DEPARTAMENTO DE INGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL
INGENIERÍA EN INFORMÁTICA
PROYECTO DE SISTEMAS INFORMÁTICOS
CURSO 2011/2012

Rebel War, Juego online multijugador de cartas

Autores:
Elena Hernández Delgado
Natalia Ronda Villora
Eduardo Sanz Curto

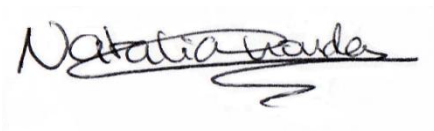
Directores:
Rubén Fuentes Fernández
María Guijarro Mata-García

Autorización

Los abajo firmantes, matriculados en la asignatura Sistemas Informáticos de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado durante el curso académico 2011-2012 bajo la dirección de la Dra. María Guijarro Mata-García y el Dr. Rubén Fuentes Fernández del Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Elena Hernández Delgado

Natalia Ronda Villora



Eduardo Sanz Curto

Resumen:

Los videojuegos online multijugador son videojuegos de navegador (se juegan a través de un navegador web) en los que el usuario juega contra otros usuarios en tiempo real. Este tipo de juegos está muy de moda en Internet por la comodidad que supone no tener que instalar ningún programa específico para poder jugar. Solo es necesario conectarse a los servidores del juego vía una dirección web accesible mediante un navegador. Entre sus inconvenientes, los videojuegos online a menudo requieren mucho tiempo de juego y son parcial o completamente de pago. Dentro de estos juegos, los juegos de cartas son aquellos en los que unas cartas representan los distintos elementos del juego. Estos juegos añaden a las partidas una importante componente de intercambio y comercio de las cartas, bien por afán de coleccionismo o para mejorar las habilidades en el propio juego.

El proyecto presentado en esta memoria consiste en el desarrollo de uno de estos juegos. *Rebel War* es un juego online multijugador de cartas con un trasfondo de fantasía épica. Lo que lo diferencia de otros videojuegos disponibles en Internet es que *Rebel War* es completamente gratuito y que ha sido diseñado de manera que no requiere una dedicación intensiva por parte de sus usuarios. Además, este juego ofrece al usuario gran cantidad de funcionalidades para gestionar sus cartas, tratando de reproducir la experiencia de poseerlas físicamente.

Palabras clave: videojuego, multijugador, navegador, cartas, tiempo real.

Abstract:

Multiplayer online games are browser games (i.e. games played through a browser) in which the user plays against other users in real time. Such games are very popular on the Internet due to, among other reasons, the convenience of not requiring installing any specific software to play. The user only needs to connect to game servers accessible via a web address using a browser. Among its drawbacks, online video games often demand long playing sessions at specific moments, and they charge some costs on the player. Among these games, card games are those in which cards represent the different elements of the game. These games add to the playing activity an important component of trade and commerce of cards, either for the sake of collecting itself or to improve skills in the game.

The project presented herein is the development of one of these games. *Rebel War* is a multiplayer online game of cards belonging to the epic fantasy genre. Its main difference with other games available on the Internet is that *Rebel War* is completely free and has been designed so that does not require an intensive commitment by its users. This game also offers to users many features to manage their cards, trying to reproduce the experience of physically possessing them.

Keywords: game, multiplayer, browser, cards, real time.

Índice

Índice de ilustraciones

1. Introducción

1.1. Planteamiento del problema

Hoy en día, los "videojuegos online" o "videojuegos en línea" están experimentando un gran auge dentro de las opciones de ocio que ofrece Internet. La posibilidad de jugar a cualquier hora, usando meramente el navegador de un ordenador, sin necesidad de conectar ninguna consola a la televisión, ni de realizar ninguna descarga ni instalación de software adicional, ha atraído a muchos internautas a esta forma de ocio interactivo.

Los videojuegos de navegador son videojuegos online con los que se puede jugar utilizando únicamente un navegador web (*Internet Explorer*, *Mozilla Firefox*, *Opera*, *Google Chrome*, etc.). Suelen estar realizados en su mayor parte mediante código HTML (Wikipedia HTML, 2012) con elementos adicionales en FLASH (Adobe Flash, 2012), PHP (PHP, 2012) o JavaScript (Wikipedia JavaScript, 2012).

Dentro de los videojuegos online existen los llamados "videojuegos de navegador persistentes" (PBBG, del inglés *Persistent Browser-Based Game*). Los PBBG son los videojuegos de navegador en los que el progreso del juego es logrado a través de múltiples sesiones, con constancia y dedicación. Este progreso se guarda de una sesión a otra, de manera que se proporcionan diferentes retos cada vez, y una complejidad y profundidad de juego de un nivel acorde con el que el propio jugador va adquiriendo. Esa profundidad que ofrece el progreso del usuario multisesión, añadida a la característica multijugador de los juegos de Internet, proporciona a los jugadores la experiencia más completa entre los videojuegos online.

Los PBBG también tienen algunos inconvenientes para sus jugadores. La mayoría de ellos requieren mucha dedicación, es decir, pasarse un gran número de horas delante de la pantalla para poder progresar en el juego. En algunos es necesario conectarse a una hora concreta para completar alguna misión o dar alguna orden vital para seguir siendo competitivos y mantener nuestro estatus; en otros, las partidas contra otros jugadores son por turnos y llevan mucho tiempo. Esto hace que para poder disfrutar de un videojuego online de estas características, el usuario tenga que plantearse de antemano si va a poder dedicarle el tiempo necesario para ser competitivo y no quedarse atrás.

En la actualidad, existen PBBG de prácticamente todo tipo: de estrategia (ej. http://www.ogame.com.es,http://medievol.es) de rol (ej. http://www.drakensang-online.es, http://www.darkonline.net), de deportes (ej. http://www.hattrick.org, http://buzzerbeater.com) coleccionable (ej. http://urban-rivals.com, http://ederon.com), etc. Es dentro de esta última categoría donde se ha centrado la atención de este proyecto. Aunque a primera vista parezca una subcategoría dentro de un mundo poco conocido como el del ocio de Internet, vamos a presentar unos datos que muestran la relevancia de este tipo de juegos.

Los juegos coleccionables suponen un gran mercado que mueve millones de euros anualmente en ventas. Por ejemplo, *Wizards of the Coast*, empresa que comercializa las cartas *Magic: the Gathering* (Magic: The Gathering, 2012) facturó en 1995, 65 millones de dólares (Wikipedia Wizards of the Coast, 2012). El catálogo de juegos va desde los caros *wargames* con sus miniaturas de plomo (ej. http://www.games-workshop.com), pasando por los más asequibles y modernos juegos de tablero con miniaturas de plástico (ej. http://heroclix.com) hasta los populares juegos de cartas o *trading card games*.

El coleccionismo de cartas a nivel físico es un mundo dominado por las cartas *Magic: The Gathering*, que recientemente han desembarcado en Internet también con su versión online. No obstante, existen otros muchos juegos coleccionables como Pokemon (http://www.pokemon.com/es/pokemon-trading-card-game/) o YuGiOh (http://yugioh-card.com) que demuestran que es una alternativa de ocio muy popular. Sin embargo, es por todos conocido (y dentro del mundo del coleccionismo muy criticado), que este coleccionismo es un hobby caro. Conlleva el intentar obtener todas o la mayoría de cartas u objetos posibles, para poder ser competitivo o solo por el hecho de poseerlas. Este comportamiento suele llevar también a que surjan mercados secundarios entre jugadores, a que se cree una microeconomía dentro del mundo del juego y a que la interactuación entre jugadores vaya más allá de las propias partidas y sea aún mucho más rica debido al intercambio de productos del propio juego.

Estas características propias de los juegos de coleccionismo se transfieren a su versión online. Gracias a Internet, los mercados de productos de la colección son más accesibles, la interacción entre jugadores es más ágil por medio de los foros, y el número de rivales con los que contactar para jugar las partidas es inmenso. Es por ello por lo que los juegos coleccionables han encontrado acomodo de una forma tan natural en la red. A modo de ejemplo, *Magic: The gathering* tiene 6 millones de jugadores, se distribuye en 52 países y es considerado el segundo juego más jugado del mundo después del póker.

Por otro lado, se está convirtiendo en algo frecuente que los videojuegos online sean gratis, pero con incentivos “de pago”. Para ello suelen tener dos tipos de dinero virtual: un dinero normal que use el juego para sus intercambios, que se gana jugando y se consigue fácilmente; y un dinero especial del juego que solo se obtiene con el consiguiente desembolso de dinero real, mediante pago con tarjeta, sms, paypal, etc. Este dinero especial suele servir para conseguir mejoras únicas que no están disponibles con el otro tipo de dinero. En la mayoría de los juegos, este aspecto es opcional y se presenta al usuario como una alternativa para avanzar más deprisa y para conseguir mejoras de otra forma inalcanzables o beneficios dentro del juego. Lo que ocurre es que de esta forma, el videojuego online pierde su espíritu de gratuidad, porque se vuelve injusto para aquellos que renuncian al pago opcional.

1.2. Objetivos del proyecto

A la vista de lo planteado en el punto anterior, se considera que existe un vacío en los videojuegos online de coleccionismo que se busca cubrir con el presente proyecto. Se

propone crear un videojuego online multijugador de cartas coleccionables, bajo las características de los PBBG, de sesiones rápidas y que no requiera mucha dedicación para avanzar y alcanzar un buen nivel o estatus. *Rebel War* pretende ser un videojuego online en el que el usuario elige el tiempo que le dedica, sin que ello vaya en perjuicio de la complejidad del mismo. Persigue ser en ese sentido igual de divertido para aquellos que puedan sentarse a jugar porque tengan 10 minutos libres como para aquellos que se pasen las noches intentando terminar su colección de cartas y ganar partida tras partida. Tanto si se quiere jugar una partida suelta, porque no se dispone de más tiempo, como si se quieren explorar en profundidad las tácticas y habilidades y dedicarle más horas, con *Rebel War* podrá ajustar el juego a sus necesidades de ocio.

Para lograr estas características se ha creado un juego dentro del mundo del coleccionismo que proporciona al usuario una experiencia completa a nivel de jugador. El juego mezcla el sistema de juego de cartas coleccionables con la riqueza del formato de rol clásico. Éste hace que las cartas del jugador evolucionen y varíen sus características en cada uno de sus diferentes niveles, multiplicando la complejidad del juego y por tanto la diversión.

Todo ello se hace de una forma totalmente gratuita. Las mejoras en *Rebel War* se consiguen cuanto más se juegue, porque hay que premiar a los más jugones, pero también cuanto mejor se juegue. De esta forma se evitan privilegios dentro del juego, desequilibrios de nivel e injusticias entre los jugadores que gastan dinero real y los que no. No obstante, el juego podría ser modificado para incluir estas variantes de pago.

1.3. Planteamiento del trabajo

Para conseguir abordar los objetivos propuestos, el trabajo se ha organizado en tres fases. Éstas pretenden organizar el desarrollo en subsistemas sucesivos que se apoyen en la funcionalidad desarrollada en las fases anteriores.

En primer lugar se van a diseñar e implementar los algoritmos de la batalla entre jugadores. Estos algoritmos tratarán de evitar en la medida de lo posible que ésta se desarrolle en múltiples turnos y se haga muy larga.

En segundo lugar se implementará la gestión de la batalla desde la interfaz de usuario. Se pondrá un tiempo límite para que cada acción (despliegue de tropas y adjudicación de órdenes) de los usuarios participantes se resuelva en un tiempo prudencial y se penalizará al usuario que exceda ese tiempo. Así se consigue que las batallas sean de corta duración y no aburran a los jugadores. También se tendrá en cuenta que cada jugador pueda combatir cuando le plazca sin tener que cumplir unos horarios preestablecidos.

En tercer lugar, se implementarán el resto de funcionalidades básicas de un juego online de cartas coleccionables. Éstas permiten a los usuarios gestionar sus cartas. Aquí se incluyen las funcionalidades para, por ejemplo, la gestión de la colección de cartas, visualización de estadísticas y compra-venta de cartas.

Ninguna de las características de la aplicación requerirá pago alguno, por lo que todos los jugadores tendrán las mismas oportunidades de conseguir completar su colección y tener un buen estatus dentro de *Rebel War*.

Para terminar, es preciso subrayar la importancia de la interfaz de usuario y el aspecto visual en una aplicación de estas características. Por ello, las fases anteriores siempre incluirán tareas específicas de desarrollo y validación de estas interfaces, buscando que resulten atractivas y profesionales.

1.4. Estructura de la memoria

Esta memoria se encuentra estructurada de la manera siguiente. En el capítulo 2 se ofrece una revisión del estado del arte. En ella se describe la evolución de los juegos online multijugador y, más en concreto, de los juegos online de cartas coleccionables. Esta sección usa está revisión para hacer una pequeña comparación de lo que ofrecen estos juegos, lo que sirve de base para establecer los requisitos de *Rebel War*. En el capítulo 3 se realiza un análisis detallado del proyecto que se va a realizar, estudiando sus requisitos, alcance, viabilidad, riesgos, estándares, tecnologías utilizadas y planificación del trabajo. El capítulo 4 presenta la arquitectura detallada de la aplicación, describiendo diseño e implementación. En el capítulo 5 se comentan y explican las pruebas de uso y rendimiento realizadas a la aplicación y las conclusiones obtenidas. El capítulo 6 recoge las conclusiones finales de la memoria y el desarrollo del proyecto, así como las posibles ampliaciones y trabajos futuros. Por último, al final de esta memoria se encuentran los apéndices.

2. Revisión del estado del arte

2.1. Videojuegos online

El "videojuego online" es un juego en el que se participa a través de un dispositivo electrónico (ej. computadora o consola) que se conecta a un servidor usando una red. Dentro de estos juegos, este proyecto se centra en los videojuegos multijugador online, generalmente de estrategia y jugados usando un navegador. En el resto de la memoria, se usará el término MWG (del inglés *Multiplayer Web Game*) para referirse a esta categoría concreta.

Los MWG poseen una serie de características que los hacen competitivos frente a otros videojuegos de consola o PC, pese a que sean juegos menos atractivos en términos gráficos o de interacción. Estas características son:

- Se compite contra otros jugadores de todo el planeta en "tiempo real".
- No necesitan ningún tipo de instalación, tan sólo conexión a Internet y un navegador.
- Son juegos gratuitos, aunque bien es cierto que la mayoría suelen poseer formas de pago para facilitar mejoras individuales o ahorro de tiempo.
- Pueden ser jugados por todo el mundo. Normalmente se encuentran en inglés, español o traducidos a varios idiomas.

En la actualidad, el abanico de MWG que podemos encontrar es inmenso y de muy diversas temáticas, para satisfacer a un público cada vez más amplio y que pasa más horas frente al ordenador. A continuación se discuten algunos de los que han tenido más influencia en la evolución de este tipo de juegos.

Ogame (http://www.ogame.com.es) fue probablemente el primer juego de esta categoría que alcanzó una gran repercusión en términos de jugadores. Se trata de un videojuego futurista ambientado en el espacio y desarrollado por la empresa alemana *Gameforge* (http://es.gameforge.com). Cada jugador dispone de una base en un planeta y a partir de ahí tiene que desarrollarse a través de la consecución de recursos, construyendo minas, defensas, naves y colonizando otros planetas. Lo más divertido llega cuando el jugador ataca a otros para destruir sus naves y robar sus recursos. También se puede entrar en una alianza o bien empezar una propia, donde se puede conversar con otros jugadores y preparar diversas estrategias o guerras contra otras alianzas. El objetivo del juego es simplemente ser el mejor, lo cual, además de ser casi imposible lo hace infinito ya que no hay un final marcado.

A raíz de la popularidad de *Ogame*, han ido apareciendo muchos otros juegos similares con el mismo o parecido motor de juego y en los que solo varía la temática y ambientación. El objetivo de estos juegos se resume en tratar de alcanzar el primer puesto en la clasificación del juego. Otros ejemplos en esta línea son *Travian* (http://www.travian.net), ambientado en la época de la gloriosa civilización romana;

Ikariam (http://es.ikariam.com), con estética griega; o el más moderno *Thirst of Night* (http://www.kabam.com/thirst-of-night) basado en el mundo vampírico.

Muy similares aunque con ciertas diferencias, son aquellos en los que el jugador tan sólo tiene que "manejar" a un personaje, y preocuparse de la evolución del mismo en todos los aspectos. Este es el caso del *Bitefight* (http://bitefight.es), en el que el jugador se mete en la piel de un hombre lobo. Los juegos de este tipo reducen aún más las posibilidades del usuario y terminan por cansar muy pronto.

Dentro del mundo deportivo, nos encontramos con juegos con una gran cantidad de usuarios registrados. El referente dentro de este subgrupo es *Hattrick* (www.hattrick.org). *Hattrick* es un juego de fútbol desarrollado hace ya más de 15 años por la empresa sueca *ExtraLives AB* (http://www.extralives.com/), en el que cada usuario lleva las riendas de un equipo de fútbol en todos los niveles. El jugador realiza todas las gestiones económicas del club, fichajes y despidos, organiza los entrenamientos y prepara las alineaciones para cada partido. También existe una infinidad de MWG de otros deportes como: automovilismo, con el *F1 portal* (www.f1portal.net/); tenis con el *Rocking Rackets* (http://rockingrackets.com/); o baloncesto con el *Buzzer Beater* (http://www.buzzerbeater.com/).

Con la aparición de las redes sociales como *Facebook* (http://www.facebook.com) y *Tuenti* (www.tuenti.com), los juegos online han aprovechado para extenderse y utilizar directamente éstas páginas. De esta forma se permite jugar con los contactos o con el resto de usuarios de la red social. Ni que decir tiene que debido a su rápida difusión y facilidad de manejo, los MWG en redes sociales han conseguido atraer a gente nueva que no es jugadora habitual pero que se ha sentido atraída al ver jugar a sus contactos.

Este punto nos lleva a considerar el perfil del usuario de MWG. En general, se trata de personas exigentes e inconstantes, que debido a la amplia gama de videojuegos que ofrece Internet no dudarán en cambiar a otro juego si el que están probando no les convence. Por ello, para destacar en el mundo de los MWG hay que aportar algo diferente a los demás, ya sea en el aspecto gráfico, de jugabilidad o de entretenimiento.

