rol que el punto anterior -> El sistema redirige al usuario a su lista personal de fichas de personaje.
7	El usuario accede a la opción de editar ficha de una de las tres últimas fichas creadas y modifica incorrectamente la ficha -> El sistema indica las modificaciones incorrectas en la

	página de modificación de esa ficha de personaje.
8	El usuario editar correctamente la ficha -> El sistema indica que la ficha ha sido modificada correctamente en la página de editar esa ficha de personaje.
9	El usuario accede a la página de lista de fichas y elimina otra de las tres últimas fichas creada que no sea la editada -> El sistema indica que la ficha ha sido eliminada en la página de lista de fichas.

Tabla 72: Prueba. Gestión de ficha de personaje

6.3.2. Plataforma de juego

Esta prueba recoge las acciones de los usuarios en la plataforma de juego. Para poder realizar esta parte del test es necesario haber realizado la del apartado anterior de Plataforma web.

Gestión de sesiones de juego y conexión de usuarios:

Paso	Acción
1	El usuario accede a la página de crear sesión de juego y crea con datos incorrectos una -> El sistema indica los datos que son incorrectos en la página de crear sesión de juego.
2	El usuario crea una sesión de juego con datos correctos -> El sistema redirige al usuario a la sesión de juego.
3	Es necesario un segundo y tercer usuario en navegadores, sesiones privadas o dispositivos diferentes. Es necesario que estén registrados y hayan accedido al sistema. Es irrelevante el navegador y el dispositivo. A continuación se los nombrará como usuario2 y usuario3.
4	El usuario2 accede a la página de listado de sesiones de juego y accede a la funcionalidad de unirse de la sesión de juego creada por el usuario -> El sistema redirige al usuario2 a la página de acceso de la sesión de juego.
5	El usuario2 introduce la contraseña incorrectamente -> El sistema indica al usuario2 que la contraseña ha sido introducida incorrectamente en la página de acceso de la sesión de juego.
6	El usuario2 introduce correctamente la contraseña -> El sistema redirige al usuario a la sesión de juego y avisa, mediante la herramienta de chat, al usuario de la conexión del usuario2.
7	El usuario3 realiza las mismas acciones que el usuario2 pero introduciendo correctamente la contraseña la primera vez -> El sistema redirige al usuario a la sesión de juego y avisa mediante la herramienta de chat al usuario y usuario2 de la conexión del usuario3.
8	El usuario accede a la funcionalidad de salir de la sesión de juego -> El sistema redirige al usuario a la página de listado de las sesiones de juego y avisa, mediante la herramienta de chat, al usuario2 y al usuario3 de que el usuario ha abandonado la sesión de juego.
9	El usuario utiliza la funcionalidad de eliminar la sesión de juego que creó -> El sistema le indica, en el listado de las sesiones de juego, que no puede ser eliminada habiendo usuarios dentro.
10	El usuario accede a la sesión de juego -> El sistema avisa, mediante la herramienta de chat, al usuario2 y al usuario3 de la conexión del usuario.
11	El usuario accede a la opción de editar los datos básicos de la sesión de juego y los cambia incorrectamente -> El sistema notifica al usuario de los datos incorrectos en la ventana de modificación.
12	El usuario introduce correctamente los cambios de los datos básicos de la sesión de juego,

	entre esos cambios debe estar la contraseña -> El sistema notifica al usuario de la realización de los cambios en la ventana de modificación.
13	El usuario utiliza la herramienta de gestionar usuarios y expulsa al usuario2 -> El sistema expulsa al usuario2 y le redirige a la página de información de desconexión de la sesión de juego. Además el sistema notifica, mediante la herramienta de chat, al usuario y usuario3 de la desconexión del usuario2.
14	El usuario2 accede a página de acceso de la sesión de juego e introduce la primera contraseña. Además se deben mostrar los datos modificados anteriormente por el usuario -> El sistema le indica que la contraseña no es correcta en la página de acceso a esa sesión de juego.
15	El usuario2 introduce correctamente la contraseña -> El sistema lo redirige dentro de la sesión de juego y notifica, mediante la herramienta de chat, al usuario y usuario3 de la conexión del usuario2.
16	El usuario3 utiliza la funcionalidad de salir de la sesión de juego -> El sistema redirige al usuario a la página del listado de sesiones de juego. Además notifica, mediante la herramienta de chat, al usuario y usuario2 de la desconexión del usuario3.
17	El usuario utiliza la funcionalidad de eliminar la sesión de juego -> El usuario y el usuario3 son redirigidos a la página de sesión de juego eliminada.
18	El usuario2 crea dos sesiones de juego, siguiendo los pasos descritos anteriormente, pero desde dos pestañas diferentes. Posteriormente elimina una sesión de juego desde dentro de la sesión de juego y la otra desde la página de listados de sesiones de juego -> El sistema elimina ambas sesiones de juego.

Tabla 73: Prueba. Gestión de sesiones de juego y conexión de usuarios

Para probar las siguientes funcionalidades el usuario debe haber creado una sesión de juego y que él, el usuario2 y el usuario3 se encuentran en ella. Además la herramienta de chat es usada en el resto de la plataforma de juego por lo que estará en constante testeo.

Funcionalidad de chat:

Paso	Acción
1	El usuario escribe en el chat un texto cualquiera -> El sistema envía a todos los usuarios el texto escrito.
2	El usuario2 envía al usuario un susurro de forma incorrecta -> El sistema informa al usuario2, a través de la herramienta de chat, del error del formato.
3	El usuario2 envía al usuario un susurro de forma correcta -> El sistema envía al usuario el susurro del usuario2 y el usuario2 también lo visualiza.

Tabla 74: Prueba. Funcionalidad de chat

Funcionalidad de mapa:

Paso	Acción
1	El usuario accede a la funcionalidad del mapa y crea cinco fichas con diferentes colores y nombres -> El sistema las crea y los usuarios las visualizan.
2	El usuario las mueve por el mapa -> Todos los usuarios ven las posiciones modificadas.
3	El usuario elimina dos fichas -> Todos los usuarios ven la eliminación de las dos fichas.
4	El usuario mueve las fichas restantes y crea una nueva -> Todos los usuarios ven la nueva ficha

	creada y la posición modificada de las otras tres.
5	El usuario elimina todas las fichas -> Todos los usuarios ven la eliminación de las fichas.

Tabla 75: Prueba. Funcionalidad de mapa

Funcionalidad de lanzamiento de dado:

Paso	Acción
1	El usuario accede a la funcionalidad, selecciona tres dados cualesquiera y les asigna una cantidad aleatoria entre 1 y 10. Posteriormente elimina uno de ellos y añade otro con una cantidad aleatoria entre 1 y 10. Finalmente realiza el lanzamiento -> El sistema muestra a todos los usuarios, a través de la herramienta de chat, el resultado del lanzamiento realizado.
2	El usuario2 realiza el mismo proceso con un solo dado, eliminándolo y volviéndolo a seleccionar -> El sistema muestra a todos los usuarios, a través de la herramienta de chat, el resultado del lanzamiento realizado.

Tabla 76: Prueba. Funcionalidad de lanzamiento de dado

Funcionalidad de importación de fichas de personaje:

Paso	Acción
1	El usuario importa sus dos fichas de personaje del juego de rol -> El sistema le provee de la funcionalidad que puede ejecutar teniendo el rol de director de juego. El sistema muestra a los demás usuarios la ficha. Además se muestra en la herramienta de chat la importación realizada.
2	El usuario2 importa sus dos fichas de personaje del juego de rol -> El sistema le provee de la funcionalidad que puede ejecutar teniendo el rol de usuario. El sistema muestra a los demás usuarios la ficha y provee al usuario la funcionalidad de director de juego. Además se muestra en la herramienta de chat la importación realizada.
3	El usuario3 importa sus dos fichas de personaje del juego de rol -> El sistema le provee de la funcionalidad que puede ejecutar teniendo el rol de usuario. El sistema muestra a los demás usuarios la ficha y provee al usuario la funcionalidad de director de juego. Además se muestra en la herramienta de chat la importación realizada.
4	Cada usuario elimina una de sus fichas y la vuelve a importar -> El sistema notifica, mediante la herramienta de chat, cada eliminación e importación de ficha de personaje.
5	El usuario y el usuario2 salen de la sesión de juego y vuelven a entrar -> El sistema les provee la sesión de juego tal y como la dejaron.

Tabla 77: Prueba. Funcionalidad de importación de fichas de personaje

Funcionalidad de ejecución de funcionalidad de fichas de personaje:

Paso	Acción
1	El usuario2 ejecuta una funcionalidad individual de una de sus fichas -> El sistema muestra el resultado a todos los usuarios por la herramienta de chat.
2	El usuario3 ejecuta una funcionalidad colectiva de una de sus fichas seleccionando como objetivo una del usuario -> El sistema muestra el resultado a todos los usuarios por la herramienta de chat.
3	El usuario va a realizar una funcionalidad individual y colectiva con una de sus fichas pero cancela el proceso.

4	El usuario realiza una funcionalidad individual y colectiva en una de las fichas del usuario2, seleccionando una de las fichas del usuario como objetivo -> El sistema muestra el resultado a todos los usuarios por la herramienta de chat.
---	--

Tabla 78: Prueba. Funcionalidad de ejecución de funcionalidad de fichas de personaje

6.4. Pruebas de sistema

Para el desarrollo de estas pruebas se ha dejado el sistema en manos de usuarios expertos en juegos de rol de mesa y con conocimientos básicos de aplicaciones web para que se puedan desenvolver con soltura al usar la aplicación sin ningún tipo de guía adicional.

Para realizar la prueba se les ha proporcionado de forma individual la aplicación desde la página principal, con un tiempo estimado de 20 minutos para probarla. Además de darle soporte por si tuviera alguna duda puntual.

