Universitat Autònoma
de Barcelona

Sistema de Soporte Para Juego de Rol

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Gestió
realitzat per
David Arjona Fernández
i dirigit per
Xavier Verge Mestre

Escola d'Enginyeria
Sabadell, *Juny* de *2010*

El sotasignat, ***Xavier Verge***
Professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció
per en ***David Arjona Fernández***

I per a que consti firma la present.
Sabadell, ***Juny*** de ***2010***

Signat: ***Xavier Verge Mestre***

Resumen

El presente proyecto consiste en el desarrollo de una aplicación que proporcione una serie de herramientas que ayuden a gestionar todos los procesos surgidos durante el transcurso de una partida de rol tradicional.

Habitualmente una persona toma el rol de Máster de partida y su función es la de proporcionar el contexto donde se sucederán los eventos y también resolver dichos procesos. Sobre esta figura está ideada la aplicación pues será capaz de mejorar la calidad del juego gracias a su eficacia y a liberar una gran carga de trabajo al Máster.

La forma de conseguir este propósito será reproduciendo el reglamento del juego en el software, otorgando herramientas que permitan trabajar con todos los diferentes elementos que lo componen.

El proyecto se compondrá de un aplicativo desarrollado en Java y una base de datos embebida para almacenar todos los datos generados.

Se trata de un proyecto satisfactorio pues se han cumplido los objetivos planteados, y concretamente su idea central ha sido resulta con solvencia.

Índice

1 Introducción

1.1 Descripción

El concepto de juego de rol nace a finales de los 60, desarrollándose en los años posteriores. Su evolución se ha diversificado, y una de las líneas que ha seguido ha sido la de los juegos de rol de mesa que existen hoy en día.

Un juego de rol de mesa no es más que una interpretación de situaciones. Como en toda historia, aquí también hay un director, que en este caso es comúnmente denominado Máster. Él es quien crea el contexto y guía a los jugadores con la ayuda del reglamento del juego.

Una partida de rol genera gran cantidad de procesos, como por ejemplo las tiradas de dados con su correspondiente comparación de números y la gran mayoría de éstas han de ser resueltas manualmente por el Máster. Se trata de una carga de trabajo importante que en muchas ocasiones empeora la experiencia y entorpece el transcurso de la partida. A menos que sea muy experimentado el Máster ha de estar más pendiente de estas operaciones que de velar para que el juego resulte ameno, y además, ralentiza el transcurso de la historia llegando incluso a aburrir.

Nos encontramos pues, ante un conjunto de procesos que, como en cualquier actividad productiva, requieren ser optimizados. El propósito del proyecto es entonces, digitalizar el trabajo del Máster, automatizando estos procesos mediante una aplicación informática. Como resultado: se incrementará la rapidez con la que serán resueltas las diferentes tareas; mejorará la eficacia y seguridad proporcionando una fiabilidad que aportará una mayor calidad; y, además, la liberación de trabajo del Máster dará lugar a una mejora de todo lo improvisado al disponer de más tiempo y calma. En definitiva, se trata del paso del rol de lápiz y papel (analógico), al rol digital.

Un beneficio adicional que otorgará el programa a la situación, será facilitar enormemente la tarea de ser Máster dado que no será necesario estudiarse el reglamento del juego para poder dirigir una partida, ya que con conocimientos básicos sobre el reglamento y habiendo aprendido a usar la aplicación podremos llevar la partida a cabo.

1.2 Objetivos

El objetivo principal es conseguir que se mejore la experiencia de juego en una partida de rol evitando errores y facilitando las tareas del Máster. Para ello se desarrollará un software en el que se digitalizarán diversos procesos que realiza el máster y que pueden llegar a ser tediosos y producir esperas relevantes a los jugadores.

Si se quiere que el salto a lo digital merezca la pena, se necesita que la información sea mostrada de forma ordenada, clara y concisa, dado que esto contribuirá a mejorar la fluidez con la que transcurre una partida. También es un factor que permitirá conseguir que el programa sea de fácil manejo, sobre todo si conseguimos que la interfaz sea extremadamente intuitiva para alguien que conozca la dinámica de un juego de rol.

Las reglas básicas son la parte intrínseca de un juego de rol pero a ningún director de partida le suele gustar que un reglamento le imponga sus reglas no básicas, ya que aquí el único que debería imponer este tipo de reglas es el propio Máster. Por lo tanto, será muy importante desarrollar un software flexible donde el Máster pueda hacer y deshacer a su necesidad.

El programa además ha de cumplir otras metas, debido a que se suele realizar esta actividad en lugares dispares. No siempre se puede reunir a grupos de gente donde se desee y ciertos jugadores prefieren lugares más apartados y tranquilos.

Se pretende que sea capaz de almacenar gran parte de la información generada por los usuarios y además la contenida en el reglamento del juego, el cual sólo será requerido en casos muy puntuales o incluso se podrá prescindir de él. A pesar de ello se necesita autonomía pues no siempre se va a disponer de una conexión a Internet.

También se quiere ligereza, pues no siempre se tiene el dispositivo más potente, y finalmente si además se puede ejecutar en cualquier sistema, crecería en gran medida el abanico de dispositivos en los que correr el programa.

1.3 Estado del Arte

La naturaleza de este proyecto no es un tema especialmente explotado. Quizás por no despertar el suficiente interés o por no aportar los suficientes beneficios. Aunque principalmente hay dos soluciones viables para solventar este problema.

La primera es que existen aplicaciones libres que cumplen diferentes funcionalidades, como por ejemplo, las que lanzan dados o las que permiten crear personajes. No he encontrado ninguna libre que satisfaga la totalidad de las facetas ocupadas por este proyecto. Algún software libre que ha sido hallado en www.sourceforge.net es:

- **PcGen Character Generator:** Como su nombre indica, es una herramienta capaz de crear personajes. Abarca varios juegos de rol distintos, entre ellos, el tratado por este proyecto. Realmente es una aplicación muy completa.
- **GameTable:** Proporciona un mapa modificable e interactivo. Funcionalidad interesante de la que carece este proyecto, cuya línea de expansión podría ser ésta.
- **RPTools:** Conjunto de utilidades muy simples, entre las que se incluyen un generador de tiradas de dados y un gestor de turnos.

La segunda alternativa se trata de un software de pago disponible por licencia de uso en el mercado llamada **Fantasy Grounds.** Incluye la mayoría de facetas de este proyecto, además de capacidades on-line y gestión de mapas. Incluso dispone la posibilidad de modificar reglas de los juegos. El inconveniente, a parte de ser de pago, es que requiere bastante aprendizaje para poder ser usado satisfactoriamente.

1.4 Motivaciones Personales

Al principio hubo mucha incertidumbre respecto a qué proyecto realizar. Mi única idea era que quería desarrollar una herramienta de gestión para alguna actividad indeterminada. Así que era necesario encontrar un contexto sobre el que trabajar, una actividad con una serie de procesos que se pudieran gestionar mediante un programa informático.

Al no saber de ninguna actividad que cumpliese lo necesario, surgió la idea de utilizar un hobby como los juegos de rol. Al conocer todo su funcionamiento, y saber que la complejidad de su

digitalización podía estar a la altura ya que no envidiaría nada a cualquier la gestión estándar, se podría convertía entonces en un proyecto viable, y además muy original.

Se trata de un proyecto con pocas referencias, no hay ejemplos que seguir, por lo que su diseño es algo ideado por mí desde cero, y ésto generaba un reto que a mí me resultaba muy atractivo.

Antes de decidir el proyecto tenía la preferencia de realizar el proyecto en Java. Ya que lo aprendido en la universidad había despertado mi ánimo de aprender más sobre el lenguaje, además de que conocerlo resulta ser una puerta abierta de cara al futuro.

Finalmente una motivación evidente es que realizar el proyecto sobre un tema del que soy aficionado hacia todo mucho más tentador, las ideas y propósitos surgían prácticamente solos. Hacer un trabajo que guste, por complicado que sea, lo hace todo mucho más fácil.

1.5 Estructura de la Memoria

Se ha seguido una organización estándar para realizar la memoria. Se podrán encontrar diferentes puntos de conexión entre varios aspectos, ya que se ha intentado que todo tenga una cierta coherencia; por ejemplo, haciendo referencia a los objetivos que se han establecido en este punto.

En el apartado que se está concluyendo con este punto, se han descrito aspectos importantes que marcarán el devenir del resto del proyecto.

En el segundo capítulo se comentarán diferentes conceptos que facilitarán su comprensión. Se explica de manera más detallada qué es el rol y qué se está haciendo en este proyecto, además de conceptos sobre las tecnologías usadas.

El tercero describe un análisis y la planificación que se realizó antes de empezar.

Durante el cuarto apartado referente al diseño, se dará una visión general sobre cómo funciona la aplicación y sobre cómo está estructurada.

En el punto número cinco se comenta cómo se ha ido realizando la codificación, qué pruebas se han realizado al finalizar y el resultado de éstas.

Finalmente, en el último apartado se comenta todas las conclusiones extraídas del proyecto tanto a nivel personal como en el cumplimiento de objetivos. También se muestra un diagrama de seguimiento final y se comentan los motivos de las desviaciones surgidas.

Como añadido a la memoria se incluirán dos anexos. El primero será una interesante recopilación de problemas surgidos con ejemplos explicativos sobre cómo se han solucionado, y el segundo anexo será un manual de usuario por si se diese el caso poco probable de que existiesen problemas a la hora de usar la aplicación.

2 Marco Teórico

2.1 ¿Qué es un Juego de rol?

2.1.1 Definición

Durante la introducción se definió a vista de pájaro, qué era un juego de rol. Ahora desarrollaré este concepto un poco más profundamente.

Los juegos de rol surgieron como una actividad didáctica para enseñar diversos aspectos sociológicos, políticos o comunicativos. Hoy en día se usan con varios fines a parte de los mencionados: terapéuticos, de selección de personal y lúdicos que son los más conocidos y sobre el que se centrará el proyecto.

Durante el juego, los participantes exploran un espacio imaginario compartido planteado por el director de la partida y el juego, y desarrollado por los propios jugadores. Ellos son los que dibujan todo lo que suceda allí, con sus acciones, descripciones, narraciones, etcétera.

Cada uno de los jugadores, asume el rol de uno de los personajes imaginarios ya propuestos o creados por ellos mismos, disponiendo de total libertad de decisión. De ahí el nombre de estos juegos. Así pues, la gran mayoría del transcurso del juego está constituido por el diálogo y la narración entre los jugadores y el Máster.

No es necesario representar teatralmente dicho personaje, ni introducirse en su personalidad. Simplemente se trata de describir como interactuaría con el entorno imaginario.

Cuando se producen las situaciones en las que las ideas de los jugadores no puedan ser trasladadas de forma trivial al mundo imaginario de la partida, porque produzcan conflicto con otra idea; entran en juego las reglas del juego. Son las encargadas de ayudar a resolver estos problemas, decidiendo qué acción prevalece y cual será su resultado. Pero al final, quien tiene la última palabra son los jugadores.

En este proceso de decisión, se está produciendo una negociación. Existen multitud de juegos que contextualizan mundos de todo tipo y aportan reglas muy dispares.

2.1.2 Sistema de un juego de rol.

Después de muchos debates en el seno del mundo de los juegos de rol, surgió lo que en su contexto se conoce como el Principio de Lumpley. Que dice que el sistema es un mecanismo utilizado para decidir qué sucede en la partida.

De manera que el sistema no tiene que coincidir necesariamente con el reglamento del juego. Descrito de otra forma, el sistema es el cómo se juega y no el cómo se debería jugar.

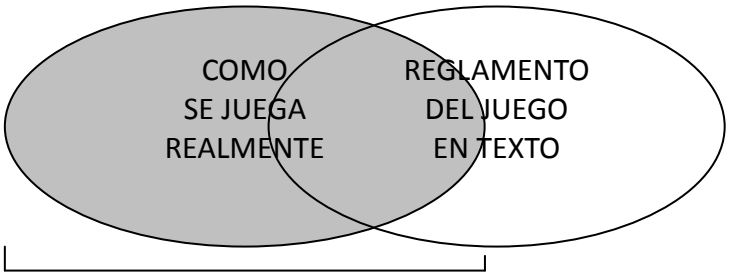


Figura 2.1: sistema de juego

Existen varias posibilidades: Un alto o bajo grado de acoplamiento, un desentendimiento total de las reglas, o un hipotético en el que coinciden completamente.

El grado de acoplamiento puede ser muy variable según los jugadores y el juego. Depende de factores como por ejemplo los gustos personales de los jugadores o de la complejidad de las reglas. Generalmente un alto grado de solapamiento implica satisfacción por el juego.

Por lo tanto podemos diferenciar entre dos tipos de reglas: Las circunstanciales que son tomadas de forma improvisada, y las mecánicas empleadas por el grupo de forma sistemática, estas incluyen las contenidas en el reglamento y las ideadas por el grupo. Si una decisión improvisada se produce de forma reiterada pasará a formar parte de una mecánica ideada por el grupo.

2.1.3 La diferencia entre Personaje y NPC

Durante la memoria se mencionará multitud de veces las palabras Individuos, Personajes y NPC. El objetivo de este apartado es aclarar estos conceptos.

En la gran mayoría de los juegos de rol cada jugador tomará el control de únicamente un personaje. Así que como todas las acciones de los personajes decidirán el devenir de la partida, ellos representan la base de todo reglamento. Siendo así, los personajes son construidos utilizando unos métodos específicos, más o menos complejos según el juego.

Cualquier partida no sólo se constituye de personajes, sino que también de todo con lo que ellos interactúan. Sería muy engorroso que cada vez que un Máster quisiese introducir un elemento animado dentro de la partida, se tuviese que crear mediante el, probablemente complejo, sistema de creación de personajes propuesto por el juego. Así que para cubrir esta necesidad existen los NPC (Non-Player Character), que pueden realizar la mayoría de las acciones de un personaje, pero siendo elementos mucho más simples que éstos. Se puede decir que existen para poder interactuar con los personajes.

La palabra individuos en este contexto, se utiliza para englobar Personajes y NPCs. Ya que como se ha dicho, son similares, pues ambos constituyen todos los elementos “vivos” del juego.

2.1.4 Funcionamiento del Reglamento

Seguidamente se explicará el funcionamiento básico del juego de rol Dungeons & Dragons que utiliza los métodos que describe un sistema basado en dados de 20 caras (Sistema D20). De esta forma será posible entender mejor qué es lo que se está informatizando en este proyecto.

Repetir que, aquí sólo se halla el funcionamiento a grandes rasgos, no se ha entrado en los entresijos del juego y quedarían muchos aspectos que no estarán aquí descritos.

Como se dijo en el apartado anterior, todo juego de rol toma como centro sus personajes. Así que se empezará por describir cómo se crean los personajes.

Creación de Personajes

La base del proceso de creación de personaje, obviando la introducción de datos descriptivos, es asignar sus seis características (Fuerza, Destreza, etc.). Existen varias formas de manejar esta selección, la más común es realizar seis tiradas compuestas de dados, cada una de ellas consta de cuatro tiradas de dados de seis caras, la menor se descarta y la suma de las demás da el resultado final. Finalmente los seis resultados se reparten a elección del jugador.
El siguiente paso es escoger los dos aspectos que más determinarán la constitución del personaje.

La raza es quizás el aspecto que menos repercusión tiene, pues solamente bonifica o penaliza alguna característica, otorga alguna habilidad especial y en algunos casos limita la selección de la clase.

La clase es el siguiente aspecto a elegir y es el más importante. Cada clase dispone de una tabla distinta que según qué nivel sea el personaje define una serie de cualidades u otras, como por ejemplo, el Ataque Base y la defensa. Además de esto su elección también decide, entre otras cosas, cuales son las habilidades de clase del personaje y qué conjuros es capaz de preparar.

Cada personaje que se cree, comúnmente suele comenzar en nivel uno para después ir subiendo según la experiencia conseguida en las partidas que juegue. Un nuevo nivel supone para el personaje una mejora al escalar en la tabla de su clase, también gana puntos de golpe (vida) y puede aumentar un característica suya en los niveles múltiples de cuatro.

Dotes

Las dotes bonifican aptitudes pasivas de los personajes y cambian ligeramente el comportamiento de los personajes al realizar según qué tipos de acciones. Hay bastante cantidad con dispares efectos. Los niveles múltiplos de tres otorgan una nueva dote. En el caso de que el personaje sea de clase *Guerrero* dispondrá de puntos especiales para dote que podrá gastar en sólo unas dotes específicas.

Habilidades

Las habilidades describen las acciones comunes que pueden realizar los personajes, como abrir una cerradura o saltar. La cantidad de puntos de habilidad de los que dispone un personaje se

determina con una fórmula dónde influye su clase, su nivel y su característica inteligencia. Lo bueno que un personaje con una habilidad se determina por el nivel de modificador que ésta dispone. Para aumentar dicho modificador se invierten los puntos de habilidad en los rangos que cada una dispone. Si la clase del personaje dice que es bueno con esa habilidad, un rango supondrá un punto de modificador, en caso contrario serán necesarios dos rangos. Es importante decir que cada habilidad está relacionada con una característica en concreto cuyo modificador también influirá en el resultado de la acción.

Conjuros

Los conjuros que puede usar un personaje dependen de la clase que este sea. Algunas clases incluso no habilitan el uso de ningún conjuro. Los conjuros se dividen en niveles de conjuro y según la clase que puede realizarlo, esta organización permite que los personajes al subir sus niveles puedan elegir unos conjuros u otros según lo que les limiten la clase y nivel que son. Es de relevancia decir que la variedad de efectos que provocan los conjuros es muy grande, algunos de ellos no provocan efectos cuantificables pues sólo modifican la conducta o el terreno, por lo que no siguen ningún estándar de aplicación.

Inventario y Equipo

Realmente la mayor regla que sigue este apartado es la que dicta el sentido común. Cada objeto está ideado para equiparse en un lugar concreto, por lo que por ejemplo, no se puede pretender hacer cosas como equiparse dos armas que requieren dos manos. Normalmente las armaduras y escudos provocan penalizadores si son pesados y hay ciertas armas que requieren disponer de una dote que habilite su uso.

Atacando

Cuando se realiza un ataque pueden suceder varias cosas. Los personajes al atacar lanzan un dado de 20 cuyo resultado será sumado al ataque del personaje. Con esto, se obtiene la efectividad del ataque. Después de este paso lo siguiente es tan simple como comprobar si supera la defensa del objetivo, y de ser así el golpe impactará. El daño que causará el impacto lo decide la tirada de dados dictada por el arma que se está usando.

Uso de Habilidades

El uso de una habilidad puede tener dos objetivos distintos, una dificultad concreta a la que debe superar o un enfrentamiento contra otro individuo. El primer caso se resuelve simplemente viendo si se supera la dificultad establecida. El segundo caso es más interesante, pues se produce una tirada enfrentada que consiste en que cada individuo hace su respectiva tirada de habilidad, el que consiga una tirada mayor será el vencedor.

Críticos y Pifias

Siempre que se ejecuta cualquier tipo de acción en el juego, al igual que la realidad, las cosas también pueden salir realmente bien o ser un fracaso estrepitoso. Esto lo describen los críticos y las pifias, y conforman uno de los aspectos más divertidos de este tipo de juegos. Cuando se sucede un crítico (Máximo resultado en el dado de 20 generalmente), aquello que se hubiese intentado hacer habrá salido bien, sea cual sea la dificultad y normalmente provoca efectos adicionales beneficiosos sobre lo que sucedería si sólo hubiera salido bien. En contraposición a esto, las pifias son completamente lo contrario (un resultado de 1 en un dado), pues provocan el fracaso automático de lo que se estuviese intentando hacer y no solo eso, sino que se fracasará de la peor manera posible.

2.2 Java como Lenguaje Orientado a Objetos

Debido a que el lenguaje de programación con el que está diseñado el proyecto es Java, procederé, para empezar, a explicar los pilares conceptuales sobre los que se apoya.

La programación orientada a objetos se define como un paradigma de la programación que hace uso de los objetos y de sus interacciones para desarrollar software. Su núcleo por tanto, son los objetos, los cuales se describen como entidades que están formadas por atributos que definen su estado, por métodos que definen su comportamiento y una identidad que los diferencia del resto. Cada objeto también es una instanciación de una clase cuya función es declarar su contenido.

Entre las características propias un lenguaje de programación orientado a objetos se encuentra:

- Herencia: Es la capacidad de los objetos de heredar atributos y métodos de la clase a la que pertenezcan permitiendo así el establecimiento de una estructura jerárquica.
- Polimorfismo: Permite definir clases distintas con atributos o métodos denominados igual, pero que actuaran de forma diferente.
- Encapsulamiento: Esconde el estado de un objeto permitiendo su acceso sólo utilizando los métodos del mismo.
- Abstracción: Se trata de convertir cada objeto en un módulo independiente y autónomo, que dispondrá de sus características propias y esenciales. La abstracción es una aliada de la reutilización.

Java es un lenguaje de muy alto nivel, ya que a pesar de estar fuertemente influenciado por C y C++, hecho que salta a la vista por su sintaxis, está simplificado y elimina herramientas de bajo nivel como los punteros y el control de memoria. Ésto último se lo puede permitir gracias a que dispone de un recolector de basura, cuya función es destruir automáticamente los objetos que se quedan sin ser referenciados, liberando la memoria que usaban.

Una característica fundamental de Java es que sus aplicaciones están normalmente compiladas en un bycode, el cual puede ser interpretado para su posterior ejecución. Esta cualidad es la que permite a un programa en Java ejecutarse en cualquier tipo de dispositivo. Se puede considerar entonces que el código bycode se sitúa entre el código máquina y el código fuente.

2.3 SQLite como Base de Datos Embebida

Estamos ante un sistema de gestión de base de datos relacional cuyo funcionamiento radica en las interconexiones entre los datos almacenados en tablas, de ahí su nombre. Este modelo relacional es el más extendido, pues entre otras virtudes, garantiza la integridad referencial. También resulta comprensible y aplicable facilitando así su normalización, y además proporciona herramientas como por ejemplo, las que evitan la duplicidad de los datos.

Las claves primarias y foráneas, las restricciones y los dominios se encuentran entre sus características propias. Sistemas de gestión tan populares como MySQL, PostgreSQL y Oracle comparten este modelo referencial.

SQLite también es un gestor de base de datos referencial y además, implementa la mayor parte del estándar SQL, pero tiene una particularidad que lo hace diferente.

Los sistemas antes mencionados funcionan bajo una estructura cliente-servidor, es decir, con un proceso independiente de la aplicación, y requiere estar establecido en una máquina que actúe como servidor. La práctica habitual consiste en que el cliente (aplicación) y el servidor (sistema gestor) estén en dispositivos distintos. En lugar de eso, SQLite resulta ser una biblioteca que se integra en la aplicación, pasando a formar parte de la misma.

Su punto más polémico es que mientras los otros sistemas establecen los tipos de valores a columnas enteras (tipo string por ejemplo), SQLite realiza las asignaciones a los valores individualmente. La comunidad está dividida pues un sector ve la particularidad como una innovación por su dinamismo, otro sector opina que es un inconveniente ya que la técnica no es portable a otros sistemas.

SQLite es un proyecto de código abierto actualmente activo y en constante evolución. Principalmente es usada por aplicaciones escritas en C/C++ así que originalmente no podía ser utilizada por software en Java. Pero afortunadamente existen diferentes proyectos independientes que han adaptado la librería para poder ser utilizada en varios lenguajes entres los que se incluye Java. Dado que SQLite es un sistema relativamente desconocido, es de relevante mencionar que es utilizado por Android y Firefox entre otros.