La mayoría de los MWG optan por intentar aportar algo más espectacular gráficamente. Cada vez se ven juegos que tienen un aspecto visual más potente y que aprovechan mejor las posibilidades que les ofrecen los nuevos navegadores y las conexiones a Internet de alta velocidad. Sin embargo, también existen otros MWG que sin hacer un gran despliegue gráfico optan por innovar en el motor estratégico y crear modos de juego realmente originales y divertidos. Este el caso de *Empire Strike* (http://www.empire-strike.com).

Empire Strike es un juego creado por un español, que ha alcanzado la nada despreciable cifra de 500.000 usuarios registrados. Con una ambientación fantástica al más puro estilo tolkieniano, en *Empire Strike* cada usuario maneja un imperio de una de las razas a elegir, construyendo diferentes ciudades con recursos que se generan diariamente. Sobre un mapa rudimentario, el jugador debe evitar que sus ciudades

sean conquistadas por otros jugadores, mientras que intenta unir a su imperio ciudades de otros usuarios asediándolas con tropas y héroes. El juego incluye un sistema de comercio bastante simple y otro de formación de clanes. La interfaz gráfica se basa en una visualización de imágenes totalmente estáticas. Estas características podrían indicar que se trata de un juego común, pero no es así. El atractivo que hace este juego diferente a los demás es una idea simple: las partidas en el *Empire Strike* duran 2 meses exactos. Tienen por tanto un principio y un final, donde todos empiezan con los mismos parámetros y por tanto posibilidades, y la clasificación a los dos meses determina el vencedor y el resto de puestos. Basándose en esta idea, inédita en el mundo de los MWG, *Empire Strike* se ha convertido en uno de los juegos de referencia dentro de la comunidad de habla hispana de Internet.

Recientemente, la mejora y el mayor uso de los móviles de última generación, ha hecho que algunos juegos hayan dado el salto al mundo de las aplicaciones para teléfonos móviles. Ello les permite ofrecer a sus usuarios la posibilidad de jugar desde cualquier lugar. *Urban Rivals* (http://www.urban-rivals.com), un juego que se comentará más adelante, dispone de una versión para jugar en *iPhone* y en *Android*. Está bastante lograda y es de fácil manejo, lo que sin duda es un aliciente para aquellos usuarios que no podían dedicarle el tiempo que querían por no estar frente a una computadora. Es innegable que la comodidad y la facilidad del usuario a la hora de acceder al MWG siempre supondrán un beneficio para aumentar el número de jugadores registrados.

2.2. MWG de cartas coleccionables

Los MWG de cartas coleccionables se suelen enmarcar dentro de los juegos donde se da una mayor relevancia a aspectos como la jugabilidad y la estrategia, y menos al aspecto gráfico. Como su propio nombre sugiere, el mayor potencial de estos juegos es el coleccionismo de sus cartas. Ésta es una actividad que atrae por sí misma a su propio público a los MWG. Además de ser entretenidos, en mayor o menor medida, los videojuegos que contienen un factor de coleccionismo brindan la posibilidad de intentar conseguir todos los elementos disponibles en el juego o aquellos que se necesiten para ciertos aspectos. Es decir, hay un objetivo concreto y a largo plazo. A esto hay que añadir que un MWG de cartas, no requiere la constancia de otros juegos que son en tiempo real. Los juegos como *Ogame*, *Travian* o *Empire Strike* requieren una conexión constante al juego, que a veces obliga a estar pendiente del mismo en momentos determinados. Frente a ello, la ventaja de los juegos de cartas es obvia: uno juega cuando y durante el tiempo que quiere, sin compromiso de ningún tipo y pudiendo estar lo que desee sin volver a jugar.

Entre los MWG de cartas, uno de los juegos con más éxito del momento y que ha servido de inspiración y referencia para la realización de este proyecto es *Urban Rivals*. *Urban Rivals* es un videojuego de cartas coleccionables creado por la empresa francesa *Boostr* en 2006. Es muy completo y a pesar de lo que pueda parecer por lo sencillo de sus reglas, es tremendamente estratégico. Se trata de un modelo a seguir que asegura casi infinitas horas de entretenimiento a sus usuarios. Además, es todo un ejemplo de

cómo desarrollar un juego gratuito con opciones de pago en el que no se cierran puertas a los jugadores que no quieran pagar.

2.3. Lecciones aprendidas para *Rebel War*

Rebel War pretende aunar algunas de las características más apreciadas por los jugadores de los juegos online existentes con algunas aportaciones novedosas. Estas características novedosas se revisan a continuación.

La primera y más importante es que fusiona el rol con las cartas coleccionables. En otros juegos, una carta era igual a muchas otras. Un jugador dispone de una carta que, de forma más o menos fácil, puede ser adquirida por otro jugador. En *Rebel War* no ocurre eso. Las cartas pueden ser iguales en su origen, pero su evolución ser diferente. Así por ejemplo, una carta de "Soldado Humano" tendrá siempre las mismas características y habilidades en su primer nivel, pero según vaya adquiriendo nuevos niveles será diferente a cualquier otra carta "Soldado Humano" que pueda tener el mismo u otro usuario. Cuando una carta alcance los puntos de experiencia necesarios para subir de nivel, se le modificarán sus atributos, aprenderá habilidades y aumentará su coste según unos algoritmos internos del juego. Este mecanismo es el de la base del rol clásico, pero aplicado a las cartas coleccionables.

En segundo lugar, y siguiendo con la línea anterior, las cartas pueden sufrir daños e incluso "morir". Las cartas representan personajes o monstruos de un mundo fantástico. Aquellas cartas que se utilizan en una partida pueden resultar heridas o destruidas para siempre.

Otra aportación clave, es el sistema de batalla. Los juegos de cartas online suelen ser por turnos y las partidas se pueden alargar demasiado, además de aumentar el riesgo de pérdida de la conexión. En *Rebel War* se ha optado por que los jugadores graben una serie de órdenes para sus tropas (cartas) y después se resuelva la batalla de forma automática. De esta forma, las batallas se planifican de forma individual por adelantado, mostrándose posteriormente el resultado final en un informe de batalla a cada uno de los jugadores. Así las partidas ganan agilidad y se evitan esperas innecesarias.

Otros aspectos de *Rebel War* siguen las convenciones del género. Por ejemplo, la idea de disponer de colecciones de cartas que se organizan en mazos para las partidas, y de cuyo mazo activo el usuario selecciona y organiza las cartas para una batalla concreta. También la existencia de un mercado donde los jugadores puedan intercambiar sus cartas.

3. Análisis

3.1. Requisitos, alcance y viabilidad

Rebel War está concebido para ser un proyecto con múltiples ampliaciones y mejoras potenciales. Sin embargo, y a fin de adaptar el desarrollo al tiempo disponible en este proyecto, aquí nos centramos en sus características principales, las que constituyen su núcleo.

3.1.1. Requisitos

La aplicación consta de 5 módulos principales, que son clave para la terminación del proyecto, y de múltiples de mejoras. Los módulos principales son:

Gestión de usuarios: El módulo de la gestión de usuarios se encarga de todas las funcionalidades relacionadas con la creación y mantenimiento de los usuarios y sus datos en la base de datos, así como del mantenimiento de las sesiones de los mismos. Esto incluye registro de nuevos usuarios, edición y gestión de sus perfiles y datos, gestión de la sesión de usuario y subidas de nivel.

Gestión de colección de cartas: Con este módulo se pretende emular la posesión física de una colección de cartas. Para ello, incluye las funcionalidades para la creación de un mazo inicial cuando el usuario se registra, visualización de las cartas del usuario (colección y mazos), creación y edición de mazos, subidas de nivel de las cartas, enfermería y cementerio.

Mercado de cartas: Para que los usuarios puedan tener acceso a más cartas, y no sólo a las que se les regalan al registrarse, el módulo de mercado incluye operaciones de compra y venta de cartas entre usuarios y de compra de sobres de cartas, todo ello utilizando dinero del juego, nunca dinero real.

Batalla básica entre dos usuarios: El módulo más importante de la aplicación es sin duda el que se encarga de las batallas entre usuarios, pues es la parte principal del juego. La batalla básica consiste en una batalla sin habilidades ni órdenes especiales de las cartas, solo intervienen los atributos de las cartas y solo se puede atacar de frente. Este módulo no solo incluye todo lo necesario para llevar a cabo una batalla básica entre dos usuarios, sino también la visualización del resultado de la batalla y de los puntos obtenidos por cada jugador.

Diseño de las cartas del juego: El aspecto visual del juego es muy importante y para que resulte atractivo a los usuarios se debe cuidar, sobre todo, el diseño de las cartas, dado que son los elementos principales del juego.

Posibles mejoras:

- Órdenes de ataque a la derecha y a la izquierda para la batalla.
- Orden de ataque a distancia para la batalla.
- Uso de habilidades básicas de las cartas en la batalla.
- Informe de batalla con información más detallada sobre el combate (órdenes de ataque, puntos de daño hechos por cada carta, cartas huidas y muertas, etc.).
- Historial de las batallas del usuario.
- Venta de sobres de cartas en el mercado.
- Historial de compras y ventas del usuario en el mercado.
- Historial completo del mercado.

Para llevar a cabo el desarrollo de los requisitos se ha seguido la metodología Scrum (Schwaber, 2001), como se explica con más detenimiento en la sección 3.7.

3.1.2. Alcance

Según la planificación realizada en la sección 3.7 se espera terminar todos los módulos principales y las mejoras propuestas en el apartado anterior.

3.1.3. Viabilidad

Dados el tiempo del que se dispone, los conocimientos de los miembros del equipo, las herramientas elegidas para el desarrollo de la aplicación, detalladas en la sección 3.3 de este capítulo, y la planificación realizada en la sección 3.7 calculamos que podemos llevar a cabo todos los módulos principales y todas las posibles mejoras descritas en el punto anterior.

3.2. Casos de uso

A partir de los requisitos expuestos en el apartado anterior se han identificado los casos de uso correspondientes a la aplicación. A continuación se van a detallar mediante diagramas y explicaciones los casos de uso englobados por cada requisito.

3.2.1. Gestión de usuarios

Dentro del módulo de la gestión de usuarios se distinguen el usuario como actor principal.

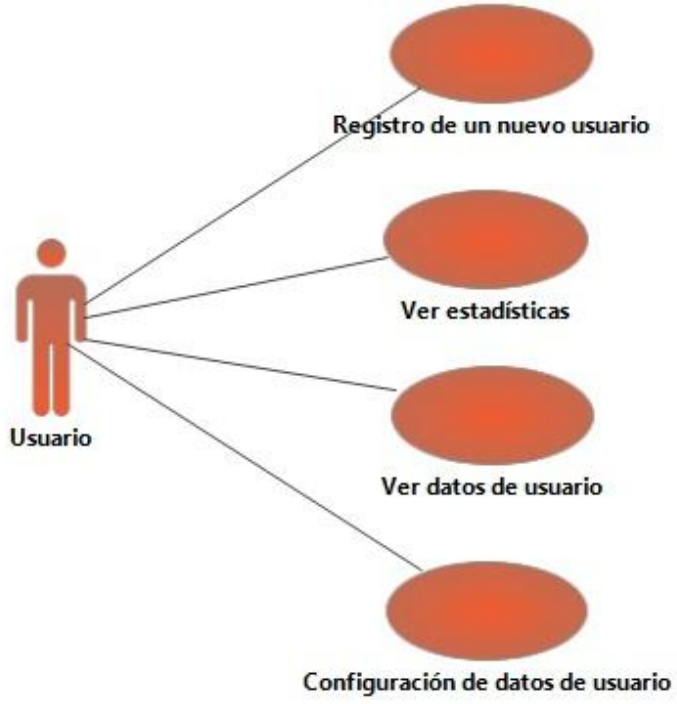


Figura 3.2.1.: Diagrama de casos de uso de usuarios

Los casos de uso pueden distinguirse según los prerrequisitos que se requieren. En el caso de que un usuario se registre, el único prerrequisito es que ese usuario no se haya registrado ya, puesto que la aplicación no permite el registro de dos usuarios con el mismo correo electrónico.

Por otro lado, una vez que el usuario ya ha accedido a la aplicación y más concretamente a su perfil de usuario, puede realizar el resto de acciones:

- **Ver estadísticas:** consultar sus estadísticas de batallas, así como el resultado de las mismas y un recuento general de batallas ganadas, perdidas o empatadas.
- **Ver datos de usuario:** obtener detalles sobre su nombre de usuario, email, nivel y monedas de oro disponibles.
- **Configuración de datos:** modificar sus datos de registro (nombre de usuario, email y/o contraseña).

3.2.2. Gestión de colección de cartas

El prerrequisito para los casos de uso de este módulo de la aplicación es que el usuario haya pulsado el botón “Colección” para poder acceder a las opciones de la misma.

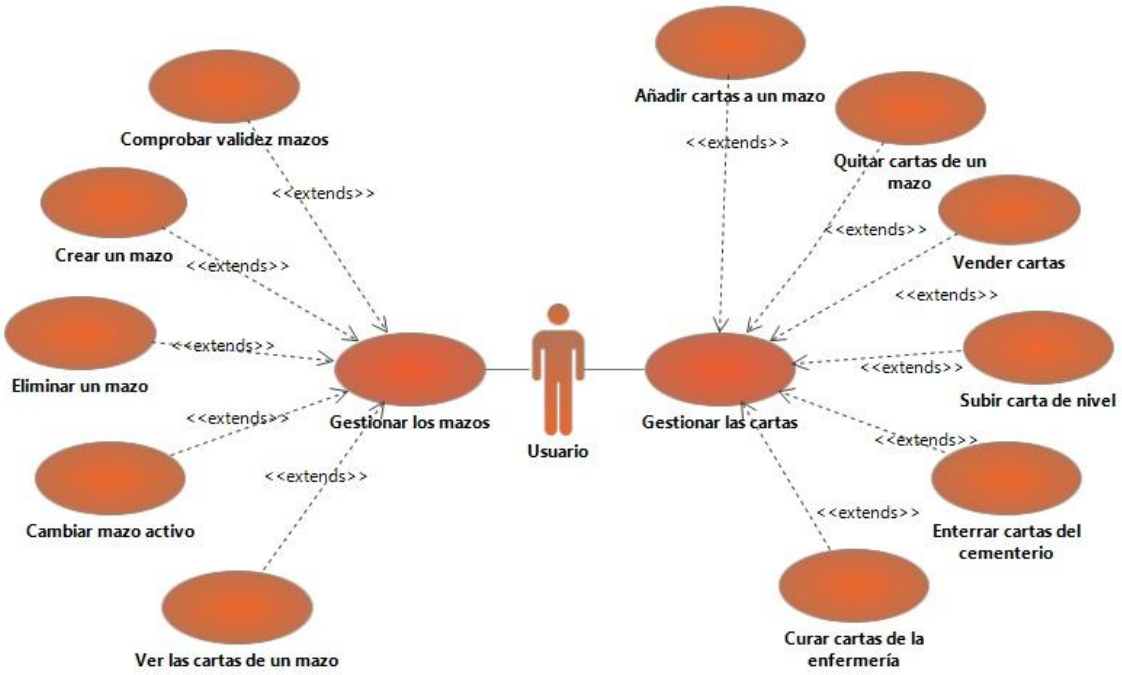


Figura 3.2.2.: Diagrama de casos de uso de colección

En este caso se pueden separar los casos de uso en dos casos generales:

- **Gestionar las cartas:** El resto de casos de uso relacionados con éste son partes opcionales para cubrir las distintas funcionalidades del usuario con las cartas. Incluyen a su vez dos grupos: gestión de cartas individuales y de las cartas de un mazo. La gestión de cartas individuales permite al usuario vender cartas de su colección, subirlas de nivel, curar cartas heridas o enterrar a las cartas muertas en combate que no puede recuperar. La gestión de las cartas de un mazo cuenta con las funcionalidades para añadir o quitar cartas de un mazo.

- **Gestionar los mazos:** Los casos de uso que amplían éste son los relacionados con las opciones que tiene un usuario respecto a sus mazos: puede crear un mazo vacío, eliminar uno existente, cambiar el mazo activo con el que combatir en las batallas, y comprobar si sus mazos son válidos, para lo que examina las cartas que componen un mazo.

3.2.3. Mercado de cartas

El prerrequisito de los casos de uso relacionados con este subsistema es que el usuario haya pulsado el botón “Mercado”.

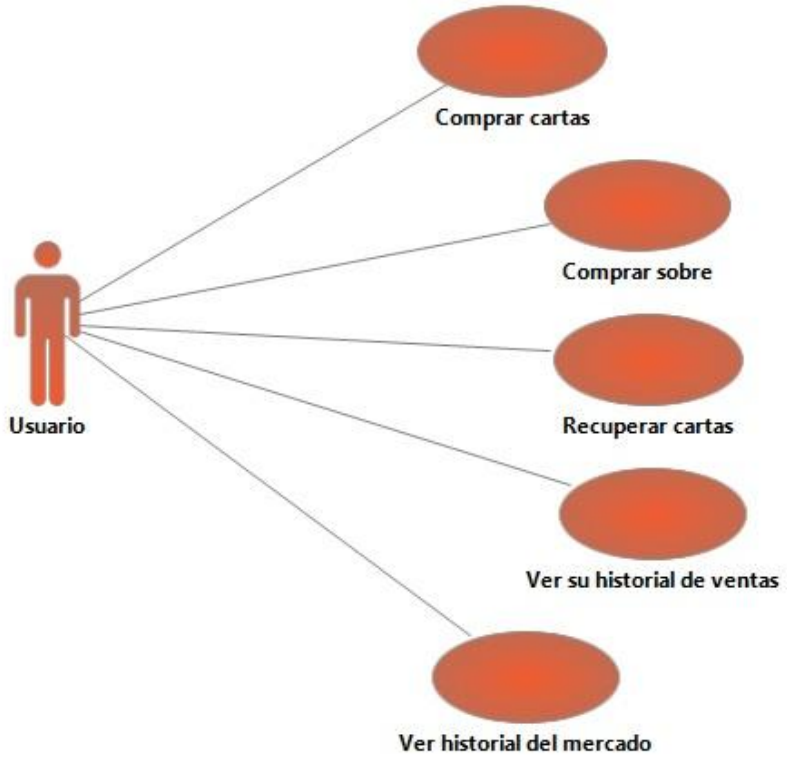


Figura 3.2.3: Diagrama de casos de uso de mercado

Los casos de uso de este apartado se corresponden con los requisitos principales del mercado. Esto incluye la compra y venta de cartas entre usuarios, así como la compra de sobres que incluyen tres cartas aleatorias de la raza elegida por el usuario.