La única guía que se les ha proporcionado, son las diferentes opciones que tienen que intentar realizar:

- Crear un usuario.
- Crear dos fichas de personaje de un juego de rol.
- Editar una ficha de personaje.
- Crear una sesión de juego del juego de rol de las fichas de personaje creadas.
- Importar las fichas de personajes creadas anteriormente.

En este punto se le añaden dos nuevos usuarios importándoles dos fichas de personaje de cada uno. A continuación se le comunica al usuario que puede mirar las diferentes pantallas de los dos nuevos usuarios para tener una visión más completa del sistema y de las interacciones. El usuario continúa probando:

- Realizar tiradas de dados.
- Ejecutar funcionalidad individual y colectiva con las fichas.
- Utilizar el mapa de la sesión de juego.
- Eliminar alguna ficha.
- Modificar la sesión de juego.
- Expulsar un jugador.
- Salir de la sesión de juego y del sistema y volver a acceder al sistema y a la sesión de juego.

Las conclusiones de mejora que se han sacado son fundamentalmente para la plataforma de juego y son las siguientes:

- La aplicación es confusa, el usuario no siempre tiene claro qué opciones tiene y como ejecutarlas.
- Demandan más funcionalidad en la plataforma de juego.

El primer punto puede solucionarse con un buen diseño gráfico de la aplicación y algunos tutoriales para su uso. Es normal en este tipo de aplicación encontrarte con funcionalidades que no sabes como usarlas o simplemente buscar una funcionalidad concreta y no encontrarla. Todo lo anterior es debido a la gran cantidad de opciones que puede tener un juego de rol de mesa.

En el segundo punto los usuarios demandan más funcionalidad para realizar sus partidas. Fundamentalmente le han dado importancia a tres puntos y por el orden que se muestra a continuación:

1. Gestionar sistema multimedia donde los directores de juego puedan subir audios, imágenes y vídeos y se reproduzcan o muestren cuando ellos lo indiquen.
2. Más opciones para el mapa.
3. Deshacer y rehacer las acciones en la partida.

Dejando a un lado las conclusiones de mejora, la aplicación les ha gustado y les ha resultado útil para realizar partidas de rol de mesa a distancia.

La tasa de error, entiendo por tasa de error como el fallo en la aplicación o que el usuario no ha podido continuar con el guión, ha sido del 33% ya que aunque no ha habido ningún error del sistema, los usuarios se han quedado un tercio de las veces sin saber dónde se encontraba una opción o qué hacer a continuación.

Estas pruebas del sistema la han realiza tres personas, obteniendo resultados similares.

Parte III

Epílogo

7. Conclusiones

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

7.1. Objetivos

Los objetivos principales que nos habíamos propuesto de **[OBJ-1] Crear un sistema de plataforma de juego** y **[OBJ-2] Crear un sistema de plataforma web**, los hemos completado con éxito después de 4 iteraciones de en la aplicación. Podemos decir que los hemos completado con éxito porque un grupo de usuarios pueden jugar una partida de rol de mesa a distancia a través de la aplicación. Aún así hay una gran cantidad de mejoras posibles pero la base del sistema es lo suficientemente grande como para abarcar las necesidades básicas de una partida de rol de mesa.

7.2. Lecciones aprendidas

La realización de este Trabajo Fin de Grado me ha permitido aprender una gran cantidad de tecnologías, buenas prácticas de desarrollo y, lo más importante, llevar a cabo un proyecto real cubriendo todas las etapas. A continuación paso a enumerar las tecnologías que he aprendido a usar de cero:

- Symfony.
- YAML.
- Twig.
- Bootstrap.
- Jquery UI.
- Web Application Messaging Protocol (WAMP).

Además he aprendido a trabajar más rápido y cometiendo menos errores con las siguientes tecnologías:

- PHP.
- JavaScript.
- jQuery.
- CSS.
- GIT.
- Apache2.

Tampoco puedo olvidar el aprendizaje de la metodología empleada Agile Unified Process (AUP). Aunque no he conseguido que los ciclos sean cortos, he aprendido su funcionamiento para aplicarlo a otros proyectos de forma más realista. Además creo que mis estimaciones serán más precisas debido a la experiencia adquirida.

He aprendido el concepto de “sprint zero” o “ciclo cero” que básicamente viene a definir un ciclo del proyecto que está únicamente dedicado a montar toda la arquitectura del sistema y que esta sea funcional para poder empezar a trabajar en ella.

Finalmente decir que he aprendido a valor el esfuerzo de las personas que desarrollan software libre y a las que resuelven dudas de forma desinteresada, sin ellos este proyecto no habría llegado a cumplir todos sus objetivos.

7.3. Trabajo futuro

Como se ha comentado a lo largo de todo el desarrollo, ha habido muchas herramientas y funcionalidades que no se han añadido por lo que contarían como elementos a desarrollar para el futuro. A continuación pasamos a numerarlas junto con otras posibles mejoras:

- Herramienta de turnos e iniciativa.
- Más opciones para las fichas de personaje colectivas, permitiendo más funcionalidad a los juegos de rol.
- Mejora en toda la interfaz gráfica, tanto en la plataforma de juego como en la plataforma web.
- Plantilla de enemigos para fichas de personaje, permitiendo importar varias veces la misma.
- Herramienta de deshacer y rehacer.
- Sistema de inventario en las fichas de personaje.
- Añadir más juegos de rol.
- Herramienta multimedia. Para permitir la ejecución de vídeos, audio y la visualización imágenes.
- Importar y exportar fichas de personaje.
- Generadores aleatorios.
- Herramienta de organización de partidas. Donde el director de juego pueda diseñar la partida con varios mapas, personajes y eventos con antelación.
- Sistema de glosario.
- Sistema de foros para organizar partidas.

Además de todo lo anterior hay muchas más funcionalidad que podrían desarrollarse como funcionalidad futura pero las aquí mencionadas son las que creemos más necesarias. Toda funcionalidad puede llegar a ser poca ya que en un juego de rol de mesa todo límite está en nuestra imaginación.

Anexos

Anexo I: Manual de usuario

En este anexo, Manual de usuario, se detallarán las instrucciones de uso del sistema para un usuario. Las capturas de pantalla que se incluyen están en Inglés, uno de los idiomas presentes en el sistema.

1. Requisitos previos

Como ya se comentó anteriormente, los requisitos para poder hacer uso del sistema son: tener un sistema capaz de soportar un navegador como Google Chrome 54 o Firefox 47, por lo que podría ser un dispositivo móvil, tablet, sobremesa y portátil. No se asegura que en las futuras versiones de Google Chrome y Firefox funcione el sistema, sobre todo si realizan cambios en los estándares de desarrollo.

2. Utilización

Aunque la mayor parte de la funcionalidad se explica por si sola como es el caso del registro y el login, se detallarán aquellos aspectos que puedan ser más confusos. Para ello se va a incluir una explicación con capturas de pantalla de estas funcionalidades. Estas funcionalidades podrán ser ejecutables por los diferentes roles de la aplicación: el usuario, el director de juego y el administrador.

2.1. Usuario no registrado

Representa al usuario no registrado en el sistema (usuario anónimo).

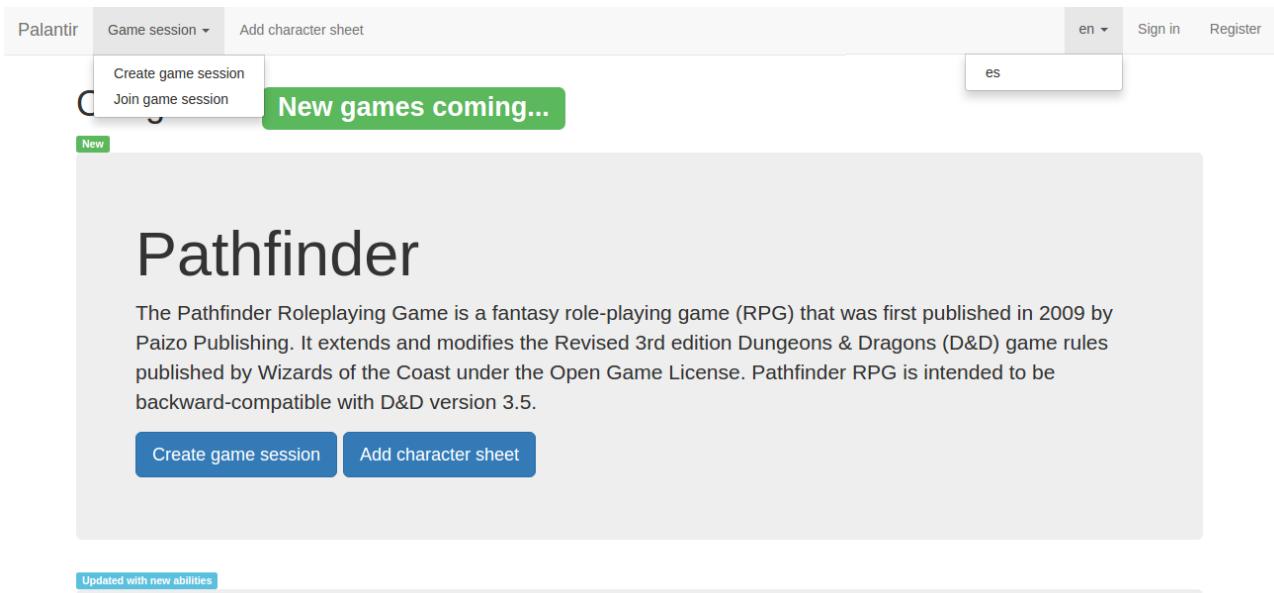
2.1.1. Página principal

El único acceso a la aplicación que tiene el usuario no registrado es la página principal. En ella se pueden ver las diferentes opciones principales que tiene el sistema:

- Create game session: Crear una sesión de juego.
- Join game session: Unirse a una sesión de juego.
- Add character sheet. Crear/añadir una ficha de personaje.
- Sign in: Acceder al sistema.
- Register: Registrarse.

