

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

**ARPEGOS: AUTOMATIZED ROLEPLAYING-GAME  
PROFILE EXTENSIBLE GENERATOR ONTOLOGY  
BASED SYSTEM**

AUTOR: ALEJANDRO MUÑOZ DEL ÁLAMO

Puerto Real, mes \_ año



TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

**ARPEGOS: AUTOMATIZED ROLEPLAYING-GAME  
PROFILE EXTENSIBLE GENERATOR ONTOLOGY  
BASED SYSTEM**

DIRECTOR: D. ALBERTO GABRIEL SALGUERO HIDALGO  
CODIRECTOR: D. PABLO GARCÍA SÁNCHEZ  
AUTOR: ALEJANDRO MUÑOZ DEL ÁLAMO

Puerto Real, mes \_ año



## DECLARACIÓN PERSONAL DE AUTORÍA

Alejandro Muñoz Del Álamo con DNI 77396804-X, estudiante del Grado en Ingeniería Informática en la Escuela Superior de Ingeniería de la Universidad de Cádiz, como autor de este documento académico, titulado ARPEGOS: Automated Roleplaying-game Profile Extensible Generator Ontology based System y presentado como trabajo final de Grado.

### DECLARO QUE

Es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara su origen, tanto en el cuerpo del texto, como en su bibliografía, y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Así mismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse.

En Puerto Real, a día \_ de mes \_ de año

Fdo: Alejandro Muñoz Del Álamo



## *Dedicatoria*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam maximus facilisis odio vel dignissim. Integer sed tincidunt neque. Cras vitae nisi odio. Morbi in pellentesque nulla. Nam sagittis leo nec ex ultricies, eget hendrerit velit tincidunt. Phasellus fringilla varius tellus, in aliquam purus convallis eget. Donec mattis nisi turpis, vitae ultrices lectus ornare at. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis in rutrum eros. Duis ut lacinia justo. Donec fringilla velit eu sem congue cursus. Etiam porttitor, justo gravida porttitor ultrices, eros sem lacinia ex, eget bibendum enim dolor non ligula. Lorem ipsum dolor sit amet, consectetur adipiscing elit.





## *Agradecimientos*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam maximus facilisis odio vel dignissim. Integer sed tincidunt neque. Cras vitae nisi odio. Morbi in pellentesque nulla. Nam sagittis leo nec ex ultricies, eget hendrerit velit tincidunt. Phasellus fringilla varius tellus, in aliquam purus convallis eget. Donec mattis nisi turpis, vitae ultrices lectus ornare at. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis in rutrum eros. Duis ut lacinia justo. Donec fringilla velit eu sem congue cursus. Etiam porttitor, justo gravida porttitor ultrices, eros sem lacinia ex, eget bibendum enim dolor non ligula. Lorem ipsum dolor sit amet, consectetur adipiscing elit.



## *Resumen*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam maximus facilisis odio vel dignissim. Integer sed tincidunt neque. Cras vitae nisi odio. Morbi in pellentesque nulla. Nam sagittis leo nec ex ultricies, eget hendrerit velit tincidunt. Phasellus fringilla varius tellus, in aliquam purus convallis eget. Donec mattis nisi turpis, vitae ultrices lectus ornare at. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis in rutrum eros. Duis ut lacinia justo. Donec fringilla velit eu sem congue cursus. Etiam porttitor, justo gravida porttitor ultrices, eros sem lacinia ex, eget bibendum enim dolor non ligula. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

## *Abstract*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam maximus facilisis odio vel dignissim. Integer sed tincidunt neque. Cras vitae nisi odio. Morbi in pellentesque nulla. Nam sagittis leo nec ex ultricies, eget hendrerit velit tincidunt. Phasellus fringilla varius tellus, in aliquam purus convallis eget. Donec mattis nisi turpis, vitae ultrices lectus ornare at. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis in rutrum eros. Duis ut lacinia justo. Donec fringilla velit eu sem congue cursus. Etiam porttitor, justo gravida porttitor ultrices, eros sem lacinia ex, eget bibendum enim dolor non ligula. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

## *Palabras clave*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam maximus facilisis odio vel dignissim. Integer sed tincidunt neque. Cras vitae nisi odio. Morbi in pellentesque nulla. Nam sagittis leo nec ex ultricies, eget hendrerit velit tincidunt. Phasellus fringilla varius tellus, in aliquam purus convallis eget. Donec mattis nisi turpis, vitae ultrices lectus ornare at. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis in rutrum eros. Duis ut lacinia justo. Donec fringilla velit eu sem congue cursus. Etiam porttitor, justo gravida porttitor ultrices, eros sem lacinia ex, eget bibendum enim dolor non ligula. Lorem ipsum dolor sit amet, consectetur adipiscing elit.



## Notación y formato

En la siguiente tabla se presenta un conjunto de convenios de notación de sintaxis.

Notación establecida	
<b>negrita</b>	Título o texto destacado
<i>cursiva</i>	Texto en otro idioma, destacado, citas o nombres de aplicaciones
<code>monoespaciado</code>	Referencias a código fuente
<u>subrayado</u>	Advertencia para el lector
color	Enlace interno ( <b>rojo</b> )



---

## Índice general

---

Índice general	XI
Índice de figuras	XV
Índice de cuadros	XVII
 <b>I Prolegómeno</b>	 <b>1</b>
<b>1. Introducción</b>	<b>3</b>
1.1. Motivación . . . . .	4
1.2. Alcance . . . . .	7
1.3. Glosario de Términos . . . . .	7
1.4. Estructuración del documento . . . . .	10
 <b>2. Antecedentes</b>	 <b>13</b>
2.1. Estado del Arte . . . . .	13
2.2. Crítica al estado del arte . . . . .	16
2.3. Propuesta . . . . .	16
2.3.1. Plataforma . . . . .	16
2.3.2. Arquitectura . . . . .	16
2.3.3. Modelo de datos . . . . .	17
2.4. Metodología . . . . .	18
2.4.1. Metodologías de desarrollo software . . . . .	18
2.4.2. Metodologías Ágiles . . . . .	20
2.4.3. SCRUM . . . . .	21

<b>3. Planificación del proyecto</b>	<b>25</b>
3.1. Planificación del proyecto . . . . .	25
3.1.1. Objetivo inicial . . . . .	25
3.1.2. División del trabajo . . . . .	25
3.1.3. Identificación de <i>sprints</i> y estimación de tiempos . . . . .	26
3.1.4. Resumen de la planificación del proyecto . . . . .	27
3.2. Organización . . . . .	28
3.2.1. Agentes involucrados . . . . .	28
3.2.2. Recursos utilizados . . . . .	28
3.3. Costes . . . . .	29
3.3.1. Costes humanos . . . . .	29
3.3.2. Costes materiales . . . . .	29
3.4. Evaluación y gestión de riesgos . . . . .	29
3.4.1. ¿Qué es un riesgo? . . . . .	29
3.4.2. Gestión de riesgos . . . . .	29
3.4.3. Identificación de riesgos . . . . .	30
3.4.4. Reducción de riesgos . . . . .	30
 <b>II Desarrollo</b>	 <b>33</b>
<b>4. Requisitos del Sistema</b>	<b>35</b>
4.1. Situación Actual . . . . .	35
4.1.1. Crítica al estado del arte . . . . .	38
4.2. Objetivos del Sistema . . . . .	38
4.2.1. Requisitos funcionales . . . . .	39
4.2.2. Requisitos no funcionales . . . . .	39
4.3. Alternativas de Solución . . . . .	40
4.3.1. Android Studio . . . . .	40
4.3.2. React Native . . . . .	41
4.3.3. Xamarin . . . . .	42
4.4. Solución propuesta . . . . .	42
<b>5. Análisis del Sistema</b>	<b>45</b>
<b>6. Diseño del Sistema</b>	<b>47</b>
<b>7. Construcción del Sistema</b>	<b>49</b>
<b>8. Pruebas del Sistema</b>	<b>51</b>
 <b>III Epílogo</b>	 <b>53</b>
<b>9. Manual de Implantación y Explotación</b>	<b>55</b>



<b>10.Manual de Usuario</b>	<b>57</b>
<b>11.Manual de Implantación y Explotación</b>	<b>59</b>
<b>12.Conclusiones</b>	<b>61</b>
<b>13.Bibliografía</b>	<b>63</b>
<b>14.Información sobre Licencia</b>	<b>69</b>



---

## Índice de figuras

---

2.1.	<b>5e Character:</b> Pantalla de selección de clases . . . . .	13
2.2.	<b>5e Character:</b> Información de la clase <i>Druida</i> . . . . .	13
2.3.	<b>RPG Simple Dice:</b> Pantalla de selección de dados . . . . .	14
2.4.	<b>RPG Simple Dice:</b> Ejemplo de lanzamiento de dados . . . . .	14
2.5.	<b>BattleTrack:</b> Pantalla de selección de grupo . . . . .	14
2.6.	<b>BattleTrack:</b> Ejemplo de combate . . . . .	14
2.7.	<b>RPGSound:</b> Menú principal . . . . .	15
2.8.	<b>RPGSound:</b> Menú de <i>Ambiente Sostenido</i> . . . . .	15
2.9.	<b>RPG Character Sheet:</b> Pantalla de <i>Estado</i> . . . . .	15
2.10.	<b>RPG Character Sheet:</b> Pantalla de <i>Features</i> . . . . .	15
2.11.	Patrón MVVM: <i>Modelo-Vista-Modelo de Vista</i> [14] . . . . .	17
2.12.	Esquema de un modelo de datos [33] . . . . .	17
2.13.	Modelo de desarrollo en cascada . . . . .	18
2.14.	Modelo de desarrollo en espiral . . . . .	19
2.15.	Modelo de desarrollo con prototipos [38] . . . . .	20
2.16.	Modelo de desarrollo incremental [40] . . . . .	20
2.17.	Logo de <i>Scrum</i> . . . . .	21
2.18.	Ciclo de desarrollo de <i>Scrum</i> . . . . .	23
2.19.	Roles en <i>Scrum</i> . . . . .	24
4.1.	<b>5e Character:</b> Pantalla de selección de clases . . . . .	35
4.2.	<b>5e Character:</b> Información de la clase <i>Druida</i> . . . . .	35
4.3.	<b>RPG Simple Dice:</b> Pantalla de selección de dados . . . . .	36
4.4.	<b>RPG Simple Dice:</b> Ejemplo de lanzamiento de dados . . . . .	36
4.5.	<b>BattleTrack:</b> Pantalla de selección de grupo . . . . .	36
4.6.	<b>BattleTrack:</b> Ejemplo de combate . . . . .	36
4.7.	<b>RPGSound:</b> Menú principal . . . . .	37
4.8.	<b>RPGSound:</b> Menú de <i>Ambiente Sostenido</i> . . . . .	37

4.9. <i>RPG Character Sheet</i> : Pantalla de <i>Estado</i> . . . . .	37
4.10. <i>RPG Character Sheet</i> : Pantalla de <i>Features</i> . . . . .	37
4.11. Logo de <i>Android Studio</i> . . . . .	41
4.12. Logo de <i>Apache Jena</i> . . . . .	41
4.13. Logo de <i>React Native</i> . . . . .	41
4.14. Logo de <i>rdflib.js</i> . . . . .	41
4.15. Logo de <i>Xamarin</i> . . . . .	42

---

## Índice de cuadros

---

1.1. <i>Lankoski, Heliö y Ekman</i> : Estructura ósea de una persona tridimensional. [5] . . . . .	5
3.1. Comparativa entre coste humano estimado y coste humano real . . . . .	29
4.1. Comparativa entre <i>JavaScript</i> y <i>C#</i> . . . . .	42



Parte I

Prolegómeno





# CAPÍTULO 1

---

## Introducción

---

*“En Arrswyd, la oscuridad ha consumido las estrellas en el cielo, dejando nada más que una sola ciudad en medio del olvido. Arrswyd es una nación amurallada donde permanece la última población del mundo. El día ha sido olvidado hace mucho tiempo, los miedos se hacen realidad, y entre las torcidas espirales, merodeando entre las sombras, hay horrores que se alimentan del miedo. El terror es todo lo que Arrswyd sabe ya que su gente es perseguida por criaturas y oprimida por el Señor inmortal, Yomon Ecuro. Algunos se animan a luchar contra el abismo y las maquinaciones del Ministerio y del Instituto, mientras que otros simplemente quieren vivir los últimos días del mundo hasta que se apague la última luz.”*

---

*Arrswyd, Regnum Ex Nihilo*

## 1.1. Motivación

Los juegos de rol, a los que llamaremos *RPG* (*Role-Playing Game*) a partir de ahora, son juegos con una meta particular, que es “*El objetivo de ambos tipos de juegos, computerizados y de otro tipo, es experimentar una serie de aventuras en un mundo imaginario, a través de un personaje avatar o un pequeño grupo de personajes cuyas habilidades y poderes crecen conforme el tiempo pasa.*” [1].