Así mismo el usuario puede consultar el historial de ventas particular y el historial general del mercado, lo que le permite conocer las ventas de otros usuarios y estudiar los precios de venta de las cartas.

3.2.4. Batalla entre dos jugadores

En este caso el prerrequisito vuelve a estar relacionado con que el usuario pulse un botón, el de “Combates”. Una vez dentro de los combates se presentan los siguientes casos de uso:

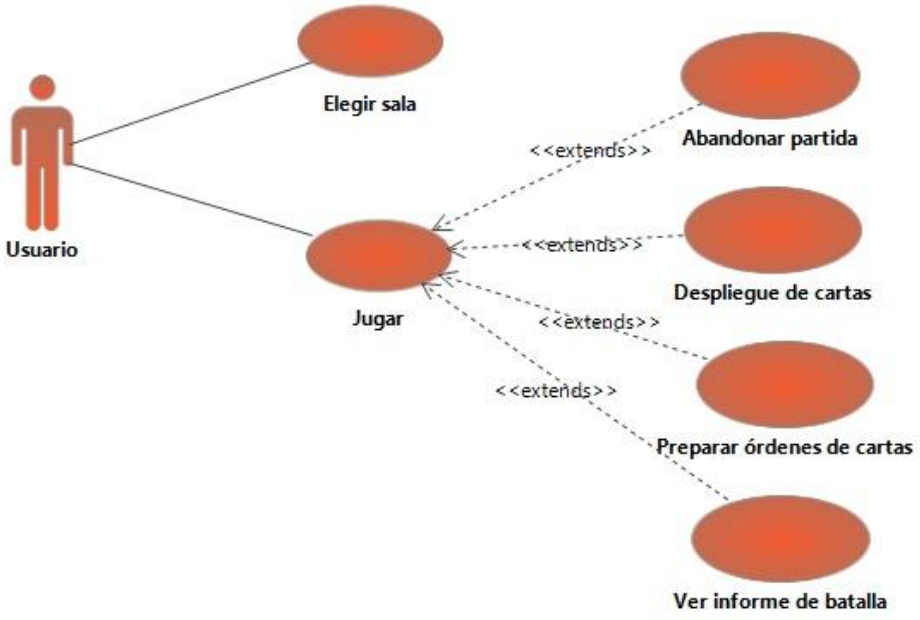


Figura 3.2.4.: Diagrama de casos de uso de batalla

El usuario puede elegir la sala donde quiere combatir (teniendo como prerrequisito que cumpla las restricciones de la misma) y jugar. Dentro del caso de uso de jugar se encuentran todas las posibilidades que tiene la batalla: o bien abandonar en cualquiera de las fases, o seguir y realizar el despliegue de cartas y la asignación de órdenes previas al combate. Una vez finalizado, el usuario puede elegir ver el informe con el resultado final detallado.

3.3. Tecnologías

Las herramientas utilizadas para desarrollar la aplicación son las siguientes.

3.3.1. Entorno de programación:

Este tipo de juegos requiere una arquitectura cliente-servidor. En este caso, se ha utilizado una basada en J2EE (Wikipedia JavaEE, 2012). Para facilitar algunos aspectos de la misma se recurre a las librerías de GWT (del inglés *Google Web Toolkit*) (Google Web Toolkit, 2012). Con estos elementos hemos elegido *Eclipse IDE for JavaEE developers* (http://www.eclipse.org/downloads/) como entorno de programación. La aplicación completa ha sido desarrollada en el entorno *Eclipse*. Se ha elegido este IDE (del inglés *Integrated Development Environment*) sobre otros disponibles por su completa integración con las librerías de GWT, que han sido claves para la implementación del proyecto, y por su sencillez y claridad a la hora de escribir código Java.

Se han utilizado además los siguientes *plugins*:

GWT Google Web Toolkit: es un framework creado por *Google* para ocultar la complejidad de varios aspectos de la tecnología AJAX. Básicamente permite crear código en Java usando cualquier IDE de Java y el compilador lo traducirá a HTML y JavaScript. Cuando la aplicación es desplegada, el compilador GWT traduce la aplicación Java a un archivo Javascript. GWT y el código que genera es compatible con varios navegadores, lo cual es notorio ya que cada navegador suele necesitar código específico para lograr un *front-end* correcto en una aplicación web.

GWT no es sólo una interfaz de programación. También proporciona un conjunto de herramientas que permiten desarrollar funcionalidades Javascript de alto rendimiento en el navegador del cliente.

Una aplicación GWT puede ser ejecutada en dos modos:

- **Modo desarrollo:** La aplicación se ejecuta como código *bytecode* de Java dentro de la Máquina Virtual de Java (JVM). Este modo es el más usado para desarrollo, soportando el cambio de código en caliente y el depurado.
- **Modo web:** La aplicación se ejecuta como código Javascript y HTML puro, compilado a partir del código Java. Este modo se suele usar para el despliegue de la aplicación.

Se han utilizado estas herramientas de *Google* en vez de otras, como por ejemplo *Apache Struts* (Apache Struts, 2012), por las facilidades que aportan a la hora de la comunicación cliente-servidor y en el desarrollo de servicios web. Además, es de código abierto y utilizado por miles de desarrolladores, por lo que su aplicación favorecerá las ampliaciones de este proyecto por parte de otros desarrolladores que no estén familiarizados con el mismo (Wikipedia Google Web Toolkit, 2012).

SVN Subversion (http://www.eclipse.org/subversive/)**:** se ha utilizado este sistema de control de versiones también por su integración con *Eclipse* y *Google code*, el repositorio utilizado para alojar el código durante la implementación de la aplicación (Wikipedia Subversión (Software), 2012).

3.3.2. Lenguaje de programación:

El lenguaje de programación base seleccionado ha sido Java.

3.3.3. Gestión de bases de datos:

Se ha elegido *MySQL* (http://dev.mysql.com/downloads/) como sistema de gestión de bases de datos por ser el más utilizado en aplicaciones web (Wikipedia MySQL, 2012). Entre las razones de esta aceptación destaca que dispone de una herramienta visual de diseño, *MySQL Workbench*. Esta herramienta simplifica enormemente el diseño de las bases de datos gracias a la posibilidad de crear esquemas entidad relación y convertirlos directamente en tablas y relaciones (MySQL, 2012).

3.3.4. Servidor web:

Como servidor web para alojar y ejecutar la aplicación se ha seleccionado *Apache Tomcat* (http://tomcat.apache.org/tomcat-7.0-doc/index.html). *Tomcat* es un servidor web con soporte de servlets y JSPs, que puede funcionar como servidor web por sí mismo. Se ha seleccionado este servidor web y no otro porque es el que mejor funciona con GWT y el más documentado en cuanto a despliegue de aplicaciones hechas con GWT (Wikipedia Tomcat, 2012).

3.3.5. Diseño gráfico:

Para el diseño gráfico de las cartas y demás imágenes del juego se ha usado *CorelDraw Graphics x5* (http://www.corel.com/). Se ha seleccionado esta herramienta de diseño en vez de otras por el conocimiento de su manejo por parte de los miembros del equipo de desarrollo.

3.4. Riesgos

3.4.1. Análisis de riesgos

En la siguiente tabla se muestran los posibles riesgos, su probabilidad y su impacto en el desarrollo del proyecto, ordenados por probabilidad:

Riesgo	Probabilidad	Impacto
Incompatibilidad de horarios entre los miembros del grupo	Alta	Tolerable
Mala estimación en el tiempo necesario para completar las características fundamentales del proyecto	Alta	Serio
Falta de tiempo para realizar características secundarias	Alta	Tolerable
Los desarrolladores no son capaces de adquirir los conocimientos requeridos	Alta	Serio
Problemas con la comunicación cliente-servidor	Alta	Serio
La aplicación es demasiado lenta como para ser usable	Moderada	Serio
Bajas permanentes del personal	Baja	Serio

3.4.2. Estrategia de gestión de riesgos

A continuación se incluyen las plantillas de control de los riesgos considerados en el punto anterior:

Incompatibilidad de horarios entre los miembros del grupo:

Probabilidad	Alta
Impacto	Tolerable
Descripción	Problemas para realizar las reuniones de planificación, toma de decisiones y puesta en común
Consecuencias	No se pueden planificar las iteraciones y el desarrollo del proyecto se ralentiza
Cómo evitarlo	Realizar un horario común para buscar las horas libres en las que coincidan todos los miembros del equipo
Cómo tratarlo	Si no se encuentran suficientes horas libres comunes utilizar el correo electrónico, para que la comunicación sea lo más fluida posible, y herramientas de videollamada, como *Skype*

Mala estimación en el tiempo necesario para completar las características fundamentales del proyecto:

Probabilidad	Alta
Impacto	Serio
Descripción	El equipo tarda más de lo estimado en cumplir los objetivos propuestos
Consecuencias	El proyecto no avanza al ritmo debido y no se termina a tiempo
Cómo evitarlo	Estudiar detenidamente cada objetivo antes de estimar la duración de su desarrollo y tener en cuenta el tiempo que puede aportar cada miembro
Cómo tratarlo	Hacer las estimaciones con cuidado y que cada miembro del equipo cumpla los tiempos de dedicación

Falta de tiempo para realizar características secundarias:

Probabilidad	Alta
Impacto	Tolerable
Descripción	Las estimaciones se hacen mal y no da tiempo a perfeccionar los aspectos principales de la aplicación y el aspecto visual de la misma
Consecuencias	La aplicación queda menos vistosa
Cómo evitarlo	Estudiar detenidamente cada objetivo y la importancia de su impresión en el usuario antes de estimar la duración de su desarrollo
Cómo tratarlo	Priorizar las características adicionales. Así se podrán abordar primero aquellas que tengan una impresión positiva importante sobre el usuario. Por supuesto, esta priorización se realizará dentro de que el esfuerzo previsto para esas características esté dentro de los márgenes disponibles en el proyecto

Los desarrolladores no son capaces de adquirir los conocimientos requeridos:

Probabilidad	Alta
Impacto	Serio
Descripción	No se consiguen entender las nuevas tecnologías que son necesarias para desarrollar la aplicación
Consecuencias	No se utilizan correctamente las herramientas que no se comprenden y la aplicación no puede desarrollarse como estaba previsto
Cómo evitarlo	Comprender bien las herramientas que se van a utilizar antes de empezar el proyecto para asegurar su correcta utilización
Cómo tratarlo	Consultar manuales, hacer tutoriales y pedir ayuda al tutor

Problemas con la comunicación cliente-servidor:

Probabilidad	Alta
Impacto	Serio
Descripción	No conseguir que la comunicación cliente-servidor, que es vital en esta aplicación, se realice rápida y correctamente
Consecuencias	La aplicación no funciona
Cómo evitarlo	Comprender bien las herramientas GWT y *Apache Tomcat* antes de empezar el proyecto para asegurar su correcta utilización
Cómo tratarlo	Consultar manuales, hacer tutoriales y pedir ayuda al tutor

La aplicación es demasiado lenta como para ser usable:

Probabilidad	Moderada
Impacto	Serio
Descripción	La aplicación final es demasiado lenta visualmente y los tiempos de respuesta del servidor y la base de datos son excesivamente largos
Consecuencias	La aplicación no cumple el objetivo de ser un juego rápido
Cómo evitarlo	Estudiar las partes de la aplicación más lentas y tener especial cuidado al desarrollarlas para agilizarlas todo lo posible
Cómo tratarlo	Hacer prototipos y pruebas de rendimiento en cada iteración. Volver a implementar las partes más lentas con algoritmos de coste menor y reducir al mínimo los accesos a la base de datos y al servidor

Bajas permanentes del personal:

Probabilidad	Baja
Impacto	Serio
Descripción	Uno o varios componentes del equipo deja de asistir a las reuniones y de participar en el desarrollo del proyecto
Consecuencias	El proyecto no se termina por falta de mano de obra
Cómo evitarlo	Motivar al equipo para que no abandone
Cómo tratarlo	Priorizar las partes del desarrollo para negociar con el cliente posibles partes a eliminar según el esfuerzo perdido por la baja

3.5. Gestión de la configuración

Todos los archivos, tanto de documentación como de código, son objeto de control de la gestión de configuración. Para controlar los cambios se utilizará la herramienta de control de versiones *Subversión* para el código de la aplicación y *Google Docs* (http://code.google.com/) para los archivos de documentación.

Para nombrar los directorios, documentos, paquetes y clases se elegirán nombres claramente identificativos y descriptivos de la información que contienen.

Para realizar cambios sobre un archivo de documentación que tengan que modificar varias personas antes de alcanzar su aspecto final se deberá establecer un turno de modificación para no provocar pérdidas e inconsistencias de la información. De esta manera será necesario avisar a las personas correspondientes antes de modificar el archivo.

3.5.1. Estándares de código

Cada clase, variable y método que se use para el desarrollo del código debe tener un nombre claramente explicativo y representativo. El nombre de las clases deberá comenzar siempre con mayúscula, y el de las variables, atributos de clase y métodos con minúscula. Se deberá incluir antes de cada uno de estos cuatro elementos nombrados una breve explicación del propósito y función que desarrollan dentro del programa, cuando no sean éstas deducibles de su nombre. Esta documentación seguirá los criterios de *Javadoc* (Oracle How to write Doc Comments for the Javadoc Tool, 2012).

3.5.2. Estándar de documentación

El nombre que identificará a los documentos debe ser a la vez claramente representativo y descriptivo de la información que contiene. Cada nueva sección aparecerá en un nuevo comienzo de hoja.

En cuanto a la presentación del texto en el documento será la siguiente:

- **Fuente para todo el documento y las cabeceras:** Calibri
- **Formato texto normal:** Calibri 12 ptos., en color negro, interlineado sencillo
- **Formato títulos:** Calibri 48 ptos., en negrita, en color negro, centrado, interlineado sencillo.
- **Formato títulos primarios:** Calibri 24 ptos., en negrita, en color negro, interlineado sencillo.
- **Formato títulos secundarios:** Calibri 18 ptos., en negrita, en color negro, interlineado sencillo.
- **Formato títulos terciarios:** Calibri 14 ptos., en negrita, en color negro, interlineado sencillo.
- **Numeración de página:** esquina inferior derecha.
- **Listas de enumeraciones:** se usará la herramienta del editor de texto correspondiente (*Microsoft Word*).

3.6. Requisitos del sistema

3.6.1. Usuario:

Esta aplicación no requiere una configuración ni requisitos especiales para el sistema del usuario, ya que es una aplicación web. El usuario que quiera jugar a *Rebel War* simplemente necesitará tener instalado el navegador *Google Chrome*[1].

3.6.2. Desarrollador:

El desarrollador que quiera ampliar la aplicación necesitará las siguientes herramientas instaladas y configuradas:

Eclipse IDE for JavaEE developers con el plugin de GWT: *Eclipse* puede descargarse de la página http://www.eclipse.org/downloads/. No es necesario instalarlo, ya que basta con descomprimir el paquete que se descarga. Para instalar el *plugin* de GWT, en Eclipse en *Help->Install New Software...->Add...* se introduce la siguiente URL (del inglés *Uniform Resource Locator*) http://dl.google.com/eclipse/plugin/3.7 y se le pone un nombre identificativo (por ejemplo GWT para saber que es la dirección de descarga del *plugin* de GWT). Se seleccionan todas las descargas (excepto la de Google *App Engine*

[1] Es posible jugar con otros navegadores, pero el juego está optimizado y probado para las características de este navegador.

Tools for Android, a no ser que se desee adaptar la aplicación o ampliarla para *Android*), se aceptan las licencias oportunas y se siguen los pasos hasta el final.

MySQL Community Server: El *MySQL Community Server* puede descargarse en http://dev.mysql.com/downloads/mysql/. Se necesita para la gestión y comunicación de la base de datos. Es útil descargar también *MySQL Workbench* para facilitar el acceso a la base de datos (http://dev.mysql.com/downloads/workbench/5.2.html), pero no es necesario. Es conveniente instalar *MySQL Community Server* en los ordenadores de desarrollo y en el servidor que alojará la aplicación para poder gestionar bases de datos locales, para los ordenadores de desarrollo, y una base de datos de producción, en el servidor.

Servidor con Apache Tomcat: Para desplegar y alojar la aplicación hace falta un servidor con *Apache Tomcat*. Se puede encontrar la última versión de *Apache Tomcat* en http://tomcat.apache.org/download-70.cgi. No es necesaria una configuración especial para la ejecución de la aplicación.

Sistema de control de versiones: Se recomienda usar un sistema de control de versiones para la mejor organización del desarrollo del proyecto si en el intervienen varias personas. En este caso se ha usado *Subversion* con su *plugin* para *Eclipse* (http://www.eclipse.org/subversive/downloads.php#indigo_stable) por su simplicidad y completa integración con el entorno de desarrollo, pero hay muchas otras opciones que también pueden ser válidas. Por ejemplo: *GIT* (http://www.eclipse.org/egit/) y *Mercurial* (http://javaforge.com/project/HGE) que también disponen de un *plugin* para *Eclipse*.

3.7. Planificación del trabajo

Para conseguir abordar los objetivos propuestos, se ha dividido el trabajo en las siguientes partes:

- **Preparación del software y pruebas del mismo:** instalación del software para alojar la aplicación en el servidor, instalación del software que se va a utilizar para implementar la aplicación en los ordenadores de desarrollo y pruebas de dicho software para cerciorarse de que se ajusta a las necesidades del proyecto.
- **Diseño de los diagramas y esquemas de requisitos básicos:** mapa de navegación, esquema E/R de la base de datos, diagrama de actividad y diagrama de la jerarquía de paquetes.
- **Implementación de la aplicación siguiendo lo diseñado en el punto 2.**

Se ha seguido una metodología *Scrum* (Schwaber, 2001) para llevar a cabo las tareas anteriores. *Scrum* es una metodología de desarrollo ágil de software basada en un proceso iterativo e incremental. En cada iteración o *sprint* se eligen los objetivos a

cumplir en dicho período de entre los objetivos globales del proyecto, teniendo en cuenta la prioridad y la dificultad de cada uno de ellos.