Además de lo anterior hay un elemento común en prácticamente todas las ventanas que es el cambio de idioma. Actualmente los idiomas soportados son el inglés “en” y el español “es”.

Aunque el usuario no registrado pueda ver las opciones, si accede a alguna de ellas, salvo cambiar el idioma, registrarse o acceder, será redirigido a la página de acceso al sistema para que introduzca sus credenciales.



Dibujo 62: Captura "página principal de usuario no registrado"

2.2. Usuario

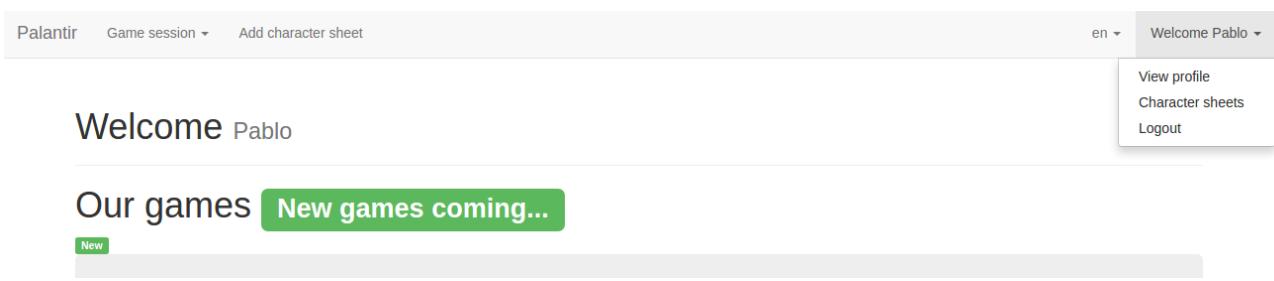
Representa al usuario registrado en el sistema.

2.2.1. Página principal

Además de la funcionalidad ya comentada con el usuario no registrado, el usuario registrado tiene las siguientes funcionalidades:

- View profile: ver su perfil.
- Character sheets: gestionar sus fichas de personaje.
- Logout: desconectarse.

Por su puesto, y al contrario que el usuario no registrado, puede acceder a toda esta funcionalidad.



Dibujo 63: Captura "página principal de usuario"

2.2.2. Ver perfil

El usuario tiene acceso a sus datos del perfil, puede editarlos y modificar su contraseña.

Pablo USER

Email: correo_de_prueba@test.com

Name: Pablo

Surname: De la Torre

[Edit profile](#) | [Change password](#)

Dibujo 64: Captura "ver perfil"

2.2.3. Gestión de fichas de personaje

Desde aquí podemos ver, editar y eliminar las fichas de personaje que hemos creado y tenemos en el sistema. Para crear una nueva recordemos que hay que acceder desde el menú principal.

Character sheets

Character sheet	Rol game	Edit	Remove
Francisco de Quevedo	Pathfinder	Edit	Remove
Vladimir	Vampire The Masquerade	Edit	Remove
Vladimira	Vampire The Masquerade	Edit	Remove

Dibujo 65: Captura "gestionar fichas de personaje"

2.2.4. Crear ficha de personaje

Para crear una ficha de personaje hay que seleccionar un juego de rol y posteriormente la plantilla de ficha de personaje de este juego de rol. Una vez seleccionado, y si continuamos, podemos llenar la ficha de personaje.

Add character sheet

Rol game	<input type="text" value="Pathfinder"/>
Character sheet template	<input type="text" value="Choose the rol game"/>

This value should not be blank.

[Continue](#)

Dibujo 66: Captura "crear ficha de personaje 1/2"

Palantir Game session ▾ Add character sheet en ▾ Welcome Pablo ▾

Add character sheet

Character sheet	<input type="text"/>
Alignment	<input type="text"/>
Attributes	
Strength actual score	<input type="text"/>
Strength actual modifier	<input type="text"/>
Strength temporary score	<input type="text"/>
Strength temporary modifier	<input type="text"/>
Dexterity actual score	<input type="text"/>

Dibujo 67: Captura "crear ficha de personaje 2/2"

2.2.5. Crear sesión de juego

Un usuario puede crear una sesión de juego de un juego de rol determinado. El campo “lenguaje”, y por tanto el idioma, es meramente informativo para los usuarios. El resto de campos podrán ser modificados después de ser creada la sesión de juego.

Cuando un usuario acceda a una sesión de juego, la interfaz será del idioma del usuario.

En el momento de crear la sesión de juego, el usuario adquirirá el rol de director de juego para esa sesión de juego y se le redirigirá al interior de esta.

El usuario podrá navegar desde otras pestañas por la aplicación pero solo podrá mantener una pestaña para cada sesión de juego, da igual el rol que desempeñe en el sistema.

Palantir Game session ▾ Add character sheet Administer en ▾ Welcome Jaime

Create game session

Name	<input type="text" value="Epic Game"/>
Password	<input type="text"/>
Rol game	<input type="text" value="Choose the rol game"/>
Language	<input type="text" value="Choose the session language"/>
Comments	<input type="text"/>
<input type="button" value="Create"/>	

Dibujo 68: Captura "crear sesión de juego"

2.2.6. Unirse a sesión de juego

Cuando accedemos a la funcionalidad de unirnos a una sesión de juego vemos una interfaz similar es ésta donde los jugadores usuarios pueden ver un resumen de todas las que están presentes en el sistema.

Si un usuario es director de juego de una de ellas, podrá eliminarla. Para editarla debe hacerlo desde el interior y para eliminarla no puede haber ningún usuario conectado a ella. Si esto fuera así, debe acceder y expulsarlo desde dentro de la sesión de juego.

Si somos los directores de juego accedemos directamente a la sesión de juego sin tener que introducir la contraseña.

Name	Rol game	Gamemaster	Language	Join
Epic Game	Pathfinder	Jaime	Español	<button>Join</button> <button>Remove</button>
Epic Game	Vampire The Masquerade	12345	Español	<button>Join</button>
Epic Game 2	Pathfinder	Jaime	English	<button>Join</button> <button>Remove</button>
Vampire game	Vampire The Masquerade	Jaime	Español	<button>Join</button> <button>Remove</button>

Dibujo 69: Captura "acceder a sesiones de juego"

Cuando accedemos a una de las sesiones de juego de las que no somos propietarios (directores de juego) vemos el resumen más detallado de la sesión de juego y si introducimos correctamente la contraseña se nos redirigirá al interior de la sesión de juego.

Epic Game

Gamemaster: Jaime

Creation date: 2016-07-28 18:27:16.000000

Rol game: Pathfinder

Language: Español

Comments: En esta partida se jugará el Trono de las Mareas primer nivel, con personajes de nivel 4.

Login game session

Password

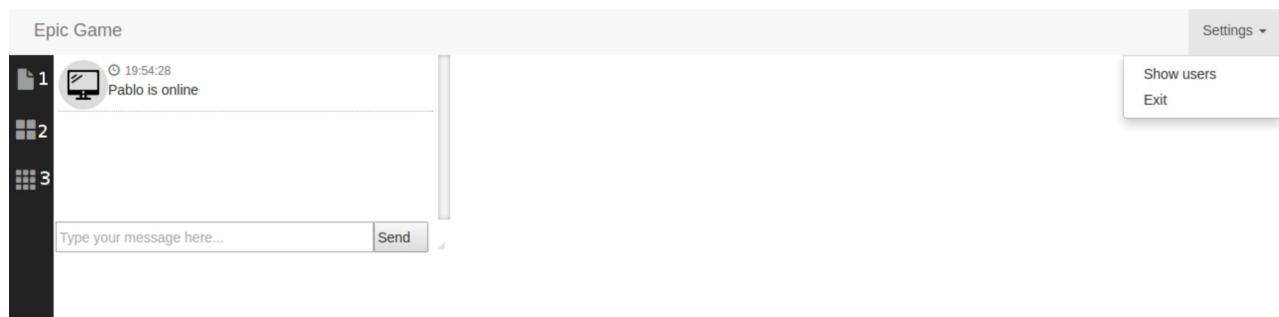
Login

Dibujo 70: Captura "acceder a sesión de juego"

2.1.7. Sesión de juego

Cuando un usuario accede a una sesión de juego que no ha sido modificada, es decir, no hay ninguna ficha de personaje importada, lo que ve es una interfaz con un chat y varias herramientas:

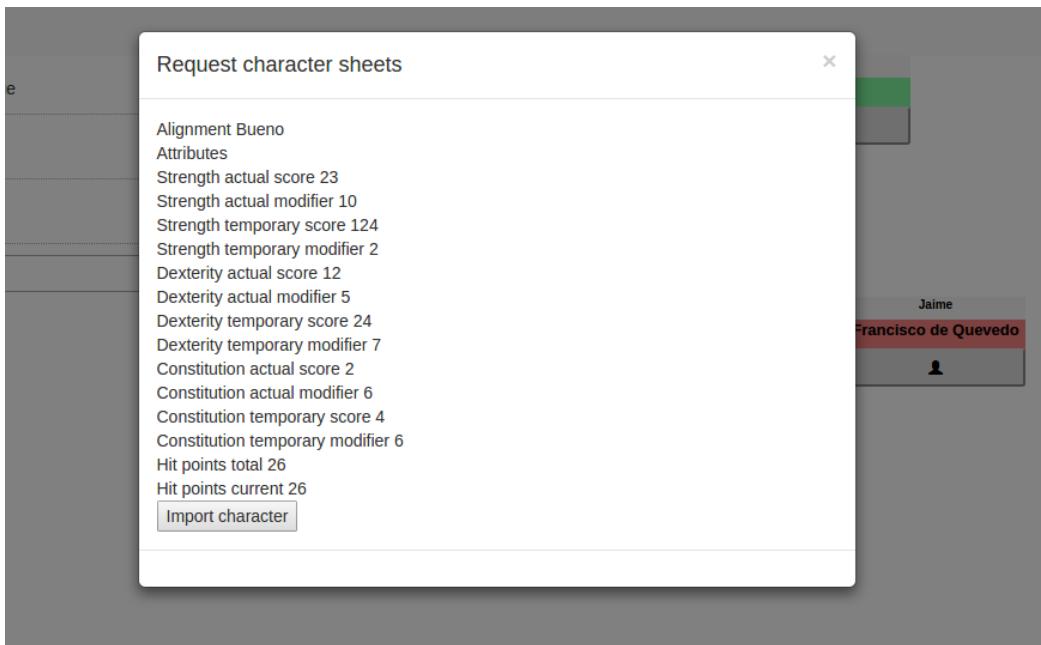
- 1. Le permite al usuario importar fichas de personaje.
- 2. Le permite hacer tiradas de dados.
- 3. Le permite acceder al mapa de la sesión de juego.
- “Show users” le permite ver los usuarios que están conectados en la sesión de juego.
- “Exit” le permite abandonar la sesión de juego. También puede abandonarla cerrando la pestaña o el navegador.