Uno de los aspectos más importantes de los *RPG* son los personajes, ya que estos son la manifestación de los jugadores dentro del universo del juego, que a partir de ahora llamaremos **personajes jugadores** o *PJs*. Ramos y Sueiro [2] plantean que, a diferencia del teatro, donde la elección de un personaje puede depender de las características físicas del intérprete, en los juegos de rol cualquier persona puede interpretar cualquier personaje, siempre que esté dentro de las posibilidades que ofrezca el juego en que se esté jugando. Los personajes deben definirse dentro de los límites establecidos por las reglas del juego, y en una historia como si se tratara de su biografía, explicando cómo ha sido la vida del personaje en la ambientación hasta el momento de comenzar la historia. A este proceso de definir un personaje para un jugador se conoce como **creación de personaje**.

Hay un amplio rango de posibilidades cuando tratamos la creación de *PJs* en diferentes mundos. En algunos juegos, el jugador sólo puede seleccionar algunas características predefinidas para el personaje, mientras que en otros juegos el usuario puede cambiar cada elemento del avatar [3]. Esto hace que la creación de personajes sea un proceso arduo y complejo, pues es necesario tener amplios conocimientos del juego para conocer todas las opciones de personalización disponibles para el jugador, y sus correspondientes características.

Egri [4] afirma en su libro *The Art of Dramatic Writing: Its Basis in the Creative Interpretation of Human Motives* que mientras todo objeto tiene tres dimensiones: profundidad, altura y anchura, los seres humanos tienen tres dimensiones adicionales: **fisiología**, **sociología** y **psicología**, y que sin el conocimiento de estas dimensiones, no se puede valorar a un ser humano. La primera dimensión, la **fisiología**, da color al punto de vista humano y le influye infinitamente, facilitando a una persona ser tolerante, desafiante, humilde o arrogante. La **sociología**, hace referencia a que no es posible realizar un análisis exacto de las diferencias entre una persona y el vecino de la puerta de al lado si no se tiene suficiente información de las circunstancias de ambas personas. La tercera dimensión, la **psicología**, es el producto de las otras dos, cuya influencia da vida a la ambición, la frustración, el temperamento, las actitudes y los complejos.

En base a esto, Lankoski, Heliö y Ekman [5] presentan una tabla, que se muestra a continuación, definiendo los aspectos de esta estructura definida por Egri, modificando y añadiendo algunos aspectos de la misma.

<i>Fisiología</i>	<i>Sociología</i>	<i>Psicología</i>
Sexo	Clase	Estándares morales y vida sexual
Edad	Ocupación	Metas y ambiciones
Altura y anchura	Educación	Frustraciones y decepciones
Color de pelo, ojos y piel	Vida familiar	Temperamento
Postura	Religión	Actitud frente a la vida
Apariencia y rasgos distintivos	Raza y nacionalidad	Complejos y obsesiones
Defectos	Posición social	Imaginación, juicios, sabiduría, gustos y estabilidad
Rasgos hereditarios	Afiliaciones políticas	Extroversión, introversión y ambiversión
Físico	Entretenimientos y aficiones	Inteligencia

Cuadro 1.1: *Lankoski, Heliö y Ekman*: Estructura ósea de una persona tridimensional. [5]

En lo referido a juegos de rol, *Tychsen, Hitchens y Brolund* [6] sostienen que “*el diseño de personajes está dividido en cuatro componentes: **personalidad**, **integración**, **apariencia** y **reglas**, que tienen su propio conjunto de restricciones para asegurar consistencia metodológica*”.

1. **Reglas (Estadísticas)**: Son las características basadas en las reglas del juego, tales como *atributos*, *aptitudes*, *habilidades* y *clase*. Los *RPG* proveen estas características en plantillas conocidas como *hojas de personaje*, que detallan todos los componentes basados en reglas del personaje, referentes al juego al que pertenezca el *PJ* en cuestión.
2. **Integración**: Los componentes de integración son aquellos que explican las circunstancias del personaje. Algunos de estos componentes son:
  - *Ubicación*: ¿Dónde está el personaje y por qué?
  - *Trasfondo*: ¿Cual es la historia detrás del personaje y qué eventos lo llevan al punto de inicio del juego?
  - *Contactos*: ¿Que contactos y relaciones tiene el personaje con otros personajes no controlados por jugadores (conocidos como *personajes no jugadores* o *PNJs*)?
  - *Conexiones*: ¿Cuál es la relación entre el personaje y los demás *PJs*?

3. **Apariencia:** El personaje requiere una definición de su apariencia, para lo que son necesarios algunos rasgos de importancia:

- *Representación:* ¿Cómo luce el personaje?
- *Comportamiento físico:* ¿Cómo se comporta el personaje? ¿Cómo puede su interacción con el mundo modificar su personalidad o comportamiento?
- *Interacción:* ¿Cómo interactúa el personaje con el entorno y como interacciona con otros personajes?

4. **Personalidad:** Este componente incluye descripciones de la psique del personaje (emociones, comportamiento) y metas, con algunos rasgos de personalidad muy definidos enfáticamente (normalmente más sutiles y que dan lugar a la interpretación personal del jugador).

De todos los apartados previos, hay uno que destaca por estar plenamente orientado a la información del juego, que es el apartado de *Reglas*, ya que es el único que tiene información objetiva del personaje, y que sigue una estructura, dada por el *RPG* al que vaya a pertenecer el personaje, que posibilita establecer un conjunto de elementos estructurado, que unido a la normativa del juego, permita establecer las características técnicas de cualquier personaje.

En relación con lo comentado previamente, y de acuerdo con el autor del blog *Ars Rolica* [7], una de las novedades que se están popularizando más es la de los **generadores de personajes**, que son herramientas que permiten crear personajes de juegos sin necesidad de invertir grandes cantidades de tiempo, evitando posibles errores y deslices. Estos generadores pueden ser bastante útiles tanto para jugadores novatos que no conocen el sistema del juego, como para jugadores más experimentados que no puedan utilizar tiempo para documentarse completamente antes de crear su álter ego.

El objetivo de este proyecto consiste en diseñar e implementar un generador de personajes de juegos de rol de mesa, para lo que se perseguirán los siguientes subobjetivos:

- Realizar un estudio del arte de los *generadores de personaje* existentes, que permita conocer el estado actual de las herramientas que se encuentran en uso en la actualidad.
- Desarrollar un sistema que permita trabajar con diferentes *RPGs*, sin tener que salir de la aplicación, siempre y cuando se disponga de los ficheros que contengan la información de dichos juegos.
- Diseñar una estructura genérica para el sistema de información, de forma que permita a la aplicación poder adaptarse de la forma más completa posible a cada juego, permitiendo profundizar de la misma manera en juegos de alta complejidad que en juegos sencillos.
- Crear una interfaz de usuario intuitiva, que facilite al usuario la interacción con la aplicación y ésta resulte cómoda y agradable de utilizar.

## 1.2. Alcance

Este proyecto consiste en una aplicación que permita al usuario generar fichas técnicas de personajes para poder participar en partidas o campañas de *RPG*. La aplicación debe servir para poder desarrollar personajes de cualquier juego, siempre y cuando disponga de la información del mismo, pero a su vez debe poder explorar todas las posibilidades de cada juego, sin importar su complejidad.

El objetivo principal que se desea conseguir con esta aplicación es facilitar a todos los jugadores de rol una herramienta que facilite la comprensión del proceso de creación de personajes y reduzca la duración del mismo, pero sin dejar de valorar todas las opciones que el juego pone a su disposición.

El otro objetivo de este proyecto es desarrollar una herramienta general que pueda ser útil para cualquier *RPG*, independientemente del sistema de creación de personajes que utilice, pues no resulta práctico tener que disponer de un *generador de personajes* por cada *RPG* que existe, ya que el coste espacial de disponer de todos ellos resultaría imposible de contener en una herramienta de gama media, de forma que no estaría disponible para todos los jugadores.

Para conseguir esto, se ha considerado que, en vez de contemplar como objetivo la creación de una base de datos que contenga la información de un juego concreto, se va a fijar como meta el diseño de una estructura general para que cualquier persona que quiera introducir la información de un nuevo juego, tenga a su disposición una metodología clara para desarrollar el banco de datos de manera que la aplicación sea capaz de trabajar con éste.

## 1.3. Glosario de Términos

- **Android:** “Sistema operativo que se emplea en dispositivos móviles, por lo general con pantalla táctil. De este modo, es posible encontrar tabletas (tablets), teléfonos móviles (celulares) y relojes equipados con Android, aunque el software también se usa en automóviles, televisores y otras máquinas. Creado por Android Inc., una compañía adquirida por Google en 2005, Android se basa en Linux, un programa libre que, a su vez, está basado en Unix.” [8]
- **C#:** “Lenguaje de programación simple, moderno, orientado a objetos y fuertemente tipado. C# tiene sus raíces en la familia de lenguajes C [...] C# está estandarizado por ECMA International con el estándar ECMA-334 y por ISO/IEC con el estándar ISO/IEC 23270.” [9]
- **Git:** “Herramienta de control de versiones particularmente potente, flexible y bajos costes generales que hace del desarrollo colaborativo un placer. Git fue inventado por Linus Torvalds para dar soporte al desarrollo del kernel Linux, pero desde entonces ha probado ser valioso a un amplio rango de proyectos” [10]

- **GitHub**: “Servidor de repositorios de código basado en el sistema de control de versiones **Git**” [11]
- **IDE**: “Un entorno de desarrollo integrado o **IDE** es una aplicación visual que sirve para la construcción de aplicaciones a partir de componentes.” [12]
- **Metodología de desarrollo**: “Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo.” [13]
- **Metodología Ágil**: “Metodologías que se derivan de la lista de los principios que se encuentran en el **Manifiesto Ágil**, y están basadas en un desarrollo iterativo que se centra más en capturar mejor los requisitos cambiantes y la gestión de los riesgos, rompiendo el proyecto en iteraciones de diferente longitud” [13]
- **Modelo**: “Las clases de modelo son clases no visuales que encapsulan los datos de la aplicación. Por lo tanto, se puede considerar que el modelo representa el modelo de dominio de la aplicación, que normalmente incluye un modelo de datos junto con la lógica de validación y negocios.” [14]
- **Modelo de Vista**: “El modelo de vista implementa las propiedades y los comandos a los que la vista puede enlazarse y notifica a la vista de cualquier cambio de estado a través de los eventos de notificación de cambios. Las propiedades y los comandos que proporciona el modelo de vista definen la funcionalidad que ofrece la interfaz de usuario, pero la vista determina cómo se mostrará esa funcionalidad.” [14]
- **MVVM**: “Patrón de arquitectura de software que ayuda a separar la lógica de negocios y presentación de una aplicación de su interfaz de usuario.” [14]
- **Ontología**: “Las ontologías computacionales tienen como objetivo modelar la estructura de un sistema, por ejemplo, las entidades y relaciones relevantes que emergen de la observación, que son útiles a nuestros propósitos.[...] Gruber [15] describe la noción de una ontología como la especificación explícita de una conceptualización compartida” [16]
- **OWL**: “El lenguaje de ontología web **OWL** es un lenguaje de marcado semántico para publicar y compartir ontologías en la World Wide Web. OWL está desarrollado como una extensión de vocabulario de **RDF** y está derivado del lenguaje de ontología web **DAML+OIL**.” [17]
- **Protégé**: “El sistema Protégé es un entorno para el desarrollo de sistemas basados en conocimiento que ha evolucionado durante más de una década. Protégé comenzó

como una pequeña aplicación diseñada para un dominio médico (planificación de terapia basada en protocolos), pero ha evolucionado en un conjunto de herramientas con un propósito mucho más general [...] El objetivo inicial de Protégé era reducir el cuello de botella de adquisición de conocimiento minimizando el rol del ingeniero de conocimiento construyendo bases de conocimiento.” [18]