3 Análisis y Planificación

3.1 Requisitos Funcionales

En este punto se enumerarán los requisitos funcionales del sistema con una pequeña descripción de cada uno de ellos. Están divididos en tres categorías: Esenciales, Condicionales y Opcionales.
Los Esenciales son básicos para el funcionamiento de la aplicación y el incumplimiento de cualquiera de ellos supondría el fracaso del proyecto. Los condicionales son considerados importantes, describen funcionalidades relevantes pero no primordiales a la hora de alcanzar los objetivos del proyecto. Finalmente los opcionales suponen funcionalidades añadidas o complementarias del proyecto, su no desempeño no tendrá un impacto real en la culminación de sus propósitos.

3.1.1 Esenciales

- **Creación de Personajes**
  Interfaz dónde se crearán los personajes que posteriormente utilizarán los jugadores. Permitirá introducir datos y restringirá la construcción de sus características y aptitudes de manera que respeten el reglamento del juego

- **Creación de NPC's**
  La aplicación permitirá crear NPC's mediante un formulario en el que se escribirán los datos y donde también se seleccionarán las distintas características. Debido al propósito por el que existen los NPC's se hallarán restricciones impuestas por el reglamento.

- **Gestión de Inventario**
  Se dispondrá de control sobre el inventario de los personajes. Aquí se añadirán y quitarán objetos y también se escogerá el equipamiento del personaje.

- **Gestión de Individuos (Selección, Exploración y Eliminación)**
  Forma de ver todos los individuos existentes en la base de datos, pudiendo disponer de todas sus características individuales. También se podrá eliminar individuos y seleccionar uno en concreto para trabajar con él cuando sea necesario.

- **Generación de Tiradas de Dados**

  Herramienta tipo de tiradas de dados y vSeer el resultado.

- **Realización de Partida**

  El software será capaz de proporcionar una interfaz para jugar una partida. En ella se podrán decidir las acciones de los individuos y realizarlas.

- **Almacenamiento en Base de Datos**

  Todos los datos generados por los usuarios durante la creación de los diferentes individuos se guardarán en una base de datos; además de los generados durante la gestión de los inventarios de los personajes en particular. En la base de datos también figurará información necesaria sobre el juego.

3.1.2 Condicionales

- **Gestión de Turnos por Iniciativa**

  Una vez iniciada una partida los individuos que en esta se hallen podrán ser colocados por orden de iniciativa para que actúen por turnos respetando ese orden.

- **Cálculo de las Acciones Automática**

  Las diferentes acciones realizadas durante la partida serán resueltas automáticamente, se habrán considerado las características correspondientes y tirado los dados adecuados sin que el Máster haya actuado en el proceso.

- **Gestión de Estado de Personajes**

  Se llevará un control de los puntos de golpe de cada individuo en todo momento ya que las acciones repercutirán directamente en ellos. También se permitirá describir el estado y ubicación asociados a cada individuo.

3.1.3 Opcionales

- **Gestión de Grupos**

  La aplicación dispondrá de una funcionalidad para gestionar diferentes grupos en la partida, moviendo los individuos a través de ellos con libertad. Cada grupo dispondrá de una descripción propia, y cuando un grupo esté seleccionado sólo se podrá interactuar con sus integrantes.

- **Mapas Localizadores**

  Serán dispuestos mapas cuadriculados donde figurará la localización de los individuos en la partida. El Master podrá editar estos mapas libremente.

3.2 Requisitos No Funcionales

- **El sistema de juego estará basado en Dungeons & Dragons 3.5**

  La aplicación respetará en la medida de lo posible este reglamento de juego de rol. Los mecanismos básicos del juego quedarán representados por el programa, mientras que otorgará herramientas para que las reglas secundarías y más especificas puedan ser llevadas a cabo.

- **La interfaz ha de ser intuitiva y de fácil manejo**

  Utilizar el programa no deber resultar complicado ni tedioso, porque de lo contrario no se estaría contribuyendo a cumplir el objetivo principal del proyecto, que es el de optimizar el desarrollo del juego.

- **La aplicación ha de ser relativamente flexible**

  El Máster ha de disponer cierta libertad para influenciar en todos los resultados calculados y podrá también resolver los conflictos surgidos –mediante la aplicación- utilizando su propio criterio. De no ser así, el proyecto no podría representar el concepto de sistema de juego de rol desagradando así a los potenciales usuarios del software.

3.3 Método de Desarrollo

El proyecto se ha desarrollando utilizando lo que se asemejaría a un modelo iterativo.

Como resultado, el desarrollo queda dividido en interacciones de unas dos semanas de duración y cada una de ellas concluía con una reunión entre el desarrollador (Alumno) y el director del proyecto.

Con el fin de constituir un buen seguimiento del proyecto, durante las reuniones se realizan varias puntos:

- Se analiza qué se ha hecho durante la iteración y comprueba si se han cumplido las expectativas. En caso de no cumplirlas se propone una solución.
- Se comprueba si se está siguiendo con éxito la planificación temporal para poder encauzar los desajustes si los hubiera
- Se estudia si puede haber cambiado algún requerimiento.
- Con todo lo comentado, se preparan objetivos para la nueva iteración que llega.

Dicho método supone también otorgar un cierto grado de adaptabilidad al desarrollador para poder utilizar como solución los nuevos conocimientos que haya adquirido durante el proyecto, ya que él estará en constante evolución.

Finalmente, hay que tener en cuenta que debido a se trata de un proyecto de final de carrera, es muy probable que el desarrollador se encuentre con problemas que no sea capaz de superar con sus conocimientos actuales. Por lo tanto el desarrollador necesitará una pequeña investigación previa para poder vencer el obstáculo. A causa de esto, la metodología no puede ser todo lo rígida que debiera, ya que será necesario ser flexible en las iteraciones, en la carga de trabajo por cada una de ellas y en la forma de proceder.

3.4 Herramientas de Desarrollo

Se explicará seguidamente los motivos por los que se ha optado por utilizar determinadas herramientas de desarrollo enfatizando en cuales son ventajas aportaban en pos de otras similares.

3.4.1 Java

El presente proyecto podría haberse desarrollado en varios lenguajes. Se alzaba como principal alternativa el usar PHP para realizar una página web que realizase las funciones del programa. Se quería utilizar la programación orientada a objeto, y PHP era capaz de satisfacer esta necesidad. A pesar de ello finalmente se optó por utilizar Java ya que ya había programado anteriormente en este lenguaje y por tanto disponía de más conocimientos.

Además Java cuenta con la biblioteca gráfica Swing que facilita en gran medida el diseño de la aplicación y tiene pocas limitaciones.

3.4.2 NetBeans

En el mundo de las plataformas de desarrollo para Java existen dos principales: NetBeans y Eclipse. Cada una con una serie de ventajas e inconvenientes así que el uso de cualquiera de las dos hubiera sido viable para este proyecto.

Finalmente se optó por utilizar NetBeans gracias a que dispone de un editor de interfaces gráficas que está en mejor consideración que los que puede ofrecer Eclipse.

3.4.3 SQLite

Se pretende que la aplicación resultante del proyecto sea potable, independiente y ligera, por lo tanto el uso de una base de datos convencional como MySQL ofrecía muchas limitaciones en estos aspectos.

SQLite cumple los requisitos nombrados de forma eficaz. Al ser una base de datos embebida, el motor de esta se halla integrado en el programa por lo que su portabilidad es la misma que

cualquier otra aplicación en Java. Se muestra también independiente ya que no necesita conexión a un servidor que gestione la base de datos. El propio sistema donde se esta ejecutando sería capaz también de realizar la función del servidor, pero choca con la necesidad de una aplicación ligera ya que seria necesario instalar el motor de la base de datos allá donde se quiera utilizar el programa.

La candidatura de SQLite pudo haber sido rechazada porque las versiones para Java andaban retrasadas respecto a la versión actual de SQLite (la que funciona para C++). Concretamente las versiones más actualizadas para Java no soportaban el uso de Foreign Keys (Claves Foráneas). También es importante decir que sólo permite recorrer hacia delante el curso en las variables de resultado de consulta, pero nada que no se pueda solucionar con un buen algoritmo.

Afortunadamente después de una búsqueda exhaustiva a través de la red se encontró una nueva versión alternativa, nacida en diciembre del 2009, proporcionada por Xerial Project.

3.4.4 SQLite Expert Personal

Un pequeño problema que se presentaba a la hora de gestionar la base de datos es que mientras que para otros motores de base de datos podíamos disponer de multitud de aplicaciones para este propósito, para SQLite no había tanta diversidad.

Después de probar varias aplicaciones -una de ellas funcionaba como un plugin de firefox- se optó por escoger SQLite Expert Personal. Los motivos fueron que su manejo era sencillo e intuitivo y que no poseía muchas más opciones de las que se necesitaban para este proyecto, su utilización era viable gracias a que disponía de una versión 'Personal' que es totalmente gratuita

3.4.5 I-Text

Se trata de una librería de java que permite manipular ficheros PDF en tiempo de ejecución desde el código.

Se barajó la posibilidad de utilizar el plugin de Netbeans iReport con JasperReports de manera con los cuales se pueden generar reportes, con la herramientas disponibles se pueden construir las plantillas que se rellenaran dinámicamente después. Tras varias pruebas se optó por utilizar i-Text debido a que su uso era menos engorroso y cumplía las espectativas de forma eficaz.

Al utilizar I-Text se ha construido todo el PDF directamente desde código, y la tarea pudo realizarse gracias a todo el soporte existente en la página oficial de I-Text.

3.4.6 Substance

Substance es un conjunto de librerías para Java que permite cambiar la apariencia de las interfaces gráficas Swing. Por defecto son cuatro aspectos distintos (llamados look and feel) los que son propuestos por Swing, Substance otorga nuevas alternativas y añade la posibilidad de personalizar muchas características mediante código. Una de las más llamativas es poder cambiar el *Watermark* (La imagen de fondo de la interfaz).

Se ha escogido substance sobre otras librerías de las mismas características por gusto personal.

3.5 Riesgos

Es importante tener en todo momento presente los posibles problemas que puedan surgir y que se conviertan en importantes obstáculos a superar si se quiere finalizar un proyecto con éxito.

- **Planificación temporal optimista**
  Posiblemente el riesgo más probable. Una mala planificación puede derivar en que no se alcancen los plazos de tiempo establecidos. La solución posible es omitir funcionalidades secundarias de la aplicación para poder finalizar lo esencial.

- **Diseño fallido**
  Debido a la poca experiencia realizando este tipo de proyectos es posible que el diseño de la aplicación no resulte el adecuado y imposibilitando la finalización del proyecto de no rediseñar la aplicación. Todo el trabajo realizado en el diseño erróneo será irrecuperable. La importancia del fallo determinará el tiempo que se empleará en solucionar el problema, rediseñando las partes afectadas. Con tal de conseguir los plazos es posible que se tenga omitir alguna funcionalidad secundaria.

- **Falta de conocimientos**

  La falta de conocimientos para desarrollar alguna parte del programa puede tener consecuencia en un estancamiento del avance del proyecto. Se deberá emplear una cantidad indeterminada de tiempo para adquirir la documentación que ayude a superar el problema.

- **Complejidad inesperada**

  Supondrá un problema si alguna funcionalidad es más compleja de llevar a cabo de que se había previsto. La forma mas viable de resolver el problema sería rediseñando esta parte para aumentar su simplicidad y que de esta manera se facilite su desarrollo.

- **Herramientas de desarrollo inadecuadas**

  Este problema tiene fácil solución pues la solución sería documentarse sobre nuevas herramientas o algunas ya analizadas que puedan satisfacer las necesidades del proyecto.

- **No se realizan correctamente las pruebas**

  Unas pruebas mal realizadas pueden provocar que no se haya detectado gran cantidad de errores del programa. Será necesario dar garantías de calidad, pero la complejidad de solucionar estos errores puede ser muy variada.

- **Cambio de requerimientos**

  Este riesgo se incrementa debido a la metodología de desarrollo utilizada. Afortunadamente este problema no tiene un impacto crítico en el proyecto ya que se soluciona volviendo a planificar estos nuevos requerimientos.

3.6 Planificación Temporal

Una de las claves en el éxito de un proyecto es una buena planificación temporal, ya que gracias a ella se facilita el llevar un control del transcurso del proyecto. Esta práctica, entre otras cosas, permitirá percibir los imprevistos y desajustes que surjan habilitando una rápida reacción que solucione el problema.

Se ha realizado una planificación poco optimista, ya que se presume que el programador va a requerir más tiempo de lo normal en el desarrollo de la aplicación dada su inexperiencia. Además la fecha de finalización queda prevista un mes antes de la fecha límite de la entrega del proyecto (29 de Junio de 2010) por si hubiera dificultades.

La organización temporal consta de las fases previas al desarrollo habituales en la informática, como son un estudio previo del contexto, las fases de análisis y las de diseño. Debido a lo mencionado sobre el programador en el párrafo anterior, también se ha incluido una fase de formación previa sobre las tecnologías a emplear y además se ha otorgado mucho tiempo a la etapa de desarrollo. Finalmente la fase de pruebas y generación de la memoria antes de concluir.

El proyecto comienza el 10 de Noviembre de 2009, y finaliza el 21 de Mayo de 2010. El calendario laboral no incluye los fines de semana y cada día de trabajo equivale a cuatro horas. El total de duración del proyecto son 780 horas.

En los siguientes gráficos queda representada la planificación temporal con su respectivo diagrama de Gantt.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	Proyecto	137,33 días	mar 10/11/09	jue 20/05/10
2	Inicio del Proyecto: Asignación y Matriculación	1 hora	mar 10/11/09	mar 10/11/09
3	Documentación Previa	2,67 días	mar 10/11/09	jue 12/11/09
4	Investigación sobre los Juegos de Rol	3 horas	mar 10/11/09	mié 11/11/09
5	Planteamiento del Sistema de Juego	2 horas	mié 11/11/09	mié 11/11/09
6	Estudio del Reglamento del Juego	3 horas	jue 12/11/09	jue 12/11/09
7	Análisis de la Aplicación	5 días	vie 13/11/09	jue 19/11/09
8	Análisis de Requerimientos	6 horas	vie 13/11/09	lun 16/11/09
9	Estudio de Aplicaciones OpenCode Similares	3 horas	mar 17/11/09	mar 17/11/09
10	Análisis de Tecnologías a Utilizar	6 horas	mié 18/11/09	jue 19/11/09
11	Formación Previa	7,67 días	vie 20/11/09	mar 01/12/09
12	Programación Orientadas a Objeto	6 horas	vie 20/11/09	lun 23/11/09
13	Aprendizaje sobre Bases de Datos	5 horas	mar 24/11/09	mié 25/11/09
14	Aprendizaje del Lenguaje de Programación	10 horas	mié 25/11/09	lun 30/11/09
15	Familiarización con el Entorno de Desarrollo	2 horas	mar 01/12/09	mar 01/12/09
16	Diseño de la Aplicación	19,33 días	mar 01/12/09	lun 28/12/09
17	Diseño de la Base de Datos	13 horas	mar 01/12/09	lun 07/12/09
18	Diseño de la Aplicación	30 horas	lun 07/12/09	lun 21/12/09
19	Diseño de la Interfaz	15 horas	lun 21/12/09	lun 28/12/09
20	Desarrollo de la Aplicación	83 días	lun 28/12/09	jue 22/04/10
21	Preparación del Entorno de Desarrollo	3 horas	lun 28/12/09	mar 29/12/09
22	Construcción de la Base de Datos	7 horas	mar 29/12/09	jue 31/12/09
23	Desarrollo de Clases Internas	25 horas	jue 31/12/09	mar 12/01/10
24	Módulo Construcción de Personajes	13,33 días	mié 13/01/10	lun 01/02/10

Página 1

Figura 3.1: Planificación + Diagrama de Gantt (1º parte)

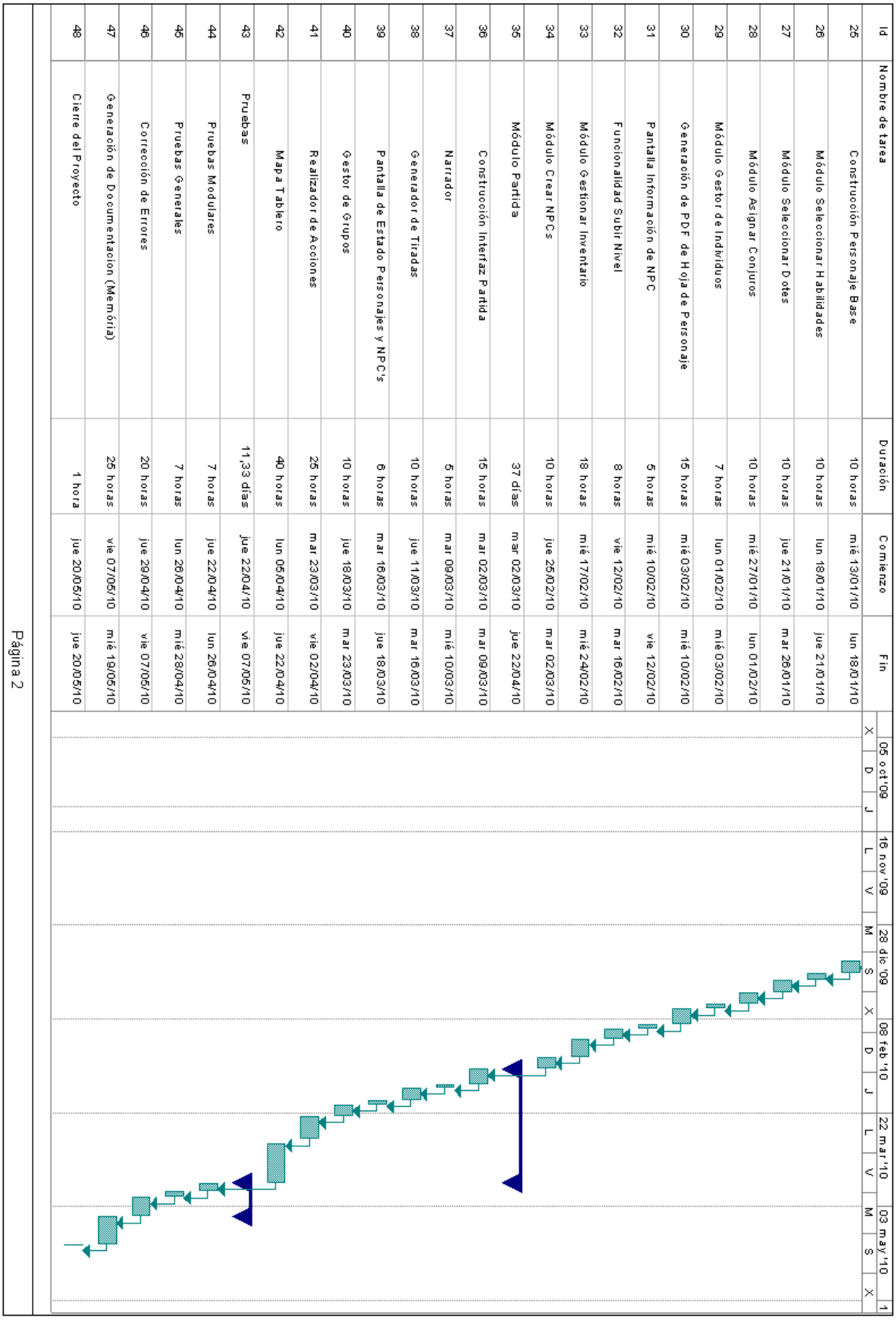


Figura 3.2: Planificación + Diagrama de Gantt (2º Parte)

4 Diseño

4.1 UML: Casos de Uso

Los casos de uso se utilizan para describir las funcionalidades de un software desde el punto de vista del usuario. Debido a esta perspectiva se puede extraer una idea general de las capacidades de la aplicación de manera sencilla. Cabe indicar que se ha intentado realizar casos de uso más generales que bien se podrían dividir en varios.

4.1.1 Diagramas de Casos de Uso

A continuación se muestran dos diagramas que cumplen el propósito arriba mencionado.

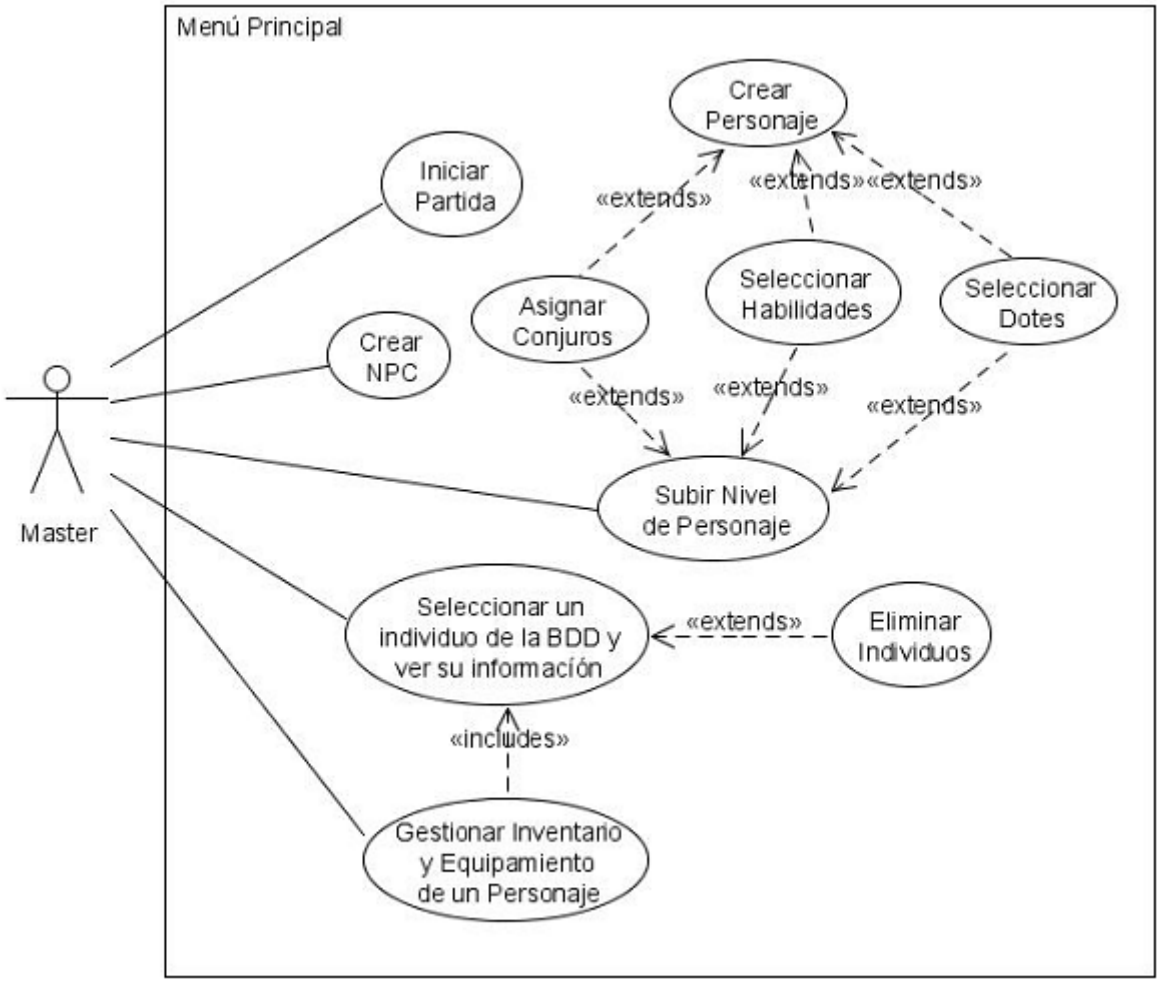


Figura 4.1: Casos de uso menú Principal

La figura 4.1 muestra las funcionalidades del menú principal sin haber iniciado una partida.

