Universidad de Cádiz
Escuela Superior de Ingeniería

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

Universidad de Cádiz
Escuela Superior de Ingeniería

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 academico, titulado ARPEGOS: Automatized 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 prejuicio 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	
negrita	Título o texto destacado
cursiva	Texto en otro idioma, destacado, citas o nombres de aplicaciones
`monoespaciado`	Referencias a código fuente
subrayado	Advertencia para el lector
color	Enlace interno (rojo)

Índice general

Índice de figuras

Índice de cuadros

Parte I

Prolegómeno

CAPÍTULO 1

Introducción

1.1. Motivación

El sector de los juegos de rol está creciendo a pasos agigantados. Hoy en día existe un extensísimo catálogo de juegos de rol, que varían en función de la ambientación del "universo" en el que nos sumergimos. Por ejemplo, *Dragones y Mazmorras* utiliza una ambientación medieval con elementos fantásticos, tales como magia y dragones, mientras que *Warhammer 40.000* está basado en un futuro distópico en el que se mezclan elementos de ciencia ficción con elementos de fantasía heroica.

Debido a la expansión de este sector, y gracias al avance de la tecnología, podemos encontrar un sinfín de aplicaciones que tratan de mejorar la experiencia de los jugadores, desde juegos de mesa que hacen uso de sistemas de reproducción (*CD*, *DVD*, *Blu-Ray*) para hacer del juego una experiencia interactiva (*Atmosfear*, *Piratas del Caribe: El Cofre del Hombre Muerto*, *Trivial Pursuit*, *Cluedo*), hasta juegos de rol completamente virtualizados que nos permiten sumergirnos en ellos, contemplar sus parajes, disfrutar de su ambientación, y en algunos casos, nos permiten participar con jugadores de cualquier parte del mundo a través de la red (*World of Warcraft*, *Diablo III*, *Black Desert Online*).

Dentro de este amplio espectro de posibilidades, los juegos de rol tradicionales también disponen de aplicaciones que enriquecen el transcurso de las partidas, y que proveen prestaciones útiles para las mismas, como lanzadores de dados virtuales, generadores de mazmorras, contadores de iniciativa, compendios de habilidades, etc.

Entre todas las utilidades que encontramos, cabe destacar las aplicaciones conocidas como *generadores de personaje*. Estas facilitan a los jugadores todo lo necesario para desarrollar rápidamente la información indispensable de un personaje para poder interpretarlo en una partida.

Por otro lado, puesto que la tecnología tiene un ritmo de avance trepidante, y más aún la informática, hoy en día podemos echar cuenta de servicios que simplifican tareas cuya ejecución resultaba impensable en el pasado.

Uno de estos servicios es la Web Semántica, que se basa en la idea de describir el contenido, el significado y la relación de los datos, de manera que sea posible evaluarlos automáticamente por máquinas de procesamiento, como pueden ser los ordenadores o los *smartphones*.

El planteamiento del presente proyecto surge de la concepción de una aplicación informática destinada a ampliar las funcionalidades de los generadores de personaje, innovando en el desarrollo de la misma al aplicar tecnologías de Web Semántica.

1.2. Alcance

El alcance de este proyecto comprende el desarrollo de una aplicación móvil para generar perfiles de personajes que puedan formar parte de una partida o campaña de un juego de rol, cuya base de datos esté incluida en la aplicación.

No forma parte del alcance de este proyecto el desarrollo de la base de datos del juego de rol, aunque ha sido necesario para poder realizar la sección de pruebas del proyecto.

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, teléfonos móviles y relojes equipados con Android, aunque el software también se usa en otros dispositivos.

- ***Blazegraph***: Base de datos de grafos de código abierto, escalable y de alto rendimiento basada en estándares. Escrito completamente en *Java*, la plataforma soporta las familias de especificaciones *Blueprint* y **RDF/SPARQL 1.1** incluyendo consultas, actualizaciones, consultas federadas básicas y descripción de servicios.