- **RDF**: “El **F**ramework de **D**escripción de **R**ecursos o **RDF** es una fundación para procesar metadatos; provee interoperabilidad entre aplicaciones que intercambian información comprensible para máquinas en la Web. **RDF** enfatiza las facilidades para permitir el procesamiento automatizado de recursos web. ” [19]
- **RDFS**: “**RDF** y su esquema de extensión, **RDF Schema Specification (RDFS)** forman las dos capas más bajas de la **Web Semántica**[...] **RDFS** provee un mecanismo estándar para declarar clases y propiedades (globales) así como para definir relaciones entre clases y propiedades, usando sintaxis de **RDF**.” [20]
- **RDFSharp**: “**RDFSharp** es un framework ligero de **C#** que puede ser utilizado para realizar aplicaciones, servicios y sitios web capaces de modelar, almacenar y consultar datos **RDF/SPARQL**.” [21]
- **Scrum**: “Marco de trabajo para la gestión y desarrollo del software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil del software.” [22]
- **SPARQL**: “**SPARQL** es un lenguaje de consulta desarrollado principalmente para consultar grafos **RDF**.” [23]
- **Sprint**: “Período en el cual se lleva el desarrollo de una tarea.” [22]
- **UML**: “El Lenguaje Unificado de Modelado (**UML**) es, tal como su nombre lo indica, un lenguaje de modelado y no un método o un proceso. El **UML** está compuesto por una notación muy específica y por las reglas semánticas relacionadas para la construcción de sistemas de software. El **UML** en sí mismo no prescribe ni aconseja cómo usar esta notación en el proceso de desarrollo o como parte de una metodología de diseño orientada a objetos. ” [24]
- **Vista**: “La vista es responsable de definir la estructura, el diseño y la apariencia de lo que el usuario ve en la pantalla. Idealmente, cada vista se define en **XAML**, con un código subyacente limitado que no contiene la lógica de negocios. Sin embargo, en algunos casos, el código subyacente podría contener lógica de la interfaz de usuario que implementa el comportamiento visual que es difícil de expresar en **XAML**, como animaciones.” [14]
- **Visual Studio**: “Microsoft Visual Studio es un entorno de desarrollo integrado (**IDE**, por sus siglas en inglés) para Windows, Linux y macOS [...] Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la

versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros.” [25]

- **Xamarin:** “Xamarin es una plataforma de código abierto para compilar aplicaciones modernas y con mejor rendimiento para iOS, Android y Windows con .NET. Xamarin es una capa de abstracción que administra la comunicación de código compartido con el código de plataforma subyacente. Xamarin se ejecuta en un entorno administrado que proporciona ventajas como la asignación de memoria y la recolección de elementos no utilizados.” [26]
- **Xamarin.Forms:** “Xamarin.Forms es un marco de interfaz de usuario de código abierto. Xamarin.Forms permite a los desarrolladores compilar aplicaciones en Xamarin.Android, Xamarin.iOS y Windows desde un único código base compartido. Xamarin.Forms permite a los desarrolladores crear interfaces de usuario en XAML con código subyacente en C#. Estas interfaces se representan como controles nativos con mejor rendimiento en cada plataforma.” [27]
- **XML:** “XML is a markup language for documents containing structured information. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays (for example, content in a section heading has a different meaning from content in a footnote, which means something different than content in a figure caption or content in a database table, etc.). Almost all documents have some structure. A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents.” [28]
- **XAML:** “XAML significa **L**enguaje de **M**arcado de **A**plicaciones **E**xtensible. Es el nuevo lenguaje declarativo de Microsoft para definir aplicaciones con interfaces de usuario. XAML provee una sintaxis accesible, extensible y localizable para definir interfaces de usuario separadas de la lógica de la aplicación. similar a la técnica orientada a objeto para desarrollar aplicaciones de múltiples niveles con la arquitectura Modelo-Vista-Controlador.” [29]
- **Web Semántica:** “Extensión de la actual web en la que a la información disponible se le otorga un significado bien definido que permita a los ordenadores y las personas trabajar en cooperación. Se basa en la idea de tener datos en la web definidos y vinculados de modo que puedan usarse para un descubrimiento, automatización y reutilización entre varias aplicaciones.” [30]

## 1.4. Estructuración del documento

El documento que se presenta está estructurado en una colección de capítulos, en los que se describen de manera precisa y detallada todas y cada una de las etapas por



las que ha pasado el proyecto, desde su inicio hasta su conclusión. A continuación se muestra un breve resumen de los contenidos de cada capítulo:

- **Capítulo 1. Introducción.** El capítulo inicial consiste en una introducción al proyecto, explicando los antecedentes a su desarrollo, así como la motivación para ponerlo en práctica, los objetivos que debe cumplir y un glosario de términos para facilitar la comprensión del presente documento.
- **Capítulo 2. Antecedentes.** Tras la introducción, se abordan los fundamentos necesarios para simplificar la lectura de este documento. A su vez, se exponen las diferentes tecnologías de las que se han aplicado para la consecución y puesta en marcha de este proyecto.
- **Capítulo 3. Planificación del proyecto.** Este capítulo engloba toda la información relevante a aspectos de suma importancia tales como la evaluación de riesgos o la planificación temporal del proyecto.
- **Capítulo 4-8. Análisis, Diseño, Codificación y Pruebas.** Este conjunto de capítulos profundizan en los diversos aspectos y fases que comprenden el desarrollo de un producto haciendo uso de una *metodología ágil*. Esto posibilita analizar y enriquecer el producto en su desarrollo, mejorando así el resultado final.
- **Capítulo 9. Manual de Instalación.** Aquí se ha desarrollado un documento con las instrucciones necesarias para realizar la instalación de la aplicación.
- **Capítulo 10. Manual de Usuario.** Se ha redactado un manual de usuario para la aplicación, cuyo objetivo es garantizar un uso eficiente y responsable de la misma por parte de los usuarios.
- **Capítulo 11. Conclusiones.** El último capítulo es un breve repaso sobre el desarrollo del proyecto, que incluye una opinión personal y un apartado de posibles mejoras que se podrían realizar al proyecto en un futuro.



## CAPÍTULO 2

### Antecedentes

#### 2.1. Estado del Arte

Con el objetivo de simplificar el proceso de creación de personajes en los juegos de rol tradicionales, se han originado multitud de aplicaciones con diferentes funciones y finalidades.

Algunas de las aplicaciones son referencias completas de los juegos, que sirven como elementos de consulta accesibles, rápidos y precisos. Un ejemplo de ello es **5e Character**, que es una referencia completa de personajes para *Dragones y Mazmorras*, 5ª Edición.

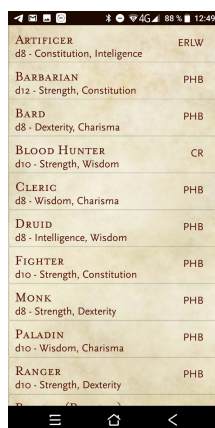


Figura 2.1: **5e Character**: Pantalla de selección de clases

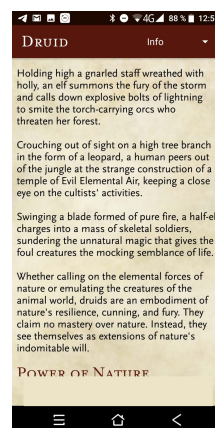


Figura 2.2: **5e Character**: Información de la clase *Druida*

También podemos encontrar aplicaciones como *RPG Simple Dice* que realizan simulaciones de lanzamiento de dados, en caso de que no dispongamos de dados físicos.

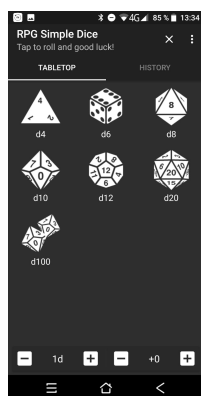


Figura 2.3: *RPG Simple Dice*: Pantalla de selección de dados

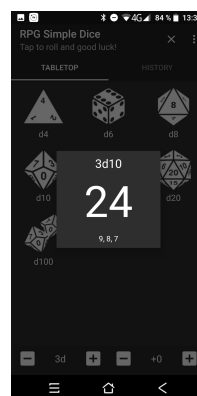


Figura 2.4: *RPG Simple Dice*: Ejemplo de lanzamiento de dados

Otras en cambio, proporcionan algunas herramientas que simplifican algunos cálculos que resultan tediosos durante el transcurso de la partida, como es el caso de *BattleTrack*.

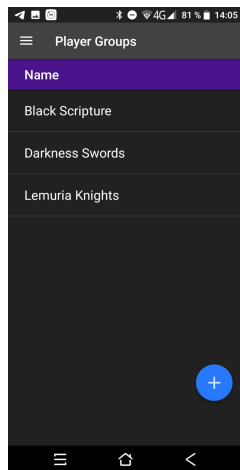


Figura 2.5: *BattleTrack*: Pantalla de selección de grupo

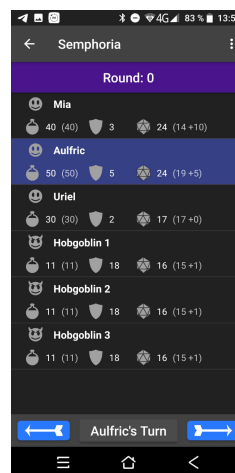


Figura 2.6: *BattleTrack*: Ejemplo de combate

Con el fin de ayudar en la ambientación, aplicaciones como *RPGSound* aportan bibliotecas de sonido que se pueden utilizar durante la representación de la partida para sumirse en ella.



Figura 2.7: *RPGSound*: Menú principal



Figura 2.8: *RPGSound*: Menú de *Ambiente Sostenido*

Finalmente, quedan las aplicaciones conocidas como *generadores de personaje*, que permiten al usuario crear personajes para formar parte de una partida de rol, y acceder a esa información de forma rápida. Un buen ejemplo de esto es *RPG Character Sheet*.

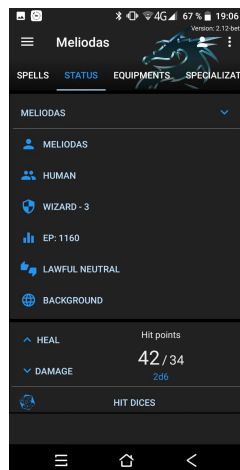


Figura 2.9: *RPG Character Sheet*: Pantalla de *Estado*

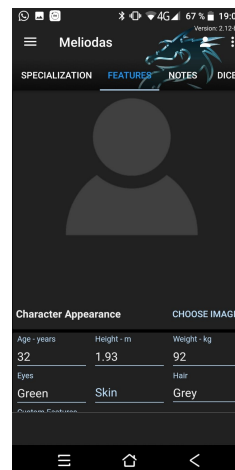


Figura 2.10: *RPG Character Sheet*: Pantalla de *Features*

## 2.2. Crítica al estado del arte

Tal y como se ha comentado previamente, existe un holgado abanico de aplicaciones cuya meta es mejorar y/o simplificar aspectos en lo referente a los juegos de rol, y aunque cumplen con su propósito, a veces no resultan tan efectivas como debieran.