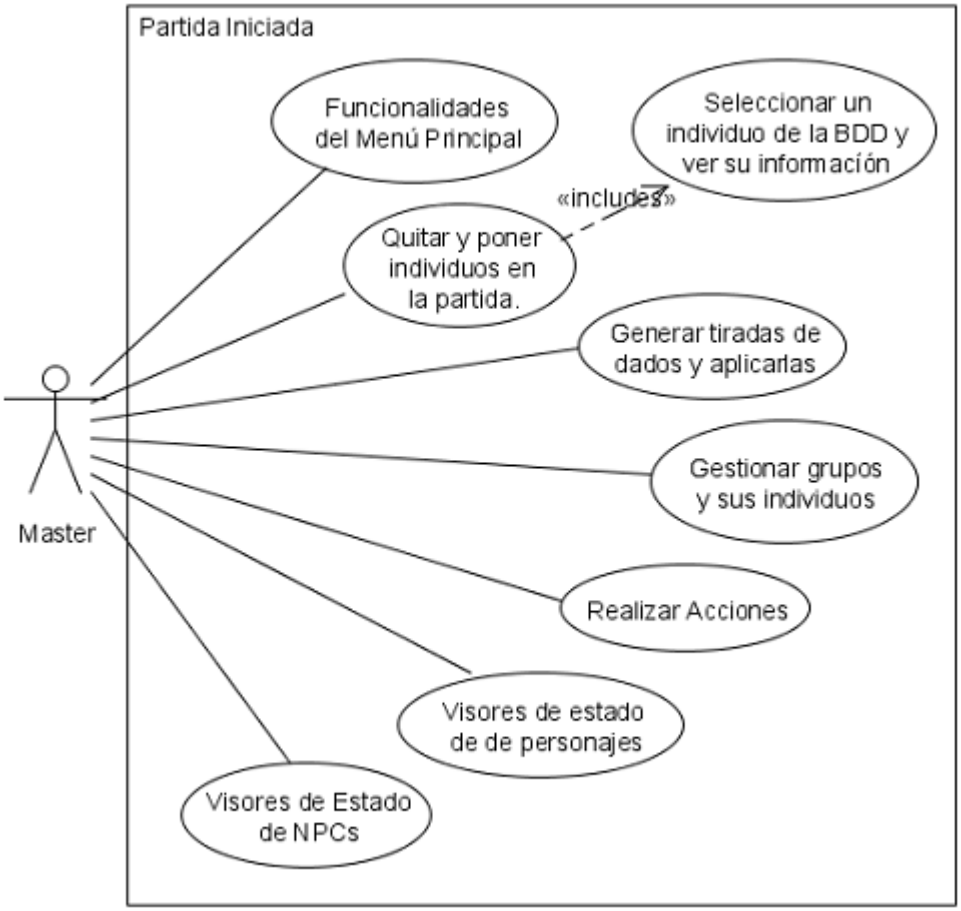


Figura 4.2: Casos de uso partida

La figura 4.2 enseña los casos de uso habidos una vez dentro de la interfaz de partida. El caso de uso '*Funcionalidades del Menú Principal*' incluye el diagrama de la figura 4.1 exceptuando el caso de uso '*Eliminar Individuos'*.

4.1.2 Especificación de Casos de Uso

A continuación se especificarán los casos de uso. Donde se detallarán los pasos necesarios para el cumplimiento de cada caso de uso. Dicha especificación constará de:

- Descripción del caso de uso.
- Flujos de eventos, en cada uno de los cuales se encuentra un flujo básico que describe el comportamiento normal del caso de uso, y sus flujos alternativos. A veces el orden de los pasos de un flujo principal no es estricto. Ya que se ha optado por intentar plasmar las posibilidades de cada caso de uso, antes que realizar una lista interminable de flujos alternativos que en realidad no dejan de ser normales en su comportamiento.
- Precondiciones, que indican las premisas para que se suceda el caso de uso y poscondiciones, que describen la situación después del caso de uso.

a) Crear Personaje

Descripción

El usuario crea un personaje introduciendo sus datos y características según las reglas.

Flujo de eventos

Flujo Básico

1. El sistema muestra un formulario con botones y campos de texto donde introducir los datos del personaje.
2. El usuario rellena el formulario principal, y si lo desea también cambia la imagen y rellena el subformulario tras el botón 'Datos de Personaje'.
3. El usuario pulsa el botón 'Método Legal'.
4. El sistema muestra un formulario con un combo de texto con 6 tiradas de característica generadas al azar.
5. El usuario distribuye las tiradas seleccionándola desde el combo y pulsando el botón 'asignar', cuando acaba pulsa el botón 'Aceptar'.
6. El Sistema, asigna las características, cierra la ventana y habilita el botón 'Aumentar Características'
7. El Usuario pulsa el botón 'Aumentar Características'.
8. El Sistema muestra una ventana con botones que permiten aumentar cierto número de características determinado por el nivel del Personaje.
9. El usuario Aumenta las características que pueda y quiera, y pulsa el botón 'Aceptar'
10. El Sistema asigna las características resultantes, cierra la ventana y habilita la opción 'Aplicar'.
11. El Usuario Pulsa el Botón 'Aplicar'.
12. El Sistema Guarda el Personaje en la Base de Datos y habilita los botones 'Seleccionar Dotes', 'Seleccionar Habilidades', 'Asignar Conjuros' y 'Salir'.

Flujo Alternativo 1: Paso 2

1. El usuario pulsa el botón 'Personalizar Características' en vez de 'Método Legal'.
2. El sistema muestra una ventana donde colocar aracterísticas de forma libre.
3. El usuario coloca los características y pulsa 'Aceptar'.
4. El sistema asigna las características, cierra la ventana y habilita el botón 'Aumentar Características'. Al paso 7.

Flujo Alternativo 2: Paso 5

1. El usuario pulsa el botón Generar Tiradas, para volver a redistribuir las tiradas.
2. Sistema rehace el combo de texto con 6 nuevas tiradas, deshabilita el botón 'Aceptar' si el usuario había finalizado y selecciona la primera característica

Postcondiciones

- El personaje creado queda guardado en la base de datos.
- El personaje ya no se podrá modificar excepto subiéndolo de nivel.

b) Crear NPC

Descripción

El usuario crea un NPC introduciendo sus datos y determinando sus atributos.

Flujo de Eventos

Flujo Básico

1. El sistema muestra un formulario con diversos campos y tablas para rellenar con los datos.
2. El usuario rellena el formulario y pulsa aceptar.
3. El sistema guarda el NPC en la Base de Datos y sale de la interfaz.

Flujo Alternativo 1: Paso 2.

1. El usuario pulsa el botón 'Crear Conjuro' mientras rellenaba el formulario.
2. El sistema muestra otra ventana dónde establecer la descripción del nuevo conjuro.
3. El usuario describe el conjuro y pulsa aceptar.
4. El sistema almacena el nuevo conjuro en la base de datos y lo muestra en la tabla para poder ser seleccionado, finalmente cierra la ventana.

Flujo Alternativo 2: Paso 2

1. El usuario selecciona un conjuro editado pulsa el botón 'Borrar Conjuro'.
2. El sistema borra el conjuro seleccionado.

Flujo Alternativo 3 : Paso 2

1. El usuario selecciona un conjuro propio de las reglas y pulsa el botón 'Borrar Conjuro'.
2. El conjuro no se borra.

Postcondiciones

- El NPC queda almacenado en la base de datos.
- El NPC ya no se podrá modificar

c) Subir Nivel Personaje

Descripción

Permite subir de nivel a un personaje, aumentando características si el nuevo nivel lo permite y otorgando acceso a Seleccionar Habilidades, Seleccionar Dotes y Asignar Conjuros.

Flujo de Eventos

Flujo Básico

1. El sistema muestra una ventana dónde permite subir puntos de característica si el nuevo nivel lo permite.
2. El usuario rellena sube las características mediante los botones y puede acceder a cambiar las características mencionadas en la descripción, finalmente pulsa el botón aceptar.
3. El sistema almacena los cambios en el personaje.

Precondiciones

- Se ha de haber seleccionado un personaje al que subir de nivel.

Postcondiciones

- Se establecerán en la base de datos los cambios realizados en el personaje.

d) Asignar Conjuros

Descripción

Dispone de una interfaz con la que asignar los conjuros de los que el personaje puede disponer mostrando la descripción de cada uno.

Flujo de Eventos

Flujo Básico

1. El sistema despliega una interfaz que contiene la lista de conjuros de cada nivel y botones para asignar estos conjuros y mostrará la descripción de cada uno.
2. El usuario selecciona los conjuros deseados mediante los botones, también puede seleccionar un conjuro de dominio, finalmente pulsa el botón aceptar.

3. El sistema almacena los nuevos conjuros seleccionados y cierra la ventana.

Precondiciones

- Ha de haber un personaje referenciado, desde la interfaz crear personaje o subir nivel

Postcondiciones

- El personaje quedará almacenado en la base de datos con los nuevos conjuros guardados.

e) Seleccionar Habilidades

Descripción

Posibilita otorgar puntos de rango a las habilidades y almacenarlas.

Flujo de eventos

Flujo Principal

1. El sistema muestra una interfaz con una lista de habilidades, las cuales disponen de botones para aumentar o reducir sus rangos, y su modificador como consecuencia. También muestra los puntos de habilidad disponibles restantes. En caso de que el personaje sea de clase 'Guerrero' se muestran también las habilidades de guerrero restantes.
2. El usuario aumenta los rangos según su preferencia mediante los botones y pulsa el botón aceptar.
3. El sistema cierra la ventana y almacena los nuevos datos.

Flujo Alternativo: Paso 3

1. El sistema no continúa si hay un número negativo de puntos de habilidad disponibles, muestra un dialogo advirtiendo de esto y no hace nada.

Precondiciones

- Ha de haber un personaje referenciado

Postcondiciones

- Se almacenan las habilidades aprendidas con sus modificadores y todas las habilidades que no requieren puntos para ser utilizadas.

f) Seleccionar Dotes

Descripción

El caso de uso describe la selección de dotes para el personaje por parte del usuario.

Flujo de eventos

Flujo Principal

1. El sistema muestra una ventana donde se muestran todas las dotes habilitadas que se pueden obtener, mostrando la descripción y requisitos de cada una.
2. El usuario selecciona las dotes deseadas.
3. Cada vez que el usuario selecciona una dote el sistema habilita las nuevas dotes disponibles.
4. El sistema almacena las dotes adquiridas y cierra la ventana.

Flujo Alternativo: paso 4

1. Si el número de dotes disponibles es inferior a 0, el sistema advierte de esto y vuelve al paso 2.

Precondiciones

- El personaje tiene que estar referenciado

Postcondiciones

- El personaje queda almacenado en la base de datos.

g) Seleccionar un individuo de la BDD y permite ver su información

Descripción

Este caso de uso permite seleccionar un individuo y ver su información, si es un NPC mediante una interfaz, y si es un personaje mediante un PDF.

Flujo de eventos

Flujo Principal

1. El sistema muestra una tabla con todos los personajes o NPC's.
2. El usuario selecciona un individuo.

3. El usuario pulsa el botón aceptar.
4. El sistema obtiene la referencia al personaje y cierra la ventana.

Flujo Alternativo 1: paso 2

1. El usuario pulsa el botón ver información en vez de aceptar.
2. El sistema generará un fichero PDF si es un personaje o mostrará los datos mediante una interfaz si es un NPC.

Flujo Alternativo 2: paso 2

1. El usuario no selecciona ningún individuo y luego pulsa ver información o aceptar.
2. El sistema no puede realizar la acción y muestra un mensaje de advertencia.

Postcondiciones

- Quedará un usuario referenciado de manera que el sistema pueda trabajar con él.

h) Eliminar Individuos

Descripción

Elimina un individuo desde la interfaz Seleccionar NPC o Seleccionar Personaje.

Flujo de Eventos

Flujo Básico

1. El usuario selecciona un individuo y pulsa el botón Eliminar Individuo/NPC.
2. El sistema elimina de la base de datos y de la tabla al individuo seleccionado.

Flujo Alternativo: paso 1

1. El usuario pulsa el botón eliminar sin seleccionar un individuo.
2. El sistema advierte de que no hay ningún individuo seleccionado.

Precondiciones

- No se puede eliminar un individuo una vez entrado en el menú partida.

Postcondiciones

- El individuo es eliminado de la base de datos.

i) Gestionar Inventario y equipamiento de un personaje

(Este caso de uso contiene muchos posibles flujos, sólo los principales son contemplados en el flujo de eventos).

Descripción

Permite al usuario agregar y quitar elementos de su inventario, y equipar los que sean equipables. El sistema no permite que se coloquen objetos donde no se puede según las reglas.

Flujo de Eventos

Flujo Principal

1. El sistema muestra una interfaz donde se muestra el inventario del usuario en una tabla y los elementos equipados.
2. El usuario pulsa el botón agregar.
3. El sistema muestra una ventana con la lista de elementos habidos en la lista de la base de datos, determinados por el tipo que el usuario quiera.
4. El usuario selecciona un elemento y pulsa el botón agregar, tantas veces como quiera.
5. El sistema agrega los elementos a la tabla.
6. El usuario pulsa el botón salir o cierra la ventana.
7. El sistema guarda los elementos habidos en el inventario en la base de datos y les otorga un numero de referencia, y cierra la ventana.
8. El usuario selecciona un elemento de la tabla y pulsa el botón quitar.
9. El sistema elimina el elemento de la tabla.
10. El usuario selecciona un elemento y lo equipa (relaciona) en una parte del cuerpo.
11. El sistema -que habilita solo los botones equipar para las partes donde se puede equipar dicho objeto- equipa el objeto en el lugar respectivo al botón pulsado sustituyéndolo si había algo equipado antes.
12. El usuario pulsa salir.
13. El sistema guarda el inventario actual en la base de datos y los objetos equipados, finalmente cierra la ventana.

Flujo Alternativo 1: Paso 4, Paso 8

1. El usuario ha pulsado el botón agregar sin un elemento seleccionado.
2. El sistema avisa que no hay un elemento seleccionado.

Flujo Alternativo 2: Paso 9

1. Si el elemento a eliminar esta equipado actualmente el sistema no permite que sea eliminado.

Precondiciones

- Se ha de seleccionar el personaje.

Postcondiciones

- Se guardarán los cambios realizados.

j) Quitar y poner individuos de la partida

Descripción

Abre la interfaz partida con todas las opciones de la misma.

Flujo de Eventos

Flujo Principal

1. El usuario pulsa el botón Partida.
2. El sistema despliega la venta principal de partida.

Postcondiciones

- La partida queda iniciada.

k) Quitar y poner individuos de la partida

Descripción

Se utiliza para quitar y poner individuos en la partida.

Flujo de Eventos

Flujo Principal

1. El usuario utiliza la opción cargar del menú.
2. El sistema acude al caso de uso 'seleccionar individuo' para escoger el individuo y lo carga en la lista de la partida.
3. El usuario utiliza la opción quitar del menú.
4. El sistema muestra una tabla con los individuos en partida.
5. El usuario selecciona un individuo y pulsa quitar.
6. El sistema elimina al individuo de la partida y cierra la ventana.

Flujo Alternativo: Paso 5

1. El usuario pulsa quitar sin seleccionar un individuo.
2. El sistema muestra un dialogo avisando de que no hay individuo seleccionado.

Precondiciones

- Es necesario haber iniciado una partida.

Postcondiciones

- Se actualizará la lista de usuarios de la partida con los cambios realizados.

l) Generar Tiradas de dados y Aplicarlas

Descripción

El sistema muestra una ventana donde se pueden realizar todo tipo de tiradas de dados, aplicar los resultados a los puntos de golpe de los individuos y mostrar lo sucedido en la interfaz narrador si se desea.

Flujo de Eventos

Flujo Principal

1. Se muestra una ventana con opciones para realizar una tirada.
2. El usuario selecciona el tipo de tirada de daos y pulsa 'Tirar'.
3. El sistema calcula el resultado y lo muestra en un campo de texto.
4. El usuario configura que hacer con la tirada, si la quiere mostrar por el narrador, aplicarla a un individuo seleccionado y si quiere que sea de curación o daño.

5. El sistema aplica la tirada según los parámetros establecidos.

Postcondición

- No sucederá nada, se añadirá la descripción y/o se aplicarán los resultados según los parámetros.

m) Gestionar Grupos y sus Individuos

Descripción

Se utiliza para crear grupos que contendrán individuos los cuales solo se podrán relacionar entre sí de estar en el mismo. También se podrá añadir una pequeña descripción a cada grupo.

Flujo de Eventos

Flujo Principal

1. El sistema dispone una interfaz con una tabla con los individuos habidos en el grupo seleccionado y otra tabla con todos los individuos de la partida.
2. El usuario pulsa el botón 'Nuevo grupo'.
3. El sistema abre una ventana con una caja de texto.
4. El usuario escribe el nombre del nuevo grupo y pulsa aceptar.
5. El sistema crea un nuevo grupo vacío y lo añade a la lista de grupos.
6. El usuario selecciona un individuo y pulsa añadir.
7. El sistema añade el individuo seleccionado al grupo.
8. El usuario selecciona un individuo de un grupo y pulsa quitar personaje.
9. El sistema quita al personaje del grupo.
10. El usuario selecciona un grupo y pulsa eliminar grupo, los individuos que hubiese en el grupo quedan desvinculados de él.
11. El sistema elimina el grupo.

Flujo Alternativo 1: Paso 4

1. El usuario introduce un nombre de grupo que ya existe.
2. El sistema avisa de que el nombre ya existe y no permite la creación del grupo.

Flujo Alternativo 2: Paso 6, Paso 8 y Paso 10

1. El usuario no ha seleccionado un elemento.
2. El sistema avisa de que no hay elemento seleccionado y avisa con una ventana de diálogo.

Postcondiciones

- Los grupos y como están constituidos quedan actualizados según los cambios.

n) Ver estado de Personajes

Descripción

Permite ver el estado de los personajes de la partida, y poner y quitar efectos.

Flujo de Eventos

Flujo Principal

1. El sistema dispone una ventana con una pestaña por cada personaje de la partida.
2. El usuario escribe en un campo dispuesto un efecto y pulsa aplicar.
3. El sistema añade el efecto al personaje y lo muestra en la tabla.
4. El usuario selecciona un efecto y pulsa eliminar efecto.
5. El sistema quita el efecto del personaje.

Flujo Alternativo: Paso 4

1. El usuario no ha seleccionado un elemento.
2. El sistema avisa de que no hay elemento seleccionado y avisa con una ventana de diálogo.

Postcondiciones

- Los efectos quedan añadidos en el personaje durante la partida.

o) Ver estado de NPC

Descripción

Permite ver el estado de los NPC de la partida seleccionándolos en una tabla, y poner y quitar efectos de ellos.

Flujo de Eventos

Flujo Principal

1. El sistema dispone una tabla con los NPC's de la partida.
2. El usuario selecciona un NPC, escribe en un campo dispuesto un efecto y pulsa aplicar.
3. El sistema añade el efecto al NPC y lo muestra en la tabla de efectos.
4. El usuario selecciona un NPC y un efecto, y pulsa eliminar efecto.
5. El sistema quita el efecto del NPC.

Flujo Alternativo: Paso 2 y Paso 4

1. El usuario ha dejado un elemento sin seleccionar.
2. El sistema avisa de que no hay elemento seleccionado y avisa con una ventana de diálogo.

Postcondiciones

- Los efectos quedan añadidos en los NPCs durante la partida.

p) Realizar Acciones

Descripción

El sistema posibilita realizar diversas acciones con o sin objetivo, realizando los cálculos de los resultados de estas y mostrándolos en el narrador si así se desea. También se muestran los efectos del personaje y permite gestionar el orden de los individuos según su iniciativa.

Flujo de eventos

Flujo Principal

1. El sistema muestra una pantalla donde se disponen todas las acciones a realizar y se puede elegir el personaje que la realizará y contra qué.
2. El usuario selecciona un grupo.
3. El sistema muestra todos los usuarios del grupo elegido.
4. El usuario selecciona un individuo y describe la acción de movimiento que se realizara si quiere.
5. El sistema muestra todas las acciones que puede realizar el individuo y coloca como objetivos posibles a todos los individuos que estén en su mismo grupo.
6. El usuario escoge una acción a realizar.
7. El sistema habilita los objetivos posibles de esa acción (Individuo, Dificultad o sin objetivo).

8. El usuario escoge un objetivo entre los disponibles y pulsa ejecutar acción.
9. El sistema realiza el cálculo de la acción mostrando el resultado y la descripción en el narrador si el usuario lo ha indicado. Seguido el sistema cambia al siguiente individuo en al lista de prioridad.

Flujo Alternativo 1: Paso 3

1. Si el usuario tiene la opción respetar turnos el sistema no permite elegir el individuo que realizará la acción ya que eso se determina según el orden establecido por la iniciativa de cada personaje del grupo.

Flujo Alternativo 2: Cualquier Paso de usuario

1. El usuario pulsa ejecutar turno.
2. El sistema selecciona el siguiente individuo en la lista del grupo.

Flujo Alternativo 3: Cualquier Paso de usuario

1. El usuario activa doble acción de movimiento.
2. El sistema deshabilita todas las acciones y individuos objetivo.
3. El usuario pulsa ejecutar.
4. El sistema muestra la descripción de la acción de movimiento en el narrador si así se quiere y cambia al siguiente individuo.

Precondiciones

- Para poder realizar acciones normales sobre un objetivo individuo ha de haber más de un individuo en el mismo grupo.
- Para realizar cualquier tipo de acción ha de haber al menos un individuo cargado en partida.

Postcondiciones

- Se realizarán todo tipo de cambios en el estado de los personajes propiciado por las acciones.

4.2 Arquitectura del Sistema

El proyecto se constituye por una aplicación y su correspondiente base de datos. La aplicación es un ejecutable de Java, lo cual otorga la versatilidad de que cualquier sistema con el software básico de Java instalado pueda ejecutarlo. Y la base de datos está embebida, o empotrada, gracias a la naturaleza de SQLite, con lo cual el programa es totalmente independiente.

Este apartado se centrará en la aplicación pues los entresijos de la base de datos serán comentados en el siguiente apartado.

Describiendo la principal base del programa, la clase individuos y sus subclases

La parte central del programa es la clase 'Individuo' de la cual también heredan las clases 'Personajes' y 'NPCs'. De esta forma se consigue poder tratar a los Personajes y NPCs de la misma forma aunando los comportamientos y características comunes en la clase 'Individuo'; de esta forma entonces, se puede acceder a sus particularidades acudiendo a su clase específica.

Así pues, por cada individuo que haya en una partida existirá una instancia de su clase con su correspondiente subclase asociada.