Dibujo 71: Captura "sesión de juego del usuario"

2.1.8. Importar ficha de personaje en la sesión de juego

Al acceder a la funcionalidad de importar ficha de personaje en la sesión de juego, se nos muestran todas las fichas de personaje que podemos importar. Éstas son las que pertenecen al juego de rol de la sesión de juego y que no están importadas actualmente.



Dibujo 72: Captura "importar ficha de personaje en la sesión de juego"

2.1.9. Lanzamiento de dados en la sesión de juego

Todo usuario puede realizar un lanzamiento de dados en la sesión de juego. El resultado será mostrado en el chat para todos los usuarios.

Se pueden seleccionar diferentes tipos de dados y distintas cantidades a la hora de realizar un lanzamiento.



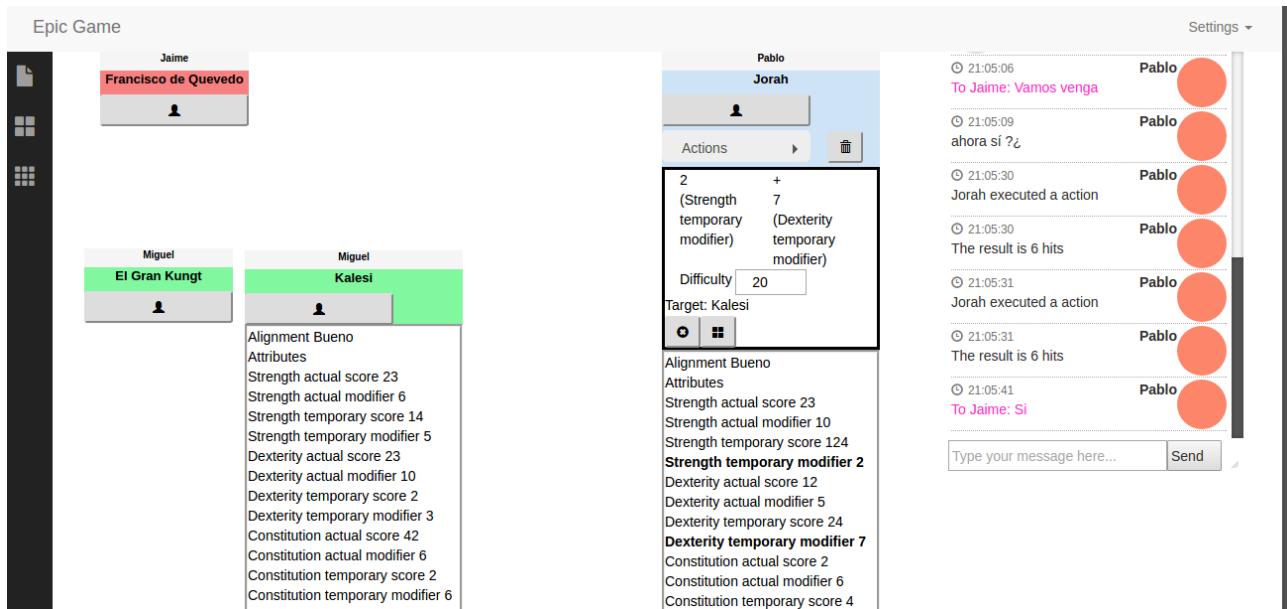
Dibujo 73: Captura "lanzamiento de dados en la sesión de juego"

2.1.10. Ejecutar funcionalidad de las fichas de personaje

Cuando importamos una ficha en la sesión de juego ya podemos ejecutar su funcionalidad. Como usuario solo podemos ejecutar la funcionalidad de nuestros personajes pero sí ver las estadísticas de los demás.

Como sabemos, hay dos tipos de funcionalidad, la colectiva y la individual. Para ejecutar una funcionalidad individual simplemente hay que seleccionar aquél atributo, habilidad o elemento que esté sobresaltado en negro. Posteriormente se nos mostrará un panel con un resumen, como el que vemos en la captura, y podremos finalmente ejecutar la funcionalidad o cancelarla.

Para ejecutar la funcionalidad colectiva hay que seleccionar primero la acción que se desea realizar. Dependiendo de la acción habrá unas opciones u otras pero la más frecuente es seleccionar una ficha objetivo. Posteriormente, y al igual que en la funcionalidad individual, se nos mostrará un resumen de la funcionalidad y podremos decidir si ejecutarla o cancelarla.



Dibujo 74: Captura "ejecutar funcionalidad de la ficha de personaje"

2.2. Director de juego

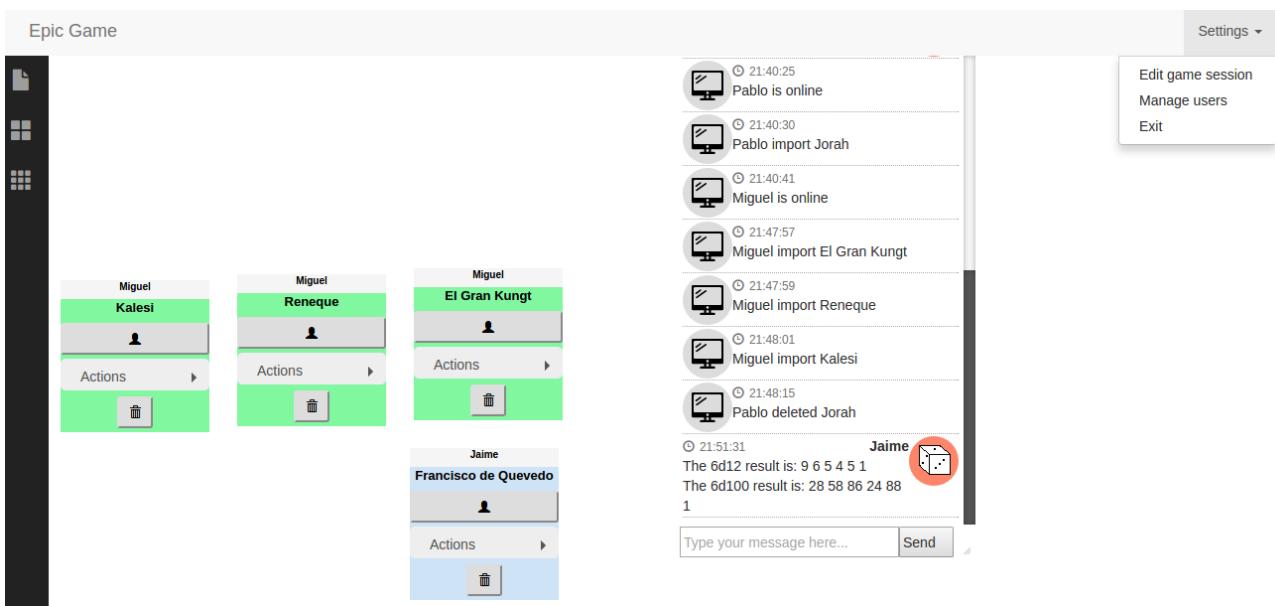
El director de juego es el usuario que ha creado la sesión de juego y se encuentra en ella dirigiéndola.

2.2.1. Sesión de juego

Para un director de juego la sesión de juego es diferente ya que posee más herramientas que le permiten moderar y dirigir la partida.

En la captura podemos ver las opciones de:

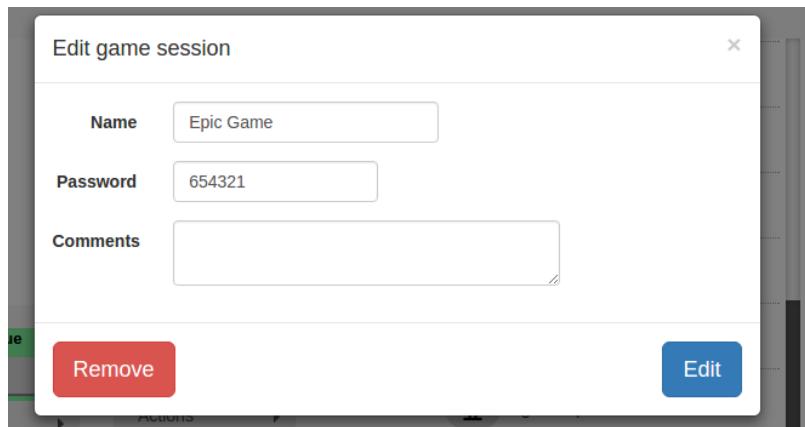
- “Edit game session” para editar la sesión de juego.
- “Manage users” para expulsar a los jugadores.
- “Exit” para salir de la sesión de juego.