- ***C#***: Lenguaje de programación multiparadigma desarrollado y estandarizado por Microsoft como parte de su plataforma **.NET**, que después fue aprobado como un estándar por la ECMA (*ECMA-334*) e ISO (*ISO/IEC 23270*). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. Su sintaxis básica deriva de *C/C++* y utiliza el modelo de objetos de la plataforma dotnet, similar al de *Java*, aunque incluye mejoras derivadas de otros lenguajes.

- ***RDFSharp***: *Framework* de código abierto **C#** diseñado para facilitar la creación de aplicaciones *.NET* basadas en el modelo **RDF**, que representa una solución didáctica directa para comenzar a trabajar con conceptos de *Semántica Web*.

- ***Git***: Software de control de versiones diseñado por *Linus Torvalds*, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

- ***GitHub***: GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones **Git**.

- ***IDE***: *(Integrated Development Environment)* Aplicación con numerosas características que se pueden usar para muchos aspectos del desarrollo de software.

- ***iOS***: Sistema operativo móvil de la multinacional *Apple Inc.* Originalmente desarrollado para el iPhone (iPhone OS), después se ha usado en dispositivos como el iPod touch y el iPad. No permite su instalación en hardware de terceros.

- ***Metodología Ágil***: Las metodologías ágiles son métodos de desarrollo de software en los que las necesidades y soluciones evolucionan a través de una colaboración estrecha entre equipos multidisciplinarios. Se caracterizan por enfatizar la comunicación frente a la documentación, por el desarrollo evolutivo y por su flexibilidad.

- ***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.

- ***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.

- ***MVVM***: Patrón de arquitectura de software que ayuda a separar la lógica de negocios y presentación de una aplicación de su interfaz de usuario.

- ***Ontología***: Definición formal de tipos, propiedades, y relaciones entre entidades que realmente o fundamentalmente existen para un dominio de discusión en particular. Es una aplicación práctica de la ontología filosófica, con una taxonomía.

- ***OWL***: *(Ontology Web Language)* es un lenguaje de marcado semántico para publicar y compartir ontologías en la World Wide Web. OWL se desarrolla como una extensión de vocabulario de **RDF** y es derivado del lenguaje DAML + OIL asi.

- ***Protégé***: Framework editor de ontologías de código abierto y gratuito para construir sistemas inteligentes.

- ***RDF***: *(Resource Description Framework)* Modelo estándar para el intercambio de datos en la Web. RDF tiene características que facilitan la fusión de datos incluso si los esquemas subyacentes difieren, y admite específicamente la evolución de los esquemas a lo largo del tiempo sin requerir que se cambien todos los consumidores de datos.
- ***RDFS***: *RDF Schema* es una extensión del vocabulario básico de *RDF* que proporciona un vocabulario de modelado de datos para los datos relativos a este modelo.
- ***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.
- ***SonarQube***: Herramienta de revisión automática de código para detectar errores, vulnerabilidades y olores de código en su código. Se puede integrar con su flujo de trabajo existente para permitir la inspección continua de código en todas las ramas de su proyecto y solicitudes de extracción.
- ***SPARQL***: *(SPARQL Protocol and RDF Query Language)* es un lenguaje y protocolo de consulta para *RDF*.
- ***Sprint***: Período en el cual se lleva el desarrollo de una tarea.
- ***Tarsier***: Herramienta para la visualización interactiva en 3D de grafos RDF.
- ***UML***: *(Unified Modeling Language)* es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.
- ***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.
- ***Visual Studio***: El *IDE* de Visual Studio es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. Más allá del editor estándar y el depurador que proporcionan la mayoría de *IDE*, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para facilitar el proceso de desarrollo de software.
- ***W3C***: *(World Wide Web Consortium)* Consorcio internacional que produce recomendaciones para la *WWW*.
- ***WWW***: *(World Wide Web)* Sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet.