Esto puede deberse a que tras dedicar el tiempo y esfuerzo necesarios para desarrollar la aplicación, el estudio del juego ha aprovechado ese tiempo de producción para revisar el juego y editarlo, realizando modificaciones que provocan que **la aplicación quede obsoleta en poco tiempo**.

Otro inconveniente es que las aplicaciones que requieren mucha información específica, como los generadores de personaje, pueden llegar a **resultar muy complejas**, y al tener una interfaz poco intuitiva, provoca que el usuario no experimentado considere que el esfuerzo que tiene que dedicar para aprender cómo utilizarla es mayor que el de realizar el proceso manualmente.

También existen aplicaciones que no contemplan la reutilización de la información que han producido para efectuar operaciones que mejoren la jugabilidad, por lo que el usuario no le ve provecho a emplear dichas aplicaciones.

## 2.3. Propuesta

Tras analizar el estado del arte, y destacar algunos de los aspectos negativos de éste, lo que prosigue es diseñar una propuesta que permita recoger las mejores ideas de las herramientas actuales, y que además pueda dar solución a los problemas comentados en el apartado 2.2 (*Crítica al estado del arte*).

### 2.3.1. Plataforma

En primer lugar, se debe considerar cuál será la plataforma en la que se implementará la aplicación, puesto que es la base sobre la que se realiza el desarrollo, y que condiciona las herramientas que se pueden utilizar para la elaboración del proyecto.

Como se puede apreciar en la sección 2.1 (*Estado del arte*), muchas de las aplicaciones existentes se desarrollan en entornos móviles, lo que supone que la aplicación pueda ser accesible desde cualquier sitio. Por otro lado, se considera una opción plausible el poder extender la aplicación a otros entornos, tales como sistemas de sobremesa, o el entorno web. Teniendo todo esto en consideración, se ha tomado la decisión de realizar una **aplicación móvil**, haciendo uso de **herramientas de desarrollo multiplataforma**, de manera que en un futuro sea posible extender el proyecto a otras plataformas.

### 2.3.2. Arquitectura

Después de seleccionar la plataforma principal de desarrollo, el siguiente paso es definir la arquitectura de la aplicación, que hace referencia a *“la estructura general*



## 2.4. Metodología

### 2.4.1. Metodologías de desarrollo software

El desarrollo de software está en constante cambio. Esto se debe en parte a la continua aparición de nuevas tecnologías que transforman los modelos teóricos vigentes. Por otro lado, existe una barrera entre las herramientas de desarrollo y la metodología que impide la puesta en práctica de muchos de los modelos propuestos. No es fácil adaptarse de manera adecuada a una metodología de desarrollo de software, lo que resulta en un proceso con posibles demoras. No obstante, *el uso de una metodología adecuada ha probado ser un pilar para el desarrollo de un proyecto de construcción de software* [34]

De aquí es posible extraer dos ideas claras: la primera es que adaptarse a una metodología es una tarea complicada, pero que de lograrse con éxito, son claros los beneficios obtenidos frente a los resultados si no se hubiera realizado dicha adaptación. La segunda es que resulta necesario realizar un estudio para conocer cuáles son las metodologías existentes, cuáles están presentes en el mercado, conocer sus ventajas e inconvenientes, conocer su proceso de implementación y conocer si su alcance está alineado con el objetivo que se desea lograr.

Actualmente existe un gran abanico de metodologías, las cuales se adaptan en mayor o menor medida al tipo de producto que se pretende desarrollar. La gran mayoría de ellas están basadas en alguno de los siguientes modelos de desarrollo de software:

- **Desarrollo en cascada:** *“Enfoque metodológico que ordena rigurosamente las etapas de proceso para el desarrollo de software, de forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior”* [35]

Como expresa Prieto [36]: *“De acuerdo a **Winston Royce**, que propuso dicho modelo, los beneficios de esta metodología surgen cuando no existen fechas inmediatas de implementación, de manera que se dispone del tiempo apropiado para desarrollar cada fase. Cabe destacar que para que este modelo tenga un índice de riesgo bajo, los requerimientos deben ser claros y deben haberse establecido oficialmente en la primera parte del proyecto.”*

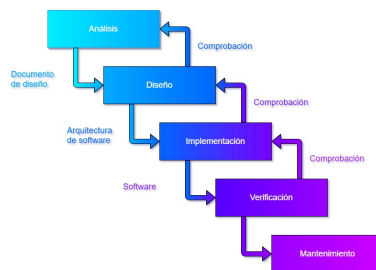


Figura 2.13: Modelo de desarrollo en cascada



- **Desarrollo en espiral:** *“Este modelo, presentado por **Barry Boehm**, permite analizar con mayor profundidad las etapas comprendidas en el desarrollo de un producto software. Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a ninguna prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior. [36]*

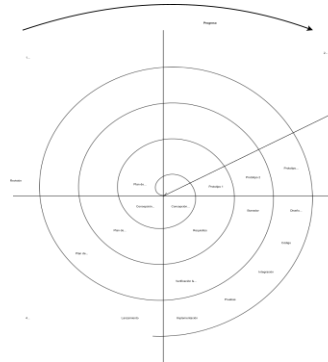


Figura 2.14: Modelo de desarrollo en espiral

- **Desarrollo con prototipos:** *El desarrollo de software basado en prototipos promueve la comunicación entre el cliente y el equipo de programadores, a la vez que logra una rápida integración de cambios y acorta el tiempo de desarrollo del proyecto. [37]* El paradigma de desarrollo basado en prototipos consiste en un proceso iterativo que tiene cinco fases:

1. **Comunicación:** Se indica un conjunto de objetivos que el software debe cumplir.
2. **Plan rápido:** Se propone una estrategia para llevar a cabo el desarrollo
3. **Diseño rápido:** Se realiza el diseño de una interfaz gráfica rápidamente.
4. **Construcción:** Se construye el prototipo del sistema software.
5. **Entrega y retroalimentación:** Se entrega el prototipo y el cliente realiza una retroalimentación al equipo, que da inicio a una nueva iteración que incorpora los ajustes indicados en la información dada por el cliente.

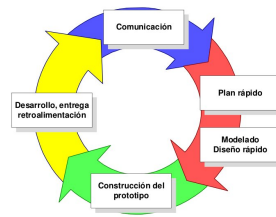


Figura 2.15: Modelo de desarrollo con prototipos [38]

- **Desarrollo incremental:** *Se basa en el desarrollo a partir del incremento de la funcionalidad del programa, se puede considerar un precursor de las modernas metodologías iterativas. El primer incremento es a menudo un desarrollo esencial, apenas con los requisitos básicos, cada incremento representa una entrega escalable. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad.* [39]

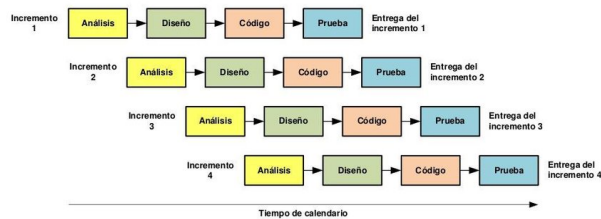


Figura 2.16: Modelo de desarrollo incremental [40]

### 2.4.2. Metodologías Ágiles

En febrero de 2001 nace el término **ágil** aplicado al desarrollo de software, tras una reunión celebrada en Utah (EEUU). El objetivo de la misma fue esbozar valores y principios que deberían permitir desarrollar software de manera rápida, dando respuesta a los cambios surgidos durante el desarrollo del proyecto. Se pretende con esto ofrecer alternativas a los procesos de desarrollo software tradicionales, rígidos y dirigidos por la documentación que se generaba en cada una de las etapas del proceso. [13]

El punto de partida para ello fue el **Manifiesto Ágil** [41]: documento que resume la filosofía ágil, en el cual se valoran los siguientes elementos:

- **El individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas:** Las personas que forman parte del proyecto son el principal factor de éxito de un proyecto, de manera que el entorno influye menos que la bondad del equipo que realiza el desarrollo. Es preferible que el entorno se adapte al equipo.

- **Desarrollo de software útil es mejor que conseguir una buena documentación:** La regla a seguir es *producir sólo documentos que sean necesarios inmediatamente para tomar una decisión importante*.
- **La colaboración del cliente sobre la negociación de un contrato:** Se propone una interacción constante entre cliente y desarrolladores, de manera que esta colaboración permita marcar el ritmo del proyecto y asegure el éxito del mismo.
- **Respuesta rápida a los cambios es mejor que seguir un plan de forma estricta:** La habilidad de responder a los cambios determina el éxito o fracaso del proyecto, de manera que lo más importante de la planificación es su flexibilidad.

Los principios del *Manifiesto Ágil* se basan en estos valores, que a su vez hacen de fundamentos de todas las metodologías ágiles, orientando el desarrollo a la rápida obtención de un producto funcional aunque no tenga todas sus funciones implementadas.

Del modelo de desarrollo ágil se pueden encontrar diversas metodologías, como **eXtreme Programming** [42], **Scrum** [43] o **Crystal Clear** [44]. Para este proyecto se ha tomado la decisión de seguir la metodología **Scrum** ya que debido a la naturaleza y complejidad del mismo, es posible que sea necesario realizar cambios en el planteamiento del proyecto durante el proceso de desarrollo.

### 2.4.3. SCRUM



Figura 2.17: Logo de *Scrum*

#### Introducción

“**SCRUM** es una de las metodologías de desarrollo ágil más reconocidas a nivel mundial, su concepción resulta de unos análisis realizados por **Ikujiro Nonaka** e **Hiroataka Takeuchi** en los años 80, resaltando el trabajo en equipo para el desarrollo de productos y la autonomía que estos deben tener (Takeuchi & Nonaka, 1986). Su diseño se debe a que en los años 90, **Jeff Sutherland** y **Ken Schwaber** formalizaron un marco de trabajo y unas reglas aplicadas particularmente al desarrollo de software de productos complejos.” [45]

### Características

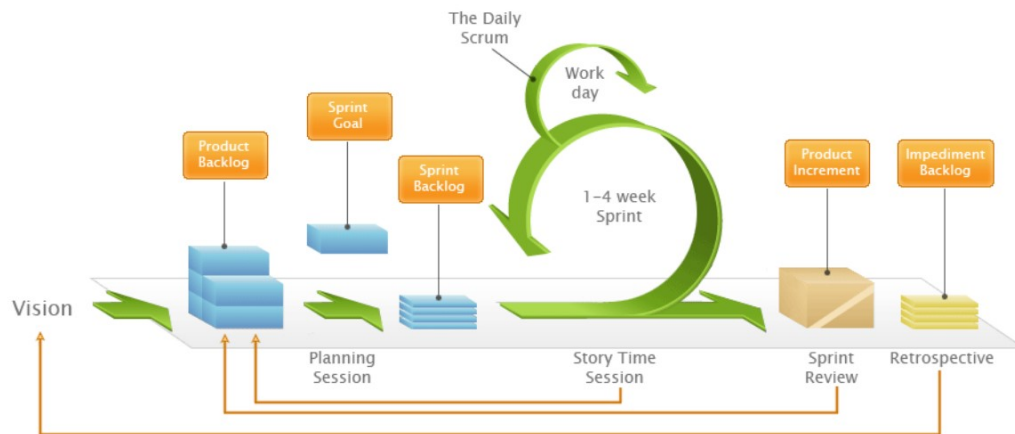
A continuación se muestra una serie de características que deben tener todos los procesos que se introducen al marco de la metodología *Scrum*:

- El desarrollo incremental de los requisitos en bloques temporales cortos y fijos.
- Se da prioridad los requisitos más valorados por el cliente.
- El equipo se sincroniza diariamente y se realizan las adaptaciones necesarias.
- Tras cada iteración se muestra el resultado real al cliente, para que tome las decisiones necesarias en relación al resultado observado.
- Se le da al equipo la autoridad necesaria para poder cumplir los requisitos.
- Fijar tiempos máximos para lograr objetivos.
- Equipos de trabajo pequeños (de 5 a 9 personas).