Dibujo 75: Captura "sesión de juego del director de juego"

2.2.2. Editar la sesión de juego

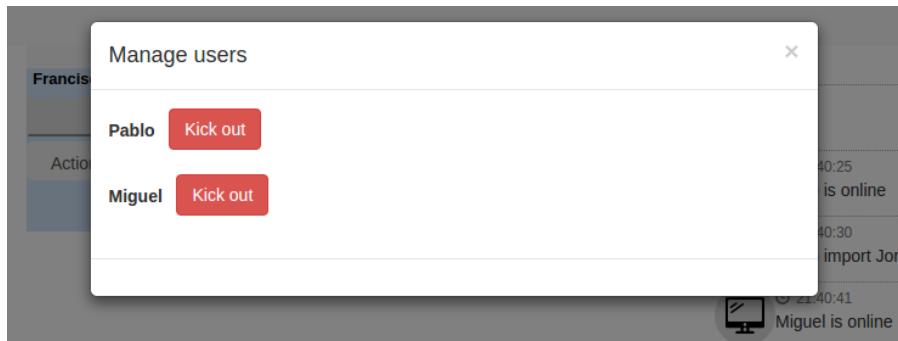
En la funcionalidad de editar la sesión de juego podemos modificar el nombre, la contraseña y los comentarios de la sesión de juego. Además tenemos la opción de eliminarla. Si eliminamos la sesión de juego, todos los usuarios serán expulsados avisándoles de este suceso, incluido el director de juego.



Dibujo 76: Captura "editar la sesión de juego"

2.2.3. Gestionar usuarios de la sesión de juego

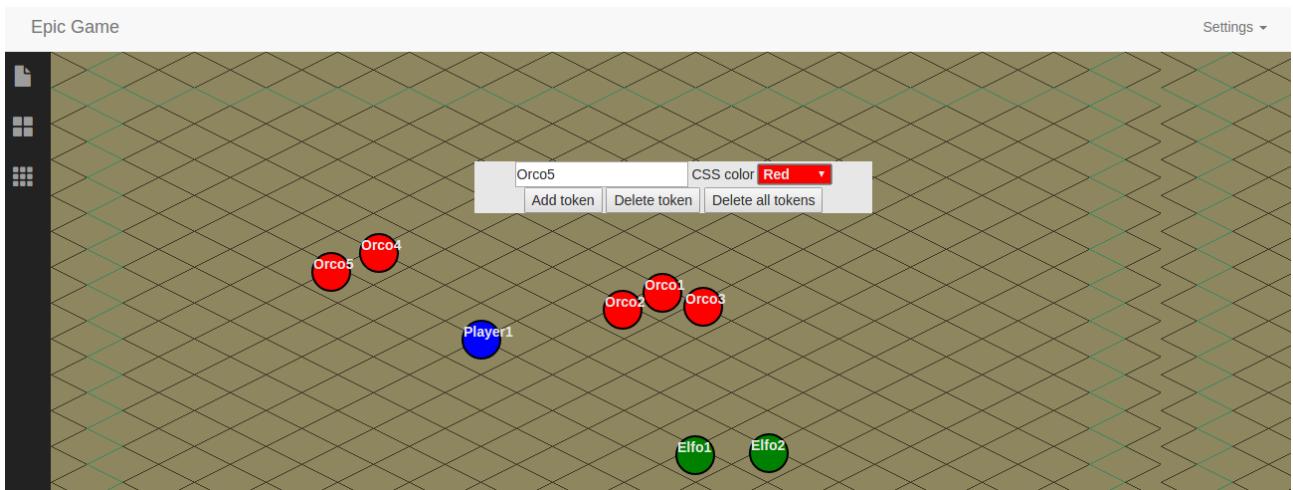
La opción de gestionar usuarios nos permite expulsarlos de la sesión de juego aunque estos podrán volver a unirse si conocen la contraseña por ello se recomienda cambiarla antes de expulsar a un jugador.



Dibujo 77: Captura "gestionar usuarios de la sesión de juego"

2.2.4. Mapa de la sesión de juego

Los usuarios y el director de juego visualizan por igual el mapa de la sesión de juego. La única diferencia es que el director de juego posee las herramientas para añadir, mover y eliminar los elementos que se encuentran en el mapa.



Dibujo 78: Captura "mapa de la sesión de juego"

2.3. Administrador

El Administrador es el usuario que posee permisos para la gestión del sistema.

Actualmente la funcionalidad de la que dispone le permite generar un entorno de prueba y eliminar el existente. Además de reiniciar las conexiones de todos los usuarios con las sesiones de juego y la gestión de las plantillas de las fichas de personaje.

Administration menu

The screenshot shows the administration menu with three main items:

- UserBundle**: Labeled "On develop". Includes an "Administer" button.
- GameSessionBundle**: Labeled "On develop". Includes an "Administer" button.
- GamingPlatformBundle**: Labeled "On develop". Includes an "Administer" button.

Dibujo 79: Captura "administración 1/3"

Administration GameSessionBundle

The screenshot shows the GameSessionBundle administration page with one main section:

- User game session connection**: Includes a "Remove all user game session connection" button.

Dibujo 80: Captura "administración 2/3"

Administration GameSessionBundle

The screenshot shows the GameSessionBundle administration page with four main sections:

- Default data**: Includes "Languages, rol games and character sheet templates.", "Add all default data" (green), and "Delete all default data" (red) buttons.
- Character Sheet templates**: Includes "Manage character sheet templates" button.
- Game sessions**: Includes "Manage game sessions" button.
- Character sheets**: Includes "Manage character sheets" button.

Dibujo 81: Captura "administración 3/3"

Anexo II: Manual de instalación y exportación

En este anexo se detallarán las instrucciones de instalación, explotación del sistema y un conjunto de buenas prácticas que facilitarán el mantenimiento del mismo.

1. Requisitos previos

Se requiere de un servidor de aceptable rendimiento (2 CPU y 4GB de RAM). Que esté en Ubuntu Server 15.10 o similar.

2. Procedimiento de instalación

A continuación se detallarán los pasos necesarios para poner en marcha la aplicación.

2.1. Configuración previa para crear/importar un proyecto Symfony2

Independientemente de si se pretende crear o importar un proyecto Symfony2, previamente hay que instalar y configurar las siguientes herramientas ya que son imprescindibles para el funcionamiento de Symfony2. A partir de ahora, cada vez que hablemos de “Symfony” nos estamos refiriendo a la versión Symfony 2.8.6.

Instalar Apache2

Es el servidor de HTTP.

```
$ sudo apt-get install apache2
```

Instalar PHP5 y PHPUnit

PHP5 es el lenguaje de programación del servidor, mientras que PHPUnit es un marco de pruebas para PHP.

```
$ sudo apt-get install php5 curl php5-curl libapache2-mod-php5 phpunit
```

Instalar Git

El propio código fuente de Symfony2 se gestiona mediante Git y las actualizaciones se realizan mediante comandos de Git. Así que aunque no utilices Git en tus aplicaciones, tendrás que instalarlo para poder utilizar Symfony2.

```
$ sudo apt-get install git
```

Instalar Composer

Composer es una herramienta para la gestión de las dependencias en PHP. Esta herramienta es indispensable para desarrollar en Symfony y se usará constantemente.

1. Abre la consola de comandos y ejecuta el siguiente comando en cualquier directorio:
\$ sudo curl -s https://getcomposer.org/installer | php
Si todo ha funcionado bien, verás un nuevo archivo composer.phar en ese mismo directorio.

2. Mueve el archivo composer.phar a algún directorio ejecutable del sistema, como por ejemplo:
\$ sudo mv composer.phar /usr/local/bin/composer

3. Abre una nueva consola de comandos y ejecuta el siguiente comando sin ninguna opción:
\$ composer
Si todo ha funcionado bien, deberías ver un listado con los comandos de Composer.

2.2. Importar un proyecto Symfony2 mediante Eclipse

En este caso nuestro repositorio se encuentra en Github. Para importar el proyecto mediante Eclipse, debemos seguir los siguientes pasos:

1. File -> Import -> Git -> Projects from Git -> Next -> Clone URI.
2. En el campo URI añadimos la URI del repositorio en cuestión ([https://github.com/*****](https://github.com/)) y se nos autocompletan los campos Host y Repository path. Rellenamos los campos User y Password y continuamos con Next.
3. Elegimos la branch que deseamos importar -> Next.
4. Indicamos el directorio donde queremos que se descargue -> Next.
5. Esperamos a que se descargue -> Next -> Finish.

En todo el proceso hay varias opciones que puedes elegir si te interesan, nosotros hemos ido a lo más básico.

En el momento que finalicemos Eclipse empezará a construir nuestro proyecto, tardará un rato y ya tendremos el proyecto importado.

Finalmente hay que ejecutar composer para configurar el repositorio mediante el siguiente comando:
\$ sudo composer install

2.3. Post configuración para crear/importar un proyecto Symfony2

Independientemente de si se ha creado o importado el proyecto, se deben realizar las siguientes post configuraciones:

Modificar .ini de PHP

Completar date.timezone = Europe/Madrid, recuerda descomentar la línea (quitar el punto y coma inicial).

\$ sudo nano /etc/php5/cli/php.ini

Modificar permisos

Es necesario que la aplicación y subprocessos puedan modificar el contenido de estas carpetas para el correcto funcionamiento de Symfony2.

\$ sudo chmod 777 -R app/cache/
\$ sudo chmod 777 -R app/logs/

Comprobar si la instalación es correcta

Hay que acceder a la carpeta raíz del proyecto y desde ahí ejecutar:
\$ php app/check.php

Configurar servidor local Apache2

Ahora hay que configurar el servidor apache2 para que se pueda iniciar el servidor en local.