- ***Xamarin***: Xamarin es una plataforma de código abierto para compilar aplicaciones modernas y de rendimiento para iOS, Android y Windows con .NET.
- ***Xamarin.Forms***: Xamarin.Forms es un marco de interfaz de usuario de código abierto. Xamarin.Forms permite a los desarrolladores compilar aplicaciones de Android, iOS y Windows desde un único código base compartido.
- ***XAML***: *(eXtensible Application Markup Language)* Lenguaje basado en XML creado por *Microsoft* como una alternativa a código de programación para la creación de instancias e inicialización de objetos, y la organización de esos objetos en jerarquías de elementos primarios y secundarios.
- ***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.*"

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. Conceptos básicos***. 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

Planificación del proyecto

2.1. Metodología

2.1.1. 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 desarrolllo de un proyecto de construcción de software* (Moyo, Fonde, Soganile, Dzawo & Madzima, 2013).

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 métodologías 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* (Pressman, 1995).

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.

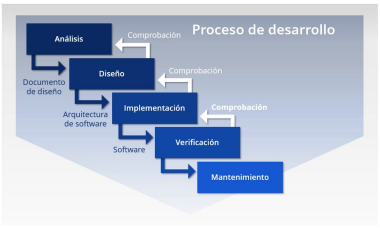

Figura 2.1: Modelo de desarrollo en cascada

- **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.

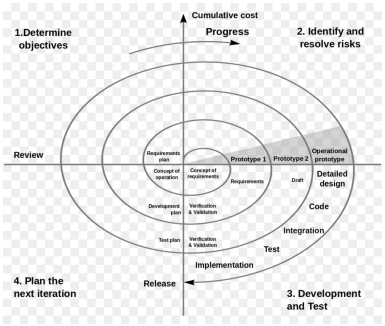

Figura 2.2: Modelo de desarrollo en espiral

- **Desarrollo con prototipos**: *El desarrollo de software basado en prototipos promueve la comunicación entre el cliente y el equipo de programadoroes, a la vez que logra una rápida integración de cambios y acorta el tiempo de desarrollo del proyecto.* (Ventura, Salinas, Álamos y Arreola, 2015). 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.

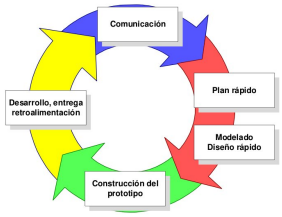

Figura 2.3: Modelo de desarrollo con prototipos

- **Desarrollo incremental**:Consiste en un modelo iterativo que cuenta con una serie de fases de desarrollo (Análisis, Diseño, Implementación, Pruebas) que se repiten en orden de manera que cada vez que se finaliza una iteración, el producto está mas refinado, siendo el objetivo llegar al producto final.

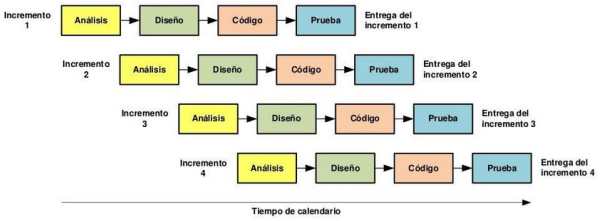

Figura 2.4: Modelo de desarrollo incremental

2.1.2. 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.

El punto de partida para ello fue el **Manifiesto Ágil**: 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** (Stephens & Rosenberg, 2003), **Scrum** (Sutherland & Schwaber, 2010) o **Crystal Clear** (Cockburn, 2004). 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.

2.1.3. SCRUM

Figura 2.5: Logo de *Scrum*

Introducción

SCRUM es una de las metodologías de desarrollo ágil más reconocidas mundialmente. Su concepción resulta de unos análisis realizados por **Ikujiro Nonaka** e **Hirotaka**

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 (Schwaber & Sutherland, 2012).

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.