La función principal de estas tres clases resulta ser la de devolver todos los datos extraídos de la base de datos referentes a los individuos, además de también, proporcionar métodos para generar nuevos datos a partir de operar con los datos base. Un paso para llegar a este fin, es que cada objeto individuo tiene un método que recoge todos sus datos de la base de datos, y los asigna a sus atributos, actualizándose de esta forma.

También son encargadas de dos objetivos más. El primero es la de controlar el estado actual de los individuos, es decir, sus Puntos de Golpe (vida) actuales y una descripción de su estado y ubicación durante la partida. Y el segundo, y muy importante, es el de otorgar una forma de conexión a la base de datos donde sea necesario acceder a ella. Esto lo consigue porque se dispone de alguna instancia de la clase 'Individuo' en cualquier clase del programa al estar trabajando en todo momento con ellas.

Estructura de la aplicación, el funcionamiento a través de sus diversas interfaces

La aplicación esta organizada con diversas interfaces de usuario con funciones específicas cada una. Se constituye entonces de editores cuya función es la de proporcionar herramientas para construir individuos y gestionar su inventario; de las interfaces con el fin de gestionar los individuos, que permiten seleccionar un individuo, o eliminarlos; y de la interfaz de la partida.

En la creación de personajes, la aplicación proporciona la interfaz que de forma mayoritariamente intrínseca contiene las reglas del juego, ya que estas reglas son las que están describiendo su comportamiento. Por ejemplo, un personaje de una clase que no sea capaz de aprender conjuros según las reglas, en la aplicación tampoco podrá asignarlos, porque la interfaz habrá inhabilitado las opciones para ello. Dicha situación no sucede en la creación de NPC's, ya que la idea de su existencia es que puedan ser construidos con total libertad y con mucha más sencillez.
Debido a que no todas las reglas habidas en el reglamento pueden representarse a través del comportamiento de las interfaces, las restantes serán obtenidas mediante consultas a la base de datos.

Continuando sobre la creación; el sistema a la hora de crear un individuo, lo que hace para empezar, es obtener un objeto individuo vacío, con referencia 0, que ni siquiera está almacenado todavía en la base de datos. Después de haber introducido las opciones básicas en el formulario y haberlas aplicado, el sistema cambia esta referencia y guarda en la base de datos todos los parámetros introducidos, para seguidamente ejecutar su método construir, actualizándose el objeto.

Cuando se requiere referenciar un objeto 'individuo' concreto, la forma de proceder es que deberá seleccionar mediante la interfaz de gestión un individuo en concreto de los que haya en la base de datos, dicha interfaz de gestión creará un objeto con el individuo seleccionado y enviará el objeto a donde corresponda.

Organizando la partida y su interfaz correspondiente

El transcurso de la partida se organiza utilizando la interfaz partida. Esta interfaz funciona a modo de escritorio, donde, a través del menú, se añaden las diferentes ventanas internas las cuales otorgan funcionalidades.

Obviamente la partida necesita disponer de los individuos para funcionar, por lo que estos se pueden cargar como se describió en el anterior subapartado. Lo que realmente está sucediendo cuando se carga un individuo es que él está siendo añadido una lista de objetos habida en la clase de la interfaz partida. De esta forma se los controla fácilmente y cuando se quiera quitar un individuo de la partida solamente se ha de quitar de la lista.

De la misma manera que existe una lista de individuos para controlar los individuos de la partida, también existe otra lista por cada grupo. Para ello existe la clase grupo donde cada objeto suyo contiene una lista de individuos, la clase proporciona métodos para devolverlos y para ordenarlos según la iniciativa de cada uno. Este último propósito se cumple gracias a la utilización del método de ordenación Selection Sort. Los grupos contenidos en la partida también se controlan gracias a que también existe una lista de objetos de la clase grupo.

Esta organización descrita permite hacer comprobaciones tales como comprobar si un individuo ya está en un grupo para que no se duplique, cuando se quita un personaje de la partida que también se quite de todos los grupos en los que está y etc.

La parte más importante es la interfaz con la cual se realizan las acciones. Debido a la organización por grupos, un individuo seleccionado sólo podrá realizar acciones contra otros que se hallen en su mismo grupo. Se sabrá pues, qué individuo va a realizar la acción y a quien, ya que se habrá seleccionado en los combo de texto, con lo cual el índice del elemento seleccionado, coincidirá con la posición que el individuo ocupe en el grupo.

Una vez se tengan todos los elementos seleccionados, el programa sabrá que acción realizar de la siguiente forma: Cuando el usuario pulsa la acción a realizar, una variable cambia indicando ese tipo de acción, finalmente cuando se pulsa ejecutar, la aplicación comprueba esa variable y ejecuta la función correspondiente. El siguiente paso sólo consistirá en obtener el valor de esa acción de los 2 individuos, realizar las tiradas de dados, y finalmente comprobar qué tirada ha ganado a cual, mostrando el resultado por el narrador.

Después de la ejecución, simplemente se cambia el índice del combo de individuos y el evento que se dispara con este cambio, actualiza el formulario según el nuevo individuo escogido. En el caso de que estuviese la opción ‘respetar turnos’ seleccionados, la aplicación simplemente no permite cambiar este combo manualmente.

4.3 Base de Datos

La configuración de una base de datos embebida como es SQLite resulta sencilla. Una librería agregada al proyecto actúa como driver de esta y la conexión con la aplización se realiza de manera similar a como se haría con una base de datos como MySQL. Como resultado se obtiene un archivo .sqlite que contiene la base de datos. Es capaz de almacenar hasta 2 Tb de memória, capacidad más que sobrante para esta aplicación.

A continuación se muestran todas las tablas de la base de datos con todos sus campos y relaciones.

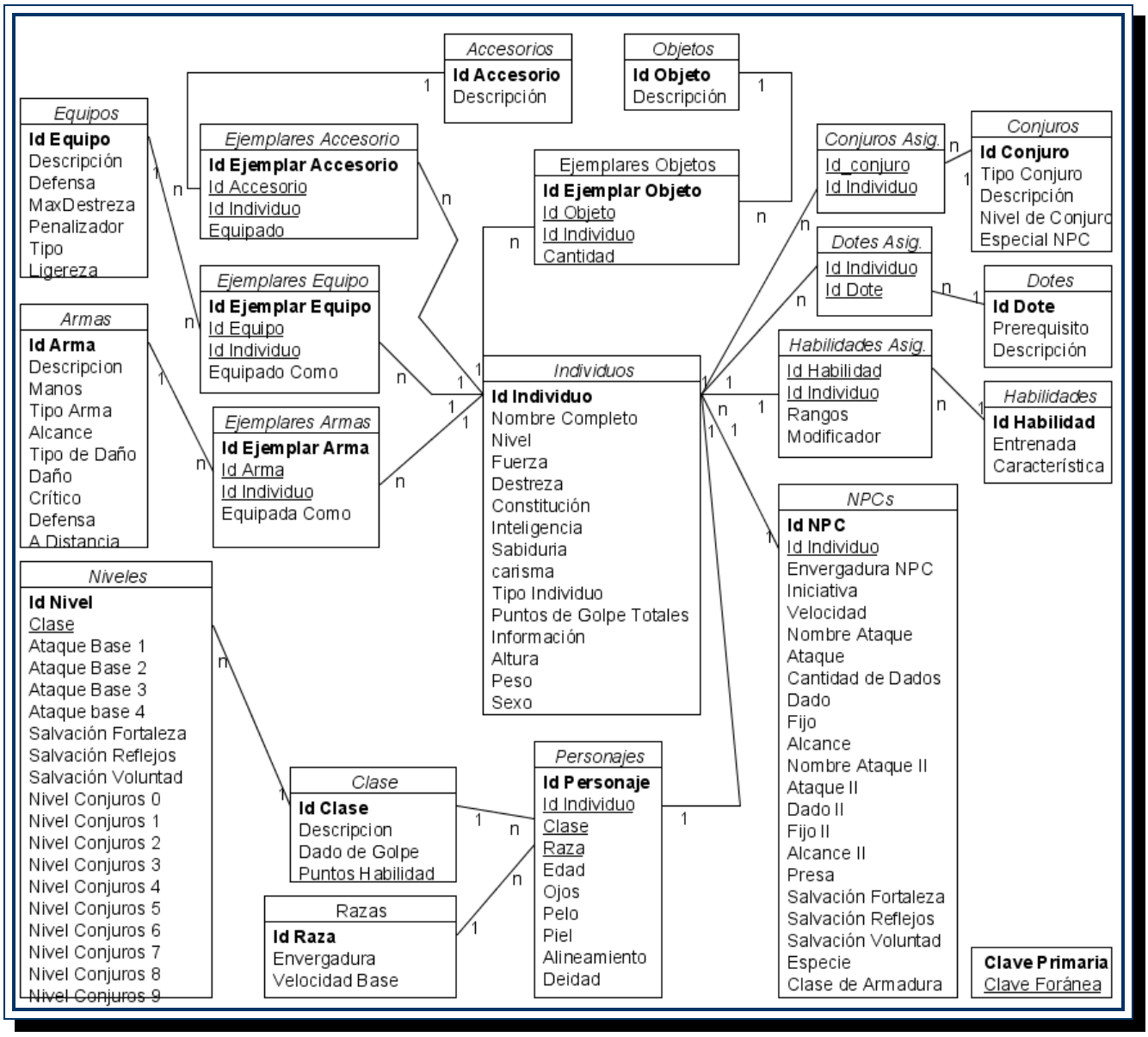


Figura 3.3: Base de Datos

Individuos como Núcleo de la Base de Datos

Al igual que sucedía en la arquitectura del sistema, el núcleo de la base de datos también es la tabla *individuos* y además la mayor complejidad del diseño de la base de datos también se halla en este lugar. La tabla dispone de dos relaciones 1 a 1, una con *NPCs* y otra con *Personajes*.

Este diseño se debe a que Personajes y NPCs comparten gran cantidad de campos por lo tanto, con lo cual los que son comunes se encuentran en la tabla individuos y los particulares en su respectiva tabla. Para que dicha estructura tenga sentido, cada vez que se cree un individuo la aplicación creará su correspondiente subtabla, es decir, si el usuario está construyendo un personaje, se construirá primero la tabla *individuos* y luego se construirá la tabla *personajes* utilizando la id del individuo creado.

También es relevante comentar el campo *Tipo Individuo* de la tabla *Individuo* ¸ la razón de ser de del campo es diferenciar qué tipo de individuo es, en otras palabras, distinguir NPCs de Personajes.

Personajes y sus Relaciones

A parte de los campos particulares de la tabla, lo más relevante de ella radica en las relaciones que establecen las claves foráneas *Raza* y *Clase.* Ambas son relaciones 1 a n, ya que ambas describen unas características que compartirán todos los personajes de esos dos tipos.

Se puede observar, que además, la tabla *Clase* está relacionada de forma 1 a n con la tabla *Niveles.* Este diseño está solucionando la problemática de que cada clase es distinta y tiene 20 niveles diferentes, cada uno de los cuales otorga ciertos atributos a los personajes.

Determinando las Capacidades de cada Individuo:

Conjuros, Habilidades y Dotes son las tres tablas que cumplen el objetivo del título. Se tratan de capacidades generales que pueden disponer los individuos; un individuo puede tener varias y una cualquiera de ellas puede ser dispuesta por varios individuos. La consecuencia es que son tres relaciones n a n.

Pero las Habilidades tienen una peculiaridad, cada un individuo no usa una habilidad de la misma forma ya que puede ser mejor o peor en ella, en otras palabras, cada relación es especial ya que los campos *rango* y *modificador* no tienen que ser iguales para todos.

Las Tablas que Gestionan el Inventario

Dicho oficio lo cumplen las tablas *Equipos*, *Armas*, *Accesorios* y *Objetos.* Como Ejemplo, la tabla Armas describe un tipo de arma en concreto, se podría interpretar como una tabla "abstracta" ya que no existe en el mundo imaginario de la partida, lo que si existe es el ejemplar de arma creado en la relación n a n. Cada ejemplar es diferente, incluso si surgen de relaciones entre un mismo arma y un mismo individuo, por lo que para tener en cuenta este inciso es necesario que la tabla ejemplar disponga de un Id propio.

El campo *equipado como* de las tablas de ejemplares indica en que lugar está equipado el elemento, exceptuando en la tabla *ejemplares objeto* que figura un campo *cantidad* el cual mostrará la cantidad de objetos de ese tipo que tiene ese individuo. A causa de esto, cuando un individuo obtenga dos objetos iguales, en vez de crearse dos ejemplares distintos, aumentará el campo cantidad.

4.4 Interfaz de usuario

La interfaz de usuario se ha desarrollado mediante las librerías gráficas que proporciona Java Swing. La aplicación está constituida por varias ventanas que proporcionan las diferentes funcionalidades de la aplicación.

Al iniciar el programa se muestra un menú principal simple donde se muestran botones con las diferentes opciones. Pulsando los botones surgirán ventanas (JDialogs) con las diferentes interfaces del programa.

‘Crear Personaje’ es un ejemplo de JDialog y se muestra a continuación:

Figura 4.4: Interfaz Crear Personaje

Algunos de los botones que se encuentran en esta ventana abrirán otros JDialog que realicen otras funciones. La estructura jerárquica tendrá que ser respetada pues no se podrá volver a la ventana antigua hasta no haber cerrado la nueva ventana emergida.

Pero lo realmente interesante respecto a las interfaces empieza una vez iniciada la partida. Se mostrará un JFrame (Ventana) dónde se realizarán las funciones principales de una partida.

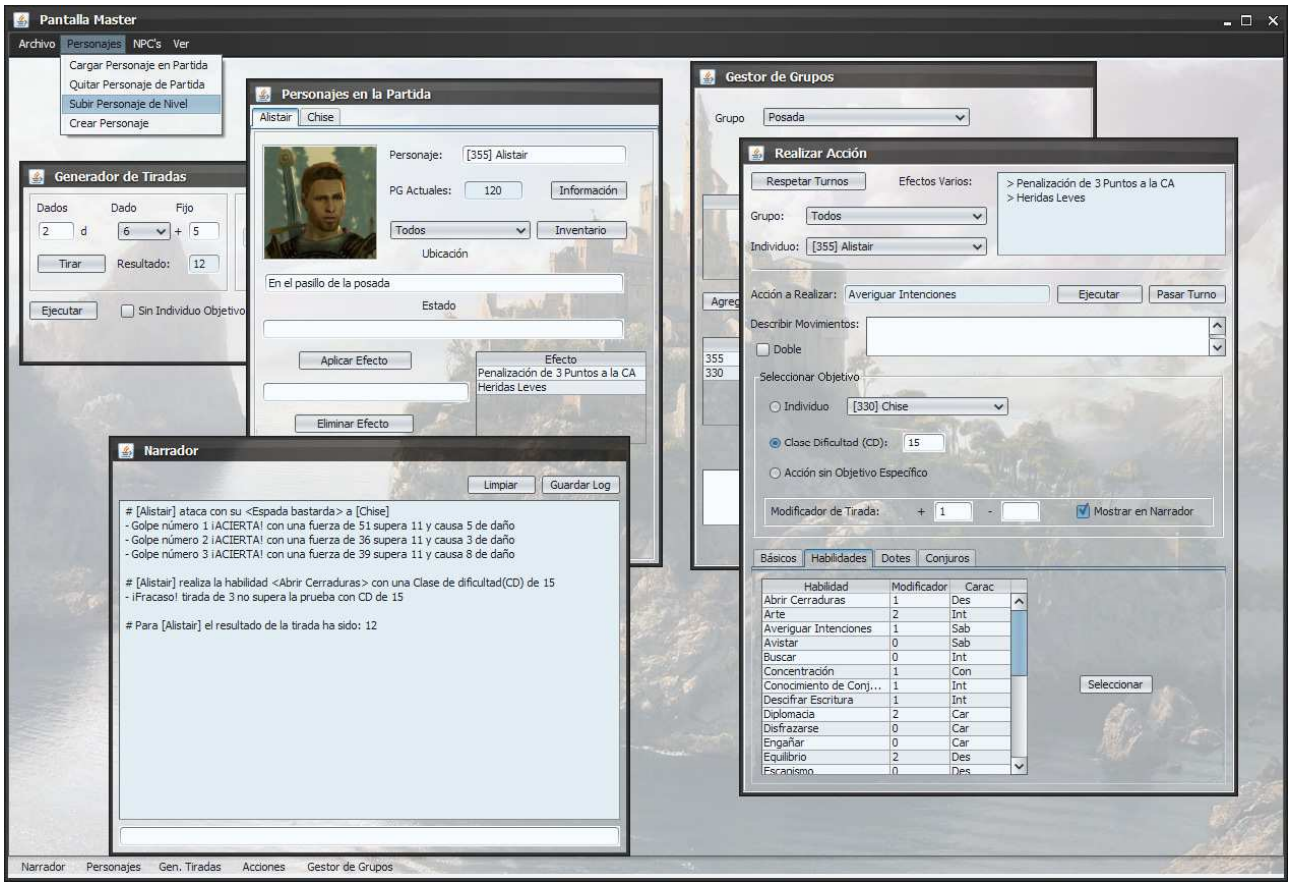


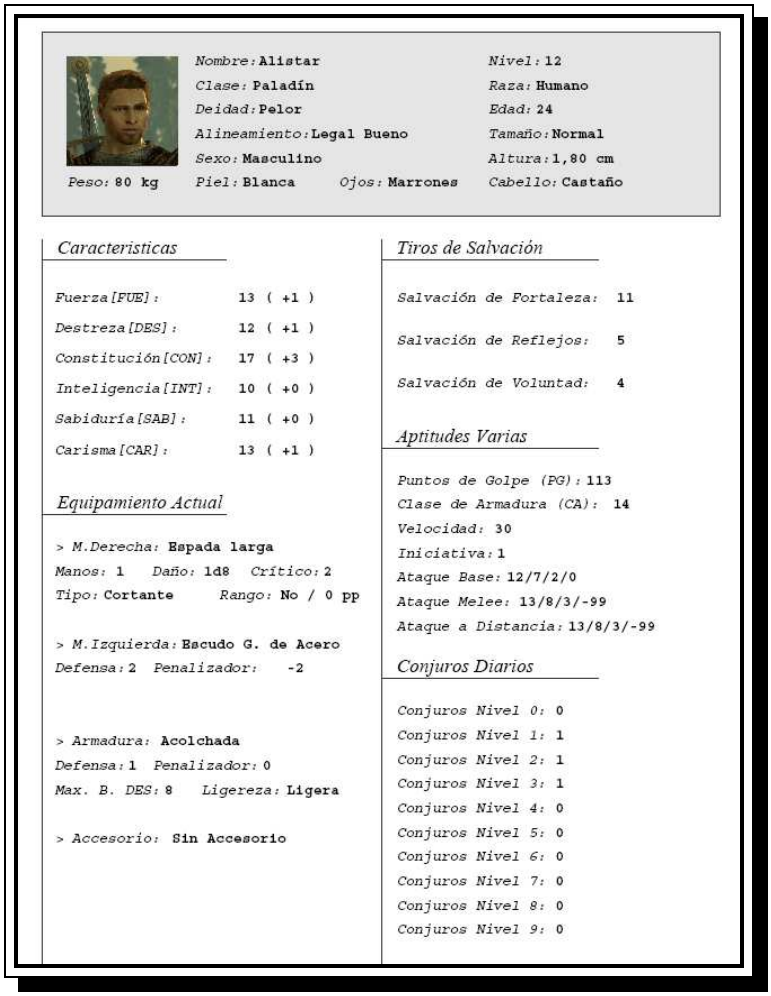
Figura 4.5: Interfaz Partida

Como se puede apreciar en la imagen el JFrame contiene un panel escritorio (JDesktopPane) donde se hallan las diversas ventanas internas (JInternalFrame) de las que dispone el programa. Cada una de ellas realiza una función específica.

Para que resulte más cómodo el uso de la aplicación se ha habilitado una barra de escritorio en la parte baja que muestra botones que representan a las ventanas que se hayan abiertas. Cuando se pulsa uno de estos botones la ventana correspondiente pasará a estar encima de las demás. Dichas ventanas pueden ser gestionadas mediante el menú ya que desde ahí el usuario podrá escoger las ventanas que quiere que se muestren.

Finalmente también en el menú se pueden encontrar accesos a más funcionalidades del programa, entre ellas las que eran accesibles desde el menú principal.

Es importante mostrar la forma en la que se pueden mostrar los datos de los Personajes, pues para acceder a toda la información al detalle es necesario acceder a la Hoja de Personaje que se genera en un Fichero PDF.



Nombre: Alistar | Nivel: 12
Clase: Paladín | Raza: Humano
Deidad: Pelor | Edad: 24
Alineamiento: Legal Bueno | Tamaño: Normal
Sexo: Masculino | Altura: 1,80 cm
Peso: 80 kg | Piel: Blanca | Ojos: Marrones | Cabello: Castaño

Caracteristicas

Fuerza[FUE]:	13	(+1)
Destreza[DES]:	12	(+1)
Constitución[CON]:	17	(+3)
Inteligencia[INT]:	10	(+0)
Sabiduría[SAB]:	11	(+0)
Carisma[CAR]:	13	(+1)

Equipamiento Actual

> M.Derecha: Espada larga
Manos: 1 Daño: 1d8 Crítico: 2
Tipo: Cortante Rango: No / 0 pp

> M.Izquierda: Escudo G. de Acero
Defensa: 2 Penalizador: -2

> Armadura: Acolchada
Defensa: 1 Penalizador: 0
Max. B. DES: 8 Ligereza: Ligera

> Accesorio: Sin Accesorio

Tiros de Salvación

Salvación de Fortaleza:	11
Salvación de Reflejos:	5
Salvación de Voluntad:	4

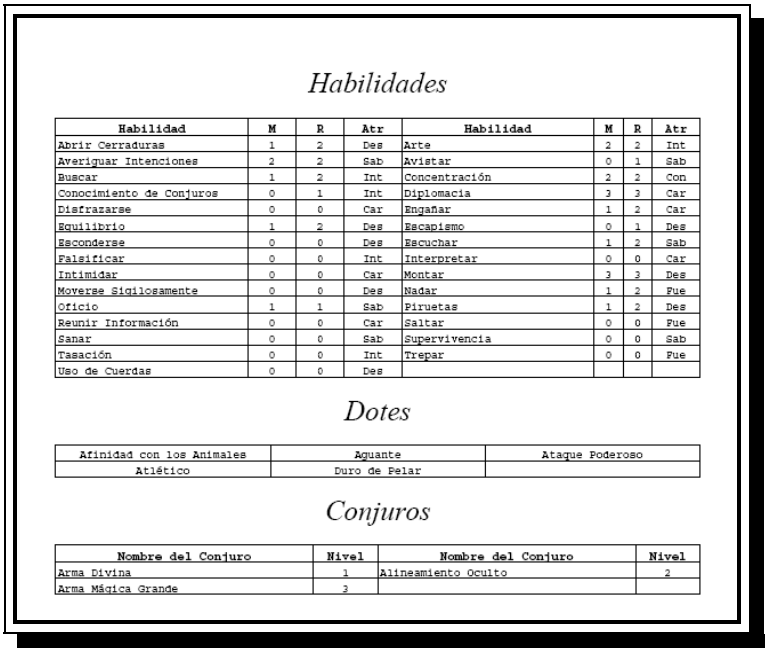
Aptitudes Varias

Puntos de Golpe (PG): 113
Clase de Armadura (CA): 14
Velocidad: 30
Iniciativa: 1
Ataque Base: 12/7/2/0
Ataque Melee: 13/8/3/-99
Ataque a Distancia: 13/8/3/-99

Conjuros Diarios

Conjuros Nivel 0: 0
Conjuros Nivel 1: 1
Conjuros Nivel 2: 1
Conjuros Nivel 3: 1
Conjuros Nivel 4: 0
Conjuros Nivel 5: 0
Conjuros Nivel 6: 0
Conjuros Nivel 7: 0
Conjuros Nivel 8: 0
Conjuros Nivel 9: 0

Figura 4.6: Hoja de Personaje (Página 1)



Habilidades

Habilidad	M	R	Atr	Habilidad	M	R	Atr
Abrir Cerraduras	1	2	Des	Arte	2	2	Int
Averiguar Intenciones	2	2	Sab	Avistar	0	1	Sab
Buscar	1	2	Int	Concentración	2	2	Con
Conocimiento de Conjuros	0	1	Int	Diplomacia	3	3	Car
Disfrazarse	0	0	Car	Engañar	1	2	Car
Equilibrio	1	2	Des	Escapismo	0	1	Des
Esconderse	0	0	Des	Escuchar	1	2	Sab
Falsificar	0	0	Int	Interpretar	0	0	Car
Intimidar	0	0	Car	Montar	3	3	Des
Moverse Sigilosamente	0	0	Des	Nadar	1	2	Fue
Oficio	1	1	Sab	Piruetas	1	2	Des
Reunir Información	0	0	Car	Saltar	0	0	Fue
Sanar	0	0	Sab	Supervivencia	0	0	Sab
Tasación	0	0	Int	Trepar	0	0	Fue
Uso de Cuerdas	0	0	Des				

Dotes

Afinidad con los Animales	Aguante	Ataque Poderoso
Atlético	Duro de Pelar	

Conjuros

Nombre del Conjuro	Nivel	Nombre del Conjuro	Nivel
Arma Divina	1	Alineamiento Oculto	2
Arma Mágica Grande	3		

Figura 4.7: Hoja de Personaje (Página 2)

5 Codificación y Pruebas

Aquí se hablará sobre el estilo de codificación empleado y la forma en la que se ha comentado el código. También incluirá, debido a que es interesante, los muchos problemas que han surgido durante el desarrollo del proyecto y como se han solucionado.