Añadir el sitio a Apache2

```
$ cd /etc/apache2/sites-available/  
$ sudo nano my_project.local.conf  
  
<VirtualHost *:80>  
    ServerName domain.tld  
    ServerAlias www.domain.tld  
  
    DocumentRoot /var/www/project/web  
    <Directory /var/www/project/web>  
        AllowOverride None  
        Order Allow,Deny  
        Allow from All  
        #Require all granted #Cuando se encuentre en desarrollo.  
  
    <IfModule mod_rewrite.c>  
        Options -MultiViews  
        RewriteEngine On  
        RewriteCond %{REQUEST_FILENAME} !-f  
        RewriteRule ^(.*)$ app.php [QSA,L] #app_dev.php  
    </IfModule>  
    </Directory>  
  
    ErrorLog /var/log/apache2/project_error.log  
    CustomLog /var/log/apache2/project_access.log combined  
</VirtualHost>
```

Habilitar el sitio en Apache2

```
$ cd /etc/apache2/sites-enabled/  
$ sudo a2dissite *  
$ sudo a2ensite my_project.local.conf  
$ sudo service apache2 restart
```

Agregar nombre de ruta a la aplicación

Ahora si queremos acceder a nuestro proyecto desde un nombre diferente a “localhost” por ejemplo “my_project.local” tenemos que acceder al archivo hosts y añadirla, de la siguiente forma:
\$ sudo nano /etc/hosts

A continuación editamos de la siguiente manera, por ejemplo:
127.0.0.1 localhost my_project.local

Configurar base de datos e instalar PDO drivers

Esto solamente es necesario si tienes pensado usar bases de datos para tu proyecto.

Instalación:

```
$ sudo apt-get install mysql-server-5.6
```

La contraseña del root de mysql debe coincidir con la de app/config/parameters.yml.

Ahora instalamos el controlador de php5 con mysql:

```
$ sudo apt-get install php5-mysql
```

Importante no olvidar el comando para iniciar mysql:

```
$ sudo service mysql start
```

Configurando permanentemente la base de datos como UTF8:

Esta configuración se hace para evitar futuras desconfiguraciones en el desarrollo, para ello hay que acceder al fichero my.cnf:

```
$ sudo nano /etc/mysql/my.cnf
```

Una vez dentro, añadiremos lo siguiente:

```
[mysqld]
collation-server = utf8_general_ci
character-set-server = utf8
```

Instalar el visualizador de mysql:

El visualizador nos permitirá acceder a la base de datos y gestionarla desde un cómodo entorno gráfico. Para instalarlo usaremos los siguientes comandos:

```
$ sudo apt-get install mysql-client
$ sudo apt-get install mysql-workbench
```

Crear base de datos y tablas:

Ahora para crear la base de datos y las tablas debemos acceder a nuestro proyecto desde la consola.

Para generar la base de datos:

```
$ php app/console doctrine:database:create
```

Para generar las tablas:

```
$ php app/console doctrine:schema:update --force
```

3. Buenas prácticas y comandos útiles

Nombramiento de los controladores

Siguiendo con la convención de nombres de Symfony, todos los métodos que retornan o generan una vista, tienen que terminar en “Action” y empezar en minúsculas. Por ejemplo: indexAction será una acción que mostrará la pantalla inicial.

Actualizar composer

No es mala idea actualizar nuestro gestor de dependencias si tenemos intención de actualizar o añadir algún bundle nuevo. Para ello accederemos a la ruta donde tengamos el ejecutable binario y ejecutaremos el siguiente comando con la ruta recomendada para su instalación:

```
$ sudo /usr/local/bin/composer self-update
```

Actualizar bundles

Es tarea de nuestro gestor de dependencias el actualizar los bundles y es una buena práctica realizar esto de vez en cuando ya una gran parte de las actualizaciones consiste en reparar errores de seguridad. Para ello accedemos a nuestro proyecto:

```
$ cd proyecto
```

Y actualizamos:

```
$ sudo composer update
```

Asegurar utilización del bundle

Aunque hayamos instalado los bundles hay que añadirlos al fichero app/AppKernel.php de nuestro proyecto, qué es donde se cargan los diferentes bundles para su uso. Comprueba que los bundles de terceros están en la carpeta vendor del proyecto. Se verán de la siguiente forma:

```
$bundles = array(
    new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
    new Symfony\Bundle\SecurityBundle\SecurityBundle(),
    new Symfony\Bundle\TwigBundle\TwigBundle(),
    ...
    new Symfony\Bundle\MonologBundle\MonologBundle(),
);
```

El Perfilador

Symfony tiene una herramienta muy potente que permite ver una gran cantidad de información de las peticiones al servidor y sus resultados. Para asegurarse de tenerlo activado hay que acceder al archivo config_dev.yml de nuestra aplicación y añadir lo siguiente:

```
web_profiler:
    toolbar: true
```

Por su puesto se pueden seguir usando los métodos habituales para mostrar información:

- alert(), método de JavaScript que muestra el contenido en el navegador.
- alert(JSON.Stringfy()), muestra con detalle el objeto JSON.
- var_dump(), método de php que muestra el contenido en el navegador.
- die(), método de php que detiene la ejecución del código php. Suele ser usado justo después de mostrar contenido con var_dump().
- {{ dump() }}, método de YAML que muestra el contenido en el navegador.

Actualizar assets en el entorno de producción

Los assets son las hojas de estilo CSS, los archivos JavaScript y las imágenes que se utilizan en la parte *front* de las aplicaciones.

Los assets se referencian de forma dinámica, lo que conlleva unos gastos de recursos elevados, por lo que en el entorno de producción se recomienda crear una tabla que refiera todos los contenidos para no tener que generarlos de forma automática y así mejorar notablemente el rendimiento.

```
$ php app/console assetic:dump --env=prod --no-debug
```

Eliminar bundle de tercero

Aún siendo fácil instalar un bundle de tercero, eliminarlo del proyecto no es tan rápido y directo. Para eliminarlo hay que seguir los siguientes pasos:

- Eliminar su llamada en el fichero app/AppKernel.php. Ej:

```
$bundles = array(
    new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
    new Symfony\Bundle\SecurityBundle\SecurityBundle(),
    new Symfony\Bundle\TwigBundle\TwigBundle(),
    ...
    new Symfony\Bundle\MonologBundle\MonologBundle(),
);

```

- Eliminar cualquier dependencia que tenga el bundle en el archivo de configuración config.yml.

- Eliminar cualquier ruta a la que sea acceda de ese bundle.

- Eliminarlo del composer.json. Ej:

```
"require": {
    "php": ">=5.3.9",
    "symfony/symfony": "2.7.*",
    "doctrine/orm": "~2.2,>=2.2.3,<2.5",
    "doctrine/dbal": "<2.5",
    "doctrine/doctrine-bundle": "~1.4",
    "symfony/assetic-bundle": "~2.3",
    ... (add "," at last)
    "symfony/monolog-bundle": "~2.4"
},

```

- Eliminar todo el contenido del bundle de la carpeta vendor:

```
$ sudo rm -r /vendor/bundle
```

- Limpiar la caché para verificar que está todo correcto:

```
$ php app/console cache:clear --env=[prod|dev]
```

- Y finalmente actualizar el composer para restablecer las dependencias:

```
$ sudo composer update
```

Generar bundle

```
$ php app/console generate:bundle --namespace=Acme/HelloBundle format=yml
```

Limpiar caché

```
$ php app/console cache:clear --env=[prod|dev]
```

Mantener permisos en las carpetas app/cache y app/logs

Durante el desarrollo de Symfony surgirán momentos en los que se requiera limpiar la caché por lo que los permisos también se eliminarán. Como limpiar la caché suele ser un procedimiento bastante común, hay que evitar tener que estar cambiando los permisos de las carpetas y su contenido, para ello utilizaremos las listas de control de acceso (ACL) mediante el siguiente comando:

```
$ sudo setfacl -dR -m u:"$HTTPDUSER":rwx -m u:`whoami`:rwx app/cache app/logs
```

Ayuda en la depuración con el directorio app/logs

En la carpeta logs podemos encontrar dos archivos: logs y dev_logs, donde se puede ver, dependiendo del entorno que se esté usando, los posibles errores, excepciones, envío y recepción de datos, etcétera. Esto resulta muy útil si se quiere seguir con detalle alguna operación en concreto.

Actualizar esquema de la base de datos

```
$ php app/console doctrine:schema:update --force
```

Eliminar base de datos

```
$ php app/console doctrine:database:drop --force
```

Error could not find driver

Este error puede originarse al actualizar/installar cambios en mysql, apache2, php o alguno de los servicios y programas relacionados; la mayoría de las veces se soluciona con el reinicio de apache2:

```
$ sudo /etc/init.d/apache2 restart
```

Generar base de datos

```
$ php app/console doctrine:database:create
```

Generar entidad

```
$ php app/console doctrine:generate:entities AppBundle\Entity/User
```

FOSUserBundle

Hay veces que se requiere modificar el rol de un usuario ya existente para darle permisos de superusuario. Esta modificación debe ser manual para evitar que por algún error de seguridad alguien consiga ponerse los permisos. Para añadir el permiso basta con acceder a la tabla de usuarios “fos_user”, ir al usuario en cuestión y modificar el campo “roles”, normalmente, si no tiene ningún rol adicional, se presenta con la cadena “a:0: {}”, ahora hay que reemplazarla por “a:1:{i:0;s:16:”ROLE_SUPER_ADMIN”;}”, vuelve a iniciar sesión con ese usuario y ya tendrá el rol asociado.