En este proyecto se ha definido el *sprint* como un período de dos semanas (14 días) y los objetivos principales se repartieron de la siguiente manera:

- **Sprint 1 (del 7 al 20 de Octubre de 2011):** Preparación del software en el servidor y en los ordenadores de desarrollo. Desarrollo de la aplicación tutorial de GWT (Google Developers Google Web toolkit Tutorial Overview, 2012) y despliegue de la misma en el servidor para comprobar su correcto funcionamiento.
- **Sprint 2 (del 21 de Octubre al 3 de Noviembre de 2011):** Diseño de los diagramas y esquemas de requisitos básicos: mapa de navegación, esquema entidad/relación de la base de datos, diagrama de actividad y diagrama de la jerarquía de paquetes.
- **Sprint 3 (del 4 al 17 de Noviembre de 2011):** Implementación de un prototipo de la interfaz básica de la aplicación completa, sin aplicar estilos, y del esquema de la base de datos usando la herramienta online *DaoGen* (http://titaniclinux.net/daogen/). *DaoGen* genera un objeto y su correspondiente DAO (del inglés *Data Access Object*) para cada tabla de la base de datos automáticamente.
- **Sprint 4 (del 18 de Noviembre al 1 de Diciembre de 2011):** Implementación de los servicios de login, logout, registro de nuevos usuarios y de los algoritmos de lucha entre dos cartas. Aplicación de estilos a las interfaces de home público, home de usuario y registro.
- **Sprint 5 (del 2 al 15 de Diciembre de 2011):** Implementación de los servicios de la colección y de los algoritmos de la batalla completa sin habilidades ni órdenes especiales. Aplicación de estilos a las interfaces de sobre inicial y colección.
- **Sprint 6 (del 16 al 29 de Diciembre de 2011):** Implementación de los servicios de creación y edición de mazos y de selección de jugadores para la batalla. Aplicación de estilos a las interfaces de creación y edición de mazos y de salas del combate.
- **Sprint 7 (del 30 de Diciembre de 2011 al 12 de Enero de 2012):** Implementación de los servicios de enfermería, cementerio, selección de cartas para el combate y despliegue. Aplicación de estilos a las interfaces de cementerio, enfermería, presentación del combate y despliegue.
- **Sprint 8 (del 13 al 26 de Enero de 2012):** Implementación de los servicios de mercado de cartas, entrega de órdenes y lanzamiento de la batalla. Aplicación de estilos a las interfaces de mercado, órdenes y pop-ups de lanzamiento de la batalla.
- **Sprint 9 (del 27 de Enero al 9 de Febrero de 2012):** Implementación de los servicios de perfil de usuario y de los algoritmos y servicios de subida de nivel para usuarios y cartas. Aplicación de estilos a las interfaces de perfil de usuario y pop-ups de subidas de nivel.
- **Sprint 10 (del 10 al 23 de Febrero de 2012):** Implementación de los servicios de alertas. Ampliación de los algoritmos de la batalla para contemplar las

habilidades básicas de las cartas. Aplicación de estilos a los pop-ups de alertas. Diseño de las cartas de Humanos.

- **Sprint 11 (del 24 de Febrero al 8 de Marzo de 2012):** Implementación de los servicios y algoritmos de cálculo de resultado e informe de la batalla y de los servicios de guardado de históricos de batallas del usuario. Aplicación de estilos a las interfaces de resultado e informe de la batalla.
- **Sprint 12 (del 9 al 22 de Marzo de 2012):** Implementación de los servicios de cálculo de estadísticas y visualización de históricos del usuario. Ampliación de los algoritmos de la batalla para incluir las órdenes de ataque a la derecha y a la izquierda. Aplicación de estilos a las interfaces de estadísticas e históricos.
- **Sprint 13 (del 23 de Marzo al 5 de Abril de 2012):** Implementación de los servicios de abandono de la batalla y ampliación de los algoritmos de la batalla para incluir la orden de ataque a distancia. Aplicación de estilos a los pop-ups de abandono. Diseño de las cartas de Elfos y Orcos.
- **Sprint 14 (del 6 al 19 de Abril de 2012):** Implementación de los servicios de ayuda y creación del manual de usuario. Aplicación de estilos a las interfaces de ayuda. Diseño de las cartas de Enanos.
- **Sprint 15 (del 20 de Abril al 3 de Mayo de 2012):** Implementación de los paneles de zoom de las cartas. Revisión y corrección de detalles en toda la aplicación.
- **Sprint 16 (del 4 al 17 de Mayo de 2012):** Desarrollo de la memoria del proyecto. Entrega de la primera versión completa.
- **Sprint 17 (del 18 al 31 de Mayo de 2012):** Desarrollo de la memoria del proyecto. Entrega de la primera versión final.

4. Arquitectura

4.1. Diseño de la aplicación

La aplicación está diseñada de forma que se cumpla con los requisitos y los casos de uso especificados anteriormente. De una forma abstracta podemos destacar los siguientes subsistemas que la componen:

- **Parte gráfica:** Incluye tanto la interfaz gráfica de usuario como las diferentes imágenes de las cartas, botones, etc.
- **Batalla:** Se compone de todo lo relacionado con los combates entre usuarios, desde las clases que implementan los algoritmos de batalla hasta las clases auxiliares de resultados.
- **Coleccionismo:** Se pueden juntar en este subsistema tanto lo relativo a la colección de cartas como la parte de mercado, ya que ambos están orientados a simular la pertenencia real de las cartas del juego.
- **Base de datos:** Engloba todas las clases que comunican la base de datos con la aplicación, así como al propio diseño e implementación de las tablas que la componen y las relaciones entre ellas.

Puesto que la base de datos es el núcleo para la gestión de los datos del juego en el resto de la aplicación, se comenzará por discutir su diseño. Las siguientes secciones abordarán el resto de los módulos que emplean la base de datos.

4.1.1. Diseño de la base de datos

En la base de datos del juego se almacenan todas las características que intervienen en el mismo y se actualiza constantemente la información para que sea consistente. Por un lado, se tienen las tablas con datos fijos que actúan como constantes. Por otro lado están las tablas que almacenan información dinámica sobre el desarrollo del juego. Ambos tipos intervienen durante la ejecución de la aplicación, ya que las del primer tipo proporcionan datos que las del segundo tipo necesitan para insertar nuevos registros si es necesario.

Un ejemplo de la distinción anterior entre tablas se encuentra en la tabla de ***CartasPrimerNivel***, que es del primer grupo. En esta tabla están almacenados todos los tipos de cartas que hay, con sus características en el primer nivel. Estos datos no se modifican mientras que un usuario está jugando. Cuando se registra un usuario nuevo, se seleccionan cartas de esta tabla dependiendo de la raza que ha elegido y se copian en la tabla ***Cartas*** insertando nuevas filas con los datos correspondientes. Estos registros describen las cartas que serán propiedad de un usuario y podrán ser modificadas a medida que éste juegue.

A fin de facilitar un adecuado mantenimiento de los datos y reducir las posibilidades de inconsistencias, la base de datos está diseñada con claves foráneas (*foreign keys*), que impiden la duplicidad de atributos que deben ser únicos y relacionan unas tablas con

otras de manera consistente. También se ha diseñado teniendo en cuenta la necesidad de evitar en ciertas tablas que varios usuarios accedan simultáneamente para su modificación. Asimismo, la aplicación se ocupa de que si una operación falla no queden datos inconsistentes en las tablas, ya que hasta que no se completan todos los accesos necesarios de un método de la aplicación no se actualiza la base de datos de forma permanente.

A continuación se muestra el diagrama entidad-relación simplificado de la base de datos, que ilustra las diversas relaciones entre las tablas que posibilitan el buen funcionamiento de la aplicación.

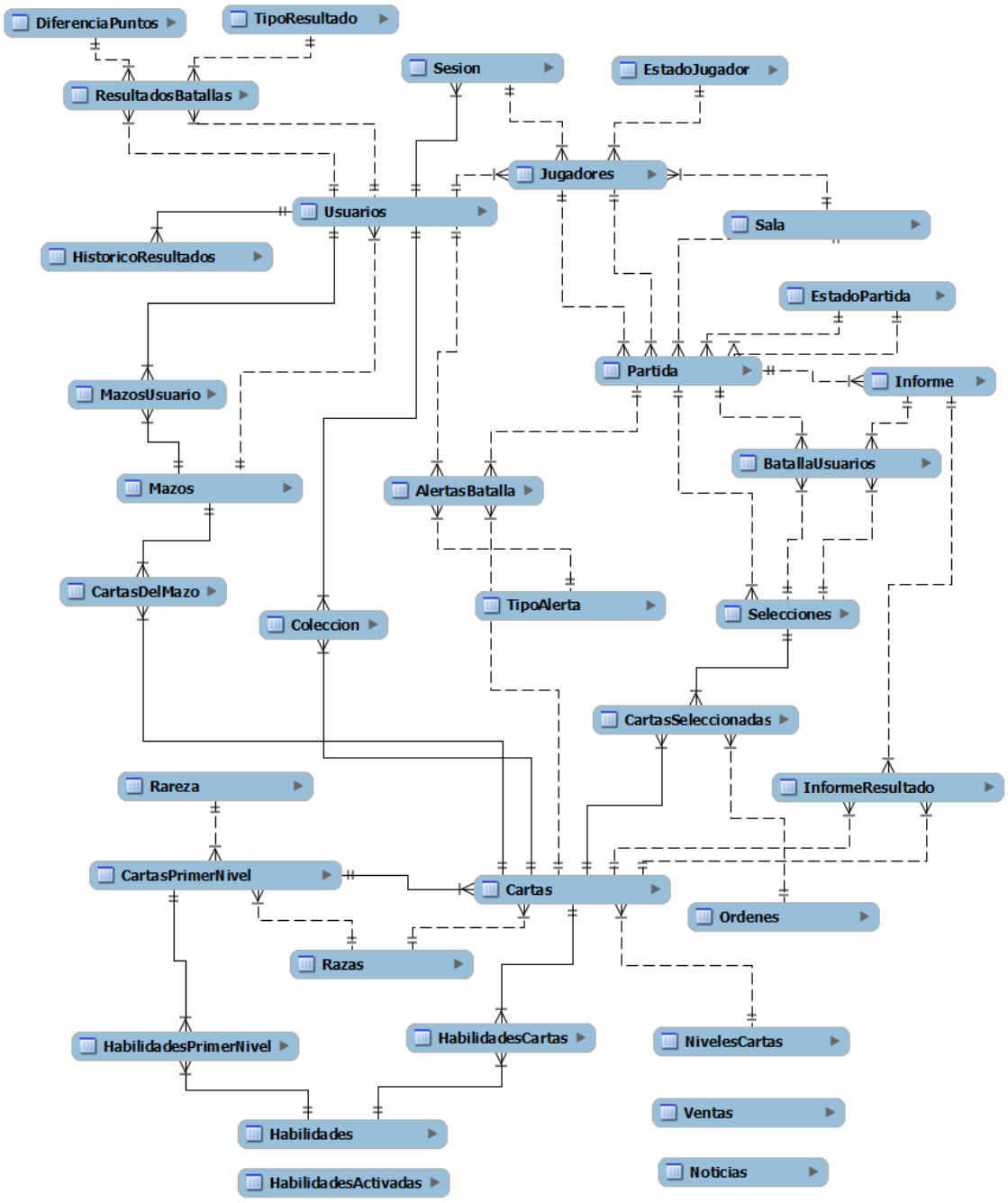


Figura 4.2.1.: Diagrama Entidad-Relación de la base de datos

Las principales tablas son las siguientes:

- **Usuarios:** es la tabla donde se almacenan los usuarios registrados y su información relevante. Sus atributos son: identificador de usuario, nombre, email, contraseña, experiencia acumulada, nivel, oro y el identificador del mazo que tiene activo en ese momento.
- **Cartas primer nivel:** contiene todas las cartas de nivel uno, que sirven como base para las cartas propias de cada usuario. Entre otros atributos, poseen nombre de la carta, raza, rareza y los parámetros de ataque, defensa y velocidad.
- **Cartas:** se almacenan las cartas propias de cada usuario, y son las que se van modificando. Por ello incluyen atributos añadidos, como el nombre propio, el número de batallas perdidas y parámetros que indican si la carta está en venta, herida o muerta.
- **Habilidades y habilidades cartas:** la primera contiene una lista con todas las habilidades que pueden tener las cartas, y la segunda relaciona cada carta con las que tiene en ese momento.
- **Colección:** relaciona al usuario con cada carta mediante sus identificadores.
- **Mazos, mazos de usuario y mazo de cartas:** permite relacionar los mazos que crean los usuarios con las cartas contenidas en cada uno de ellos.
- **Sesión:** guarda los datos de la sesión del usuario, incluida la fecha.
- **Partida:** en esta tabla se añade una fila por cada partida entre dos usuarios. Incluye los identificadores de los jugadores, sus estados y la sala en la que se encuentran.
- **Batalla usuarios:** incluye información acerca de la partida, las cartas seleccionadas por cada contrincante y el informe de la batalla.
- **Informe resultado:** almacena información detallada sobre cada lucha individual de dos cartas, incluyendo quién es el atacante y quién el defensor, los daños realizados, los puntos de cada carta, etc.
- **Resultados batallas:** su función es almacenar las estadísticas de cada usuario, almacenando su identificador e información de cada batalla realizada (contrincante, el tipo de resultado, la experiencia y el oro adquiridos, etc.).
- **Ventas:** guarda información sobre las ventas realizadas para poder mostrarlas en el historial del mercado: vendedor, comprador, precio, y fecha de venta.

4.2. Estructura

Para desarrollar una aplicación que sea un proyecto GWT hay que cumplir la estructura de paquetes y clases que viene definida básicamente por 3 partes diferenciadas:

- Un paquete **client**, que implemente toda la funcionalidad asociada al lado cliente de la aplicación.
- Un paquete **server**, que haga lo propio para el lado del servidor de la aplicación.
- La carpeta **war**, que es donde se despliega la aplicación una vez compilada para subirla al servidor y que los usuarios puedan acceder a ella.

Pero esos elementos son sólo la base de cualquier aplicación desarrollada en GWT, así que a medida que se van añadiendo nuevas funcionalidades se han de incorporar clases que permitan desarrollar todos los casos de uso. Para mantener una estructura organizada, se han incluido nuevos paquetes donde diferenciar las clases relacionadas con la parte visual, la funcionalidad referente a la batalla, la comunicación con la base de datos, etc. A continuación vamos a comentar la estructura final de paquetes resultante, así como los detalles de las clases más significativas.

4.2.1. Paquete com.ssii.rebelwar.client

Es el paquete propio del lado del cliente e incluye únicamente la clase ***RebelWar***, que implementa el punto de entrada de la aplicación (método `onModuleLoad`). Desde aquí se gestionan los diferentes paneles que se muestran en las distintas pantallas, los eventos de los botones, las llamadas a los servicios... En definitiva, es la clase que actúa como controlador de la aplicación. Aunque gran parte del resto de clases también son de la parte del cliente de la aplicación, se han estructurado en otros paquetes para mejorar la organización.

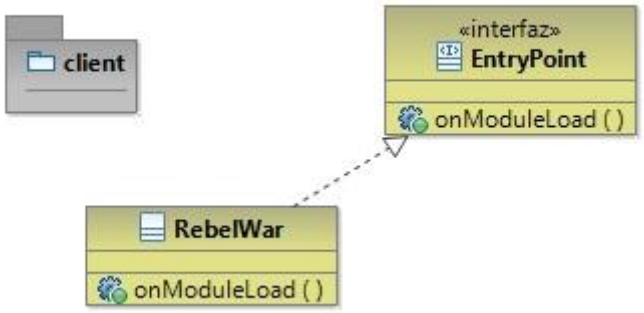


Figura 4.2.2.: Diagrama de clase RebelWar

4.2.2. Paquete com.ssii.rebelwar.client.batalla

En este paquete se incluyen todas las clases relacionadas con los combates entre usuarios, y las luchas individuales de las cartas.

Clase *Batalla*

Es la clase que implementa la batalla entre los mazos de los dos jugadores. Para ello recibe el tablero con las cartas posicionadas, las órdenes de cada una, y los datos de los 2 jugadores. Los algoritmos específicos que se usan para resolver las batallas se explican detalladamente en el Apendice A.

Clase *SeleccionMazoBatalla*

Selecciona las cartas, de entre las que componen el mazo activo, dependiendo de los puntos de la sala en la que está el jugador y del valor de sus cartas. El algoritmo de selección de cartas para la batalla se explica detalladamente en el Apéndice A.

Clase *Tablero*

Implementa el campo de batalla como una matriz de 20 posiciones. Para ello utiliza 4 matrices como atributos: tablero, que es la matriz de cartas; órdenes, que es una matriz donde se guardan las órdenes de cada carta; y muertos y huidos que son matrices que sirven para reflejar las cartas que huyen o mueren en la lucha.

Clase *HabilidadesLucha*

En esta clase se calculan los atributos para la lucha de cada carta dependiendo de sus habilidades y las de la carta adversaria. Tras este cálculo, invoca a la propia lucha.

Clase *Lucha*

Implementa el algoritmo básico de lucha entre dos cartas, o bien cuerpo a cuerpo o bien a distancia. También va generando el informe del resultado de cada lucha.

Clase *Heridas*

Calcula las huidas, las heridas y las bajadas de atributos de cada carta durante su lucha individual contra otra carta.

Clase *SubidaNivel*

Mediante unos algoritmos que emplean entre otros datos la raza y el tipo de carta, se calculan las diferentes mejoras que obtiene una carta al subir de nivel. Estas mejoras cambian entre una carta y otra.

4.2.3. Paquete com.ssii.rebelwar.client.batalla.habilidades

Las cartas de los jugadores tienen "habilidades especiales". Estas habilidades pueden venir dadas por el tipo de carta, y son asignadas por tanto en el momento de creación de la misma desde la carta prototipo, o haber sido adquiridas al subir la carta de nivel. Estas habilidades son muy diferentes unas a otras, por lo que se implementan cada una por separado, influyendo en los ataques y defensas de las cartas. En futuras ampliaciones se irán añadiendo nuevas habilidades.