### Ciclo de desarrollo

Para entender el ciclo de desarrollo de *Scrum* es necesario conocer las fases que lo definen:

1. **Planificación:** Reunión de los involucrados en la que se definen los requisitos prioritarios para la iteración actual y se elabora una lista de tareas necesarias para lograr los requisitos previamente seleccionados.
2. **Scrum diario:** Evento del equipo de desarrollo de quince minutos, que se realiza diariamente durante la ejecución de la iteración para explicar lo que se ha alcanzado desde la última reunión, lo que se hará antes de la siguiente y los obstáculos que se han presentado.
3. **Revisión:** El equipo presenta al cliente los requisitos completados en la iteración. En función de los resultados mostrados y de los cambios habidos en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, replanificando el proyecto.
4. **Retrospectiva:** El equipo analiza cómo ha sido su manera de trabajar y qué problemas podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad.

Figura 2.18: Ciclo de desarrollo de *Scrum*

## Roles

Los roles presentes en *Scrum* son los siguientes:

- **Product Owner:** Tiene la responsabilidad de decidir qué trabajo necesita hacerse, y maximizar el valor del proyecto o producto que se esté llevando a cabo. Para ello debe tener las siguientes cualidades:
  1. *Saber gestionar prioridades:* Es responsable de gestionar los presupuestos, de contratar al equipo de desarrollo y de explicar cuál es el valor que produce el producto en el que está invirtiendo.
  2. *Toma de decisiones:* Debe ser capaz de tomar decisiones por su cuenta.
  3. *Coordinador:* Tiene que poder medir el valor generado y utilizar la flexibilidad de entregar cada *sprint* para incrementar ese valor.
- **Scrum Master:** Persona que ayuda al equipo y a la organización a optimizar el uso de la metodología. Traslada la visión del proyecto al equipo, y elimina los obstáculos que impiden que el equipo alcance el objetivo del *sprint*.
- **Development Team:** Grupo de profesionales con los conocimientos técnicos necesarios y que desarrollan el proyecto de manera conjunta llevando a cabo los requisitos a los que se comprometen al inicio de cada *sprint*.

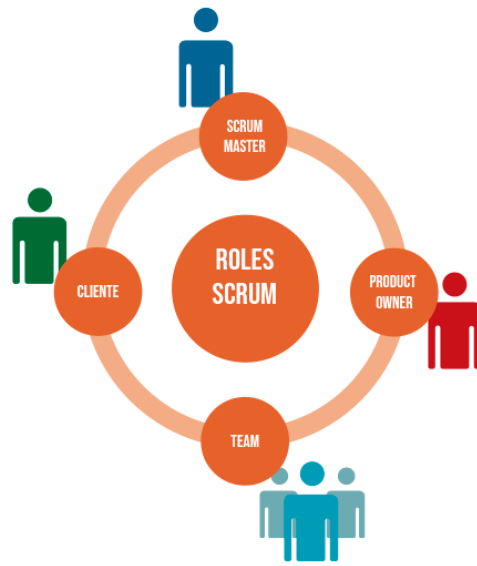


Figura 2.19: Roles en *Scrum*

### 3.1. Planificación del proyecto

En este apartado se van a establecer el orden de las tareas que se deben realizar y la estimación del tiempo que se considera necesario para la elaboración de este proyecto. Finalmente se mostrará un análisis de la desviación temporal que indicará la diferencia entre los tiempos estimados con los empleados realmente.

#### 3.1.1. Objetivo inicial

El objetivo inicial de este proyecto era desarrollar una aplicación móvil para la generación y manipulación de información de personajes de juegos de rol, extensible a cualquier juego de rol cuya información esté almacenada en la aplicación.

#### 3.1.2. División del trabajo

La estructura de desglose del trabajo está dispuesta a continuación:

- **Planificación:** Inicialmente, se realiza una planificación *grosso modo*, que facilita una visión global de los hitos que se deben conseguir, y los pasos adecuados para alcanzarlos.
- **Conocimientos:** Al inicio del proyecto, el equipo de desarrollo de este proyecto no estaba relacionado con algunas de las herramientas que han sido utilizadas. Por otra parte, otras herramientas sí eran conocidas por el equipo, pero no con la profundidad que ha exigido el proyecto. Es precisamente esto lo que ha motivado al equipo a recuperar la información que ya obtenía sobre las herramientas y ampliarla mediante la búsqueda y lectura de documentación que pudiera beneficiar en el desarrollo.

- **Desarrollo:** Tras un breve período para ponerse al día con los conocimientos básicos para comenzar el proceso de desarrollo, se realizarán los diferentes *sprints* con las siguientes tareas:
  - **Planificación:** Cada *sprint* tendrá su planificación específica seleccionando los requisitos *necesarios* para su desarrollo con el cliente. Como este proyecto *no tiene un cliente claramente definido, se ha entrevistado a potenciales clientes* de la aplicación para poder conocer las necesidades que consideran oportunas como posibles usuarios de la aplicación. Posteriormente, se procede a la planificación del *sprint* elaborando una lista de tareas que deben llevarse a cabo para considerar que se ha finalizado el *sprint* de manera satisfactoria.
  - **Análisis**
  - **Diseño**
  - **Implementación:** Se realizarán pruebas de funcionalidad de manera simultánea a la codificación del proyecto, pudiendo comprobar que las funciones finalizadas cumplen su función de forma correcta y completa, sin errores.
  - **Documentación:** Al finalizar cada *sprint* se actualizará la documentación del proyecto.
- **Memoria:** Aunque la memoria se ha realizado de forma conjunta con el proyecto desde su inicio, algunos aspectos como los manuales, han requerido que otros aspectos del proyecto estuvieran finalizados para poder realizarse.
- **Presentación:** Elaboración de una presentación para la defensa del proyecto ante el tribunal.

### 3.1.3. Identificación de *sprints* y estimación de tiempos

El equipo de desarrollo procede, en este punto de la memoria, a describir las iteraciones que se han dado a lo largo de la elaboración del proyecto.

1. ***Sprint 0: Planificación inicial:*** Forman parte de esta iteración tanto las reuniones con los clientes potenciales, como el estudio y elección de las tecnologías a emplear.
2. ***Sprint 1: Preparación del equipo:*** Durante el primer *sprint* se pondrá a punto el entorno de trabajo, instalando las aplicaciones necesarias para el desarrollo. También se procederá a la instalación de aplicaciones que, no siendo necesarias, serán de ayuda durante el proceso.
3. ***Sprint 2: Creación de la ontología:*** Este *sprint* es el más extenso de todos, debido a que abarca desde el esbozo inicial del modelo conceptual de la ontología, que debe contener toda la información de un juego de rol, hasta el momento en que ésta queda totalmente operativa. Como el sistema debe poder procesar otras



ontologías además de la elaborada por el equipo que lleva a cabo este proyecto, no se ha considerado finalizado el *sprint* hasta que se han realizado todas las modificaciones necesarias, tanto en la lógica de negocio de la aplicación como en la ontología, para que sea posible procesar y disponer correctamente de toda la información de la ontología.

4. ***Sprint 3: Creación de personajes:*** El objetivo principal de la aplicación se abordará en este *sprint*, ya que requiere que la lógica de negocio de la aplicación acceda a la ontología de un juego de rol, y extraiga información de esta para mostrar un formulario de creación de personajes paso a paso, personalizado para el juego de rol en cuestión, y que se genere un fichero con la información del personaje que ha sido seleccionada por el usuario en el formulario.
5. ***Sprint 4: Visualización, modificación y eliminación de personaje:*** La aplicación tiene que poder mostrar la información del personaje creado previamente, y permitir al usuario editar parte de la misma si así lo desea.
6. ***Sprint 5: Validador de ontologías:*** El proyecto dispondrá de una aplicación de consola que permita comprobar a un desarrollador si la ontología del juego que está elaborando podría funcionar en la aplicación principal del proyecto.
7. ***Sprint 6: Cálculo de habilidades de personaje:*** La aplicación podrá realizar cálculos con la información almacenada en la hoja de personaje. Este campo se considera opcional, debido a que al tener una fecha límite y un proceso complejo, es posible que no se pueda alcanzar este objetivo, el cual quedaría pendiente como trabajo futuro.

#### 3.1.4. Resumen de la planificación del proyecto

A continuación se muestra un resumen de la planificación previamente descrita en un diagrama de Gantt. Como el diagrama es demasiado extenso como para ser mostrado en una única página, se va a separar por *sprint*, de manera que cada hoja pueda mostrar *sprints* completos.

Los tiempos estimados han estado expuestos a imprevistos, ya fueran de tipo técnicos o tecnológicos:

- ***Librería dotNetRDF no permite trabajar con propiedades del formato OWL:*** La primera librería que el equipo encontró para poder trabajar con el formato *RDF* fue **dotNetRDF**. El problema sucedió cuando, una vez con el proyecto encaminado, resulta que la librería no es compatibles con las propiedades del formato *OWL*, de manera que no era posible realizar tuplas *sujeto-predicado-objeto*, y por tanto, no es una librería compatible con la aplicación que el equipo tiene como objetivo elaborar. Esto supuso un retraso de al menos un mes de trabajo para el equipo, ya que no sólo se había perdido el tiempo dedicado en aprender a utilizar la librería, sino que era necesario buscar otra que permitiera realizar aquello que

la primera no podía, lo que podría hacer retroceder todo el proyecto, ya que de no encontrar una, habría que cambiar las herramientas base para el desarrollo. Después de buscar bastante, el equipo encontró la librería **RDFSharp**, que aún estando en desarrollo, sí es compatible con las propiedades necesarias para realizar tuplas *sujeto-predicado-objeto*, y por tanto, permite trabajar con el formato *OWL*, siendo así compatible con el producto final.

- *Error en el SO causa error en repositorios del proyecto*: Durante una de las actualizaciones del proyecto, el sistema operativo del ordenador que realizaba la actualización, causó la corrupción de gran parte de los ficheros del proyecto, resultando afectados los repositorios local y remoto del proyecto. El equipo entonces hizo uso de una copia de seguridad almacenada en un dispositivo externo, para recuperar la mayor cantidad de trabajo posible, pero al no estar actualizado completamente, fue necesario invertir entre 7 y 8 horas para recuperar el trabajo perdido.

## 3.2. Organización

### 3.2.1. Agentes involucrados

Las personas involucradas en la elaboración de este proyecto son:

- Los directores (o tutores) del proyecto: La labor de los directores del proyecto consiste en revisar el estado del proyecto durante todo el proceso de desarrollo y contribuir con las ideas que estimen oportunas para perfeccionarlo.
- Los clientes, que son los potenciales usuarios finales de la aplicación. Ellos indican los requerimientos que consideran necesarios para que el producto final sea atractivo para el público objetivo (*target*) del producto.
- El equipo de desarrollo, formado en este caso por una persona, que se encarga de la elaboración del proyecto.

### 3.2.2. Recursos utilizados

Los recursos que han sido utilizados en el desarrollo de la aplicación son:

- Un ordenador personal, en el que se ha constituido el entorno de trabajo.
- Visual Studio 2019 Community, como entorno de desarrollo para la aplicación.
- Protégé, un editor de ontologías de código abierto
- Xamarin, como plataforma para el desarrollo de aplicaciones multiplataforma.
- LaTeX, como herramienta para la elaboración de la memoria.
- Dispositivo móvil con *Android*, para probar que la aplicación funciona en la plataforma deseada.

Como todos los recursos software previamente citados son gratuitos, su uso no supone coste alguno en licencias de aplicaciones. No se tendrá en cuenta el precio del ordenador personal en los costes del proyecto, ya que el equipo de desarrollo disponía de uno con las capacidades necesarias para poner en funcionamiento los recursos software.

### 3.3. Costes

#### 3.3.1. Costes humanos

Para calcular los costes humanos, debemos tener alguna referencia salarial de un desarrollador, utilizándolo como base para realizar los cálculos en tiempo de desarrollo. Hemos considerado como una referencia aceptable las *tablas salariales para personal investigador encargado del desarrollo de proyectos de investigación científica o técnica a través de un contrato por obra y servicio*. Estas tablas indican que el *coste anual total* de un ingeniero son 27.664,14 €, que al dividirse en 14 pagas (12 meses y 2 pagas extra), resulta que el coste mensual sería de 2.305,35 €.