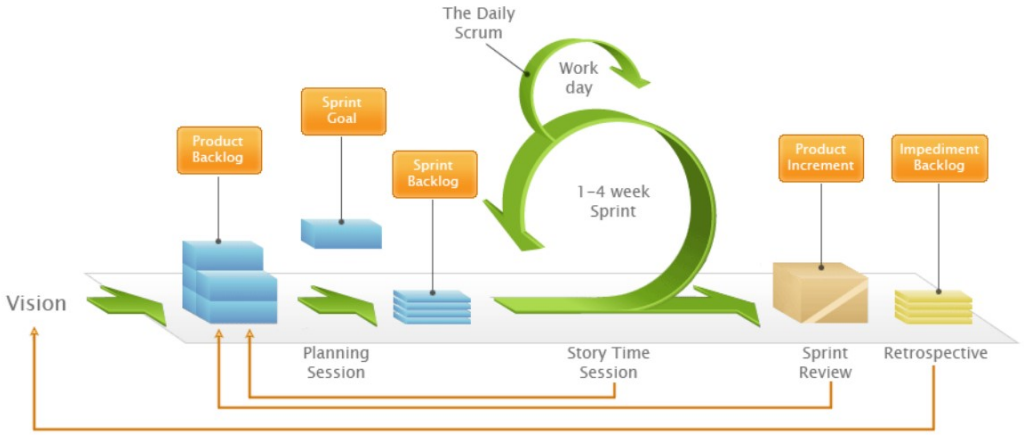

Figura 2.6: Ciclo de desarrollo de *Scrum*

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*.

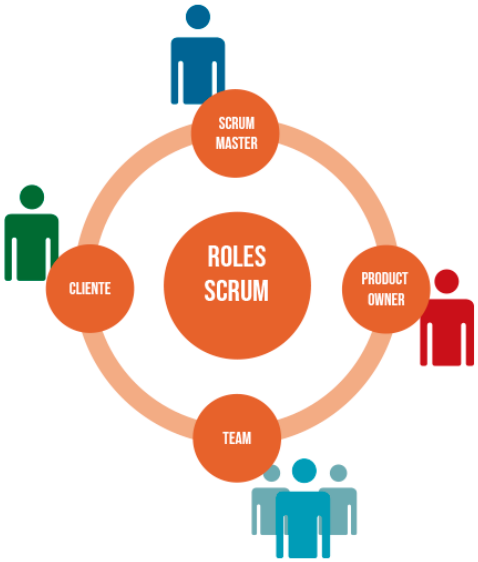

Figura 2.7: Roles en *Scrum*

2.2. 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.

2.2.1. 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.

2.2.2. 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.

2.2.3. 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.

2.2.4. 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.

2.3. Organización

2.3.1. 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.

2.3.2. 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.

2.4. Costes

2.4.1. 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 2.1: Comparativa entre coste humano estimado y coste humano real

2.4.2. 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.

2.5. Evaluación y gestión de riesgos

2.5.1. ¿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.

2.5.2. 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.

2.5.3. 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.

2.5.4. 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

CAPÍTULO 3

Requisitos del Sistema

3.1. 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 3.1: ***5e Character***: Pantalla de selección de clases

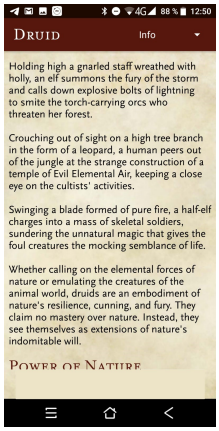

Figura 3.2: ***5e Character***: Información de la clase *Druida*

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

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

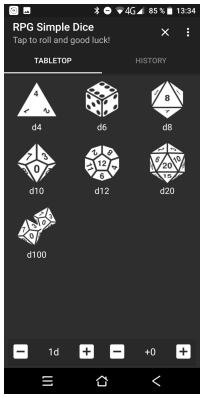

Figura 3.3: ***RPG Simple Dice***: Pantalla de selección de dados

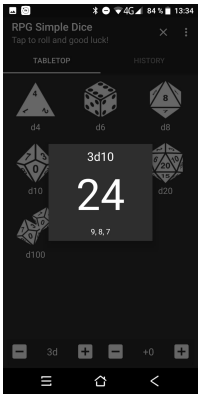

Figura 3.4: ***RPG Simple Dice***: Ejemplo de lanzamiento de dados

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***.