4.2.4. Paquetes com.ssii.rebelwar.client.dao y daoBeans

Para realizar un correcto manejo de la base de datos se han utilizado los DAOs (*Data Access Object*), que son objetos intermedios entre la aplicación y los accesos a la base de datos. Su uso permite poder manejar las tablas en la aplicación como si fueran clases Java, donde utilizar los valores de los parámetros y modificarlos. Se utilizan las clases DAO tanto para crear los objetos como para guardar sus cambios en la base de datos, de forma que la comunicación es más eficiente y segura que si se accediera directamente de los servicios a la base de datos.

Por cada tabla existente se genera una clase denominada ***nombreDeLaTablaDao*** y otra llamada ***nombreDeLaTabla***. La primera es la que contiene todas las sentencias de

acceso, diferenciadas por métodos. Por ejemplo, el método `load` llama a la sentencia *select* de *MySQL*; el método `create` prepara una sentencia *insert* para agregar una nueva fila a la tabla; el método `save` guarda los cambios mediante la cláusula *update*; etc. Estos métodos utilizan como parámetro una conexión a la base de datos, de la que hablaremos más adelante, y un objeto de la segunda clase. Estos otros objetos están implementando un patrón transferencia, que es el que se usa en combinación con los DAOs, y son clases de Java que tienen como atributos todos los de la tabla, junto a constructoras, accesoras, etc.

4.2.5. Paquete com.ssii.rebelwar.client.beans.enumerados

Se incluyen aquellas clases que definen tipos enumerados, sobre todo las que se corresponden con información de la base de datos. Son clases auxiliares para poder comparar y manejar resultados. Algunos ejemplos de clases incluidas en este paquete son:

Clase *ComprobacionesMazos*

Esta clase se utiliza como el tipo resultado de la comprobación que se hace de si un mazo es o no válido. También se usa a la hora de mostrar al usuario ese resultado.

Clase *TipoAlerta*

Define un tipo enumerado que indica las diferentes alertas que aparecen al final de una batalla: una carta ha muerto, una carta está herida, una carta herida se ha curado o el jugador sube de nivel.

Clase *EstadoPartida*

Enumera los diferentes estados por los que pasa la partida (inicio, selección, despliegue, órdenes, abandono y fin), explicados con detalle en un diagrama de estados posterior.

Clase *NombreHabilidades*

Incluye un enumerado con los nombres de todas las habilidades de la base de datos para facilitar la implementación de la batalla. Al igual que esta clase se han incluido otras como Razas, Rarezas, etc., que reproducen los datos de algunas de las tablas de la base de datos que no cambian.

4.2.6. Paquete com.ssii.rebelwar.client.exceptions

Para la depuración de las excepciones durante el desarrollo se han utilizado dos clases: la clase NotFoundException, necesaria para los posibles errores de los DAOs al intentar encontrar objetos que no existen en la base de datos; y otra clase más general, RebelWarException, utilizada en diferentes clases de la aplicación.

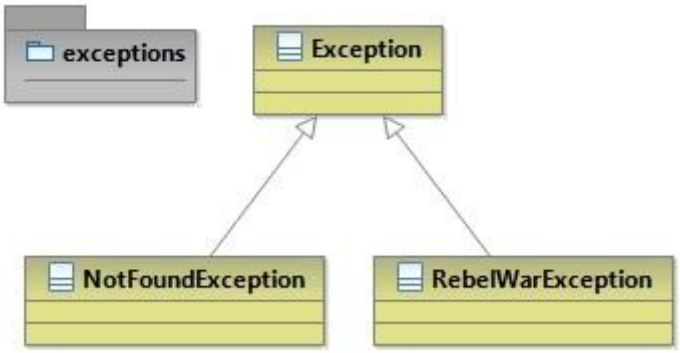


Figura 4.2.3.: Diagrama de clase del paquete exceptions

4.2.7. Paquete com.ssii.rebelwar.client.servicios

La implementación del lado del servidor en GWT implica que por cada servicio se crean dos interfaces, ***nombreServicio*** y ***nombreServicioAsync*** (incluidas en el lado del cliente), y una clase, ***nombreServicioImpl*** (en el lado del servidor). La primera es una interfaz que extiende de la clase ***RemoteService*** y define el servicio incluyendo el código `@RemoteServiceRelativePath("nombreServicio")` en la cabecera de la clase, además de definir los métodos relacionados con dicho servicio.

La segunda es la interfaz que se usa para hacer llamadas asíncronas a los servicios desde el lado cliente. En este caso, todos los métodos son de tipo `void`, y tienen la misma signatura que los de la otra interfaz añadiendo un parámetro de tipo `AsyncCallback<Tipo> callback`, donde Tipo es el tipo que devuelve el método original.

La tercera clase se detallará más adelante, en la descripción del paquete server, junto a los diferentes servicios y su funcionalidad.

4.2.8. Paquete com.ssii.rebelwar.client.utils

Dentro de este paquete se encuentran un gran número de clases auxiliares cuya función es facilitar la implementación de la funcionalidad de la aplicación. Por ejemplo, para lo referente a la batalla se encuentran la clase ***Aleatorios*** (usada para los algoritmos de batalla); la clase ***VelocidadAtaqueComparador*** (que compara la velocidad de las cartas en cada lucha); las clases ***Casilla***, ***Posición*** y ***CartaFilaCol*** (para saber la colocación de las cartas en el tablero); la clase ***ComprobacionMazo*** (que comprueba si un mazo es correcto para la batalla); etc.

Asimismo, se encuentran varias clases relacionadas con el registro de un nuevo usuario. Entre ellas destacan las siguientes: ***ColeccionInicial*** selecciona las cartas iniciales de un usuario siguiendo las restricciones; ***NomAleatorios*** asigna nombres aleatoriamente a las nuevas cartas según su raza; ***UrlImagen*** asigna a cada carta su imagen correspondiente según su tipo.

4.2.9. Paquete com.ssii.rebelwar.client.view

En este paquete se concentran las clases relacionadas con la parte gráfica de la aplicación. La interfaz está implementada en el `RootPanel` de la clase ***RebelWar***, al que se añade un fondo que incluirá los diferentes paneles correspondientes a cada sección del juego.

Cuando se produce una acción en el juego que requiere cambiar la interfaz, en la clase ***RebelWar*** se crea un nuevo objeto de una de las clases de este paquete, se actualizan sus botones y se llama al método `load` con el panel de fondo como parámetro. Este método crea y coloca los diferentes elementos del panel correspondiente y devuelve el panel completo que se mostrará al usuario en el navegador.

Cada panel necesita unas acciones distintas previas a su carga. Por ejemplo, al crear un objeto de la clase ***MercadoPanel***, hay que actualizar las cartas a la venta, el historial de ventas del usuario y el historial general del mercado. Una vez que la clase ***RebelWar*** pasa estos datos al ***MercadoPanel***, ya en el método `load` se procesan para mostrarle al usuario las tablas correspondientes a la sección del mercado. En cambio, otras clases son más sencillas, como ***PopUpSubidaDeNivelUsuario***, que sólo necesita conocer el nuevo nivel del jugador.

4.2.10. Paquete com.ssii.rebelwar.client.view.ayuda

Al igual que en el caso anterior, este paquete está orientado a la parte gráfica de la aplicación, en este caso a la ayuda del juego. La principal diferencia con las clases anteriores es que el fondo de la ayuda es un *DialogBox*, para así conseguir que sea independiente de la propia aplicación y que se pueda mantener la ayuda abierta mientras se realizan otras funciones del juego. A este elemento se añaden los distintos paneles representados por cada clase de este paquete mediante los métodos `load`. Ello permite ir mostrando las diferentes secciones de la ayuda según lo requiera el usuario, sin modificar el contenido del panel principal de la aplicación.

4.2.11. Paquete com.ssii.rebelwar.server

Este es el paquete propio del servidor, donde se encuentran las clases que implementan todos los servicios de la aplicación. Debido a la estructura marcada por GWT, estas clases tienen que nombrarse como ***nombreServicioImpl.java***, y extender a la clase ***RemoteServiceServlet***. Esto les permite heredar toda la funcionalidad necesaria para posibilitar el funcionamiento de la parte del servidor y los accesos a la base de datos. Además cada clase implementa su interfaz correspondiente del paquete *com.ssii.rebelwar.client.servicios*.

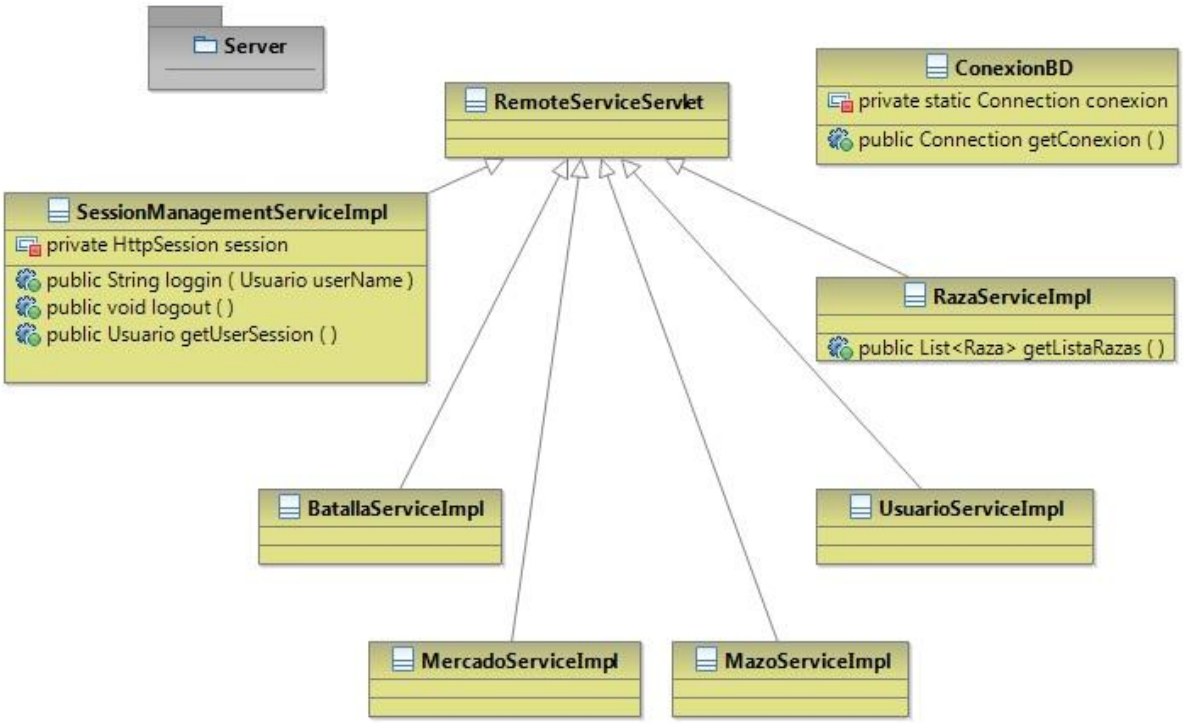


Figura 4.2.4.: Diagrama de clases del paquete server

A continuación se van a detallar las diferentes clases de este paquete y la función que tiene cada servicio dentro de la aplicación.

Clase *ConexionBD*

Es la clase que realiza la conexión con la base de datos, utilizando la librería *mysql-connector-java-5.1.18-bin.jar*. Ésta contiene un driver o controlador que permite, al registrarlo en la clase, iniciar la comunicación entre la aplicación y la base de datos, y poder tanto consultarla como modificarla. En este caso se ha aplicado un patrón *singleton* para que únicamente haya una instancia del objeto `conexion` de tipo ***Connection***, que compartirán el resto de clases. Así se consigue que la conexión se cree una sola vez y no cada vez que hay que acceder a la base de datos.

Clase *SessionManagementServiceImpl*

Esta clase es la encargada del manejo de sesiones de la aplicación. Cada vez que un usuario entra en la aplicación, en el método de login se registra la sesión actual en la variable `session`, en la que se almacena como atributo el usuario. Así se permite que si un usuario no se desconecta, durante 2 semanas (duración especificada de las *cookies*) permanezca su sesión iniciada. A su vez, esta sesión junto con la fecha de la misma se almacena en la base de datos.

Clase *RazaServiceImpl*

Este servicio únicamente obtiene las razas de la base de datos, lo que permite que si en ampliaciones futuras se introducen más razas no haya que modificar el código.

Clase *UsuarioServiceImpl*

- Todas las funcionalidades relacionadas con los usuarios se implementan en esta clase, usando el servicio usuarioService declarado en la interfaz ***UsuarioService***. A continuación se detallan sus métodos más importantes:
- **getUsuarioDatabase:** comprueba que el usuario, con el email y la contraseña introducidos en el formulario de entrada de la aplicación, existe en la base de datos, y se trae el usuario completo para usarlo en el resto del programa.
- **getRegistroUsuario:** crea un nuevo usuario en la base de datos con los campos introducidos en el formulario de registro, además de asignarle su colección inicial de cartas y el mazo activo correspondiente.
- **getEstadisticas:** calcula todas las estadísticas de batalla del usuario con idUsuario para luego mostrarlas en el perfil.
- **actualizaUsuario:** modifica el usuario con nuevos valores de experiencia, oro, nivel, etc.

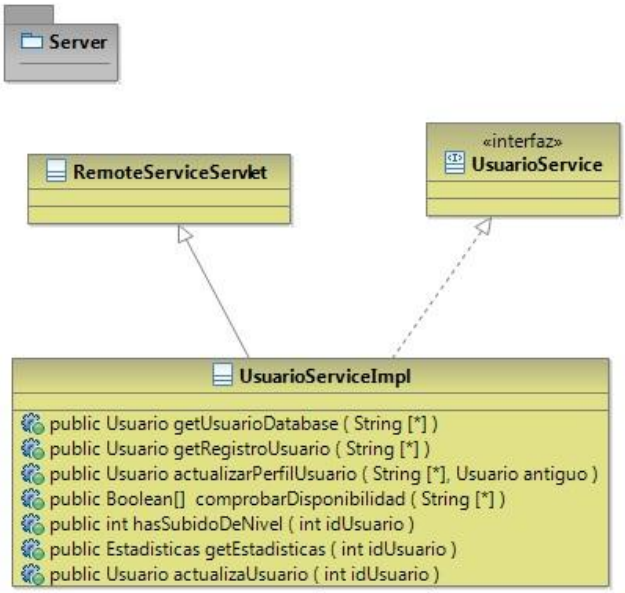


Figura 4.2.5.: Diagrama de clases de UsuarioServiceImpl

Clase *MazoServiceImpl*

En esta clase se implementa la parte relacionada a la colección y todas las acciones que puede realizar el usuario sobre ella, usando el servicio **mazoService**. Los métodos más destacados son:

- **getMazoActivo:** devuelve un `ArrayList` con las cartas del mazo activo del usuario, que es con el que lucha en las batallas.

- **getColeccionUsuario:** devuelve todas las cartas que tiene el usuario en su colección, y que se muestran cuando éste selecciona ver su colección completa.
- **getMazosUsuario:** devuelve todos los mazos del usuario con sus correspondientes cartas actualizadas.
- **comprarSobre:** añade 3 cartas aleatorias a la colección del usuario si éste tiene dinero suficiente para comprar el sobre de cartas.
- **borrarCartaMuerta:** elimina definitivamente una carta que ha muerto en combate de la colección del usuario.
- **curarCartaEnferma:** si han transcurrido el número de batallas necesario para que una carta se cure, cambia el estado de la carta en la colección del usuario y en todos los mazos en los que apareciera.
- **crearMazo/eliminarMazo:** crea un mazo vacío o elimina un mazo existente del usuario, sin eliminar las cartas de la colección y siempre que no sea el mazo activo.
- **añadir/quitarCartasMazo:** añade cartas al mazo comprobando que éste mantenga las restricciones que deben cumplir los mazos para combatir; en cambio al quitar no comprueba las restricciones, permitiendo así al usuario quitar cartas muertas o enfermas de un mazo.

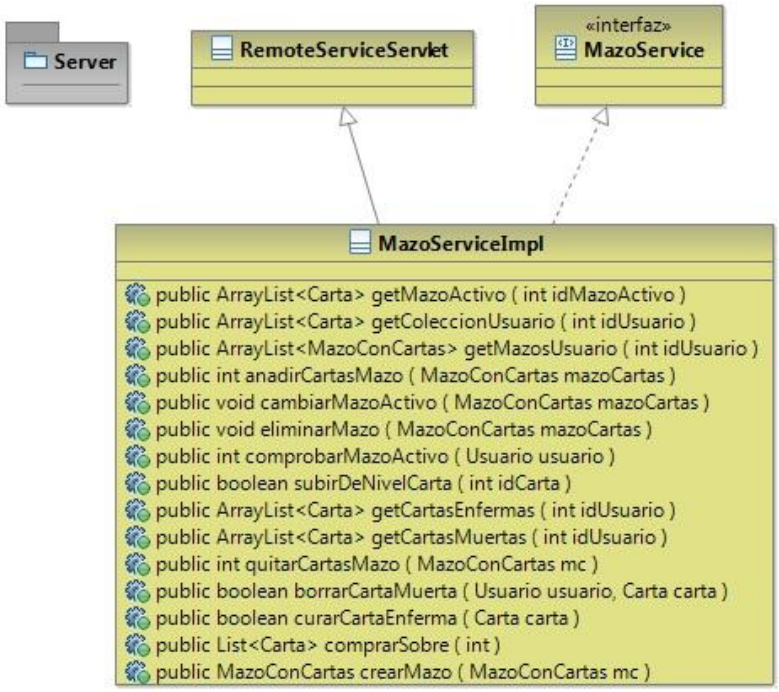


Figura 4.2.6.: Diagrama de clases de MazoServiceImpl

Clase *MercadoServiceImpl*

En este caso se actualizan en la base de datos todas las transacciones que se pueden efectuar en el mercado de cartas mediante el servicio mercadoService. Se pueden destacar los siguientes métodos:

- **venderCartas:** pone en venta en la base de datos las cartas seleccionadas por el usuario, y las elimina de los mazos en los que se encuentren ya que no puede utilizarlas mientras estén en venta.
- **comprarCartas:** compra la carta seleccionada del mercado, eliminándola de la colección del usuario vendedor y añadiéndola a la del usuario comprador. Además actualiza el oro de ambos usuarios y quita el estado de "en venta" de la carta.
- **recuperarCartas:** quita del mercado la carta seleccionada por el usuario, y vuelve a aparecer en su colección para poder añadirla a mazos y jugar con dicha carta.