Fallos en el fichero .gitignore

Algo que hay que tener en cuenta en es que el .gitignore puede no funcionar al crear el proyecto o al realizar algún cambio en la configuración, para hacer que nuestro repositorio local, github o algún servidor se reconfigure, hay que realizar los siguientes pasos:

```
$ git rm -r --cached .
$ git add .
```

```
$ git commit -m "fixing .gitignore"  
$ git push
```

Visualizar fallos de ejecución de MySQL

Debido a malas configuraciones tanto en Symfony2 como en el propio mysql, muchas veces mysql falla a la hora de lanzar el servicio pero se desconoce el error que lo ha causado. El siguiente comando indica los fallos a la hora de ejecutarlo:

```
$ sudo mysql -uowncloud -p
```

Ejecutar tests

Para ejecutar todos los tests, tanto unitarios como funcionales, del proyecto, hay que situarse en la raíz del proyecto y ejecutar:

```
$ phpunit -c app/
```

- -C indica que se lea el archivo de configuración.

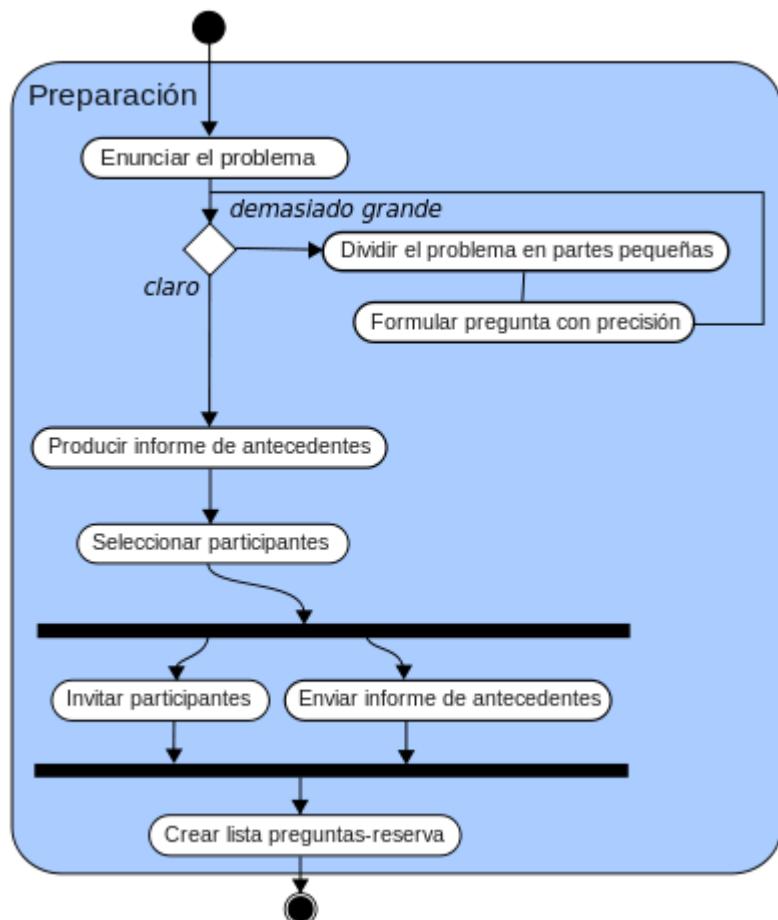
Lanzar servidor WebSocketBundle

```
$ php app/console gos:websocket:server
```

Si se está usando el servidor en el entorno de producción, a través del método {{ ws_client() }}, hay que asegurarse que assetic esté funcionando en el entorno de producción:

```
$ php app/console assetic:dump --env=prod --no-debug
```


Anexo III: Tormenta de ideas, fase de requisitos



23-11-2014

Emilio José Ciprés Nuñez (Participante)
Jaime Jesús Serrano Rodríguez (Organizador y participante)
Juan Carlos Trejo Fernández (Participante)

1. Introducción

Este *brainstorming* se ha realizado con motivo de detallar la funcionalidad necesaria que requiere una plataforma de juego para poder realizar partidas de rol a distancia a través de ella.

Para ello se ha reunido a tres personas para realizar esta tormenta de ideas. Con una duración de una hora y media.

1.1. Participantes

Los participantes de esta tormenta de ideas son:

- Emilio José Ciprés Núñez, participante y amigo del organizador. Experto en dirigir partidas y participar en ellas.
- Jaime Jesús Serrano Rodríguez, organizador y participante. Desarrollador del proyecto actual.
- Juan Carlos Trejo Fernández, participante y amigo del organizador. Experto en dirigir partidas y participar en ellas.

1.2. Problema a tratar

Creación de una plataforma de juego de rol, entendiendo por plataforma como un sistema *online* vía web donde los usuarios acceden e interaccionan en él de forma concurrente.

1.3. ¿Divisiones del problema necesarias?

No.

1.4. ¿Informe de antecedentes?

No.

1.5. Preguntas propuestas

1. ¿Qué debería permitir el sistema a los usuarios?
2. ¿Qué herramientas debería haber?

2. Ejecución de la tormenta de ideas

2.1. Restricción

Debido al conocimiento del estado del arte se ha puesto la restricción de que las ideas que se vayan a proponer no contemplen ningún mapa interactivo debido a su gran complejidad y falta de conocimiento para el desarrollo.

Según los asistentes, el añadido de esta restricción supone un punto negativo importante a la visualización de la aplicación y por ello a la interacción del usuario con el sistema, cuya importancia media, de no añadir un mapa, es de 4 siendo la escala del 0 al 10 donde 0 es sin importancia y un 10 es imprescindible que se añadiera.

2.2. Inicio

1. ¿Qué debería permitir el sistema a los usuarios?

- Que las reglas de los juegos sean modificables.
- Sistema genérico con fórmulas simples.
- Juegos concretos: Pathfinder, Vampiro...
- Modificar en cualquier juego una fórmula de tirada.
- Debería permitir al Director de Juego, a partir de ahora DJ, que es lo que no puede tocar y ver el jugador.

2. ¿Qué herramientas debería haber?

- Deshacer y rehacer.
- Editor de fórmulas.
- Crear de forma rápida mapeados básicos, predeterminados.
- Opción a crear mapeados.
- Casillas modificables.
- Modo dios para el DJ: añadir/quitar puntos de vida.
- Personalización del apartado visual al jugador, imágenes...
- Importar y exportar, workshop.
- Tiradas ocultas.
- Crear partida, contraseña, expulsar/invitar jugador.
- Añadir música e imagen.
- Capacidad de controlar a más de un personaje.
- Generadores automáticos: personajes no jugadores y criaturas.
- Generar tesoro aleatorio para X nivel.
- Editor de partida, preparar partida con antelación.
- Cola de iniciativa.

- Sistema de ventanas.
- Buscador de elementos en inventario y cadáveres.

2.3. Debate

Además de debatir sobre los diferentes puntos sacados anteriormente se ha realizado un análisis técnico superficial sobre la viabilidad en tiempo y complejidad que tendría desarrollar cada una de las ideas. Se ha realizado un comentario en cada una de ellas con este propósito.

Algunos puntos han sido modificados y se han añadido algunos nuevos, fruto del propio debate.

Como aún no se conoce lo que abarcará la primera versión, solamente se llamará como funcionalidad necesaria aquello que sea solamente imprescindible para el mínimo funcionamiento del sistema. Así mismo todo lo que no se presente con la etiqueta de “viable” es que no se conoce si puede ser desarrollado de forma simple, por lo que se requerirá de un análisis más exhaustivo para conocer ese dato.

2.3.1. ¿Qué debería permitir el sistema a los usuarios?

Sistema de juego

- Que las reglas de los juegos sean modificables y editables.
 - No es viable, es muy dinámico.
- Sistema genérico con fórmulas simples. Donde puedas hacer tiradas con fórmulas especificadas por el usuario.
 - Viable y necesario para la primera versión.
- Juegos concretos: Pathfinder, D&D, Vampiro, Hombre Lobo, Aquelarre y Cthulhu.
 - No apta para la primera versión.
 - Se recomienda primero el más fácil y luego el que más cuota de mercado tenga.
- Modificar en cualquier juego una fórmula.
 - No es viable, es muy dinámico.

Restricciones y permisos

- Debería permitir al director de juego que es lo que no puede tocar y ver el jugador.
 - Complejo pero necesario para la primera versión.
 - Especial atención con “la selección de qué ve quién”.
- Ojo automatizado con niveles/rangos de permisos para la visualización.
 - Viable para versiones futuras.
- El director de juego puede ponerse en vista de cada jugador.
 - Viable y recomendado para la primera versión.

2.3.2. ¿Qué herramientas debería haber?

Propias del DJ

- Deshacer y rehacer los cambios del juego.

- Viable y recomendado para la primera versión.
- Modo dios para el DJ que le permita:
 - Añadir/quitar puntos de vida, estados...
 - Poder modificar la ficha del pj directamente.
 - Crear partida, contraseña, expulsar/invitar jugador, expulsar personajes. modo espectador...
 - Alguna de estas características son viables y recomendables para la primera versión.
- Editor de partida, preparar partida con antelación.
 - Recomendado para versiones futuras.
- Generar tesoro aleatorio para X nivel.
 - Muy específico, recomendado para versiones futuras.
- Cola de iniciativa.
 - Viable y necesario para la segunda versión.
- Generadores automáticos: personajes no jugadores y criaturas.
 - Recomendado para versiones futuras.
- Buscador de elementos en inventario y cadáveres.
 - Recomendado para las siguientes versiones.
- Baúl con pnjs y criaturas de la partida para el DJ.
 - Recomendado para versiones futuras.

Generales

- Personalización del apartado visual del jugador con:
 - Bordes con dibujos, fondos con transparencia.
 - Imágenes de perfil.
 - Secundario y recomendado.
- Partidas co-dirigidas.
 - Secundario, para versiones posteriores.
- Importar y exportar, workshop.
 - Recomendado para versiones posteriores.
- Estados del pj.
 - Categorías de estado.
 - Recomendado para versiones posteriores.
- El jugador puede modificar la ficha de sus pjs en la partida.
- El DJ podrá hacer lo propio sin restricción.
 - Mediante un permiso que le de el DJ al jugador de hacer esa modificación, como un candado por ejemplo.
 - Necesario y viable para la primera versión.
- Tiradas ocultas.