La duración del proyecto ha sido de “introducir número aquí” días, resultando en 12 meses aproximadamente. A continuación se muestra una tabla comparativa entre los costes estimados y los costes reales:

	Tiempo (días)	Coste
Estimado		
Real		

Cuadro 3.1: Comparativa entre coste humano estimado y coste humano real

#### 3.3.2. Costes materiales

Como los dispositivos empleados son propiedad del equipo de desarrollo y el software utilizado en el proceso de elaboración del proyecto es gratuito, se asume un coste de 0 € en cuanto a costes materiales se refiere.

### 3.4. Evaluación y gestión de riesgos

#### 3.4.1. ¿Qué es un riesgo?

Se considera **riesgo** en un proyecto de implantación de software cualquier eventualidad que pueda suponer una desviación del plan previsto y que posibilite el fracaso del proyecto.

#### 3.4.2. Gestión de riesgos

La gestión de riesgos permite al equipo de desarrollo definir de forma lógica una serie de actividades para analizar los riesgos que se puedan presentar a lo largo del ciclo

de vida del proyecto, calcular su exposición y priorizarlos, de manera que se puedan establecer estrategias de control, resolución y supervisión de los mismos.

### 3.4.3. Identificación de riesgos

En primer lugar, se procederá a realizar una lista que identificará los posibles riesgos que puedan surgir a lo largo del proyecto:

- **Riesgos de planificación:** La planificación mal realizada puede retrasar enormemente el proyecto, pudiendo resultar en el fracaso del mismo. Posibles motivos para esto son una mala estimación de los tiempos de desarrollo, imposición de plazos por parte del cliente o una planificación optimista.
- **Riesgos de organización y gestión del proyecto:** Es posible que una parte del personal abandone el proyecto, lo que puede derivarse en el fracaso del proyecto si no se encuentra un sustituto adecuado rápidamente.
- **Riesgos de infraestructura hardware y software:** Es posible que surjan problemas con las herramientas, tales como incompatibilidad entre herramientas o incompatibilidad entre herramientas y dispositivos.
- **Riesgos de requisitos:** Es posible que los requisitos generen contratiempos, por ser añadidos constantemente o por modificar drásticamente lo previamente desarrollado.
- **Riesgos de diseño e implementación:** Es posible la falta de un diseño adecuado provoque obstáculos, ya sea por no realizar el diseño y pasar directamente a la implementación, porque el diseño esté demasiado simplificado para la complejidad del proyecto, o por tratar de implementar funciones no soportadas por las herramientas de trabajo.

### 3.4.4. Reducción de riesgos

En esta sección se van a describir las medidas que el equipo ha considerado oportunas para minimizar la aparición de los riesgos previamente indicados.

- **Riesgos de planificación:** Se realizará una planificación objetiva, considerando que es susceptible a tener problemas, que en el caso de darse, se estudiarán para modificar la planificación existente de manera que se puedan sortear minimizando su efecto en el desarrollo del proyecto.
- **Riesgos de organización y gestión del proyecto:** Es posible que una parte del personal abandone el proyecto, lo que puede derivarse en el fracaso del proyecto si no se encuentra un sustituto adecuado rápidamente.

- **Riesgos de infraestructura hardware y software:** Se hará un estudio previo de las herramientas para comprobar que es posible trabajar con ellas sin problema alguno. También se realizarán copias de seguridad periódicamente, de manera que de darse un riesgo imprevisible como que el dispositivo de almacenamiento sufra un error y la información sea irrecuperable, haya alguna manera de obtener una versión anterior del proyecto, lo más actualizada posible, desde la que se pueda partir.
- **Riesgos de requisitos:** Se realizará una recolección exhaustiva de requisitos del proyecto al comienzo, para evitar en la medida de lo posible la aparición de requisitos extra durante el desarrollo del proyecto. Todos los requisitos quedarán documentados.
- **Riesgos de diseño e implementación:** En cada etapa del desarrollo, se comprobará de forma periódica si el diseño puede obviar alguno de los requisitos establecidos.



# Parte II

## Desarrollo





---

Requisitos del Sistema

---

### 4.1. Situación Actual

Con el objetivo de simplificar el proceso de creación de personajes en los juegos de rol tradicionales, se han originado multitud de aplicaciones con diferentes funciones y finalidades.

Algunas de las aplicaciones son referencias completas de los juegos, que sirven como elementos de consulta accesibles, rápidos y precisos. Un ejemplo de ello es *5e Character*, que es una referencia completa de personajes para *Dragones y Mazmorras*, 5ª Edición.

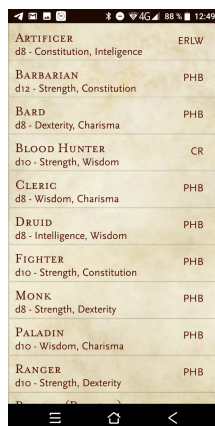


Figura 4.1: *5e Character*: Pantalla de selección de clases

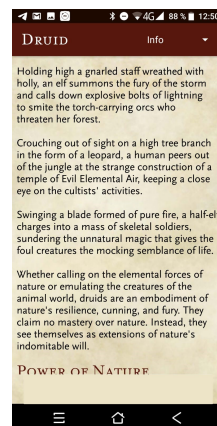


Figura 4.2: *5e Character*: Información de la clase *Druida*

También podemos encontrar aplicaciones como *RPG Simple Dice* que realizan

simulaciones de lanzamiento de dados, en caso de que no dispongamos de dados físicos.

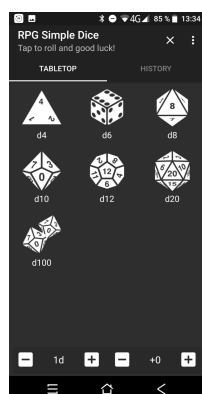


Figura 4.3: *RPG Simple Dice*: Pantalla de selección de dados

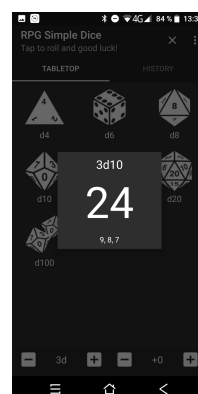


Figura 4.4: *RPG Simple Dice*: Ejemplo de lanzamiento de dados

Otras en cambio, proporcionan algunas herramientas que simplifican algunos cálculos que resultan tediosos durante el transcurso de la partida, como es el caso de *BattleTrack*.

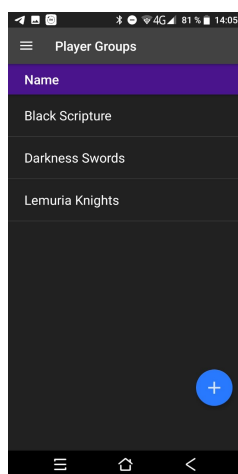


Figura 4.5: *BattleTrack*: Pantalla de selección de grupo

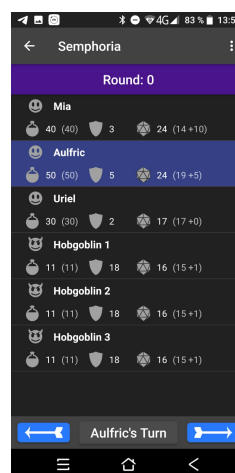


Figura 4.6: *BattleTrack*: Ejemplo de combate

Con el fin de ayudar en la ambientación, aplicaciones como *RPGSound* aportan

bibliotecas de sonido que se pueden utilizar durante la representación de la partida para sumirse en ella.



Figura 4.7: *RPGSound*:  
Menú principal

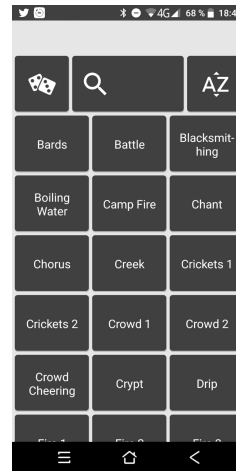


Figura 4.8: *RPGSound*:  
Menú de *Ambiente Sostenido*

Finalmente, quedan las aplicaciones conocidas como *generadores de personaje*, que permiten al usuario crear personajes para formar parte de una partida de rol, y acceder a esa información de forma rápida. Un buen ejemplo de esto es *RPG Character Sheet*.

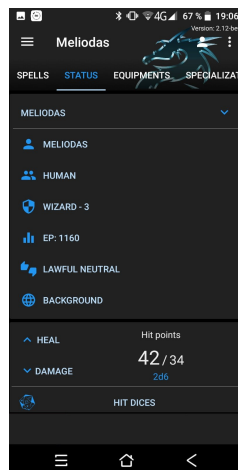


Figura 4.9: *RPG Character Sheet*: Pantalla de  
*Estado*

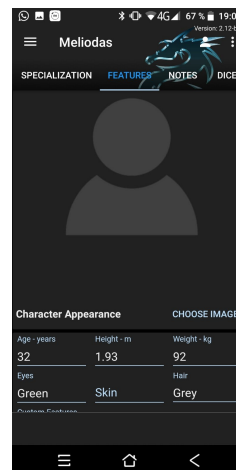


Figura 4.10: *RPG Character Sheet*: Pantalla de  
*Features*

#### 4.1.1. Crítica al estado del arte

Tal y como se ha comentado previamente, existe un holgado abanico de aplicaciones cuya meta es mejorar y/o simplificar aspectos en lo referente a los juegos de rol, y aunque cumplen con su propósito, a veces no resultan tan efectivas como debieran.

Esto puede deberse a que tras dedicar el tiempo y esfuerzo necesarios para desarrollar la aplicación, el estudio del juego ha aprovechado ese tiempo de producción para revisar el juego y editarlo, realizando modificaciones que provocan que *la aplicación quede obsoleta en poco tiempo*.

Otro inconveniente es que las aplicaciones que requieren mucha información específica, como los generadores de personaje, pueden llegar a *resultar muy complejas*, provocando que el usuario considere que el esfuerzo que tiene que dedicar para aprender cómo utilizarla es mayor que el de realizar el proceso manualmente.

También existen aplicaciones que no contemplan la reutilización de la información que han producido para efectuar operaciones que mejoren la jugabilidad, por lo que el usuario no le ve provecho a emplear dichas aplicaciones.

## 4.2. Objetivos del Sistema

El sistema va a tener tres finalidades claramente diferenciadas:

- **Selección de juego:** La aplicación podrá dar acceso a varios juegos, de manera que se pueda alternar entre éstos, permitiendo utilizar el mismo mecanismo para el catálogo de juegos disponible.
- **Creación/Modificación de personaje:** La aplicación permitirá al usuario seleccionar la información necesaria para la creación de un personaje a su gusto mediante un proceso guiado paso a paso. En caso de que el personaje ya exista, permitirá al usuario visualizar y realizar modificaciones a la información mostrada.
- **Automatización del proceso de cálculo de habilidades:** La aplicación facilitará al usuario una interfaz en la que, al indicar la habilidad que desea utilizar, y el resultado de su lanzamiento de dados, se devuelva el resultado total que se debe aplicar en el combate. También podrá calcular el resultado de tiradas enfrentadas.

Además, la aplicación deberá hacer muestra de las siguientes cualidades:

- **Generalidad:** La aplicación no estará directamente vinculada a la información específica de un juego, de forma que sea posible procesar diferentes bancos de datos, y por tanto, se pueda utilizar la misma aplicación para varias versiones distintas del mismo juego, o incluso para juegos completamente diferentes.
- **Sencillez:** La aplicación dispondrá de una interfaz simple y agradable, que permita al usuario hacer uso de sus funciones de forma asequible, sea cual fuere la complejidad del juego seleccionado.

En la presente sección vamos a proceder a realizar un análisis de requisitos del sistema, que recoge y describe el conjunto de requisitos específicos del sistema que se va a desarrollar.