5.1 Estilo de codificación y comentarios

No se ha empleado ningún estilo de codificación estricto, pero se ha intentado llevar a cabo prácticas que faciliten su comprensión.

Debido a que mucha de la programación del programa se halla en las interfaces gran cantidad de código se encuentra distribuido por los diferentes eventos habidos en ellas. De esta forma todo queda con una organización impuesta donde se puede diferenciar qué hace cada evento.

Se ha agrupado en la medida de lo posible las partes de código que se complementen para realizar una misma cosa agrupándolo en métodos. De esta forma también se ha intentado que se repita la mínima cantidad de código posible que haga funciones similares.

Por último se ha contribuido a la comprensión realizando prácticas habituales como nombrar a las variables de forma similar y entendible y no escribir líneas de código demasiado largas. Todos los elementos de interfaz, como por ejemplo los botones o campos de texto se han nombrado especificando el tipo de elemento y su nombre, por ejemplo: BotonEjecutar.

No se ha seguido un patrón específico a la hora de colocar comentarios. Pero sí se ha separado las partes de código con un comentario que explicaba la función de lo siguiente que se iba a realizar. Finalmente se han realizado numerosos comentarios explicativos en líneas de código que puedan tener una comprensión conflictiva.

Gran cantidad de ejemplos de codificación que resuelven diversos problemas pueden ser encontrados en el Anexo I (Pág 64).

5.2 Pruebas

Aquí se va a explicar las diferentes pruebas que se han realizado para asegurar el correcto funcionamiento de la aplicación en todas sus facetas.

A medida que se ha ido desarrollando el proyecto se han realizado dos tipos de prueba. Cada vez que se añadía código se probaba esa pequeña nueva porción funcionase como se esperaba de ella, seguidamente se realizaba una prueba a un nivel un poco mayor que comprobase el correcto funcionamiento de todas aquellas funcionalidades que podrían haber sido afectadas. Este método demostraba una eficacia ejemplar pues permitía avanzar con seguridad y además destapaba pequeños errores que se hubieran vuelto mucho más escurridizos en un análisis a posteriori gracias a tener muy limitada la zona de error.

Las pruebas modulares venían después de la finalización de un modulo, se comprobaba su correcto funcionamiento ejecutando todas las posibilidades que se pudieran dar en el uso del programa para no dejar cabos sueltos.

Con la finalización de cada nuevo módulo se realizaban todas las pruebas de integración pertinentes. Se probaban todas las relaciones posibles del nuevo módulo con los que ya existían, asegurándose de que todas funcionasen correctamente.

Una vez finalizada toda la aplicación se realizaban todas las pruebas de caja negra pertinentes. Se comprobaba funcionamiento de todos los módulos en cooperación asegurándose de que todo funcionase correctamente y que se sucediesen todas las respuestas esperadas por parte de la aplicación.

Se considera que el grueso de las pruebas de caja blanca ya se había realizado durante el desarrollo de la aplicación y con las pruebas modulares. Pero aún así se comprobaron las partes nuevas que pudieran haber derivado de la total integración de los módulos y nuevamente se volvió a probar las partes de código más conflictivas.

Finalmente se realizaron pruebas de aceptación para ver si la aplicación cumplía todos los objetivos establecidos al inicio del proyecto.

Errores Resueltos

Para demostrar la efectividad de las pruebas realizadas se comentarán algunos de los problemas encontrados gracias a ellas.

Mediante las pruebas realizadas durante el desarrollo se encontraron numerosos pequeños errores producidos por campos vacíos, tipos de variable inadecuadas, eventos que no respondían como se esperaba, etc.

Una de las mayores problemáticas que emergieron gracias a las pruebas fue el tratamiento de las consultas. Originalmente la clase Individuo y sus herederas, disponían de una variable ResultSet que se actualizaba con la información de la base de datos cada vez que se llamaba un método. Entonces cada vez que se quería acceder a los datos, la clase extraía la información del ResultSet y la devolvía.

Pero cuando se comenzó a trabajar con los Individuos, surgía un error que decía que el ResultSet estaba cerrado. Y es que jamás se recomienda dejar estas variables abiertas y además también es necesario que para cada ResultSet se cree desde un Statement único. Finalmente se optó por cargar todos los datos en los atributos de la clase de una sola vez y luego trabajar con ellos exclusivamente, de esta forma inmediatamente se podía cerrar el ResultSet y se evitaban más problemas.

Otro error bastante curioso fue a la hora de actualizar las múltiples tablas que hay en el programa. La forma más sencilla de actualizar una tabla es borrándola completamente y volviéndola a rellenar con los datos actualizados. Pues el error consistía en que se repetían campos de las tablas al actualizarse.

Resultó ser porque cuando nunca se llegaban a borrar completamente porque bucle encargado de eso era incorrecto. Dicho Bucle recorría la tabla a través de una variable i que se incrementaba y se iba borrando cada elemento. El error era porque al borrar el índice número 0 todos los índices se actualizan (pasando el elemento del índice 1 a tener índice 0) así que como en el siguiente paso se iría a borrar el elemento del índice 1 el segundo elemento hallado ahora en el índice 0 jamás se borraría.

6 Conclusiones

6.1 Cumplimiento de Objetivos

Varias son las metas que se han alcanzado con el proyecto, pero como se especificó en la introducción de esta memoria sólo había un objetivo que se perfilaba como el punto crítico a cumplir.

Se puede afirmar con seguridad que se ha alcanzado la cima a la que se aspiraba, pero una vez desde ahí se distingue que existen múltiples cimas más a las que escalar. En otras palabras, que a pesar de haber finalizado cumpliendo el objetivo principal, el proyecto no queda cerrado pues existen varias posibles mejoras y complementos en los que seguir trabajando.

Optimizar los procesos era el logro que llevaría al cumplimiento del objetivo. Con dicho fin se han proporcionado herramientas para crear y almacenar los diferentes individuos. Y después de este paso, es cuando se ha podido automatizar la manera de resolver todas las acciones propias de una partida de rol. La optimización es la que ha aligerado la carga de trabajo del Master, la que ha hecho que llevar a cabo sus funciones en la partida es más sencillo y obviamente es la que ha acelerado en gran medida la rapidez con la que se resuelven todas las operaciones.

Al echar un vistazo al resto de las finalidades puede considerarse que el proyecto también ha cumplido con solvencia sus pretensiones.

El programa dispone de herramientas que cierta flexibilidad a la aplicación, modificando resultados o generando tiradas propias por ejemplo. Así se cumple la intención de acercarse en la medida de lo posible al concepto de sistema de juego, no obligando a los jugadores a guiarse solamente por lo que propone el reglamento.

En cuanto a la autonomía que se deseaba que tuviese la aplicación. El haber usado una base de datos embebida en la propia aplicación ha permitido poder almacenar todos los datos necesarios a la vez que se prescinde conexión a Internet. La utilización de Java también ha sido relevante pues sumada al tipo de base de datos empleada ha aportado una aplicación que puede ser ejecutada en cualquier sistema operativo y además resulta ser ligera.

La aplicación dispone de una interfaz intuitiva que realmente guía a los usuarios a la hora de realizar sus diferentes funciones. Se debe a que sólo habilita las funciones válidas en todo momento evitando así incoherencias y haciendo más sencillo su uso. En adicción a esto, como ya se ha mencionado, toda la información necesaria se puede encontrar en el propio programa por lo que no depende del libro con el reglamento.

Finalmente se quería que el software dispusiese de recursos para exhibir la información de la forma más adecuada posible, y en gran medida los objetivos se han cumplido pues existen gran cantidad de formularios en los que se indican todos los datos que se requieran de forma ordenada. Pero a pesar de ello, el software carece de una funcionalidad, que aunque sea secundaria, empaña el resultado general. Esta funcionalidad era la de revelar un mapa donde se señalasen las posiciones de todos los personajes e incluso que se pudiese modificar alguna característica de dicho mapa. Afortunadamente dicha carencia se ha compensado con el gestor de grupos que aporta muchas posibilidades de uso. Y sobretodo con un muy acertado generador de hojas de personaje a través de un fichero PDF cuyas hojas impresas pueden repartirse entre los jugadores para complementar a la aplicación.

En conclusión, queda demostrado que se ha alcanzado con éxito el objetivo principal y también los secundarios, sólo cabe la duda acerca de la falta del mapa pero que personalmente considero que se ha sabido contrarrestar eficazmente.

6.2 Desviaciones sobre la Planificación

Nada transcurre nunca exactamente según lo planeado. Si este hecho es una certeza universal, en los proyectos de desarrollo informático todavía cobra más sentido, y sobretodo si se trata de un proyecto de final de carrera.

Se muestra a continuación el Gantt de seguimiento que representa la consecución real del proyecto, a su vez las barras oscuras independientes indican cual fue la planificación del proyecto a priori (Línea Base). Así es posible mostrar de manera muy visual las desviaciones del proyecto respecto a la planificación,

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Proyecto**	**162,67 días**	**mar 10/11/09**	**jue 24/06/10**
2	Inicio del Proyecto: Asignación y Matriculación	1 hora	mar 10/11/09	mar 10/11/09
3	**Documentación Previa**	**2,67 días**	**mar 10/11/09**	**jue 12/11/09**
4	Investigación sobre los Juegos de Rol	3 horas	mar 10/11/09	mié 11/11/09
5	Planteamiento del Sistema de Juego	2 horas	mié 11/11/09	mié 11/11/09
6	Estudio del Reglamento del Juego	3 horas	jue 12/11/09	jue 12/11/09
7	**Análisis de la Aplicación**	**5 días**	**vie 13/11/09**	**jue 19/11/09**
8	Análisis de Requerimientos	6 horas	vie 13/11/09	lun 16/11/09
9	Estudio de Aplicaciones OpenCode Similares	3 horas	mar 17/11/09	mar 17/11/09
10	Análisis de Tecnologías a Utilizar	6 horas	mié 18/11/09	jue 19/11/09
11	**Formación Previa**	**7,67 días**	**vie 20/11/09**	**mar 01/12/09**
12	Programación Orientadas a Objeto	6 horas	vie 20/11/09	lun 23/11/09
13	Aprendizaje sobre Bases de Datos	5 horas	mar 24/11/09	mié 25/11/09
14	Aprendizaje del Lenguaje de Programación	10 horas	mié 25/11/09	lun 30/11/09
15	Familiarización con el Entorno de Desarrollo	2 horas	mar 01/12/09	mar 01/12/09
16	**Diseño de la Aplicación**	**23 días**	**mar 01/12/09**	**jue 31/12/09**
17	Diseño de la Base de Datos	13 horas	mar 01/12/09	lun 07/12/09
18	Diseño de la Aplicación	30 horas	lun 07/12/09	lun 21/12/09
19	Diseño de la Interfaz	26 horas	lun 21/12/09	jue 31/12/09
20	**Desarrollo de la Aplicación**	**103 días**	**vie 01/01/10**	**mar 25/05/10**
21	Preparación del Entorno de Desarrollo	3 horas	vie 01/01/10	vie 01/01/10
22	Construccion de la Base de Datos	10 horas	lun 04/01/10	jue 07/01/10
23	Desarrollo de Clases Internas	35 horas	jue 07/01/10	vie 22/01/10
24	**Módulo Construcción de Personajes**	**28,33 días**	**lun 25/01/10**	**jue 04/03/10**

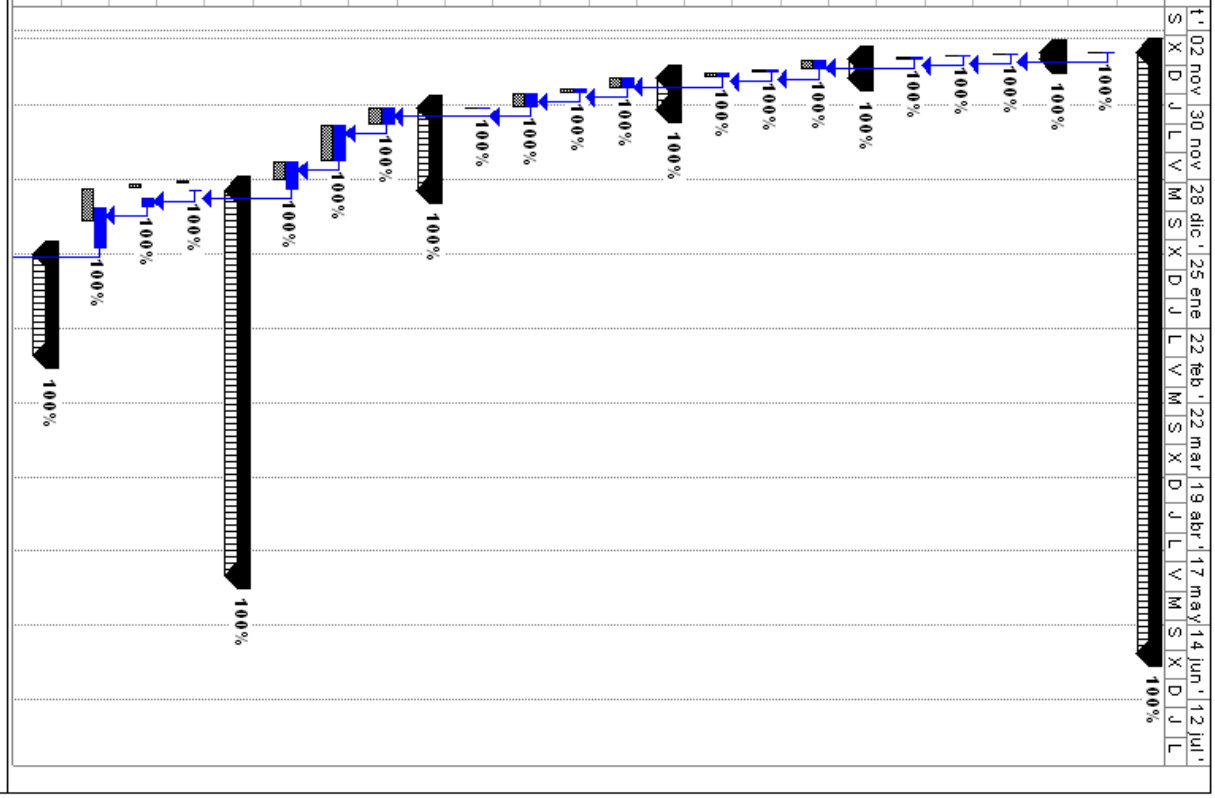


Figura 6.1: Diagrama de Seguimiento (Parte 1)

Id	Nombre de tarea	Duración	Comienzo	Fin
25	Construcción Personaje Base	25 horas	lun 25/01/10	jue 04/02/10
26	Módulo Seleccionar Habilidades	20 horas	jue 04/02/10	vie 12/02/10
27	Módulo Seleccionar Dotes	20 horas	lun 15/02/10	mar 23/02/10
28	Módulo Asignar Conjuros	20 horas	mar 23/02/10	jue 04/03/10
29	Módulo Gestor de Individuos	10 horas	jue 04/03/10	mar 09/03/10
30	Generación de PDF de Hoja de Personaje	20 horas	mar 09/03/10	jue 18/03/10
31	Pantalla Información de NPC	5 horas	jue 18/03/10	vie 19/03/10
32	Funcionalidad Subir Nivel	8 horas	lun 22/03/10	mié 24/03/10
33	Módulo Gestionar Inventario	25 horas	mié 24/03/10	lun 05/04/10
34	Módulo Crear NPCs	10 horas	mar 06/04/10	vie 09/04/10
35	**Módulo Partida**	**32,67 días**	**vie 09/04/10**	**mar 25/05/10**
36	Construcción Interfaz Partida	15 horas	vie 09/04/10	vie 16/04/10
37	Narrador	5 horas	vie 16/04/10	lun 19/04/10
38	Generador de Tiradas	13 horas	mar 20/04/10	lun 26/04/10
39	Pantalla de Estado Personajes y NPC's	10 horas	lun 26/04/10	jue 29/04/10
40	Gestor de Grupos	20 horas	jue 29/04/10	lun 10/05/10
41	Realizador de Acciones	35 horas	lun 10/05/10	mar 25/05/10
42	**Pruebas**	**11,33 días**	**mié 26/05/10**	**jue 10/06/10**
43	Pruebas Modulares	7 horas	mié 26/05/10	vie 28/05/10
44	Pruebas Generales	7 horas	vie 28/05/10	mar 01/06/10
45	Corrección de Errores	20 horas	mar 01/06/10	jue 10/06/10
46	Generación de Documentacion (Memória)	30 horas	jue 10/06/10	jue 24/06/10
47	Cierre del Proyecto	1 hora	jue 24/06/10	jue 24/06/10

Página 2

Figura 6.2: Diagrama de Seguimiento (Parte 2)

El cómputo final indica que el tiempo total son 488 horas finalizando el día 24 de Junio, 5 días antes de la fecha límite. El incremento de la duración se ha sucedido a pesar de que la planificación temporal fue muy generosa en cuanto al tiempo ya que se tuvo en cuenta que el riesgo de no cumplir con el plazo. La lectura positiva es que se ha cumplido el plazo gracias a haber podido soportar la presión generada por los imprevistos y la presteza que se requería.

Se tenia constancia de que el riesgo de haber realizado una planificación optimista podía darse con facilidad, incluso casi se tenía la certeza de ello

El diagrama dice muchas cosas acerca de cómo se han sucedido los acontecimientos. Se puede apreciar que las fases previas al comienzo del desarrollo han transcurrido según lo previsto, exceptuando el diseño de la interfaz, que es donde se produce el primer desajuste. Entonces a partir de este punto, cuando se entra en la fase de programación de la aplicación, todas las tareas adoptan una tendencia generalizada a prolongarse.

La explicación al retraso del diseño de la interfaz es que fue difícil encontrar un diseño a la altura de lo que se deseaba. Es sencillo crear una interfaz incómoda y enrevesada pero cuando se trata de que sea intuitiva y práctica todo resulta mucho más complejo.

Pero la causa del importante retraso es, como ya queda evidenciada en los diagramas, la etapa de desarrollo de la aplicación.

Ya se mencionó anteriormente en esta memoria, que el programador iba a necesitar constantemente buscar conocimientos nuevos para avanzar y que eso iba a tener una repercusión. Pero no se había tenido lo suficientemente en cuenta que el riesgo de una gran complejidad emergiese, provocando consecuencias todavía más relevantes.

La causa de dicha complejidad ha sido que el juego resultó más difícil programar de lo que al inicio se había anticipado. Esto se ve reflejado en el aumento de tiempo de de desarrollo de las diferentes interfaces de creación respecto a lo que se planificó. Lo sucedido demuestra que la base del juego radica en los personajes como se explicó durante el Marco Teórico.

Además de lo dicho en el párrafo anterior hay que sumar que durante esos primeros compases del desarrollo también se evidenciaron algunos diseños fallidos realizados que obligaron a volver hacia

atrás para rehacer lo desarrollado. Se puede decir que esta fue la etapa más problemática del proyecto.

Aunque se sufrieron todos estos inconvenientes, se logró encauzar el desarrollo del proyecto por el buen camino, pero no sin haber sufrido grandes retrasos. A causa de esto se creyó conveniente omitir la funcionalidad del mapa de partida porque se dedujo que su realización supondría superar la fecha límite de finalización de proyecto. La omisión fue viable porque se trata de una funcionalidad secundaria.

6.3 Líneas Futuras

Llegados a este punto es de suponer que nos hallamos ante un proyecto finalizado, pero personalmente considero que, son tales las líneas abiertas que se podría hablar de sólo el comienzo de un proyecto mucho mayor.

Dentro de la cantidad de líneas abiertas algunas se perfilan como mejoras o adiciones de funcionalidades de poco calibre, mientras que otras pueden significar el ascenso a un nuevo escalón en lo que a prestaciones se refiere pero que requieren una gran dedicación.

El frente abierto más inminente es el de concluir la funcionalidad del mapa de personajes, ya que ha quedado pendiente de realizar debido a circunstancias anteriormente descritas.

Con el fin de mejorar la calidad del proyecto sin revolucionarlo existen varias posibilidades como representar más elementos del reglamento en cuanto a métodos y posibilidades en el programa que no estén ya realizados, intentar realmente programar el efecto de cada conjuro o también automatizar en la medida de lo posible y de lo conveniente varias funcionalidades. Todas ellas constituyen acciones que se agradecerían enormemente y que probablemente habría que llevar a cabo si se quisiese llegar más lejos con este proyecto.

Posiblemente la ampliación más rentable en lo que a la relación trabajo-repercusión se requiere sería la de programar una partida que pudiese ser llevada a cabo a través de Internet. La forma en la que está diseñada la aplicación siempre pretendió dejar abierta la puerta del on-line por lo que su arquitectura da bastantes facilidades para ello.

Este propósito se llevaría a cabo realizando una conexión a través de Internet siguiendo el modelo que ha usado años atrás gran cantidad de videojuegos. La forma entonces de llevarlo a cabo, sería que el Master crease la partida y con ella hiciese de servidor improvisado, los demás jugadores se conectarían a esta partida. El Master dispondría de la misma interfaz de la que ahora dispone, la novedad es que se habría creado una nueva interfaz orientada desde el punto de vista de los jugadores, donde ellos sólo tomarían el control de su personaje y realizarían acciones previo permiso del Master. Estos jugadores conectados verían lo que sucede en la partida a través de lo publicado por el Narrador y se comunicarían utilizando un Chat proporcionado.