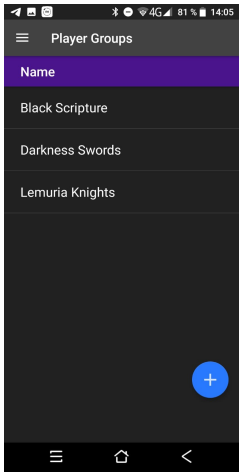

Figura 3.5: ***BattleTrack***: Pantalla de selección de grupo

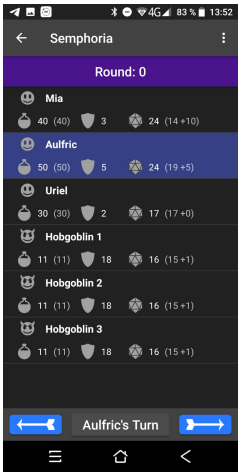

Figura 3.6: ***BattleTrack***: Ejemplo de combate

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.

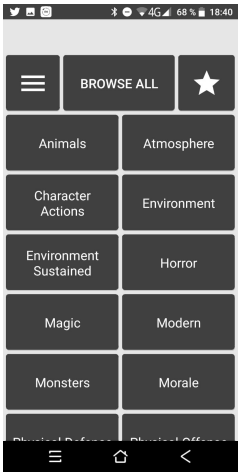

Figura 3.7: ***RPGSound***: Menú principal

Figura 3.8: ***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***.

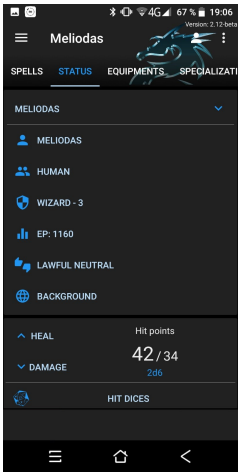

Figura 3.9: ***RPG Character Sheet***: Pantalla de *Estado*

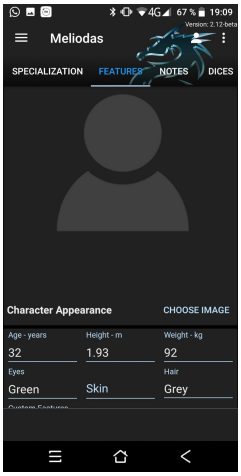

Figura 3.10: ***RPG Character Sheet***: Pantalla de *Features*

3.1.1. 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.

3.2. 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.

3.2.1. 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.

3.2.2. 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*.

3.3. 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.

3.3.1. 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 lenguanje *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 3.11: Logo de *Android Studio*

Figura 3.12: Logo de *Apache Jena*

3.3.2. 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.

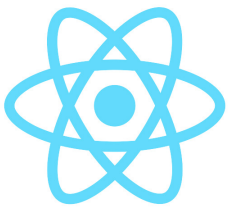

Figura 3.13: Logo de *React Native*

Figura 3.14: Logo de *rdflib.js*

3.3.3. 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 3.15: Logo de *Xamarin*

3.4. 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 3.1: Comparativa entre *JavaScript* y *C#*

	JavaScript	***C#***
Tipo de Lenguaje	Scripting	Orientado a Objetos
Tipado	Débil	Fuerte
Detección de errores	Ejecución	Compilación y ejecución
Compilación	No	Sí
Mantenibilidad	Complejo	Sencillo
Soporte de IDE	No	Microsoft Visual Studio
Sintaxis	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 4

Análisis del Sistema

CAPÍTULO 5

Diseño del Sistema

CAPÍTULO 6

Construcción del Sistema

CAPÍTULO 7

Pruebas del Sistema

Parte III

Epílogo

CAPÍTULO 8

Manual de Implantación y Explotación

CAPÍTULO 9

Manual de Usuario

CAPÍTULO 10

Manual de Implantación y Explotación

CAPÍTULO 11

Conclusiones

CAPÍTULO 12

Bibliografía

CAPÍTULO 13

Información sobre Licencia