En primera instancia, se presentarán los requisitos agrupados en conjuntos funcionales del sistema. Posteriormente, se describirán los casos de uso en el próximo capítulo.

Para ello, se hará una diferenciación entre *requisitos funcionales*, que son aquellos que detallan la funcionalidad del sistema, y *requisitos no funcionales*, que refieren a otros aspectos del software que deben ser satisfechos.

#### 4.2.1. Requisitos funcionales

Un requisito funcional especifica una función concreta del sistema o de alguno de sus componentes. A continuación se muestran los requisitos funcionales, estructurados según el módulo del sistema al que refieren.

El sistema precisa de una lógica estructurada y compleja que permita procesar información de diferentes fuentes, de manera que los procesos del sistema se adecuen a su contenido.

- **OBJ-001:** El usuario podrá seleccionar un juego concreto (juego activo) para poder acceder a la información relacionada con el mismo.
- **OBJ-002:** El usuario podrá crear un personaje para el juego activo, mediante un proceso guiado paso a paso
- **OBJ-003:** El usuario podrá seleccionar un personaje (personaje activo) de todos los existentes para el juego activo
- **OBJ-004:** El usuario podrá visualizar la información del personaje seleccionado del juego activo.
- **OBJ-005:** El usuario podrá eliminar un personaje ya creado del juego activo.
- **OBJ-006:** El usuario podrá realizar cálculos con los valores de las habilidades del personaje activo.

#### 4.2.2. Requisitos no funcionales

Un requisito no funcional es una propiedad o cualidad que no forma parte de los fundamentos del sistema, pero es necesario para que el producto cumpla con su cometido apropiadamente.

Para la declaración de requisitos no funcionales, se establecerán como base los requisitos indicados en las normas *IEEE Std. 830* e *ISO/IEC 25010 (SQuaRE)*:

- **Adecuación funcional:** La aplicación debe cumplir con todos los requisitos necesarios, de manera que sea completo y correcto funcionalmente.

- **Seguridad:** El sistema no requiere asegurar la información que procesa, debido a que no contiene información sensible del usuario en ningún momento, ni realiza conexión externa alguna para obtener información.
- **Compatibilidad:** La aplicación deberá ser compatible con los ficheros que contienen la información de los juegos que formarán parte del sistema.
- **Usabilidad:** El sistema debe disponer de una interfaz de usuario intuitiva y fácil de manejar, de manera que pueda ser utilizado por usuarios sin conocimientos técnicos ni avanzados de informática. La curva de aprendizaje deberá ser lo más reducida posible, de manera que personas de cualquier ámbito puedan hacer uso del mismo.
- **Fiabilidad:** La aplicación deberá estar libre de errores que influyan negativamente en su uso normal. Debido a que la aplicación depende de información incluida por terceros, será necesario comprobar que dicha información es compatible con la aplicación.
- **Eficiencia:** El sistema debe evitar en la medida de lo posible el uso de información redundante para poder asegurar su funcionamiento cuando se introduzcan juegos de alta complejidad que requieran un elevado uso de recursos.
- **Mantenibilidad:** Este apartado representa la capacidad del producto software para ser modificado efectiva y eficientemente. Esto será posible debido al desarrollo de código limpio y bien documentado, al diseño y la implementación modular del mismo. Se plantea el uso de patrones de arquitectura de software, tales como MVVM.
- **Portabilidad:** El sistema está diseñado para su uso en dispositivos móviles, aunque no se descarta una futura ampliación para introducirlo en otro tipo de dispositivos. La implementación está realizada únicamente para sistemas *Android*, ya que no se dispone de las herramientas necesarias para el despliegue en *Mac OS*.

## 4.3. Alternativas de Solución

En esta sección, se ofrece un estudio del arte de las diferentes alternativas tecnológicas que permitan satisfacer los requerimientos del sistema, para optar por una de las opciones planteadas, que será dispuesta como base para el software a desarrollar.

Con este motivo, hemos optado por recoger algunas de las tecnologías existentes para realizar desarrollo de aplicaciones móviles.

### 4.3.1. Android Studio

Android Studio es el entorno de desarrollo integrado (*IDE*) oficial para el desarrollo de apps para Android, basado en *IntelliJ IDEA* de *JetBrains* y ha sido publicado

gratuitamente a través de la Licencia Apache 2.0. Disponible para las plataformas *Windows*, *macOS* y *GNU/Linux*. Basado en el lenguaje *Java*, no tiene herramientas nativas para trabajar directamente con *RDF* y *OWL*. Para suplir este obstáculo, haríamos uso del framework libre *Apache Jena*, cuya API permite trabajar con *RDF*, consiguiendo vincular el desarrollo en aplicaciones móviles con el uso de ontologías.



Figura 4.11: Logo de *Android Studio*



Figura 4.12: Logo de *Apache Jena*

#### 4.3.2. React Native

React Native es un *framework* para el desarrollo de aplicaciones móviles de código abierto desarrollado por *Facebook*. Se utiliza para desarrollar aplicaciones para *Android*, *iOS*, *Web* y *UWP* permitiendo a los desarrolladores usar *React* con funcionalidades nativas de las plataformas. Al igual que *Android Studio*, React Native no tiene herramientas nativas para el desarrollo de ontologías, de manera que haríamos uso de bibliotecas tales como *rdflib.js* para poder proceder al tratamiento de las ontologías.

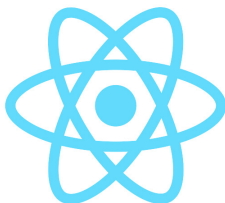


Figura 4.13: Logo de *React Native*



Figura 4.14: Logo de *rdflib.js*

### 4.3.3. Xamarin

Xamarin es una plataforma de código abierto para compilar aplicaciones modernas y con mejor rendimiento para *iOS*, *Android* y *Windows* con *.NET*. Xamarin es una capa de abstracción que administra la comunicación de código compartido con el código de plataforma subyacente. Xamarin dispone de una biblioteca conocida como *RDFSharp*, que permite generar y procesar ontologías en formatos *RDF* y *OWL*.



Figura 4.15: Logo de *Xamarin*

## 4.4. Solución propuesta

Tras considerar las opciones planteadas en el apartado anterior, se ha considerado descartar *Android Studio* en primer lugar, ya que sólo permite el desarrollo en *Android*, mientras que las otras soluciones permiten realizar el desarrollo en varias plataformas.

Una vez desechada una de las opciones, se ha comprobado que las soluciones restantes son compatibles con el proyecto, y prácticamente generan el mismo resultado. Por ello, en vez de considerar las plataformas, se ha realizado una comparación en función al lenguaje de programación con el que se trabaja en cada una de ellas, que son **JavaScript** en *React Native*, y **C#** para *Xamarin*. Esta comparativa se realizará en forma de tabla.

Cuadro 4.1: Comparativa entre *JavaScript* y *C#*

	<i>JavaScript</i>	<i>C#</i>
<b>Tipo de Lenguaje</b>	Scripting	Orientado a Objetos
<b>Tipado</b>	Débil	Fuerte
<b>Detección de errores</b>	Ejecución	Compilación y ejecución
<b>Compilación</b>	No	Sí
<b>Mantenibilidad</b>	Complejo	Sencillo
<b>Soporte de IDE</b>	No	Microsoft Visual Studio
<b>Sintaxis</b>	OBSL	OOP

En la comparativa se puede observar que JavaScript es un lenguaje de scripting débilmente tipado que no requiere ser compilado, pero resulta difícil de mantener en sistemas complejos. Por otro lado, C# es un lenguaje orientado a objetos fuertemente tipado que requiere ser compilado, que permite una mayor facilidad a la hora de mantener el código en sistemas de alta complejidad.



Dado que la aplicación a desarrollar es de una complejidad considerable y que puede ser propensa a generar errores, uno de los factores que más peso ha tenido ha sido de que la mejor herramienta para el desarrollo del presente proyecto es ***Xamarin*** con la biblioteca ***RDFSharp***.



## CAPÍTULO 5

---

### Análisis del Sistema

---



## CAPÍTULO 6

---

Diseño del Sistema

---



## CAPÍTULO 7

---

### Construcción del Sistema

---





## CAPÍTULO 8

---

### Pruebas del Sistema

---



Parte III

Epílogo



## CAPÍTULO 9

---

Manual de Implantación y Explotación

---



## CAPÍTULO 10

---

Manual de Usuario

---





## CAPÍTULO 11

---

Manual de Implantación y Explotación

---



## CAPÍTULO 12

---

Conclusiones

---



## CAPÍTULO 13

---

### Bibliografía

---

- [1] E Adams. «Fundamentals of game design». En: *Choice Reviews Online* 47.08 (2010), págs. 47–4462–47-4462. ISSN: 0009-4978. DOI: 10.5860/choice.47-4462. URL: <https://books.google.es/books?hl=es%7B%5C%7Dlr=%7B%5C%7Ddid=1vQqAwAAQBAJ%7B%5C%7Ddoi=fnd%7B%5C%7Dpg=PT5%7B%5C%7Ddq=Fundamentals+of+Role-Playing+Game+Design%7B%5C%7Dots=4vPCfiy19v%7B%5C%7Dsig=G74otw1TBCWnxm8J4Y6YrqbmtC0>.
- [2] Pedro Ramos Villagrasa y Manuel Sueiro Abad. «Personalidad y elección de personaje en los juegos de rol: dime quién eres y te diré quién prefieres ser». En: *Education in the knowledge society (EKS)* 11.3 (2010), págs. 8-26. ISSN: 1138-9737. URL: <https://www.redalyc.org/pdf/2010/201021093002.pdf>.
- [3] Susanne Isaksson. «Character Creation Processes in MMORPGs -A qualitative study of determining important factors». En: (2012). URL: <https://www.diva-portal.org/smash/record.jsf?pid=diva2:543179>.
- [4] Lajos Egri. *The Art of Dramatic Writing: Its Basis in the Creative Interpretation of Human Motives. With an Introd. by Gilbert Miller*. 1960.
- [5] Petri Lankoski, Satu Heliö e Inger Ekman. «Characters in Computer Games: Toward Understanding Interpretation and Design». En: *DiGRA '03 - Proceedings of the 2003 DiGRA International Conference: Level Up* (2003), págs. 1-12. URL: <http://www.digra.org/wp-content/uploads/digital-library/05087.10012.pdf>.
- [6] Anders Tychsen, Michael Hitchens y Thea Brolund. «Character play - The use of game characters in multi-player role-playing games across platforms». En: *Computers in Entertainment* 6.2 (jul. de 2008). ISSN: 15443574. DOI: 10.1145/1371216.1371225. URL: <http://doi.acm.org/10.1145/1371216.1371225>.