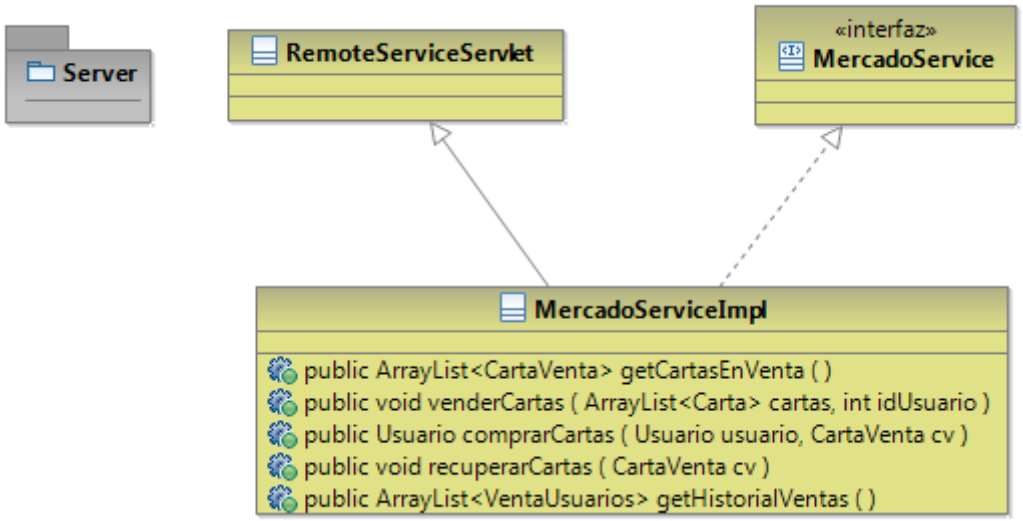


Figura 4.2.7.: Diagrama de clases de MercadoServiceImpl

Clase *BatallaServiceImpl*

Es la clase más importante de los servicios, ya que implementa todos los métodos necesarios para que se pueda llevar a cabo la batalla entre dos usuarios, todo ello mediante el servicio batallaService. Los métodos más importantes de esta clase son:

- **quiereJugar:** es el método que busca si hay otro jugador disponible, y si es así crea la partida con los datos de los dos jugadores y la batalla.
- **seleccionaCartas:** hace la selección de cartas de cada contrincante, que se almacenará en la partida común.
- **setEstadoPartida:** va modificando los estados de la partida según en que fase de la batalla se encuentre. Posteriormente se explicarán estos estados en el diagrama de estados de la sección 4.4.
- **esperarSelecciónContrario:** espera mediante comprobaciones de la tabla *Partida* a que el contrincante haya seleccionado sus cartas. Para ello se hace una comprobación de la tabla, y si aún no está listo, se duerme el proceso (*thread.sleep*) durante un segundo y se vuelve a comprobar el estado del otro jugador. Esta operación se repite en `esperarDespliegueContrario` y

`esperarOrdenesContrario`, para que la partida de forma síncrona entre los 2 jugadores y se compruebe que el contrincante no ha abandonado la partida.

- **lanzarBatalla:** es el método que ejecutará la batalla propiamente dicha. Debido a su importancia se detalla en el diagrama de secuencia de la sección 4.5.1.
- **calcularResultado:** se encarga de calcular cuál de los jugadores gana dependiendo de los puntos obtenidos, y el margen de victoria y derrota.
- **comprobarAlertas:** una vez finalizada la batalla, comprueba si existen cartas muertas, heridas o recuperadas, y si el usuario ha subido de nivel al ganar experiencia por la batalla.
- **usuarioAbandona:** cuando un usuario abandona la partida en cualquiera de sus fases es penalizado, y en este método se actualizan las penalizaciones en el usuario, las bonificaciones en el contrincante y los históricos de resultados de ambos.

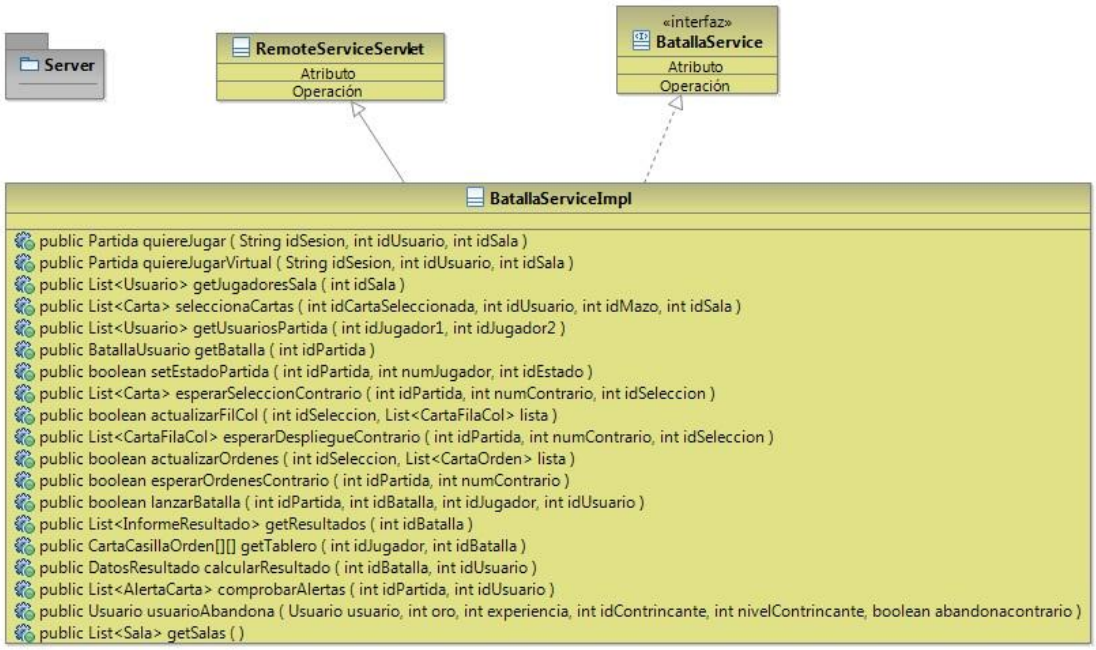


Figura 4.2.8.: Diagrama de clases de BatallaServiceImpl

4.3. Diagrama de actividad.

En la fase previa, al comienzo de la implementación de la aplicación, y una vez definidos los requisitos y los casos de uso, se diseñó un diagrama de actividad que comprendiera todo lo que podía hacer el usuario a nivel general dentro de la aplicación. En él se muestran de una manera resumida todas las posibles acciones que se pueden realizar desde que un usuario entra en el sistema hasta que se desconecta. Se ha omitido, por simplificar, el retorno al estado en el que el usuario elige qué quiere hacer, pero cada actividad una vez finalizada permite al usuario elegir libremente a que parte de la aplicación se dirige. Las diferentes posibilidades se desarrollan a partir de los cuatro botones principales de la aplicación: de “Combates” (correspondiente a

la sección de las batallas), “Colección” (relativo a la gestión de cartas y mazos del usuario), “Mercado” (correspondiente a la sección de compra y venta de cartas y sobres de cartas) y “Perfil” (asociado a la parte de información y cambios del perfil de usuario). Debido a que, excepto durante una batalla, los botones se encuentran visibles, el paso de una actividad a otra es inmediato, dando al usuario libertad de movimientos a la hora de usar la aplicación.

Inicialmente el diagrama contenía únicamente algunas características básicas, y a medida que se iba añadiendo funcionalidad al sistema se iba ampliando el número de actividades a realizar por el usuario. Igualmente se descartaron otras características que no se han incluido, como subir una foto de perfil o habilitar/deshabilitar el sonido, pero que en futuras ampliaciones se podrían incluir.

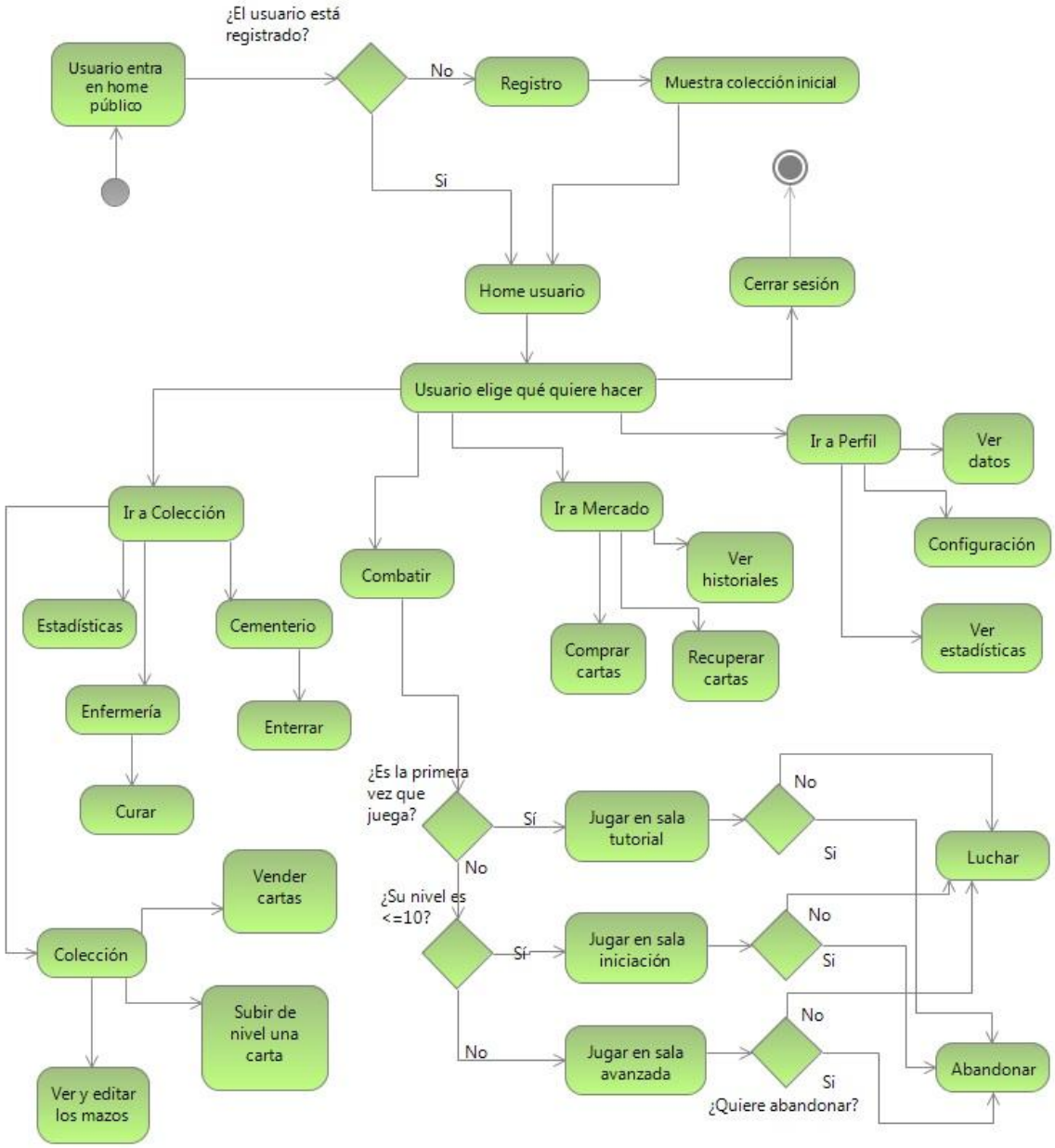


Figura 4.2.9.: Diagrama de actividad del usuario

4.4. Diagrama de estado.

En el siguiente diagrama se representan los diferentes estados por los que pasa la partida desde que un jugador entra en la sala de combates y le da al botón “Jugar” hasta que el jugador sale al finalizar el combate.

La partida pasa por los estados que se corresponden con las diferentes pantallas que se le van presentando al jugador para que ejecute las acciones pertinentes. Estas acciones forman parte de los pasos que sigue cada batalla, desde la selección aleatoria de las cartas, pasando por la colocación y la asignación de órdenes a las mismas, hasta la lucha y el resultado final.

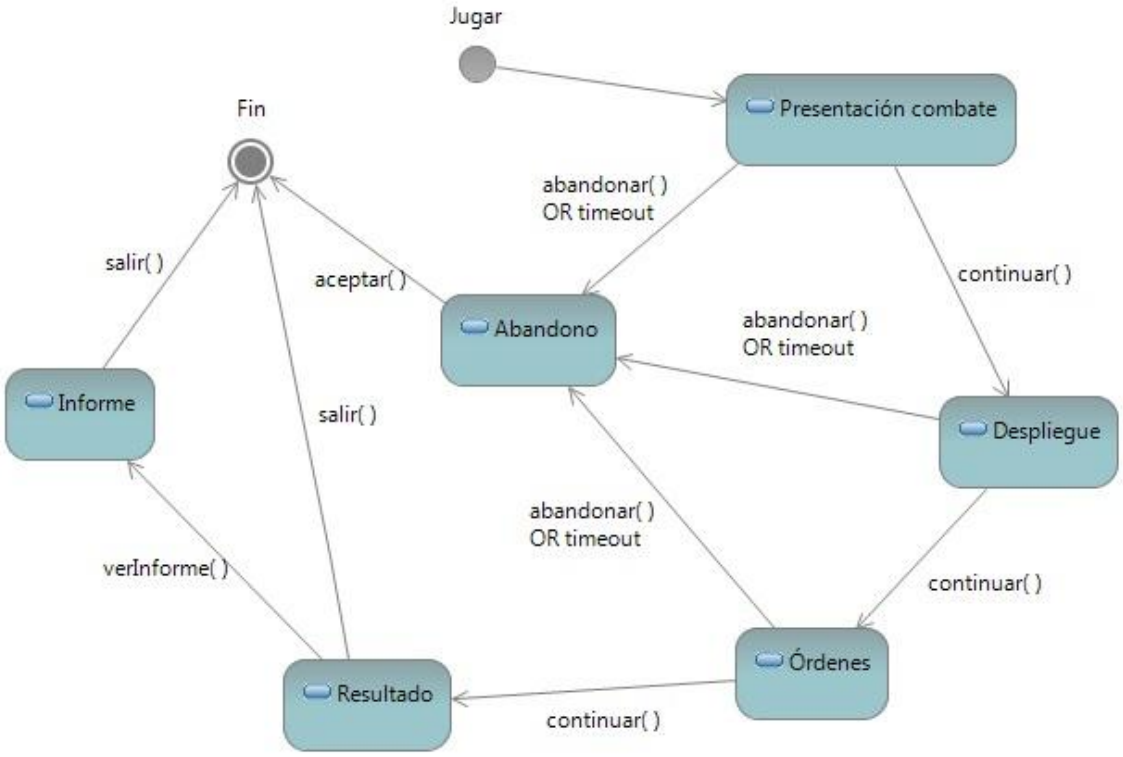


Figura 4.2.10.: Diagrama de estados de la partida

Los distintos estados son:

- **Presentación del combate:** En esta pantalla, al jugador se le presentan las cartas con las que va a combatir, que han sido seleccionadas mediante un algoritmo entre las cartas de su mazo activo. También se le muestran 3 cartas elegidas al azar del contrincante, así como su nombre de usuario y su nivel. Cuando se presiona el botón “Continuar”, la partida cambia al siguiente estado.
- **Despliegue:** En el siguiente estado se muestra al usuario el tablero y la lista con su selección de cartas y sus características. El usuario coloca entonces sus cartas en el tablero estratégicamente para organizar su ataque. En este caso, el usuario desconoce la posición de las cartas de su adversario, y puede o “Continuar” o “Abandonar” la partida.

- **Órdenes:** Se presenta el tablero completo, con las cartas del usuario y las del contrincante. Aquí el usuario puede dar las órdenes a cada carta que ha colocado dependiendo del lugar donde quiera que ataque: de frente, izquierda, derecha o a distancia (si la carta tiene esa habilidad). En este caso también se puede "Abandonar" o "Continuar" la partida.
- **Resultado:** Si el usuario no abandona, la partida pasa al estado en el que se muestra el resultado de la batalla (victoria, empate o derrota), por cuanta diferencia lo ha hecho, y los puntos de cada contrincante. En este momento se muestran las alertas del combate, y el usuario elige si quiere ver el informe completo de batalla o salir de la misma, con lo que la partida llega a su estado final.
- **Informe:** El informe detalla el desarrollo de la batalla. Indica sus aspectos globales, las rondas de la misma, con cada lucha entre cartas que se ha producido, los puntos de la carta atacante, los de la defensora, etc. De esta manera el usuario puede visualizar el comportamiento de cada carta en concreto para futuras batallas. Desde aquí únicamente se puede pulsar el botón "Salir", que finaliza la partida.
- **Abandono:** Cuando uno de los usuarios pulsa el botón "Abandonar" en cualquiera de las pantallas intermedias de la partida, el estado de ésta pasa a ser de abandono, y se informa a ambos usuarios de que la partida ha finalizado por esta causa. Al usuario que abandona se le penaliza con una cantidad de oro y experiencia dependiente de en qué estado de la partida se retire, restándose estas cantidades de sus datos de usuario. Al contrincante se le da la partida como ganada por abandono del contrario, y recibe una pequeña recompensa, aunque sus cartas no ganan experiencia. Al pulsar "Aceptar", se finaliza completamente la partida

4.5. Diagramas de secuencia

4.5.1. Método lanzarBatalla

Uno de los métodos principales relativo a los combates entre usuarios es el `lanzarBatalla` de la clase ***RebelWar***. Este método, que está asociado a la acción de pulsar el botón "A luchar" posterior al despliegue y órdenes de las cartas, hace una llamada al método del mismo nombre del servicio BatallaService, que está implementado en la clase ***BatallaServiceImpl***.