Es muy previsible la cantidad de ventajas que proporcionaría dicha funcionalidad al uso del programa, por lo que este punto sería una de las prioridades a la hora de tener en cuenta las líneas futuras.

Hoy en día el concepto de “hagámoslo entre todos” está extendiéndose por todas partes irrefrenablemente, y esto se debe a que realmente siempre será más poderosa la mente colectiva que la individual. La Web 2.0 con ejemplos como la Wikipedia son prueba de ello. Así que la última y más ambiciosa línea de ampliación del proyecto camina por esta linde.

Dicho itinerario de ampliación consiste en que los usuarios sean capaces de crear y modificar reglamentos a su antojo y que sean compartirlos con el resto de la comunidad. Para ello habría que proporcionar complejas herramientas de creación y que el programa sea capaz de seguirlas. Empresas dedicadas a los juegos de rol también dispondrían de su tajada del pastel, ya que podrían utilizar la plataforma para comercializar sus juegos en el formato electrónico del programa. Para que esta línea se haga realidad se podría plantear un cambio a una plataforma web debido a la accesibilidad que esta proporciona. La idea realmente entrañaría una revolución para este proyecto, puesto que alcanza a modificar hasta sus propias bases.

En conclusión, indagando más decididamente todavía se podrían extraer más posibilidades con las que seguir evolucionando.

6.4 Consideraciones Personales

Personalmente creo que este proyecto ha estado a la altura como culminación a una carrera que es. Son muchas cosas las que se han aprendido a todos los niveles durante los estudios, y yo he podido encontrar pequeñas representaciones de todas ellas durante la realización de este trabajo; empezando por adquirir conocimientos hasta crecer personalmente.

Como no podía ser de otra manera al girar la vista atrás son muchas cosas las que se cambiarían, cosas que se hubieran afrontado forma. Pero que suceda es una prueba de todo lo que se ha avanzado ya que desde esta perspectiva se ve todo aquello que antes no se veía.

Voy a comenzar por lo que no ha ido precisamente sobre ruedas. Considero que la elección de proyecto ha sido realmente un acierto, pero el problema radica en el juego de rol escogido para él. D&G es un juego con un reglamento realmente complejo que dificulta y vuelve muy tediosa su programación por sus muchas vertientes para resolver los problemas y porque dispone de toneladas de información en tablas. La elección de un reglamento más sencillo hubiera facilitado su desarrollo y se podría haber invertido más tiempo en añadir nuevas funcionalidades.

Finalmente no todos los entresijos del juego se han aplicado, porque cualquier que lo conozca sabrá que eso es un trabajo salomónico, pero se ha intentado que los funcionamientos básicos queden fielmente reflejados de manera coherente en conjunto.

Quería decir que previamente, yo tenía conocimiento de que no era un juego simple, pero al haberlo usado hace ya bastante tiempo lo recordaba de otra manera. Hubo un momento en el que me planteé dar un paso atrás y cambiar de reglamento, pero eso supondría perder bastante trabajo así que al final se prosiguió. Quizá se puede considerar un error el no haber dedicado más esfuerzo en las tareas de planteamiento y estudio del juego que se planificaron.

Llegados a este punto y cambiando de tema me reitero en decir que con más conocimiento en mi haber habría afrontaría el proyecto de otra forma. Pero no puedo culparme por este hecho, porque esto ha sucedido debido a mi desconocimiento inicial de las posibilidades que me ofrecía Java.

Las bondades del uso de un framework como Struts para aplicar un Modelo Vista Controlador, la aplicación de una herramienta como es Hibernate para la relación con la base de datos o un mejor diseño del programa mediante el uso de algún patrón de diseño adecuado. Si yo ahora soy capaz de descubrir mejores alternativas para el desarrollo de este proyecto que antes desconocía, muchas soluciones mejores más sería capaz de aportar cualquier persona con mayores conocimientos que los míos.

En definitiva, hubo puntos negativos, pero yo considero que realmente no fueron tan negativos y concretamente este último se puede decir que prácticamente inevitable.

Se puede deducir que ha sido un proyecto que me ha resultado largo y difícil, y la verdad es que así lo ha sido, pero no puedo estar más satisfecho de cómo ha concluido todo. Como he demostrado he cumplido mis objetivos mientras aplicaba muchos de los conocimientos que he aprendido durante la carrera, además de los nuevos que he adquirido por la necesidad de superar los pequeños retos que se me presentaban.

Todo aquello que he aprendido y he puesto en práctica, me ha ayudado a poner en orden mis ideas. Ahora puedo decir que tengo claro qué es lo que quiero de mi futuro profesional y además ya tengo los conocimientos para saber por donde empezar.

Bibliografía

§ **Página Web Oficial de SQLite**,
Disponible en http://www.sqlite.org/

§ **Programas, manuales, código y más** (2007),
Disponible en http://todojava.awardspace.com/

§ **Foro de Java,**
Disponible en http://www.forodejava.com/

§ **Xerial SQLite JDBC,** Taro L. Saito (2009, Diciembre)
Disponible en http://www.xerial.org/trac/Xerial/wiki/SQLiteJDBC/

§ **Página Web de NetBeans,** (2010)
Disponible en http://www.netbeans.org/

§ **Wiki de programación ChuWiki,** (2010)
Disponible en http://chuwiki.chuidiang.org/

§ **Página Web de I-Text,** (2010)
Disponible en http://itextpdf.com/

§ **Java™ 2 Platform Standard Edition 5.0 API,** Sun MicroSystems (2004)
Disponible en http://java.sun.com/j2se/1.5.0/docs/api/

Anexo I: Problemas y Ejemplos de Codificación

Se comentarán los problemas con su solución y su correspondiente ejemplo de código si es necesario.

Conexión a la Base de Datos

Para poder conectar a la base de datos se ha creado una clase *ConexionLiteDB* la cual contiene un método que recibe como parámetro un boleano. En caso de ser *true* realiza la conexión sino la cierra.

```
public void conectado(boolean Conectar) throws Exception{
    if(Conectar==true){
        config = new SQLiteConfig();
        config.enforceForeignKeys(true);
        Class.forName("org.sqlite.JDBC");
        Conexion = DriverManager.getConnection(
                "jdbc:sqlite:liteDB.sqlite", config.toProperties());
        if(Conexion!=null){
            System.out.println("Conexion establecida");
        }//fin if conexion
    }else{
        Conexion.close();
    }//Fin if-else Conectar
}
```

Figura 7.1: Función Conectado (ConexionLiteDB)

La utilidad de la línea *config.enforceForeignKeys(true)* garantiza el control de las claves foráneas.

Ejecutar Sentencias

La clase *ConexionLiteDB* también dispone de funciones para ejecutar las diferentes sentencias.

```
public int ejecutarInsert(String SentenciaSQL) throws Exception{
    Statement Sentencia = Conexion.createStatement();
    Sentencia.executeUpdate(SentenciaSQL);
    ResultSet Resultado = Sentencia.executeQuery("" +
                          "SELECT last_insert_rowid();");
    int Rowid=Resultado.getInt(1);
    Resultado.close();
    Resultado.getStatement().close();
    return Rowid;
}
```

Figura 7.2: Función ejecutarInsert (ConexiónLiteDB)

Este caso especifico esta pensado para ejecutar Inserts. Para pode realizar una sentencia primero hay que crear un Statement mediante la variable *Conexión* y después utilizar dicho Statement para realizar la consulta, el resultado se guardará en una variable tipo ResultSet.

Aquí surge un problema pues no pueden haber dos ResultSets abiertos al mismo tiempo. De lo contrario al realizar la consulta nos encontrariamos la variable ResultSet cerrada y daría error. Lo aconsejable es crear un Statement nuevo cada vez que se quiera realizar una consulta, y cerrar el ResultSet y el Statement al finalizar como se puede apreciar en el código.

Dicha solución es necesaria para impedir dos conexiones al mismo tiempo a la misma base de datos.

Se puede apreciar que después de ejecutar la sentencia entrada por parámetro también se realiza un Quero. Esto facilita la vida en gran medida, pues el query es de un comando que devuelve el identificador de la última fila insertada en la base de datos, el *Rowid.* Irremediablemente el Rowid va a coincidir con el identificador asignado (si es autoincremental) así que la función lo devuelve para poder trabajar con él.

Generador de Tiradas de Característica

Durante el método de generación legal de características (Fuerza, Destreza, etc.) se tiran cuatro dados de seis caras, se escogen las 3 tiradas mayores y se suman, obteniendo así el resultado final.

```
private int GenTiradaCar()
{
      int minimo;
      minimo= (int) Math.floor(Math.random()*(6-1+1)+1);
      int resultado=minimo;
      for(int i=0; i<3; i++)
      {
           int nuevaTirada= (int) Math.floor(Math.random()*(6-1+1)+1);
           resultado= resultado+nuevaTirada;
           if(minimo>nuevaTirada)
           {
                minimo=nuevaTirada;
           }
      }
      resultado= resultado - minimo;
      return resultado;
}
```

Figura 7.3: Función GenTiradaCar (DialogCrearPersonaje)

El algoritmo es sencillo. Se realizan los cuatro cálculos aleatorios y se van sumando, manteniendo una variable con el más pequeño, finalmente el resultado menor se le resta a la suma total.

Almacenamiento de Avatar

Para almacenar la imagen perteneciente al avatar de los individuos no se ha optado por guardarla en la base de datos en un campo BLOB. La solución ha sido que después de haber seleccionado la ruta de la imagen, la aplicación hacia una copia en una carpeta del programa llamada Avatares, renombrando la imagen con el nombre completo del individuo más su identificador. De esta forma no ha resultado necesario ni siquiera almacenar la ruta en la base de datos. Cuando se requiera obtener a la imagen para mostrarla en el cuadrado donde debería aparecer simplemente se accede mediante la ruta relativa, el id y el nombre. La imagen se guarda en su resolución original.

Selección de Dotes

Cada uno de los dotes tiene una serie de prerrequisitos que el personaje ha de cumplir para poder seleccionarlos, además de que también se ha de tener en cuenta que existen ciertos dotes que pueden ser aprendidos por la clase *Guerrero*.

```
    private void CheckHendeduraItemStateChanged(java.awt.event.ItemEvent evt) {
        if(CheckHendedura.isSelected()) {
            if((DotesGuerrero>0)&&
                    (objetoPersonaje.getClase().equals("Guerrero"))){
                DotesGuerrero--;
                EspecialHendedura=true;
            }else{
                DotesDisponibles--;
            }
            if(objetoPersonaje.getAtaqueBaseUno() >= 4){
                CheckGranHendedura.setEnabled(true);
            }
        }else{
            if(EspecialHendedura==true){
                DotesGuerrero++;
                EspecialHendedura=false;
            }else{
                DotesDisponibles++;
            }
            CheckGranHendedura.setSelected(false);
            CheckGranHendedura.setEnabled(false);
        }
        CampoDotesDisp.setText(DotesDisponibles+"");
        CampoDotesGuerrero.setText(DotesGuerrero+"");
}
```

Figura 7.4: Evento de Dote (DialogDotes)

Se trata de un evento ItemStateChanged lo cual significa que cuando un CheckBox (casilla de selección) cambia se ejecuta el código. A grandes rasgos lo que hace es comprobar si el elemento queda seleccionado o no. Si el personaje es un guerrero restará o aumentará un punto a las dotes disponibles especiales de guerrero y no los genéricos. En caso contrario, o que no queden dotes disponibles de guerrero modificara los puntos genéricos. También tanto si es una selección como una deselección, habilita o deshabilita los CheckBox que tenían como prerrequisito a la dote escogida.

Debido a que todas las dotes tienen el mismo tipo de evento, cuando se modifica el estado de un checkbox se produce un efecto en cadena sobre todos los checkbox que también cambiarían.

Selección de Habilidades

Uno de los problemas que surgió en este formulario es que según qué clase sea el personaje tendrá más facilidad para aprender unas habilidades u otras. El reglamento refleja esto provocando que cueste el doble de puntos de rango aumentar las habilidades en las que no es bueno el personaje.

Dicha problemática se ha solucionado cambiando la propiedad *selected* (que no tiene ningún tipo de repercusión en nada que nos concierna) según que clase sea el personaje. Gracias a esto cuando se sube un punto a una habilidad se puede saber si el personaje es bueno o no con esa habilidad, aplicando la restricción si es necesario.

Pero el gran problema surgido en este formulario ha sido al usar la funcionalidad e 'Subir Nivel' ya que esto requería cargar todos los rangos que tiene el personaje en el formulario y a controlar los rangos disponibles.

El problema ha quedado solucionado recorriendo una consulta de todas las habilidades del Personaje, por cada una de ellas se colocaban los respectivos puntos de rango en su habilidad correspondiente en el formulario y finalmente se restaban del total de puntos disponibles del personaje incluyendo el nuevo nivel obtenido.

Gestor de Grupos

Se trata de una de las partes en las que se ha encontrado una complejidad mayor de la esperada. El concepto de grupos de individuos era muy sencillo a primera vista, pero los problemas empezaron a surgir a la hora de añadir individuos a los grupos y a quitar individuos a grupos.

El problema de agregar individuos fue más sencillo de solucionar, porque básicamente lo único que se hubo de hacer es buscar si ese individuo ya existía en dicho grupo. Pero esto no era más de un anticipo. El siguiente fue el verdadero problema más problemático, porque cada vez que quitábamos un individuo de la partida suponía tener que recorrer todos los grupos, quitando todas las instancias de ese individuo que se encontrasen.

Construyendo el Fichero PDF

La construcción del Fichero PDF con la información del Personaje se podría haber realizado utilizando algún programa con el que construir la plantilla pero finalmente se generó de la forma más "manual", utilizando la librería I-Text directamente. El siguiente es el responsable de crear la cabecera.

```
//CONSTRUCCION CUADRADO
cb.setColorStroke(new GrayColor(0.2f));
cb.setColorFill(new GrayColor(0.9f));
cb.moveTo(20, 822);
cb.lineTo(575,822);
cb.lineTo(575,671);
cb.lineTo(20, 671);
cb.lineTo(20, 822);
cb.closePathFillStroke();
//CuadradoInterior
cb.newPath();
cb.setColorFill(BaseColor.WHITE);
cb.moveTo(40, 802);
cb.lineTo(40,712);
cb.lineTo(130,712);
cb.lineTo(130,802);
cb.lineTo(40,802);
cb.closePathFillStroke();
```

Figura 7.5: Dibujando el fichero PDF

Como se puede apreciar en el código no resulta especialmente complicado el dibujar un PDF mediante I-Text, lo cual supone uno de los motivos por los cuales se escogió este método. Principalmente en las dos primeras líneas se configura el color de la línea y el relleno. Después simplemente se indica la posición del cursor imaginario y a partir de ahí se trazará la línea a los siguientes puntos que se indiquen. Cuando se haya acabado con esa línea en concreto, sólo restará llamar a la función closePathFillStroke() que hará varias cosas: Ordenar el fin de la línea, pintar la línea y finalmente pintar el relleno.

Devolver Modificador

La forma de actuar de las características sobre las acciones es a través de sus modificadores. Los modificadores se calculan desde las características utilizando la siguiente tabla:

Característica	Modificador
1	-5
2-3	-4
4-5	-3
6-7	-2
8-9	-1
10-11	0
12-13	+1
14-15	+2
16-17	+3
...	...

Figura 7.6: Tabla Modificadores

Para resolver la problemática, se creó una función que realizaba el algoritmo deducido de la tabla para extraer el modificador de la característica que se quisiese.

```
int getModificador(int Carac){
    int Modif=-5;
    int i=1;

    while(i<=Carac){
        if(i%2==0){
            Modif++;
        }
        i++;
    }
    return Modif;
}
```

Figura 7.7: Función getModificador

La función realmente ejecuta un algoritmo muy simple. Como se observa en la tabla sólo hay aumento de modificador en el paso de número impar a par, así que la función empezando por -5 cada vez que se encuentra con un número par aumenta el modificador.

Gestionando el Inventario

En el gestor de inventario, se añade objetos a partir de un botón que abre una ventana con los objetos que añadir. Fue difícil decidir en qué momento almacenar los objetos en la base de datos. Primeramente cada vez que se pulsaba el botón agregar se escribía en la base de datos y ya se le otorgaba su ID, pero para minimizar el número de accesos se optó por cambiar esto y pasar almacenarlos todos juntos de una vez después de cerrar la ventana. Pero el problema es en el momento de guardar se sucedía esta situación:

Inventario
Objeto A (Ya guardado)
Objeto B (Ya guardado)
Objeto C (Ya guardado)
Objeto D (Nuevo objeto sin guardar)
Objeto E (Nuevo objeto sin guardar)

Figura 7.8: Tabla Inventario

Se habían pasado los objetos a la tabla del inventario, ahora sólo restaba guardar los dos nuevos objetos. El inconveniente es que no se había guardado ningún tipo de referencia sobre los objetos a guardar, no se sabía distinguir qué objetos había que coger de la tabla para guardar. Se pensaba que se podría solucionar simplemente borrando todo el inventario y después escribiendo la versión actualizada del mismo que es la que se tenía en la tabla. No fue tan sencillo pues de hacerlo así, se corría el riesgo de eliminar un objeto que estuviese equipado en el personaje, provocando un error bastante grande ya que habríamos reasignado todas las IDs y la relación que se establece al equipar utilizaría una ID errónea que ya no existiría.

Finalmente se resolvió el asunto guardando el índice del último objeto del inventario al entrar en el menú agregar, así que a la hora de guardar, solamente habría que recorrer la tabla a partir del índice que se había tomado como referencia.

Mencionar que como se estaban controlando también los objetos que eran tipo *objeto* según la cantidad, valga la redundancia, era necesario recorrer la tabla antes de guardar cada elemento para ver si ya había uno igual y aumentar su cantidad en 1 si ya existía en vez de crear una nueva instancia.

Ordenando por Iniciativa

Cada uno de los grupos que se crean durante la partida son ordenados según la iniciativa de los individuos que lo conforman. Para realizar esto se ha usado el algoritmo de ordenación Selection Sort, debido a que responde muy rápidamente en listas de pocos elementos.

```
public void OrdenarGrupo(){
    int[] vector= new int[ListaIndividuos.size()];
    for(int i=0; i<ListaIndividuos.size(); i++){
        vector[i]=DevIni((Individuo)ListaIndividuos.get(i));
    }
            for (int i = 0; i < vector.length - 1; i++)//SELECTION SORT
            {
            int min = i;
            for (int j = i + 1; j < vector.length; j++)
            {
                    if (vector[j] < vector[min])
                    {
                            min = j;
                    }
            }
            if (i != min)
            {
                    int aux= vector[i];
                    vector[i] = vector[min];
                    vector[min] = aux;
                    Object auxiliar=ListaIndividuos.get(i);
                    ListaIndividuos.set(i, ListaIndividuos.get(min));
                    ListaIndividuos.set(min, auxiliar);
            }
    }
}
```

Figura 7.9: Selection Sort con Iniciativas

Primeramente se han extraído todas las iniciativas a un array. Se ha aplicado el algoritmo de ordenación sobre el array, como los índices del array coinciden con los índices de la lista del grupo, cada vez que se realiza un cambio en el array se realiza también en la lista del grupo con los mismos índices. Como resultado se obtiene el grupo ordenado.

Ejecutando Acciones

Se le puede considerar el núcleo de la aplicación, se trata de cómo va a interactuar todo lo que se ha creado previamente. Hay varios tipos de acciones, así que cuando se seleccione la acción la aplicación lo que hará será obtener la referencia de la acción que se quiere realizar. Para después cuando se ejecute llamar a la función adecuada para resolver la acción.

El ejemplo siguiente es el caso de la realización de un ataque normal.

```
if(RadioIndividuo.isSelected()){
    if(ComboIndividuos.getSelectedIndex()<=ComboIndividuoContra.getSelectedIndex()){
       IndObjetivo=(Individuo)GrupoActual.getListaIndividuos()
               .get(ComboIndividuoContra.getSelectedIndex()+1);
    }else{
       IndObjetivo=(Individuo)GrupoActual.getListaIndividuos()
               .get(ComboIndividuoContra.getSelectedIndex());
    }
    ASuperar=DevuelveCA(IndObjetivo);
    TextObjetivo=" a ["+IndObjetivo.getNombreCompleto()+"]";
}else if(RadioCD.isSelected()){
    try{
    ASuperar=Integer.parseInt(CampoCD.getText());
    TextObjetivo="";
    }catch(Exception ex){
        ASuperar=0;
    }
}
```

Figura 7.10: Seleccionando un objetivo

Primeramente si está seleccionada la opción *RadioCD* sólo se ha de recoger el número de dificultad, pero en caso contrario ha de asignarse un individuo objetivo. El individuo objetivo estará seleccionado en un *ComboBox* y el que realiza la acción en otro, y por fuerza ambos estarán en el mismo grupo. Pero el Individuo que va a realizar la acción no se encuentra en los individuos objetivo posibles, ya que no se puede actuar contra sí mismo. La comparación lo que hace es ver si el índice del objetivo seleccionado es mayor al índice del objetivo que está realizando la acción.

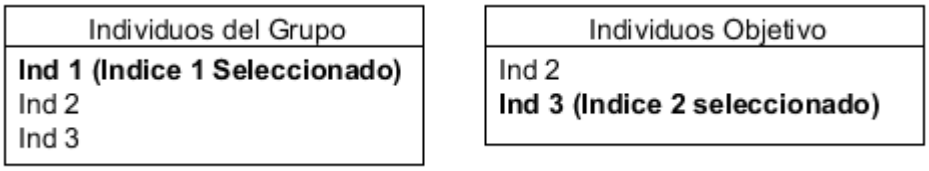


Figura 7.11: Individuos Seleccionados

Como se observa en la figura 7.11 de no ser por el +1 al índice de los objetivos estaríamos realizando la acción contra el segundo individuo del grupo actual y no contra el tercero que es el que queríamos Una vez ya con el objetivo seleccionado es cuando se realiza el ataque.

```
String TextEjecutor="# ["+pj.getNombreCompleto()+"] ataca con su <"+pj.getIdArmaDerecha()+">"+TextObjetivo+"\n";
AreaNarrador.setText(AreaNarrador.getText()+TextEjecutor);
for(int i=0; i<4; i++){//es el ataque de la mano derecha
    int MultiplicadorDano=1;
        if(pj.getAtaqueEfectivo(i)!=-99){//esto quiere decir que si puede realizar ese ataque en la consecución
           int ResultDado=Dado(20);
           int FuerzaGolpe=0;
          if(IndividuoActual.getDeRangoDerecha().equals("No")){//Si fuese de rango usaria otra funcion
               FuerzaGolpe=pj.getAtaqueEfectivo(i)+ResultDado+ResultDado
                       +DevuelveModificadorCampo(CampoBonificador)-DevuelveModificadorCampo(CampoPenalizador);
          }else{
               FuerzaGolpe=pj.getAtaqueRangoEfectivo(i)+ResultDado+ResultDado
                       +DevuelveModificadorCampo(CampoBonificador)-DevuelveModificadorCampo(CampoPenalizador);
          }
           if(ResultDado!=1){
               if(ResultDado==20){//CRITICO
                   MultiplicadorDano=2;
                   FuerzaGolpe=1000;
               }
               if(FuerzaGolpe>ASuperar){//Acierta el ataque
                    int FijoDan=pj.getModificador(pj.getFuerza())+
                            DevuelveModificadorCampo(BonificadorDanoPrimario)-DevuelveModificadorCampo(PenalizadorDanoPrimario);
                    if(!IndividuoActual.getDeRangoDerecha().equals("No")){//SI ES DE RANGO NO SE SUMA EL MODIFICADOR DE FUERZA
                        FijoDan=FijoDan-pj.getModificador(pj.getFuerza());
                    }
                    int DanoCausado=CausarDano(Dado(pj.getDanoArmaDerecha()), FijoDan, IndObjetivo, MultiplicadorDano);

                    String TextGolpe="";
                    if(FuerzaGolpe==1000){
                        TextGolpe="- Golpe número "+i+" ¡CRÍTICO! x2 de daño: " + DanoCausado;
                    }else{
                        TextGolpe="- Golpe número "+i+" ¡ACIERTA! con una fuerza de "+FuerzaGolpe+" supera "+ASuperar+" y causa "+
                    }

                    AreaNarrador.setText(AreaNarrador.getText()+TextGolpe);
                }else{//FALLA
                    String TextGolpe="- Golpe número "+i+"¡FALLA! con una fuerza de "+FuerzaGolpe+" NO supera "+ASuperar+"\n";
                    AreaNarrador.setText(AreaNarrador.getText()+TextGolpe);
                }
           }else{//PIFIAAAAAAAAAAAAAAAAA
               String Pifia="- ¡¡¡PIFIA!!!";
               AreaNarrador.setText(AreaNarrador.getText()+Pifia+"\n");
               i=4;
           }
        }
}
```

Figura 7.12: Codificación de Ataque

A grandes rasgos lo que hace el código es realizar tantos ataques como la habilidad atacante permita (máximo 4), porque cuando encuentra un -99 ya no ataca, porque este número indica que el personaje no es capaz de realizarlo.