- 
- [7] Varios Autores. *Generadores de personajes online — Ars Rolica*. URL: <https://arsrolica.wordpress.com/2018/12/19/generadores-de-personajes-online/> (visitado 26-06-2020).
  - [8] Julián Pérez Porto y María Merino. *Definición de Android - Qué es, Significado y Concepto*. 2015. URL: <https://definicion.de/android/> (visitado 29-06-2020).
  - [9] Peter Golde Anders Hejlsberg, Scott Wiltamuth. *The C# Programming Language - Anders Hejlsberg, Scott Wiltamuth, Peter Golde - Google Libros*. 2003. URL: <https://books.google.es/books?hl=es%7B%5C%7Dlr=%7B%5C%7Ddid=ICe7ea4RscUC%7B%5C%7Ddoi=fnd%7B%5C%7Dpg=PT14%7B%5C%7Ddq=c%7B%5C%7D23+definition%7B%5C%7Ddots=YbRMHVEciW%7B%5C%7Dsig=ubbZnQxhuG1kxQd6xcLQjVS0b78%7B%5C%7Dv=snippet%7B%5C%7Dq=c%7B%5C%7D23%20%7B%5C%7Ddf=false%20https://books.google.es/books?hl=es%7B%5C%7Dlr=%7B%5C%7Ddid=6L1Rm031qCkC%7B%5C%7Ddoi=fnd%7B%5C%7Dpg=PA3%7B%5C%7Ddq=c%7B%5C%7D23%7B%5C%7Ddots=5w0X7GQ1yK%7B%5C%7Dsig=JKvKouviIAS1HH-RJZUG-056sEg%7B%5C%7Dv=onepage%7B%5C%7Dq=c%7B%5C%7D23%7B%5C%7Ddf=false> (visitado 29-06-2020).
  - [10] Matthew McCullough Jon Loeliger. *Version Control with Git: Powerful tools and techniques for collaborative ... - Jon Loeliger, Matthew McCullough - Google Libros*. 2012. URL: <https://books.google.es/books?hl=es%7B%5C%7Dlr=%7B%5C%7Ddid=aM7-0xo3qdQC%7B%5C%7Ddoi=fnd%7B%5C%7Dpg=PR3%7B%5C%7Ddq=git+svn%7B%5C%7Ddots=39CfLEWhxa%7B%5C%7Dsig=H0H3ujbnqorC25sxQaZfTNZOP5E%7B%5C%7Dv=onepage%7B%5C%7Dq=git%20svn%7B%5C%7Ddf=false%20https://books.google.com.bo/books?id=qIucp61eqAwC%7B%5C%7Dprintsec=frontcover%7B%5C%7Ddq=git%7B%5C%7Dhl=es-419%7B%5C%7Dsa=X%7B%5C%7Dved=0ahUKEwiTp4TKq4zpAhW0LLkGHcA5CYsQ6AEINTAB%7B%5C%7Dv=onepage%7B%5C%7Dq=git%7B%5C%7Ddf=false> (visitado 29-06-2020).
  - [11] Laura Dabbish y col. «Social coding in GitHub: Transparency and collaboration in an open software repository». En: *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*. 2012, págs. 1277-1286. ISBN: 9781450310864. DOI: 10.1145/2145204.2145396. URL: <http://git-scm.com/>.
  - [12] María Dolores; Lozano Pérez. *Ingeniería del software y bases de datos: tendencias actuales - María Dolores Lozano Pérez - Google Libros*. 2000. URL: <https://books.google.com.ar/books?id=bNDzMt6dwNsC%7B%5C%7Dlpg=PA78%7B%5C%7Ddq=%7B%5C%7D22Entorno%20de%20desarrollo%20integrado%7B%5C%7D22%7B%5C%7Dpg=PA78%7B%5C%7Dv=onepage%7B%5C%7Dq%7B%5C%7Ddf=false> (visitado 29-06-2020).
  - [13] Yohn Daniel Amaya Balaguera. «Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual». En: *Revista de Tecnología* 12.2 (2015). ISSN: 1692-1399. DOI: 10.18270/rt.v12i2.1291. URL: <https://revistas.unbosque.edu.co/index.php/RevTec/article/view/1291>.

- [14] *Patrón Model-View-ViewModel - Xamarin — Microsoft Docs*. URL: <https://docs.microsoft.com/es-es/xamarin/xamarin-forms/enterprise-application-patterns/mvvm> (visitado 29-06-2020).
- [15] Thomas R. Gruber. «Toward principles for the design of ontologies used for knowledge sharing». En: *International Journal of Human - Computer Studies* 43.5-6 (1995), págs. 907-928. ISSN: 10959300. DOI: 10.1006/ijhc.1995.1081. URL: <https://www.sciencedirect.com/science/article/pii/S1071581985710816>.
- [16] Nicola Guarino, Daniel Oberle y Steffen Staab. *Handbook on Ontologies*. 2009. DOI: 10.1007/978-3-540-92673-3.
- [17] S Bechhofer y col. *OWL Web Ontology Language Reference*. 2004. URL: <http://www.w3.org/TR/2004/REC-owl-ref-20040210/><http://www.w3.org/TR/owl-ref/>.
- [18] John H Gennari y col. «The evolution of Protégé: An environment for knowledge-based systems development». En: *International Journal of Human Computer Studies* 58.1 (2003), págs. 89-123. ISSN: 10715819. DOI: 10.1016/S1071-5819(02)00127-1. URL: <https://www.sciencedirect.com/science/article/pii/S1071581902001271>.
- [19] O Lassila y R R Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. Inf. téc. 19990222. 1999. URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.6030>.
- [20] Zoi Kaoudi, Iris Miliaraki y Manolis Koubarakis. «RDFS reasoning and query answering on top of DHTs». En: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 5318 LNCS. 2008, págs. 499-516. ISBN: 3540885633. DOI: 10.1007/978-3-540-88564-1-32. URL: <http://atlas.di.uoa.gr>.
- [21] Marco De Salvo. *RDFSharp Reference Guide (v2.13)*. 41. URL: <https://github.com/mdesalvo/RDFSharp/releases/download/v2.13.0/RDFSharp-2.13.0.pdf>.
- [22] Carmen Lasa Gómez Alonso Álvarez García, Rafael de las Heras del Dedo. *Métodos Ágiles y Scrum*. 2012, pág. 352. ISBN: 978-84-415-3104-8.
- [23] Evren Sirin y Bijan Parsia. «SPARQL-DL: SPARQL query for OWL-DL». En: *CEUR Workshop Proceedings*. Vol. 258. 2007. URL: <http://www.w3.org/Submission/RDQL/>.
- [24] Geoffrey Sparks. «Una Introducción al UML». En: (2008), págs. 1-47. URL: [www.sparxsystems.com.ar-www.sparxsystems.cl](http://www.sparxsystems.com.ar-www.sparxsystems.cl).
- [25] *Microsoft Visual Studio - Wikipedia, la enciclopedia libre*. URL: [https://es.wikipedia.org/wiki/Microsoft%7B%5C\\_%7DVisual%7B%5C\\_%7DStudio](https://es.wikipedia.org/wiki/Microsoft%7B%5C_%7DVisual%7B%5C_%7DStudio) (visitado 30-06-2020).
- [26] *¿Qué es Xamarin? - Xamarin — Microsoft Docs*. URL: <https://docs.microsoft.com/es-es/xamarin/get-started/what-is-xamarin> (visitado 30-06-2020).

- [27] ¿Qué es Xamarin.Forms? - Xamarin — Microsoft Docs. URL: <https://docs.microsoft.com/es-es/xamarin/get-started/what-is-xamarin-forms> (visitado 30-06-2020).
- [28] Norman Walsh. *A Technical Introduction to XML What is XML?* Inf. téc. URL: <https://www.math.unipd.it/~%7B~%7Dbasidati/docs/introduction.pdf>.
- [29] Lori MacVittie, A. *XAML in a Nutshell - Lori MacVittie - Google Knjige*. 2006, pág. 284. ISBN: 0-596-52673-3. URL: <https://books.google.es/books?hl=es%7B%5C%7Dlr=%7B%5C%7Ddid=v03e1G0y9ogC%7B%5C%7Ddoi=fnd%7B%5C%7Dpg=PT4%7B%5C%7Ddq=Xaml%7B%5C%7Ddots=WPWD0YfrfS%7B%5C%7Dsig=E0J9CZ%7B%5C%7DfC9k2gizr4pq8n2oGmcI%7B%5C%7Dv=onepage%7B%5C%7Ddq=Xaml%7B%5C%7Df=false%20https://books.google.si/books?hl=s1%7B%5C%7Dlr=%7B%5C%7Ddid=v03e1G0y9ogC%7B%5C%7Ddoi=fnd%7B%5C%7Dpg=PT4%7B%5C%7Ddq=xaml%7B%5C%7Ddots=W0UC007ogY%7B%5C%7Dsig=Fgs08I%7B%5C%7DXLtwYM-ygLBDW%7B%5C%7DgiDVEY%7B%5C%7Dredir%7B%5C%7Ddesc=y%7B%5C%7Dv=onepage%7B%5C%7Ddq=mvc%7B%5C%7Df=false>.
- [30] James Hendler, Tim Berners-Lee y Eric Miller. *Integrating Applications on the Semantic Web*. 2002. URL: <https://www.w3.org/2002/07/swint> (visitado 30-06-2020).
- [31] Mary Shaw y David Garlan. «Formulations and formalisms in software architecture». En: vol. 1000. 1995, págs. 307-323. DOI: 10.1007/bfb0015251.
- [32] Gaston Addati. *Escuela Superior Técnica Facultad de Ingeniería*. Inf. téc. 2013, pág. 132. URL: [www.econstor.eu](http://www.econstor.eu).
- [33] A. Pérez, J. & Gardey. *Definición de modelo de datos - Qué es, Significado y Concepto*. 2012. URL: <https://definicion.de/modelo-de-datos/> (visitado 05-07-2020).
- [34] Benson Moyo y col. «Empirical evaluation of software development methodology selection consistency : A case study using Analytical Hierarchy Process». En: *search.proquest.com* (). URL: <http://search.proquest.com/openview/3df28c8b722e916cac59dabc6be8b03d/1?pq-origsite=gscholar%7B%5C%7Dcbl=1976341>.
- [35] RS Pressman y JM Troya. «Ingeniería del software». En: (1988). URL: <http://fondoeditorial.uneg.edu.ve/citeg/numeros/c02/c02%7B%5C%7Ddart10.pdf>.
- [36] Carlos Gerardo Prieto Álvarez. «Universidad tecnológica de la mixteca». En: (2013).
- [37] Ramón Ventura Roque Hernández y col. «Comparación empírica entre el Proceso Unificado y el desarrollo de software por Prototipos». En: *Western Hemispheric Trade Conference*. 2015, págs. 235-244. ISBN: 0123566789. URL: <http://www.academia.edu/download/54580094/19Conf-Sessions.pdf%7B%5C%7Dpage=245>.



- [38] *Modelo de Prototipos - Portafolio de Evidencia Erika*. URL: <https://sites.google.com/site/portafoliodeevidenciaerika/tema-3---desa/modelo-de-prototipos> (visitado 02-07-2020).
- [39] Johanna Patricia Zumba. «Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software». En: *INNOVA Research Journal* 3.10 (2018), págs. 20-33. ISSN: 2477-9024. DOI: 10.33890/innova.v3.n10.2018.651. URL: <https://dialnet.unirioja.es/servlet/articulo?codigo=6777227%7B%5C%7Dinfo=resumen%7B%5C%7Didioma=SPA%20https://dialnet.unirioja.es/servlet/articulo?codigo=6777227%7B%5C%7Dinfo=resumen%7B%5C%7Didioma=ENG%20https://dialnet.unirioja.es/servlet/articulo?codigo=6777227>.
- [40] Alejandro Aldás Alarcón. *Modelo de proceso incremental Fuente:... — Download Scientific Diagram*. 2016. URL: <https://www.researchgate.net/figure/Figura-10-Modelo-de-proceso-incremental-Fuente%7B%5C%7Dfig6%7B%5C%7D326571456> (visitado 02-07-2020).
- [41] Kent Beck y Et.al. *Principios del Manifiesto Ágil*. 2001. URL: <https://agilemanifesto.org/iso/es/principles.html> (visitado 02-07-2020).
- [42] Matt Stephens y Doug Rosenberg. *Extreme Programming Refactored: The Case Against XP*. 2003. DOI: 10.1007/978-1-4302-0810-5.
- [43] K Schwaber y Jeff Sutherland. «The scrum guide suomeksi». En: *Scrum.org, October* 2.July (2011), pág. 17. ISSN: 00195847. DOI: 10.1053/j.jrn.2009.08.012. URL: <http://www.scrum.org/scrumguides/%7B%5C%7D5Cnhttp://pdf4420.psxbook.com/scrum%7B%5C%7D1868546.pdf>.
- [44] Alistair Cockburn. *Crystal Clear: A Human-Powered Methodology for Small Teams (The Agile Software Development Series)*. 2004. URL: <https://www.researchgate.net/publication/234820806%20http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20%7B%5C%7Dpath=ASIN/0201699478>.
- [45] César Rodríguez y Rubén Dorado. *Porque implementar Scrum?* Inf. téc. URL: <https://journal.universidadean.edu.co/index.php/Revistao/article/view/1253>.



## CAPÍTULO 14

---

Información sobre Licencia

---