Como se observa en el diagrama, el comportamiento es distinto si se trata del jugador 1 o el jugador 2. Desde el punto de vista interno de la aplicación, siempre es el objeto vinculado al primer jugador el que ejecuta propiamente la batalla y almacena los resultados, mientras que el del segundo jugador solo espera a que haya finalizado.
En el caso del jugador 1, en el método `lanzarBatalla` se crea el tablero completo (las filas 0 y 1 son del jugador 1, y las 2 y 3 del jugador 2), que se usará para colocar las cartas de ambos jugadores. El tablero guarda una matriz de cartas seleccionadas (que tienen una posición asignada según el despliegue del jugador), otra matriz de órdenes, y otras dos de booleanos, una para saber qué cartas se van muriendo y otra igual para los huidos.

A continuación, se crea la batalla con el tablero y los usuarios, y se ordenan las cartas para la batalla por velocidad en orden descendente, ya que las cartas más rápidas son las que atacan primero. Después se lanza la propia batalla.
Tras la batalla, se guardan en la BD los resultados de las luchas para generar el informe y se actualizan las cartas con los puntos de experiencia y victoria que han ganado, las heridas, etc. También se crean las diferentes alertas que se le mostrarán al usuario al finalizar la batalla. Esta parte se ha resumido en los métodos `crearAlertas` y `actualizarInforme` del diagrama para simplificarlo.

Para entender el comportamiento del jugador 2 hay que explicar como se han implementado los hilos de ejecución. Dado que solo es necesario que uno de los jugadores lance la batalla, puesto que la información de ambos contrincantes ya está preparada, el segundo jugador únicamente espera a que la batalla iniciada por el primero finalice. Debido a que el jugador 2 sólo puede saber si ha finalizado la batalla consultando la partida en la base de datos, mientras el estado del jugador 1 no sea igual a 4 (partida finalizada), el proceso de batalla del segundo jugador se duerme durante un segundo (`thread.sleep`) antes de volver a consultar la base de datos. Así se garantiza que ambos jugadores estén sincronizados y se actualicen los resultados de la batalla y puedan verlos a la vez.

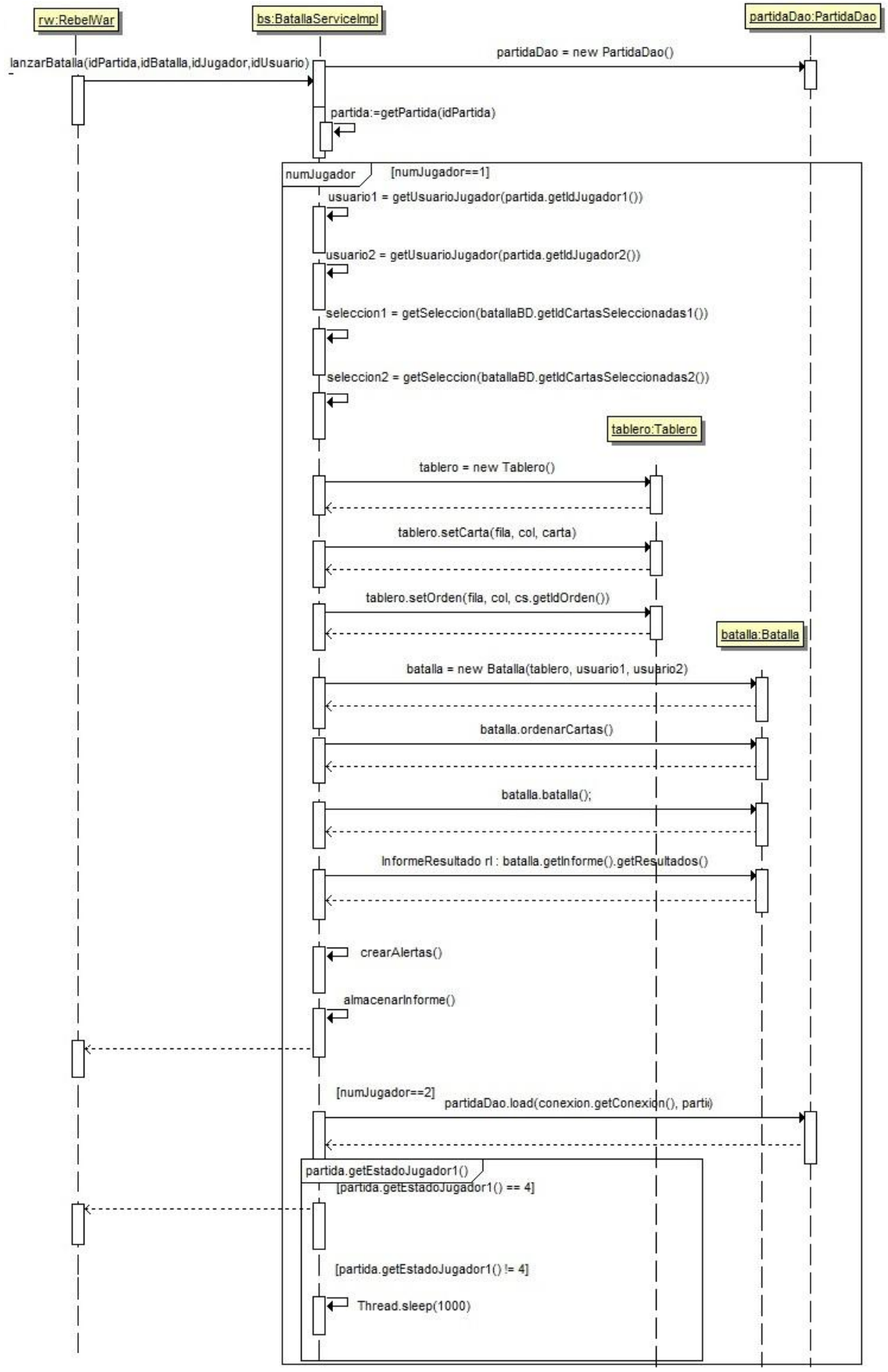


Figura 4.2.11.: Diagrama de secuencia de lanzarBatalla

5. Pruebas

Se han realizado pruebas funcionales y de rendimiento de la aplicación. Las primeras han sido pruebas manuales, para comprobar el correcto funcionamiento de las funcionalidades básicas de la aplicación desarrollada. Las segundas han sido pruebas para comprobar que el servidor puede soportar más de dos sesiones simultáneas y que los tiempos de resolución de los algoritmos de la batalla no se ven afectados de manera drástica por el número de jugadores en el sistema.

5.1. Pruebas manuales

A continuación se detallan las pruebas manuales realizadas, las cuales pueden dividirse en tres grupos según el módulo de la aplicación que se prueba.

5.1.1. Usuarios:

Registro de un nuevo usuario

Pasos	• Hacer clic en el botón "¡Regístrate!" del formulario público inicial. • Rellenar todos los textbox cuidando que la contraseña y la repetición de la contraseña sean iguales, seleccionar una de las razas disponibles y hacer clic en el botón "Registro". • Comprobar que aparecen todas las imágenes de las cartas que se le dan al usuario cuando se registra (20 cartas) y hacer clic en el botón "Continuar". • Hacer clic en el botón "Colección" y seleccionar "Colección completa" en el combobox de la derecha. • Comprobar que en la colección están las 20 cartas que se le han dado al usuario al registrarse. • Hacer clic en el botón de desconexión y volver a conectarse con el email y contraseña del usuario recién registrado. • Comprobar que se realiza la conexión correctamente, es decir, no sale ningún mensaje de error.
Resultado	Ok

Cambio de los datos personales de un usuario

Pasos	• Conectarse a la aplicación con uno de los usuarios registrados. • Hacer clic en el botón “Perfil” y en la pestaña “Configuración”. • Cambiar el nombre de usuario y hacer clic en el botón “Guardar cambios”. • Ir a la pestaña “Datos” y comprobar que aparece el nuevo nombre de usuario.
Resultado	Ok

Cambio de la contraseña de un usuario

Pasos	• Conectarse a la aplicación con uno de los usuarios registrados. • Hacer clic en el botón “Perfil” y en la pestaña “Configuración”. • Cambiar la contraseña de usuario y hacer clic en el botón “Guardar cambios”. • Desconectarse y volverse a conectar usando la nueva contraseña. • Comprobar que se realiza la conexión correctamente, es decir, no sale ningún mensaje de error.
Resultado	Ok

5.1.2. Mercado

Puesta de una carta a la venta

Pasos	• Conectarse a la aplicación con uno de los usuarios registrados. • Hacer clic en el botón “Colección” y seleccionar “Colección completa” en el combobox de la derecha. • Marcar la primera carta de la lista y hacer clic en el botón “Vender”. • Hacer clic en el botón “Vender” del pop-up emergente. • Desconectarse de la aplicación y volver a conectarse con otro usuario registrado. • Hacer clic en el botón “Mercado” y comprobar que la carta que se ha puesto en venta aparece en la lista de cartas en venta acompañada de un botón “Comprar”.
Resultado	Ok

Recuperación de una carta puesta a la venta

Pasos	• Conectarse a la aplicación con uno de los usuarios registrados. • Hacer clic en el botón “Colección” y seleccionar “Colección completa” en el combobox de la derecha. • Marcar la primera carta de la lista y hacer clic en el botón “Vender”. • Hacer clic en el botón “Vender” del pop-up emergente. • Hacer clic en botón “Mercado” y comprobar que la carta que se ha puesto en venta aparece en la lista de cartas en venta acompañada de un botón “Recuperar”. • Hacer clic en el botón “Recuperar”. • Hacer clic en el botón “Colección” y seleccionar “Colección completa” en el combobox de la derecha. • Comprobar que la carta que hemos recuperado aparece en la lista de cartas de la colección.
Resultado	Ok

Compra de una carta en venta

Pasos	• Conectarse a la aplicación con uno de los usuarios registrados. • Hacer clic en el botón “Colección” y seleccionar “Colección completa” en el combobox de la derecha. • Marcar la primera carta de la lista y hacer clic en el botón “Vender”. • Hacer clic en el botón “Vender” del pop-up emergente. • Desconectarse de la aplicación y volver a conectarse con otro usuario registrado. • Hacer clic en el botón “Mercado” y comprobar que la carta que se ha puesto en venta aparece en la lista de cartas en venta acompañada de un botón “Comprar”. • Hacer clic en el botón “Comprar” y comprobar que en el panel de la izquierda que muestra los datos del usuario se descuenta el precio de la carta de las monedas de oro que posee el usuario. • Hacer clic en el botón “Colección” y seleccionar “Colección completa” en el combobox de la derecha. • Comprobar que la carta recién comprada aparece en la lista de cartas de la colección.
Resultado	Ok

Compra de un sobre de cartas

Pasos	• Conectarse a la aplicación con un usuario registrado. • Hacer clic en el botón “Mercado” a la pestaña “Comprar sobres” y hacer clic sobre una de las imágenes de los sobres de las diferentes razas. • Hacer clic en el botón “Aceptar” del pop-up emergente. • Comprobar en el panel de la izquierda, que muestra los datos del usuario, que se han descontado 100 monedas de oro. • Hacer clic en el botón “Colección” y seleccionar “Colección completa” en el combobox de la derecha. • Comprobar que las cartas que contenía el sobre se encuentran en la lista de cartas de la colección.
Resultado	Ok

5.1.3. Colección

Creación de un nuevo mazo y adición de cartas al mismo

Pasos	• Conectarse a la aplicación con un usuario registrado. • Hacer clic en el botón “Colección” y hacer clic en el botón “Crear nuevo mazo”. • Poner un nombre al nuevo mazo y hacer clic en el botón “Crear”. • Seleccionar “Colección completa” en el combobox de la derecha. • Seleccionar las primeras 15 cartas de la colección y hacer clic en el botón “Añadir a un mazo”. • Seleccionar el mazo que acabamos de crear y hacer clic en el botón “Añadir al mazo”. • Seleccionar el nuevo mazo en el combobox de la derecha y comprobar que están las 15 cartas que se acaban de añadir.
Resultado	Ok

Cambio del mazo activo

Pasos	• Conectarse a la aplicación con un usuario registrado. • Hacer clic en el botón “Colección” y crear un mazo y añadirle cartas. • Hacer clic en el botón “Cambiar mazo activo”. • Seleccionar el mazo que se acaba de crear y hacer clic en el botón “Cambiar”. • Seleccionar “Mazo activo” en el combobox de la derecha y comprobar que se ha cambiado el mazo activo correctamente.
Resultado	Ok

Eliminación de un mazo

Pasos	• Conectarse a la aplicación con un usuario registrado. • Hacer clic en el botón “Colección” y crear un mazo y añadirle cartas. • Hacer clic en el botón “Eliminar mazo” y seleccionar el mazo que acabamos de crear. • Comprobar que el mazo eliminado ya no aparece en el combobox de la derecha.
Resultado	Ok

5.2. Pruebas de rendimiento

Se han realizado cuatro pruebas de combates aumentando el número de jugadores conectados simultáneamente.

Prueba con dos jugadores

Pasos	• Dos usuarios se conectan a la aplicación simultáneamente. • Los dos usuarios entran en la sala “Tutorial” haciendo clic en el botón “Combates” y después en el botón de dicha sala. • Los dos jugadores hacen clic en el botón “Jugar”. • Los dos jugadores juegan la batalla hasta el final, controlando el tiempo de respuesta del servidor.
Resultado	Ok. Las respuestas del servidor son inmediatas.

Prueba con cuatro jugadores en la misma sala

Pasos	• Cuatro usuarios se conectan a la aplicación simultáneamente. • Los cuatro usuarios entran en la sala "Tutorial" haciendo clic en el botón "Combates" y después en el botón de dicha sala. • Los cuatro jugadores hacen clic en el botón "Jugar". • Los cuatro jugadores juegan la batalla hasta el final, controlando el tiempo de respuesta del servidor.
Resultado	Ok. Las respuestas del servidor son inmediatas.

Prueba con cuatro jugadores en dos salas distintas

Pasos	• Cuatro usuarios se conectan a la aplicación simultáneamente. • Dos usuarios entran en la sala "Tutorial" y otros dos en la sala "Prácticas de maniobras" haciendo clic en el botón "Combates" y después en el botón de la sala correspondiente. • Los cuatro jugadores hacen clic en el botón "Jugar". • Los cuatro jugadores juegan la batalla hasta el final, controlando el tiempo de respuesta del servidor.
Resultado	Ok. Las respuestas del servidor son inmediatas.

Prueba con seis jugadores en la misma sala

Pasos	• Seis usuarios se conectan a la aplicación simultáneamente. • Los seis usuarios entran en la sala "Tutorial" haciendo clic en el botón "Combates" y después en el botón de dicha sala. • Los seis jugadores hacen clic en el botón "Jugar". • Los seis jugadores juegan la batalla hasta el final, controlando el tiempo de respuesta del servidor.
Resultado	Ok. Las respuestas del servidor son inmediatas.

5.3. Conclusiones de las pruebas

Con los resultados obtenidos en las pruebas manuales y en las pruebas de rendimiento se puede afirmar que la aplicación cumple los requisitos planteados en cuanto a funcionalidades y tiempos de respuesta con un alto grado de certeza.

6. Conclusiones

El mundo de los MWG presenta un cierto vacío en cuanto a un sector de aplicaciones que se ha intentado solventar con el desarrollo de *Rebel War*. Inicialmente se ha planteado la necesidad de hacer un juego que permitiera al usuario decidir libremente el tiempo a emplear en él, sin que su desempeño se viera afectado, y que fuese capaz de ofrecerle novedades en su funcionamiento, de forma que mantuviese su interés en el tiempo.

Los objetivos anteriores han sido cubiertos en el tiempo de desarrollo disponible. *Rebel War* ha sido además diseñado teniendo en mente que se quiere que sea un juego fácil de evolucionar y concebido para incorporar las múltiples ampliaciones y mejoras potenciales que se apuntan en el trabajo futuro.

El desarrollo de las funcionalidades de *Rebel War* se llevó a cabo en el tiempo estimado en la planificación, exceptuando la comunicación de los usuarios en la batalla. Conseguimos que se comunicaran con relativa facilidad, pero la depuración de métodos de la batalla resultó más costosa de lo que esperábamos y nos llevó más tiempo del previsto.

Además del desarrollo en sí, como grupo nos ha resultado muy interesante llevar a cabo este proyecto por su temática y las técnicas y herramientas empleadas. *Rebel War* nos ha servido para aprender el manejo de las aplicaciones con arquitectura cliente-servidor, de las librerías de GWT y la integración de bases de datos en aplicaciones. También nos ha permitido aplicar ingeniería del software en un proyecto realista. Por ello y por el resultado final consideramos que este proyecto será una experiencia destacable a incluir en nuestro currículo.

6.1. Trabajo futuro

Como trabajo futuro, a continuación se proponen mejoras y ampliaciones. Las mejoras pretenden aumentar la complejidad las batallas y así hacer el juego más estratégico e interesante para los usuarios. Las ampliaciones persiguen mejorar la experiencia del usuario con la aplicación añadiendo nuevos módulos a los ya implementados.

6.1.1. Mejoras

Las posibles mejoras futuras en *Rebel War* están principalmente orientadas al módulo de la batalla. Las batallas que se han implementado para este proyecto son muy básicas y pueden ampliarse en muchos sentidos:

> **Más habilidades**: Debido al tiempo del que se disponía sólo se han implementado cinco habilidades básicas. Para añadir más habilidades a la batalla sólo habría que definirlas implementando la interfaz Habilidad.

Más órdenes de ataque: La batalla actualmente dispone de ataque de frente, ataque a la derecha, ataque a la izquierda y ataque a distancia. Se podrían implementar también otros ataques más complejos como, por ejemplo: ataque a distancia diagonal a derecha e izquierda y ataque salto de caballo (que imita el movimiento del caballo en ajedrez).

Más cartas: Cada raza tiene veinte cartas diferentes. Este número podría incrementarse añadiendo cartas con nuevos atributos y habilidades.

Bonificadores de raza: Ahora la selección de raza cuando un usuario se registra en *Rebel War* no tiene más trascendencia que jugar con la raza preferida. Podría añadirse un bonificador de raza a cada una de ellas, esto es una habilidad especial para cada raza. Así los usuarios no pensarían solo en la que más les gusta sino también en las ventajas que pueden conseguir eligiendo una determinada raza.

Más razas: *Rebel War* dispone de cuatro razas (humanos, elfos, enanos y orcos). Podrían añadirse más razas del mundo de fantasía, como por ejemplo: elfos oscuros, espectros, medianos, etc. Cada una de ellas se añadiría con su correspondiente bonificador de raza.