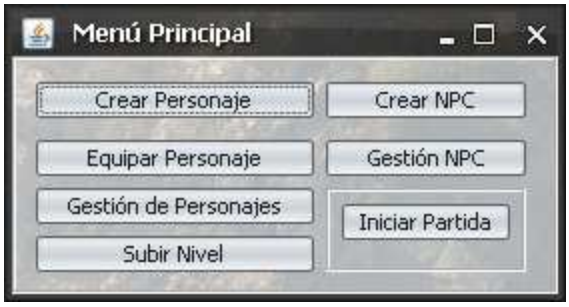
En cada iteración obtiene la fuerza del golpe a realizar sumando su ataque a la tirada de un dado de 20 (Función *Dado(20)*), y comprobará si este golpe supera o no la defensa del rival de superarla causar el daño. Como se puede observar, en el caso de que se suceda un crítico causara el doble de daño y acertara automáticamente gracias a asignar una Fuerza de 1000 al golpe, si por lo opuesto sucediese una pifia, el golpe falla inmediatamente y se cancela la sucesión de golpes.

Revelar que si el personaje portase otra arma en su mano no diestra, se provocaría después otra sucesión semejante pero aplicando los penalizadores por no usar su mano buena.

Anexo II: Manual de usuario

Generalmente la aplicación se ha creado para que su uso resulte realmente para cualquiera con una mínima base sobre el funcionamiento del juego.

El ***Menú principal*** resulta ser muy sencillo, en él se muestran las diferentes opciones de creación y gestión de personajes y NPCs; Finalmente el botón situado en un panel individual sirve para ***Iniciar Partida*** y llevarnos a una nueva interfaz donde realizar todas las acciones propias de una partida.



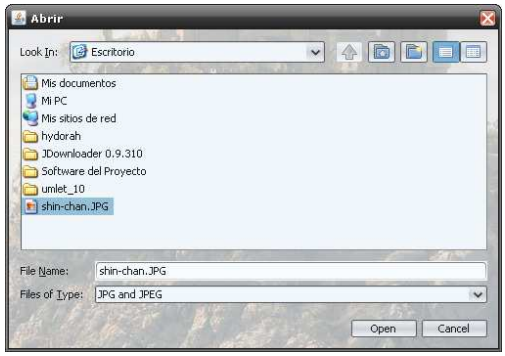
Crear Personaje

Se muestra la posibilidad de escribir el nombre, elegir raza y clase, y seleccionar el nivel que se quiere, además pulsando el botón ***Datos Personaje*** se accederá a otra ventana donde colocar toda la información adicional. Con el Botón ***Cambiar Imagen*** se mostrará una pantalla para seleccionar la imagen del avatar del Personaje. El siguiente paso consiste en seleccionar un método de generación de puntuaciones de característica, el ***Método Legal*** sigue las reglas del juego, mientras que ***Personalizar*** permite elegir las características con total libertad, en el momento que se selecciona un método se borran lo construido con el otro, y viceversa.

Los mencionados pasos se pueden realizar en cualquier orden. Después de haber otorgado lss características con cualquiera de los dos métodos se habilitará el botón ***Aumentar*** cuya pulsación

mostrará una nueva ventana dónde bonificar los puntos de característica Al finalizar este paso se habilitará el botón ***Aplicar*** que creará el personaje y habilitará el panel inferior.

La ventana ***Datos*** de la izquierda surge al pulsar el botón ***Datos Personaje*** desde la ventana anterior, el formulario sirve para rellenar datos adicionales que no influyen en el transcurso de la partida, pues sólo son mera información contextual. No es necesario rellenar todos los campos, se pueden dejar en blanco. Pulsar el botón ***Aceptar*** al finalizar.



El Dialogo ***Abrir*** aparecerá al pulsar el botón ***Cambiar Imagen***. Se trata de un explorador de archivos para escoger la imagen de avatar que queramos. Sólo acepta imágenes en los formatos JPG y JPEG. Al finalizar finalizado pulsaremos el botón ***Open***

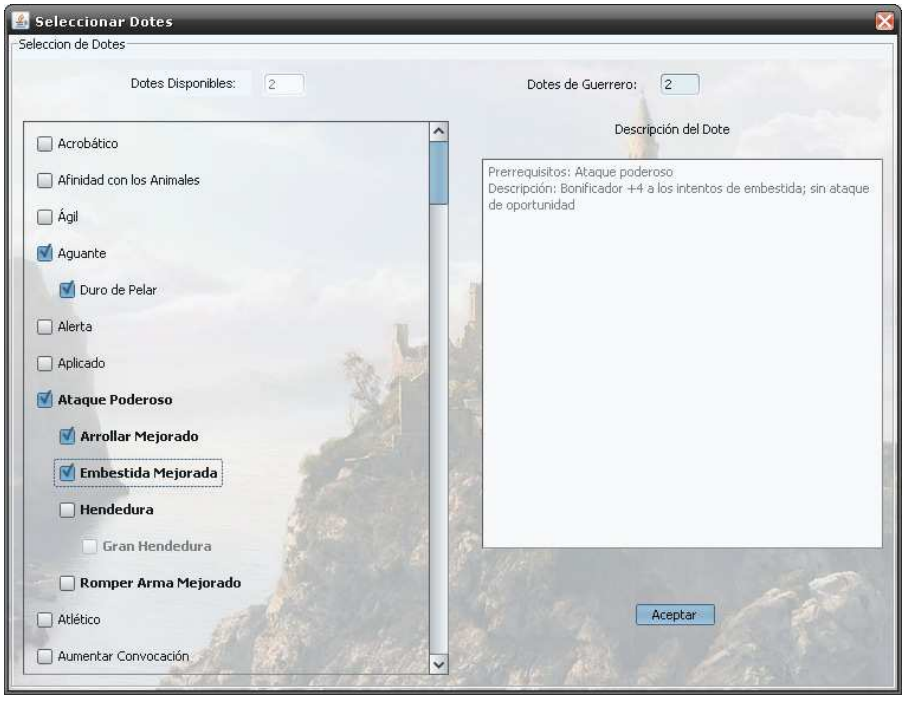
Se accede al dialogo ***Método Legal*** a través de del botón igualmente denominado. Aquí se asignan las características del personaje. En el combo seleccionado aparecen 6 números distintos generados aleatoriamente, el cuadro en amarillo indica la característica a la que va a relacionarse el número que escojamos. Con el botón ***Asignar*** ejecutaremos dicha acción. Así seguidamente hasta rellenar las 6 características. Si no nos gustan los números que nos han tocado podemos generar una nueva ristra de 6 con el Botón ***Volver a Generar,*** pero se reseteará el formulario. Cuando hayamos finalizado pulsaremos el botón ***Aceptar***, en caso de arrepentirnos pulsaremos ***Cancelar***

Si en vez de usar el método legal queremos seleccionar con total libertad las características pulsaremos el botón ***Personalizar.*** La ventana nos permite

escoger mediante las flechas las cantidades que queramos. Cuando hayamos decidido pulsaremos el botón ***Aceptar.*** Al haber finalizado el uso este método de decisión o el legal, prevalecerá el último utilizado y se habilitará el Botón ***Aumentar Características.***

Para poder continuar es necesario haber pulsado accedido a esta ventana. Aquí se muestran los puntos disponibles de los que disponemos para subir las características que queramos (la cantidad de puntos depende del nivel). Con los botones **+** otorgaremos la bonificación, finalmente pulsaremos aceptar. Seguidamente se habilitará el botón ***Aplicar*** del menú ***Crear Personaje,*** al pulsarlo se almacenará lo que llevamos construido en la base de datos así que aunque cerremos la ventana después de ese paso, existirá la instancia del personaje.

Tras los pasos anteriores la interfaz quedará de la siguiente forma. Se habrá inhabilitado todas las opciones anteriores y quedará utilizar las opciones restantes para acabar de completar el personaje que acabamos de crear. Cuando hayamos finalizado de hacer todo pulsaremos aceptar para cerrar la ventana.

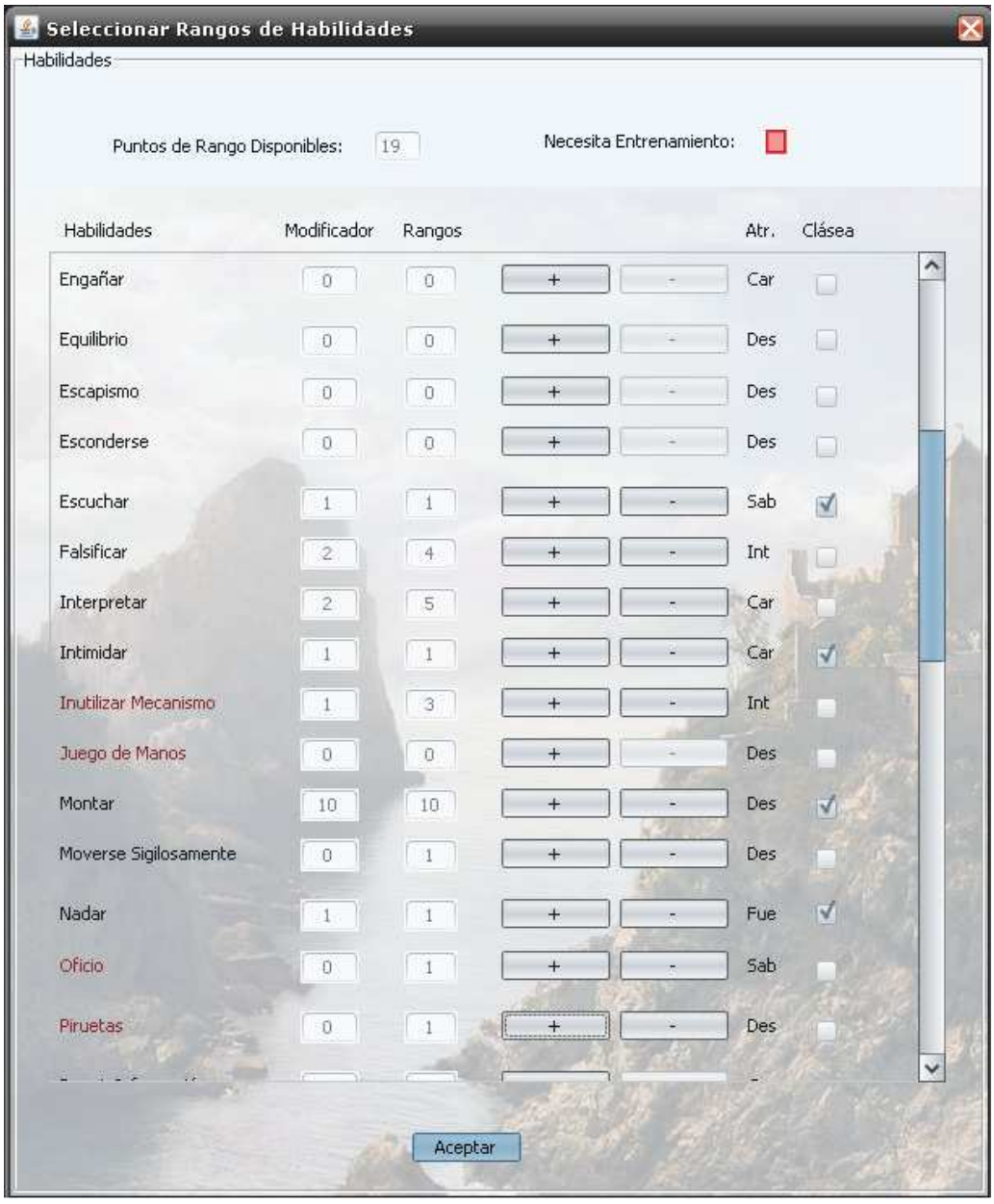


Seleccionar Dotes

A esta interfaz se accede cuando se crea un personaje o subiendo nivel. Se observa una lista de checkbox, y los puntos disponibles se van agotando a medida que seleccionamos dotes. Al pasar por encima de uno se puede ver su descripción en

el área de texto de la derecha. Para poder seleccionar algunas dotes se han de cumplir ciertos prerrequisitos, como pueden ser otras dotes o cierto nivel de característica. Este caso en particular es especial, pues se trata de un personaje de clase Guerrero, al seleccionar una dote en negrita, se gastaran puntos de guerrero y no normales, a no ser que ya no se dispongan de más de ellos.

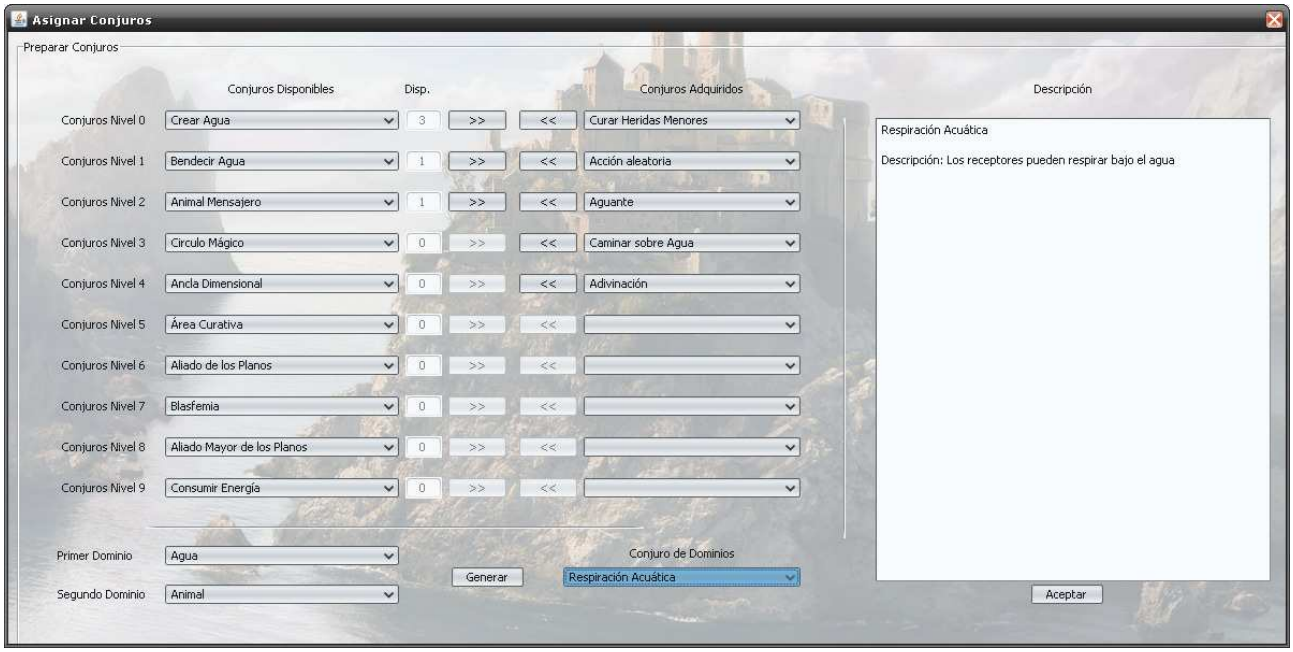
Seleccionar Habilidades



A la ventana para seleccionar los puntos de habilidad se accede de la misma forma que para las dotes. Gracias los botones **+** y **–** se pueden distribuir los rangos a través de las habilidades. Al igual que sucedía con las Dotes, no puede quedar un número negativo de Rangos Disponibles.

Como se indica, las dotes en rojo necesitan entrenamiento para ser aprendidas. Esto quiere decir, que si no se gasta ningún punto en ellas, luego no se podrán usar. Y finalmente aquellas dotes marcadas como claseas no gastan el doble de puntos para ser aprendidas como sucede con las demás. Cuando se pulse ***Aceptar*** se cerrará la ventana y se almacenarán los cambios.

Asignar Conjuros



La última de las interfaces que acaban de construir a los personajes es la que permite asignar los conjuros. Se accede de la misma forma que las dos anteriores. Si el personaje seleccionado es capaz de realizar conjuros se mostrarán los que pueden asignar a la izquierda y los ya asignados quedarán a la derecha. La cantidad de conjuros que sea capaz de aprender dependerá de la clase y el nivel. Mediante los botones **>>** y **<<** se pueden elegir los conjuros deseados, como se puede ver en la imagen, la aplicación no dejará asignar más conjuros de un nivel si se ha superado el máximo (mirar Conjuros Nivel 4).

Solamente la clase clérigo tiene derecho a seleccionar un conjuro de dominios, con los combos de abajo se pueden seleccionar los dominios preferidos y el conjuro que quede seleccionado en ***Conjuro de Dominios*** será almacenado**.** Cada vez que cambiemos uno de los dos dominios, o ambos, con el botón ***Generar*** se actualizará el combo con los nuevos conjuros.

Con este paso se finaliza todo el proceso de creación de personajes.

Gestión de Personajes

La siguiente interfaz tiene como función principal el referenciar personajes, así que siempre que se necesite cargar un personaje surgirá esta ventana para dicho propósito. Su funcionamiento es muy es sencillo, basta con seleccionar el personaje deseado haciendo click sobre él para seguidamente pulsar ***aceptar***.

Id	Nombre	Nivel	Raza	Clase
357	ShinChan	6	Elfo	Clérigo
368	Gandalf	6	Humano	Mago
369	Legolas	8	Elfo	Explorador
370	Manolo	6	Humano	Monje
371	SalvajeO	4	Semiorco	Bárbaro
372	Danny	4	Mediano	Druida
373	Eric	1	Humano	Paladín
374	Atila	6	Humano	Bárbaro
375	Promet	1	Gnomo	Pícaro
376	David	10	Gnomo	Druida
377	Merlin	6	Humano	Mago
378	Miliki	5	Elfo	Bardo
379	Galindo	8	Enano	Paladín
380	Osaka	6	Semielfo	Hechicero
381	Robin Hood	6	Humano	Explorador
382	Gingo	7	Gnomo	Guerrero
383	Ferdi	1	Elfo	Druida
384	Aragón	1	Humano	Explorador
385	Carlos Jesus	7	Semielfo	Hechicero
386	Miniyo	6	Gnomo	Paladín

Para utilizar las funcionalidades de borrar personaje o ver hoja de personaje (Generará un fichero PDF) sólo se ha de pulsar en sus respectivos botones después de haber procedido de la misma forma que se hizo para cargar un personaje.

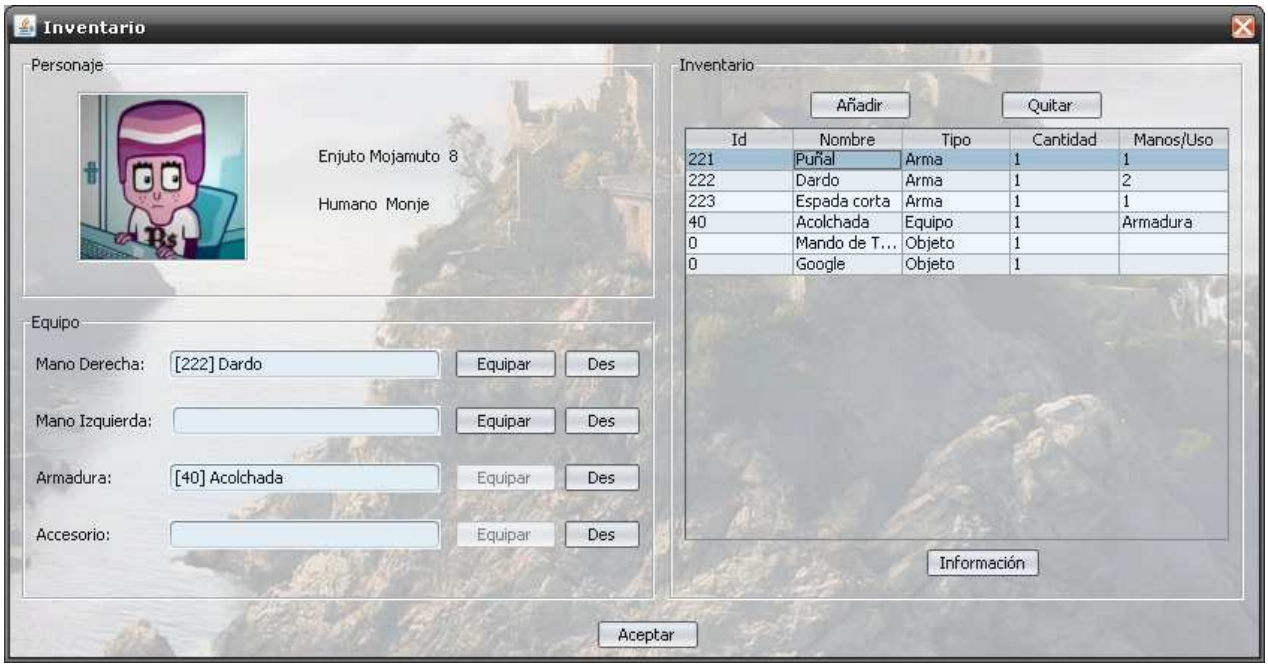
La funcionalidad Eliminar quedará inhabilitada en el caso de acceder a esta ventana desde la interfaz Partida. Y finalmente resta decir que el ***gestor de NPC's*** funciona exactamente de la misma forma sólo que en vez de disponer de la posibilidad de generar un fichero PDF, se mostrarán los datos en otra ventana.