- Necesario y viable para la primera versión.
- Capacidad de controlar a más de un personaje.
 - Viable y necesario para la primera versión.
- Guardar y cargar partida.
 - X personajes y partidas guardadas en el servidor.
 - Muy relacionado con el departamento de comercialización.
 - Recomendado para versiones futuras.
- Lanzador de dados.
 - Viable y necesario para la primera versión.
- Gamificación.
 - Recomendado para versiones futuras.
- Añadir música e imágenes para ambientar.
 - Recomendado y secundario.
- Posit con datos, ya sean de música, imágenes o texto.
 - Recomendado para la primera versión.
- Crear de forma rápida mapeados básicos, predeterminados.
- Opción a crear mapeados.
- Casillas modificables.
- Imágenes a las que se le puedan solapar cuadrículas para hacer mapas, además de permitir mover fichas que representen los pjs y criaturas de los juegos.
 - Buena idea que permite suplir el defecto de no tener un mapa interactivo pero que permite una visualización de los escenarios del juego.
 - Añadir imágenes e insertar cuadrículas o de otras formas con escala. Fichas con coordenadas y poder mover, eliminar, crear.
 - Recomendado para la primera versión.

2.4. Conclusión

Como conclusión destacaría la gran cantidad de herramientas y funcionalidad que debe tener el sistema para asemejarse lo máximo posible a una partida de rol de mesa. Además gran parte de esta funcionalidad no podrá ser desarrollada, debido a su coste, para la primera versión del sistema. Finalmente indicar que se trata de una aplicación con un posible gran crecimiento de funcionalidad por lo que hay que desarrollarla lo más mantenible posible.

Bibliografía

- [1] © The Orr Group, LLC. Roll20. Consultado el 7 de agosto de 2016. Página web: <https://roll20.net/>
- [2] SmiteWorks USA, LLC. Fantasy Ground. Consultado el 7 de agosto de 2016. Página web: <http://www.fantasygrounds.com/>
- [3] A HAPPY DREAMHOST CUSTOMER. D20PRO. Consultado el 7 de agosto de 2016. Página web: <http://d20pro.com/>
- [4] ©2006 Ian Toltz. Combattracker. Consultado el 7 de agosto de 2016. Página web: <http://www.asmor.com/scripts/CombatTracker/combattracker.html>
- [5] Jerome Dumonteil. Dicelog. Consultado el 7 de agosto de 2016. Página web: <https://dicelog.com/>
- [6] WHOISGUARD, INC. Battlegrounds Games. Consultado el 7 de agosto de 2016. Página web: <http://battlegroundsgames.com/>
- [7] © 2016 RPTools. RPTools. Consultado el 7 de agosto de 2016. Página web: <http://www.rptools.net/>
- [8] Craig Yack. TTToprpg. Consultado el 7 de agosto de 2016. Página web: <http://www.tttoprpg.com/>
- [9] Wikipedia. Juego de rol. Consultado el 7 de agosto de 2016. Página web: es.wikipedia.org/wiki/Juego_de_rol
- [10] EcuRed. AUP. Consultado el 7 de agosto de 2016. Página web: http://www.ecured.cu/index.php/Agile_Unified_Process
- [11] Copyright 2005-2014 Scott W. Ambler. AUP. Consultado el 7 de agosto de 2016. Página web: <http://www.ambysoft.com/unifiedprocess/agileUP.html>
- [12] Salario Mínimo. Salario Mínimo Interprofesional. Consultado el 18 de agosto de 2016. Página web: <http://www.salariominimo.es/>
- [13] Amador Durán Toro. REM (REquirements Management). Consultado el 7 de agosto de 2016. Página web: www.lsi.us.es/descargas/descarga_programas.php?id=3
- [14] Bulmahn, J., Gygax, G., & Arneson, D. (2009). Pathfinder roleplaying game: Core rulebook. Paizo Publishing
- [15] Greenberg, A., Greenberg, D., Rein-Hagen, M., & Davis, G. (1995). Vampiro la Mascarada. Guía del jugador.
- [16] Lockhart, J.. (2015-03-01). Modern PHP: New Features and Good Practices. O'Reilly Media, Inc.
- [17] Potencier, F., & Zaninotto, F. (2008). Symfony, la guía definitiva. Libros Web.[Online] www.librosweb.es
- [18] Nixon, R.. (2014-12-14). Learning PHP, MySQL & JavaScript. O'Reilly
- [19] Wellman, D. (2011). JQuery UI 1. 8: The User Interface Library for JQuery. Packt Publishing Ltd
- [20] Chacon, S. (2009). Git community book. The Git Community
- [21] Debrauwer, L. & Evain, Y.. (noviembre 2015). Patrones de diseño en PHP. ediciones ENI

- [22] APA Tupe, C., & Cisneros, J. (2008). Evaluación y Selección de Framework de Desarrollo PHP: Symfony, Kumbia, CakePHP y Zend.
- [23] Ben-Kiki, O., Evans, C., & Ingerson, B. (2005). YAML Ain't Markup Language (YAML™) Version 1.1. yaml. org, Tech. Rep
- [24] Copyright © 1997-2016 The Apache Software Foundation. Apache. Consultado el 13 de agosto de 2016. Página web: <https://httpd.apache.org/>
- [25] PHP. PHP.net. Consultado el 13 de agosto de 2016. Página web: <http://php.net/>
- [26] PHP. PHP. Consultado el 13 de agosto de 2016. Página web: <https://github.com/php/php-src>
- [27] Symfony. Symfony. Consultado el 13 de agosto de 2016. Página web: <https://github.com/symfony/symfony>
- [28] © 2016 - symfony.es. Symfony.es. Consultado el 13 de agosto de 2016. Página web: <http://symfony.es/>
- [29] Composer. Composer. Consultado el 13 de agosto de 2016. Página web: <https://github.com/composer/composer>
- [30] Doctrine. Doctrine. Consultado el 13 de agosto de 2016. Página web: <https://github.com/doctrine>
- [31] © 2010-2016 SensioLabs. Twig. Consultado el 13 de agosto de 2016. Página web: <http://twig.sensiolabs.org/>
- [32] Clark C. Evans. YAML. . Página web: <http://yaml.org/>
- [33] Copyright © 2001-2016 Sebastian Bergmann. PHPUnit. Consultado el 13 de agosto de 2016. Página web: <https://phpunit.de>
- [34] Copyright © 2012-2014 Tavendo GmbH and contributors. WAMP. Consultado el 13 de agosto de 2016. Página web: <http://wamp-proto.org/>
- [35] GeniusesOfSymfony. WebSocketBundle. Consultado el 13 de agosto de 2016. Página web: <https://github.com/GeniusesOfSymfony/WebSocketBundle>
- [36] Copyright © 2011-2014 Tavendo GmbH. Autobahn. Consultado el 13 de agosto de 2016. Página web: <http://autobahn.ws/js/>
- [37] Copyright © 2012-2014 Tavendo GmbH and contributors. Ratchet. Consultado el 13 de agosto de 2016. Página web: <http://socketo.me/>
- [38] FriendsOfSymfony. FOSUserBundle. Consultado el 13 de agosto de 2016. Página web: <https://github.com/FriendsOfSymfony/FOSUserBundle>
- [39] Copyright 2011-2016 Twitter, Inc. Bootstrap. Consultado el 13 de agosto de 2016. Página web: <https://github.com/twbs/bootstrap>
- [40] Developed by Florian Eckerstorfer and amazing contributors. Bootstrap Bundle. Consultado el 13 de agosto de 2016. Página web: <https://github.com/braincrafted/bootstrap-bundle>
- [41] less.php was originally ported to php by Matt Agar and then updated by Martin Jantošovič.. Less. Consultado el 13 de agosto de 2016. Página web: <https://github.com/oyejorge/less.php>
- [42] KnpLabs. KnpMenuBundle. Consultado el 13 de agosto de 2016. Página web: <https://github.com/KnpLabs/KnpMenuBundle>
- [43] Learning Center Forum API Twitter IRC GitHubCopyright 2016 The jQuery Foundation. jQuery. Consultado el 13 de agosto de 2016. Página web: <https://jquery.com/>

- [44] Learning Center Forum API Twitter IRC GitHubCopyright 2016 The jQuery Foundation. jQuery UI. Consultado el 13 de agosto de 2016. Página web: <https://jqueryui.com/>
- [45] © 2016, Oracle Corporation and/or its affiliates. MySQL. Consultado el 13 de agosto de 2016. Página web: <https://www.mysql.com/>
- [46] © 2016, Oracle Corporation and/or its affiliates. MySQL Workbench. Consultado el 13 de agosto de 2016. Página web: <https://www.mysql.com/products/workbench/>
- [47] Copyright © 2016 W3C ®. HTML. Consultado el 13 de agosto de 2016. Página web: <https://www.w3.org/html/>
- [48] Copyright © 2016 W3C ®. CSS. Consultado el 13 de agosto de 2016. Página web: <https://www.w3.org/Style/>
- [49] © 2005-2016 Mozilla Developer Network and individual contributors. JavaScript. Consultado el 13 de agosto de 2016. Página web: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [50] Van Lancker, L. & Vigouroux, C.. (octubre 2015). jQuery y JavaScript. Ediciones ENI
- [51] Standard ECMA-262 3rd Edition - December 1999. JSON. Consultado el 13 de agosto de 2016. Página web: <http://www.json.org/>
- [52] Copyright © 2016 The Eclipse Foundation. Eclipse. Consultado el 15 de agosto de 2016. Página web: <https://eclipse.org/>
- [53] Git is a free and open source. Git. Consultado el 15 de agosto de 2016. Página web: <https://git-scm.com/>
- [54] © 2016 GitHub, Inc. GitHub. Consultado el 15 de agosto de 2016. Página web: <https://github.com/>