Subir Nivel

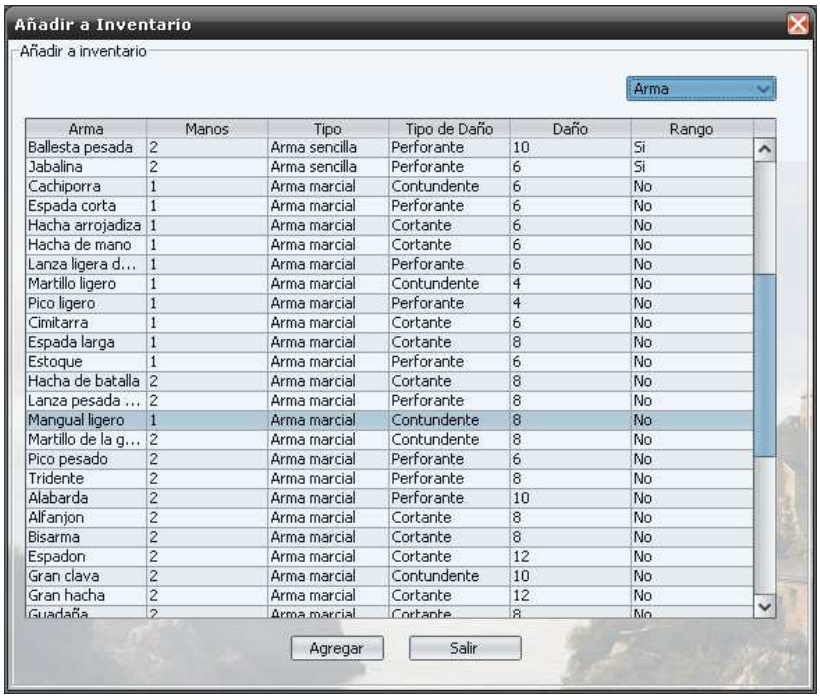
Se puede subir nivel desde el menú principal y desde el menú de la interfaz partida. Para ello también surgirá el ***gestor de personajes*** para poder indicar a cual subirle nivel.

El menú subir nivel permite subir un punto de característica si toca, y dispone las opciones que había en la creación de personajes para otorgar lo que aporte el nuevo nivel.

Gestionar el Inventario

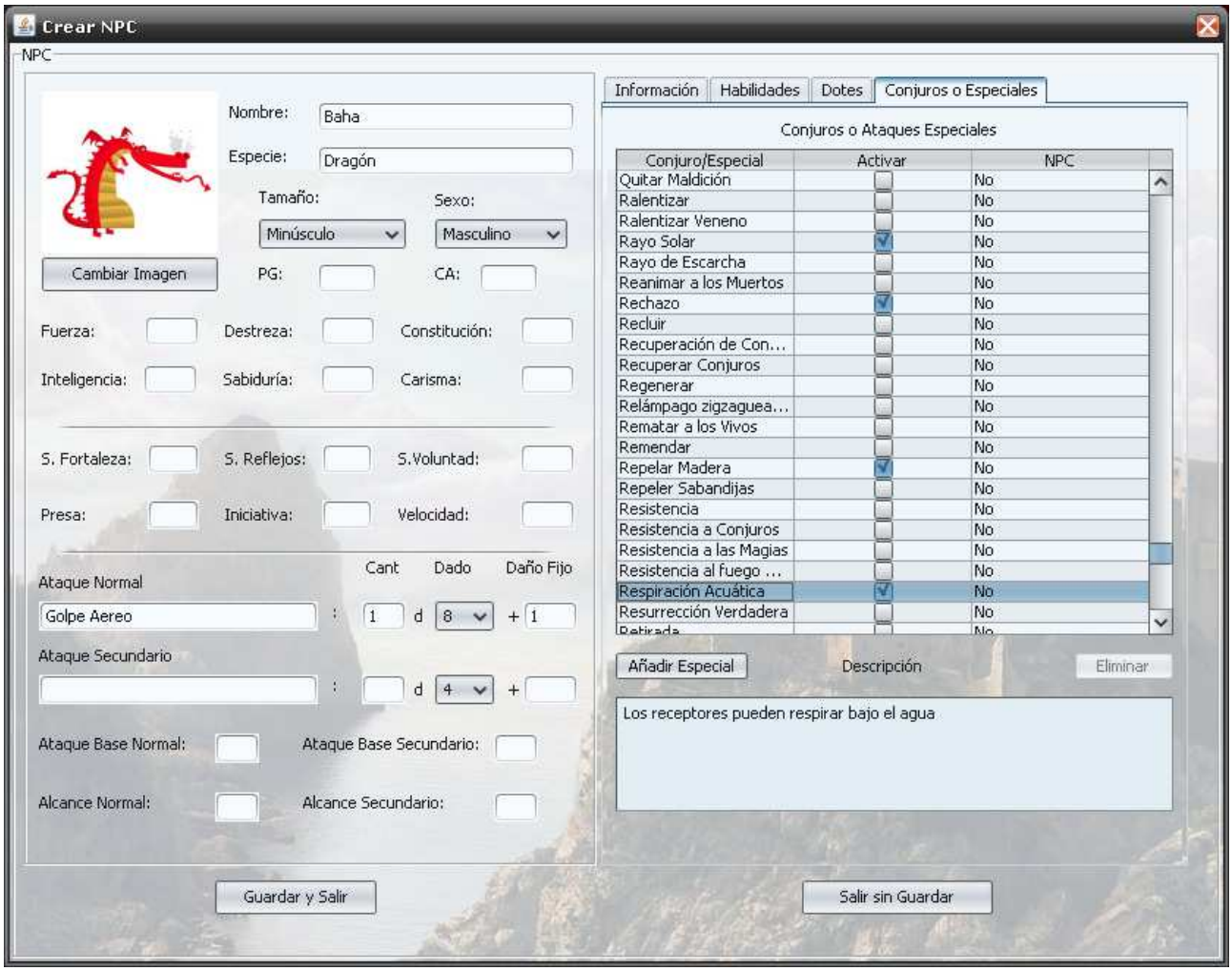


En la tabla de la izquierda se muestran todos los objetos que posee ese personaje en su inventario. Se puede ver información más detallada de cada uno seleccionándolo y pulsando el botón ***Información***. Algunos elementos son equipables así que seleccionándolos se habilitarán los botones ***Equipar*** de las partes del cuerpo correspondientes. Cuando se quiere Desequipar un objeto basta con utilizar los botones ***Des.*** Para quitar elementos del inventario hay que utilizar el botón ***Quitar*** posteriormente a haber seleccionado uno. Para añadir objetos hay que utilizar el botón ***Añadir*** que mostrará una ventana.



Esta ventana aparece para añadir objetos. Con el ComboBox de arriba se puede seleccionar los tipos de objeto que se quiere que se muestren. Seguidamente basta con seleccionar el objeto que se quiere añadir al inventario y pulsar ***Agregar,*** así tantas veces como se desee. Cuando se haya acabado pulsar el botón ***Salir***.

Crear NPC



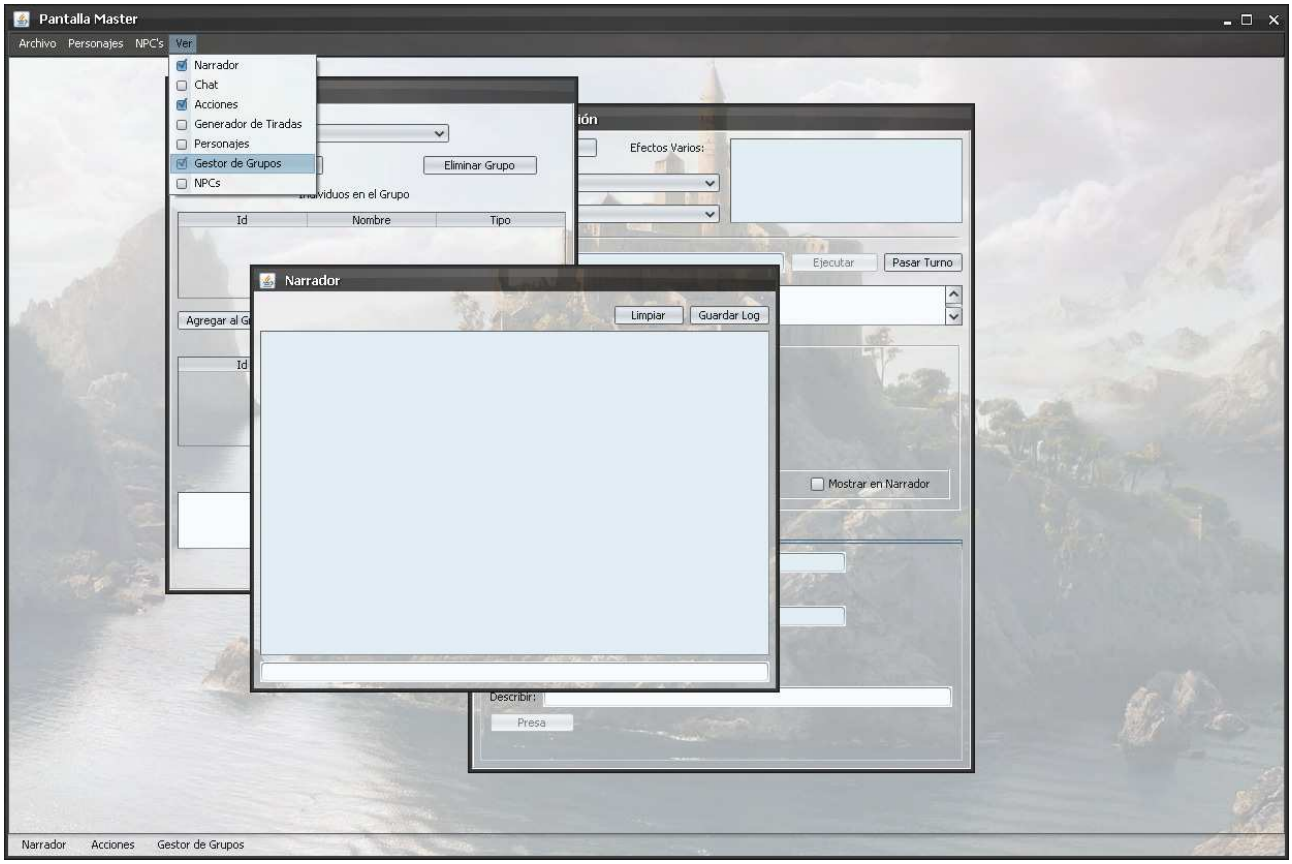
La interfaz para crear NPCs no tiene limitaciones como pasa con los Personajes.. En las pestañas de la derecha se encuentran tablas para seleccionar dotes, habilidades y conjuros. Después de rellenar todo el formulario pulsando ***Guardar y Salir*** se habrá finalizado.

Si se quiere crear un conjuro personalizado, se puede acceder a esta ventana con el botón ***Añadir Especial***. Este pequeño formulario cumplirá dicho propósito, en el momento que se pulse crear aparecerá el nuevo conjuro en la tabla.

Se pueden eliminar los conjuros creados por algún usuario mediante el Botón ***Eliminar*** de la interfaz crear, pero no se permite eliminar conjuros propios del juego.

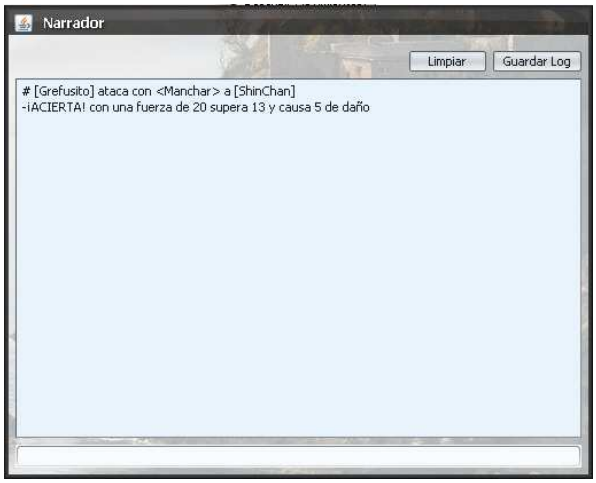
Partida Iniciada

La interfaz partida está compuesta de varias funcionalidades las cuales se irán explicando después.



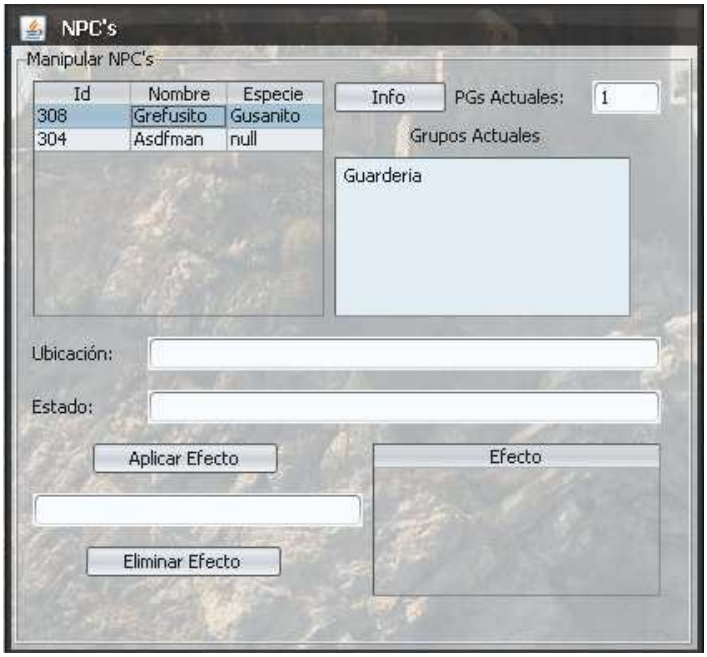
La forma de manejar la interfaz es activando las ventanas que queramos que se muestren mediante el menú ***ver,*** si existen muchas ventanas y resulta complicado acceder a una, podemos utilizar el botón correspondiente de la barra inferior para que la ventana se coloque encima de todas las demás.

Para poder jugar es necesario cargar los personajes y NPC's que van a participar, para ello se puede usar el menú ***Personajes*** y el menú ***NPC's***. Allí se encuentran las opciones correspondientes, incluyendo también las que permiten quitarlos de la partida.



La utilidad del Narrador es la de mostrar todo lo que sucede en la partida. Todas las acciones en las que se marque la acción ***Mostrar en Narrador*** saldrán reflejadas aquí. Con el botón ***Limpiar*** se vaciará el área de texto. Y con el botón ***Guardar Log*** se creará un archivo de texto con toda la información de la partida.

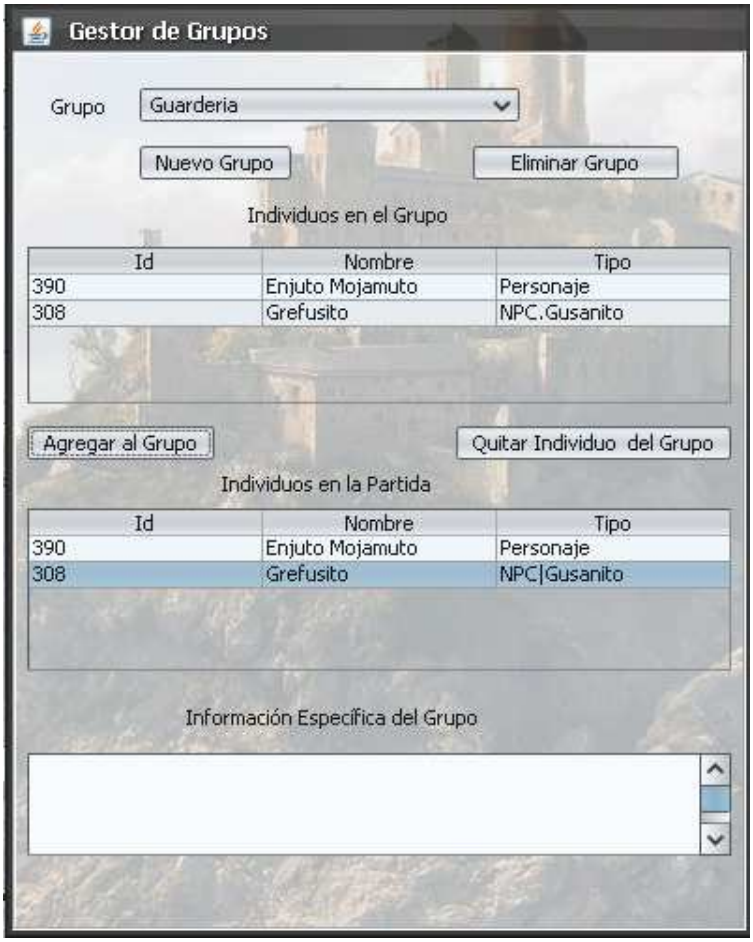
La ventana interna de Personajes permite ver el estado de los mismo en la partida y los PG's actuales. También proporciona acceso a la hoja de personajes y al inventario con sus respectivos botones. La única función del combobox es mostrar los grupos en los que se encuentra el personaje. Se puede seleccionar los diferentes personajes a través de las pestañas. Para aplicar efectos basta con rellenar el campo con el efecto y pulsar el botón ***Aplicar Efecto***. Para Eliminar un efecto sólo hay que seleccionarlo y pulsar el botón ***Eliminar Efecto***.



Los NPC's también disponen de una ventana en la que ver toda la información necesaria sobre ellos. Su funcionamiento es el mismo que la ventana anterior, exceptuando que la herramienta utilizada para navegar a través de los NPC's es una tabla donde se verá la información del NPC seleccionado.

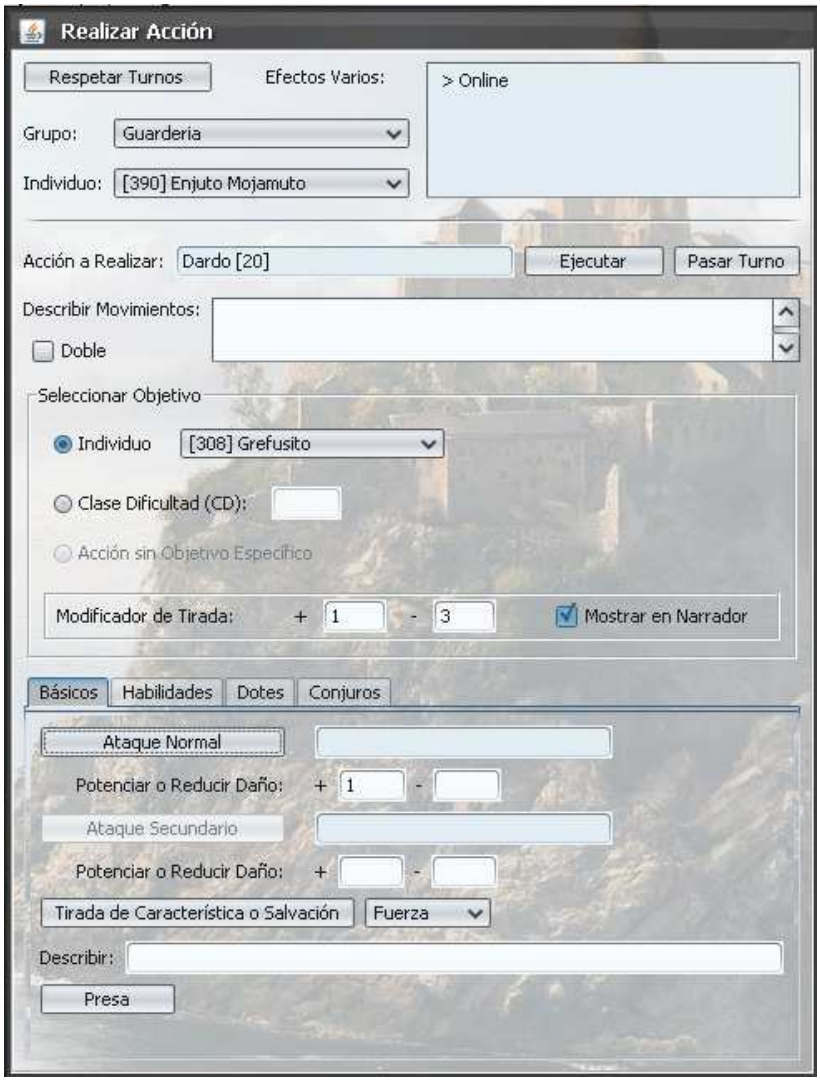
Para poder realizar tiradas aleatorias se ha de configurar dicha tirada primero. Se coloca el número de dados, el dado y el

resultado fijo que se sumará. Con el botón ***Tirar*** se calculará la tirada. A partir de este punto son varias las opciones. Si queremos que se aplique a un individuo basta con seleccionarlo en el ComboBox, y luego decidimos si queremos que se le aplique la tirada activando ***aplica***r y si se trata de daño o de curación mediante el checkbox ***Curación.*** Mediante ***Mostrar en Narrador*** se decide si se quiere que se muestre la tirada en el narrador. Cuando hayamos decidido pulsando el botón ***Ejecutar*** sucederá lo que hayamos configurado.



Lo primero que haremos en el gestor de grupos será crear un grupo, para ello usaremos el botón ***Nuevo Grupo***, nos aparecerá un pequeño dialogo donde colocar el nombre. Por el contrario si queremos eliminar un grupo, lo seleccionaremos en el Combo, y pulsaremos ***Eliminar Grupo***. En la primera tabla se muestran los individuos en el grupo seleccionado, y en la de más abajo se muestran todos los de la partida. Podemos agregarlos a los grupos y quitarlos de los grupos mediante los botones ***Agregar al Grupo*** y ***Quitar Individuo del Grupo.*** Finalmente en la parte de abajo podremos introducir toda la información específica de cada grupo que queramos. Hay que tener cuidado porque cuando quitemos individuos de la partida también desaparecerán de todos los grupos en los que se encuentren.

Después de que hayamos introducido al menos un Personaje o NPC en la partida podremos realizar acciones mediante esta interfaz. Si activamos el ***Mostrar en Narrador*** veremos los resultados de las acciones descritos en la ventana Narrador.



La forma de proceder empieza por seleccionar un grupo específico, se disponen de grupos genéricos también donde se hallan todos los individuos si así se desea. El siguiente paso será escoger el Personaje que realizará la acción. Según el personaje que hayamos escogido en las pestañas de abajo veremos las acciones de las que dispone el personaje, pulsando por ejemplo, Ataque Normal quedará fijado ese tipo de acción. En el caso del ataque por ejemplo podemos bonificar o penalizar el daño si así lo queremos. Mencionar que en todos los casos se puede añadir modificadores a las tiradas mediante los campos de texto de debajo de la Selección de Objetivo.

Con la acción decidida queda seleccionar un objetivo, según el tipo de acción podrá tener unos objetivos u otros. El caso común es poder enfrentarse a una Clase de Dificultad o a un Individuo, habrá que activar la opción que queramos. Para el primer caso sólo hay que colocar la dificultad deseada, en el otro caso será necesario indicar el Individuo objetivo, pero hay que tener en cuenta que sólo se verán los individuos habidos en el grupo del ejecutor de la acción. Sólo resta entonces pulsar el botón ***Ejecutar*** para llevar a cabo la acción.

Si se quiere se puede activar el orden de turnos, sólo hay que activar la opción ***Respetar Turno.*** Con esto al final de cada acción se pasará automáticamente al siguiente individuo en orden de iniciativa, y no se podrá seleccionar el individuo activo manualmente.

David Arjona Fernández

Sabadell, 29 de Junio de 2010