Todas estas mejoras harían el juego mucho más variado lo que, unido a su gratuidad total, aumentaría el interés de los usuarios. Haciendo juegos gratuitos más interesantes de los ya existentes de pago y, por lo tanto, más jugados, se podría conseguir que los juegos que nos encontramos en el mercado empezaran a ser gratuitos para competir con la nueva vertiente.

6.1.2. Ampliaciones

Las posibles ampliaciones que proponemos se pueden dividir en dos tipos según la funcionalidad a la que están orientadas:

Batalla: La batalla, tal cual está implementada ahora, se realiza entre dos jugadores que entran en la misma sala y son emparejados aleatoriamente. Podría ampliarse este módulo incluyendo batallas individuales contra una inteligencia artificial. Llamaremos misiones a este tipo de batallas. El usuario que completara estas misiones ganaría premios extra (más monedas de oro y experiencia que en una batalla normal, cartas exclusivas...). Las misiones estarían disponibles para todos los usuarios, así que conseguir estos premios extra estaría al alcance de todos.

Otras funcionalidades de Rebel War: En cuanto al resto de módulos de *Rebel War* proponemos las siguientes ampliaciones:

- **Gremios**: Los usuarios podrían formar asociaciones para intercambiar sabiduría y cartas de manera más ventajosa que usando el mercado normal.

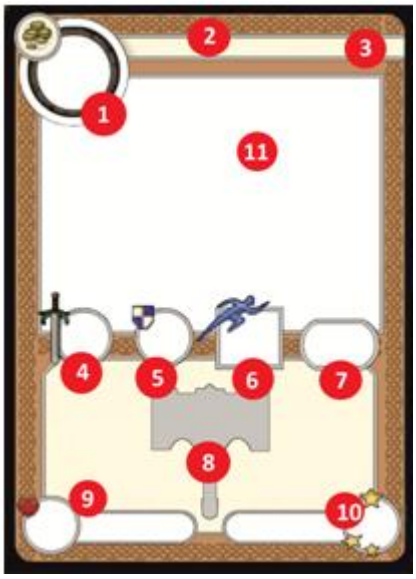
En un futuro también podrían organizarse torneos de combates en las asociaciones y premiar a los ganadores.

- **Foros**: En un MWG, sobre todo en uno con gremios, es importante que los usuarios puedan comunicarse unos con otros.
- **Chat durante la batalla**: Para aquellos jugadores que quieran relacionarse con sus contrincantes, siempre respetándose mutuamente.
- **Regateo en el mercado**: Ofrecer la posibilidad de regatear a vendedores y compradores en el mercado, para llegar entre ambos a un precio justo. Por supuesto si uno de los dos, vendedor o comprador, no está de acuerdo con el regateo puede rechazarlo.
- **Más datos en el perfil de usuario**: Dar la posibilidad al usuario de añadir más datos sobre sí mismo e incluso una imagen a su perfil.
- **Consultar perfiles de otros usuarios**: Los usuarios podrían buscar y consultar los perfiles de otros usuarios.

7. Apéndices

7.1. Apéndice A: Reglas del juego

7.1.1. Cartas



1. Coste en oro
2. Nombre
3. Rareza
4. Ataque
5. Defensa
6. Velocidad
7. Moral
8. Habilidades, tipos de ataque y daños
9. Vida total y barra de vida
10. Experiencia total y nivel de experiencia
11. Imagen

Figura 7.A.1.: Diagrama de una carta

Coste de oro: Cada carta tiene un coste en monedas de oro. Este coste no tiene nada que ver con el mercado de cartas, es el coste de la tropa para el ejército. Las batallas se realizan a un determinado límite de puntos para que estén lo más equilibradas posibles, y la suma de las monedas de oro de todas las tropas de tu bando que sean seleccionadas por el juego para que participen en la batalla no superarán ese límite. Tu mazo no tiene un límite en cuanto al coste en monedas de oro, pero la selección de cartas para cada batalla será la que se ciña a ese límite.

Nombre: Nombre de la carta. Se divide en el nombre de la tropa y el nombre único de la carta.

Rareza: Indica la frecuencia con la que aparece la carta y lo difícil que es que aparezca en los sobres.

- = Ultra rara (personajes)
- = Rara
- = Infrecuente
- = Común

Ataque: Valor de ataque de la carta para el desarrollo de los combates (atributo necesario para infligir golpes a los enemigos).

Defensa: Valor de defensa de la carta para el desarrollo de los combates (atributo para defenderse de los ataques enemigos y armadura de la tropa).

Velocidad: Valor de velocidad de la carta para el desarrollo de los combates (el orden con el que actúan las cartas en los combates lo marca este atributo).

Moral: Valor de moral de la carta para el desarrollo de los combates (% de posibilidades que tiene la tropa de NO huir tras un ataque. Hay valores por encima de 100% porque este valor se ve modificado por la cantidad de daños sufridos).

Habilidades: Diferentes habilidades que tendrán las tropas y que harán el juego mucho más variado. En esta zona de la carta también se mostrarán los ataques de la tropa, ya sean de disparo, de cuerpo a cuerpo o cualquier otro tipo de ataque especial que posea.

Experiencia: El valor numérico corresponde al nivel de experiencia en el que se encuentra la carta. Los 2 valores de la barra adyacente indican los puntos de experiencia actuales y el límite que se tiene que alcanzar para pasar al siguiente nivel.

Vida: El valor numérico indica la cantidad de puntos de vida que le restan a la tropa. Los 2 valores de la barra adyacente indicarán los puntos de vida actuales y los puntos de vida máximos que resiste la tropa antes de morir.

Imagen: La ilustración que muestra cómo es la tropa.

7.1.2. Razas

- Humanos.
- Elfos.
- Orcos.
- Enanos.

7.1.3. Niveles de usuario

A lo largo de las partidas en Rebel War, el usuario irá recibiendo puntos de combate. Estos puntos de combate sirven para ir mejorando el nivel del usuario. Estos niveles aumentan de forma similar a los niveles de experiencia de las cartas, cada cierta cantidad de puntos de combate se sube de nivel. Esta cantidad va en aumento de forma exponencial, por lo que es más costoso subir del nivel 25 al 26 que del nivel 2 al 3. Los niveles de usuario permiten valorar de una forma bastante fiable lo veterano que es un jugador y por tanto lo peligroso que puede ser enfrentarse a él en una batalla.

7.1.4. Mazos

El mazo para combatir debe cumplir unos requerimientos:

Entre 15 y 30 cartas en total, con las siguientes limitaciones:
- 1 personajes como máximo (0 – 1).
- 6 cartas comunes como mínimo y 30 como máximo (6 – 30).
- 6 cartas infrecuentes como máximo (0 – 6).
- 3 cartas raras como máximo (0 – 3).
- 4 cartas iguales como máximo.
- Todas las cartas de la misma raza.

7.1.5. Experiencia

Rebel War es un juego de enorme variedad en cuanto a jugabilidad. Las tropas de nivel 1 son iguales para todos los jugadores que las tengan en su colección, pero cuando alcancen niveles posteriores, cada carta tiene unas tablas de subida de experiencia (diferentes en cada tropa) que le permitirá modificar positivamente sus atributos, su precio en oro y en ocasiones sus habilidades.

Por lo general las tropas suben hasta el nivel 5, donde alcanzan el nivel máximo, pero existen 2 excepciones a esta regla:
- Los personajes (cartas ultra raras) pueden subir hasta el nivel 10.
- Las tropas de la raza de los elfos, debido a su extraordinaria longevidad, pueden alcanzar un nivel extra, es decir que sus tropas normales alcanzan nivel 6 y los personajes nivel 11.

7.1.6. Habilidades

Hay una gran cantidad de habilidades diferentes en el juego, todas ellas tienen diferentes efectos, algunas en la batalla y otras en otros aspectos del juego. Algunas cartas pueden empezar con ellas o ganarlas según suban de nivel.

Las habilidades solo pueden obtenerse cuando las tropas suben de nivel. Todas las habilidades que modifiquen de alguna forma los atributos u otras habilidades de las cartas, lo harán únicamente por el periodo de duración de la batalla, volviendo a su estado original al finalizar esta.

7.1.7. Salas

Cada una de las diferentes salas de juego permite disfrutar de batallas bajo algunas reglas y condiciones distintas:

> **Sala tutorial:** Niveles 0-5. Sala de 400 puntos. Sala de aprendizaje, para que el usuario se acostumbre al modo de juego combatiendo contra jugadores en su misma situación.

Sala de prácticas de maniobras: Niveles 1-10. Sala de 700 puntos. El lugar perfecto para que el usuario perfecciones sus tácticas de combate.

Sala campo de batalla: (1000 puntos) – Donde se desarrollan los combates serios de verdad. Las consecuencias se pagan y las tropas pueden sufrir heridas o incluso morir, pero la recompensa será mayor en experiencia, puntos de combate y dinero.

7.1.8. Batallas

En las batallas se enfrenan dos usuarios que previamente hayan ingresado en la sala de combate y que habrán sido emparejados de forma aleatoria para realizar su combate. Cada usuario participará con su mazo activo, que tendrá que cumplir los requisitos de la sala.

Las batallas se van a dividir en varias fases para explicar el proceso de una forma más clara.

1. Emparejamiento: Los usuarios se emparejarán de forma aleatoria.

2. Fase de selección de cartas: Aleatoriamente se seleccionarán las cartas del mazo de cada usuario que van a participar en el combate. El juego seleccionará las cartas de tal forma que parará cuando llegue al límite de 10 cartas (máxima cantidad de cartas que caben en el tablero) o no quede ninguna carta sin seleccionar que pueda unirse al ejército dentro del límite de puntos.

Supongamos una partida en la sala de batallas a 500 monedas de oro en la que entramos con el siguiente mazo
1 Noble (coste = 80 oro)
1 Paladín (coste = 90 oro)
2 Ballesteros (coste = 60 oro cada uno)
1 Piquero (coste = 63 oro)
4 Milicianos (coste = 29 oro cada uno)
2 Arqueros (coste = 56 oro cada uno)
1 Mastín de Guerra (coste = 38 oro)
1 Alabardero (coste = 55 oro)
1 Jinete (coste = 60 oro)
1 Soldado (coste = 45 oro)

El juego realizará el siguiente algoritmo para elegir los participantes en el combate:
Noble (80 oro –> total = 80) [Solo hay un personaje por lo que hay que seleccionarlo]
Arquero (56 oro -> total = 136) [Ha tocado aleatoriamente]
Piquero (63 oro -> total = 199) [“]
Soldado (45 oro -> total = 244 [“]
Arquero (56 oro -> total = 300 [“]
Paladín (90 oro -> total = 390 [“]
Ballestero (60 oro -> total = 450) [“]
Ballestero (60 oro -> total = 510) [Se descarta porque se excede las 500 monedas de oro]
Miliciano (29 oro -> total = 479) [Ha tocado aleatoriamente]
Se termina el algoritmo porque se comprueba que no falta ninguna carta de valor menor de 21 de oro que pueda completar el ejército, por lo que para esta partida nuestro ejército será de 479 puntos y estará compuesto por:

Noble + Arquero + Piquero + Soldado + Arquero + Paladín + Ballestero + Miliciano

3. Fase de despliegue: Una vez el juego ha seleccionado las cartas (tropas) que participarán en la batalla, los usuarios tienen un tiempo para distribuirlas sobre el campo de batalla, sin saber cómo va a distribuirlas el rival.

El campo de batalla está compuesto por 4 filas y 5 columnas, donde las 2 filas inferiores corresponden a tu zona de despliegue y las 2 superiores a la zona de despliegue del enemigo.

Siguiendo con el ejemplo, en la fase de despliegue optamos por distribuir las tropas de la siguiente manera:

	Piquero	*Soldado*	*Paladin*	*Miliciano*
Arquero		*Noble*	*Ballestero*	*Arquero*

4. Fase de adjudicación de órdenes: Una vez se consuma el tiempo estipulado para el despliegue o ambos jugadores hayan terminado, se mostrará la disposición del rival y se pasará a otro periodo en el que ambos jugadores podrán asignar órdenes para la batalla a sus tropas. Esta es la fase de adjudicación de órdenes o mando y es la fase principal del juego, donde se ganan y se pierden batallas. Las posibles órdenes a tener en cuenta en la fase de mando son:

Ataque: Las tropas pueden atacar a cualquier enemigo que esté en contacto, teniendo en cuenta que si atacan en diagonal teniendo una tropa enemiga delante, el daño causado será la mitad.
Defensa: La tropa no hace nada y se defiende únicamente
Distancia: La tropa puede disparar a diferentes casillas del campo de batalla dependiendo del tipo de arma y por tanto del tipo de habilidad de disparo que tenga. Hay que resaltar que los ataques a distancia de las tropas modifican el valor del atributo de velocidad para la batalla, una tropa tiene una velocidad básica que se ve incrementada cuando su orden es de realizar un ataque a distancia.

Siguiendo con el ejemplo vemos que el rival ha desplegado sus tropas de la siguiente forma:

	Caudillo	*Saqueador*		*HIena*
Huargo		*Uruk-Hai*	*Huargo*	
	Piquero	*Soldado*	*Paladin*	*Miliciano*
Arquero		*Noble*	*Ballestero*	*Arquero*

OPONENTE

NUESTRO EJÉRCITO

Y en la fase de mando grabamos las siguientes órdenes a cada tropa:

	Piquero [Ataque ↗]	*Soldado [Ataque ↑]*	*Paladin [Ataque ↑]*	*Miliciano [Defensa]*
Arquero [Disparo ↑]		*Noble [Defensa]*	*Ballestero [Disparo ↑]*	*Arquero [Defensa]*

Las posibles órdenes de combate que se podrán dar a cada tropa son las siguientes:

ÓRDENES BÁSICAS	**ABREVIATURA**	**FLECHA**
Atacar al frente	Frente	↑
Atacar a la derecha	Derecha	↗
Atacar a la izquierda	Izquierda	↖

ÓRDENES AVANZADAS	**ABREVIATURA**	**HABILIDAD DEPENDIENTE**	**FLECHA**
Disparo 1 casilla de frente	Disparo [1] Frente	- Ataque a distancia - Arma arrojadiza	↑
Disparo 1 casilla a la derecha	Disparo [1] Derecha	- Ataque a distancia + Arquería - Arma arrojadiza	↗
Disparo 1 casilla a la izquierda	Disparo [1] Izquierda	- Ataque a distancia + Arquería - Arma arrojadiza	↖
Disparo 2 casillas de frente	Disparo [2] Frente	- Ataque a distancia	⇑

5. Combate: Las batallas se desarrollan por rondas de combate o “rounds”, que se numeran y desarrollan en orden descendente. En cada round actuarán las tropas (de ambos bandos) cuya velocidad sea igual al round que esté en esos momentos activo, por lo que el primer round a disputarse será el round correspondiente a la tropa con más velocidad que haya sobre el campo de batalla y el último será el round correspondiente a la tropa con menos velocidad que quede en el campo de batalla. Si dos o más tropas tienen la misma velocidad, se determinará aleatoriamente el orden de actuación dentro del round.

Siguiendo con el ejemplo, los rounds de combate serian los siguientes:

ROUND 15 – Ballestero (disparo)
ROUND 14 – Arquero (disparo), Arquero (disparo), Huargo, Huargo
ROUND 13 – Hiena
ROUND 12 – Piquero
ROUND 11 - Paladín
ROUND 10 – Noble, Soldado, Caudillo, Uruk-Hai, Miliciano
ROUND 9 – Saqueador

(En los ROUNDS 14 y 10 se elegirá aleatoriamente el orden de actuación)

Los algoritmos de combate funcionan de la siguiente forma:

- El atacante realiza tantos disparos como indica su perfil

El perfil del Ballestero indica un disparo de 4X d10, por lo que realiza 4 ataques

- El porcentaje de probabilidades básico de impactar con cada uno de los ataques es del 50%.

- Se comparan los atributos de ATAQUE de la tropa que dispara y la DEFENSA del objetivo y se modifica el porcentaje de impacto en un 3% por cada punto de diferencia que haya.

El Ballestero dispara al Huargo: *- ATAQUE Ballestero = 13*
- DEFENSA Huargo = 9

La diferencia entre los atributos es 4 (13 - 9 = 4), a favor del atacante, por lo que el porcentaje de impacto será (4 x 3 = 12%) del 62%, que es el resultado de sumar 50% básico + 12%. Por lo tanto cada uno de los disparos tiene un 62% de posibilidades de impactar en el Huargo.

- Una vez se saben los disparos que han impactado, se calcula cuantos daños hace cada uno, según indica el atributo de la carta.

Supongamos que el Ballestero impacta con 3 de sus disparos, entonces se deberá tirar 3X d10, ya que cada disparo del Ballestero causa d10 daños. Los resultados de las tres tiradas del dado son: 3, 7 y 2 = 12 daños

- Cuando ya se ha calculado la cantidad de daños que causa el atacante, el defensor tendrá que realizar un chequeo de moral (una tirada en la que usa como referencia su atributo de moral) para comprobar si huye del campo de batalla como consecuencia del ataque o mantiene la posición. Los chequeos de moral son una tirada de d100 que se efectúa comparando el resultado con el atributo de moral modificado. Si el resultado es menor que el valor con el que se compara la tropa aguanta, si es mayor huye. El atributo de moral se modifica para los chequeos una parte proporcional igual a los daños sufridos respecto a la vida restante antes del ataque.

En el ejemplo del Ballestero, la hiena ha sufrido 12 daños por los disparos de ballesta, y como la Hiena tenía un total de 24 puntos de vida y ha sufrido una pérdida de la mitad de puntos de vida por el disparo; tendrá que realizar el chequeo con la mitad del atributo de moral, que en este caso será 80 \ 2 = 40.
Por lo tanto la Hiena tendrá que sacar 40 o menos en una tirada de d100 para mantenerse en el campo de batalla.

Los siguientes pasos solo tendrán lugar en el combate cuerpo a cuerpo.

Llegados a este punto si la tropa atacada no huye, tiene derecho a responder al ataque, realizando una ronda extra de ataques, que se desarrolla igual que la ronda de ataques del atacante.

Cuando todas las rondas de combate hayan terminado de ejecutarse, la batalla se dará por concluida y se procederá al recuento de puntos, el reparto del botín y se determinará quién es el ganador. Ambos jugadores recibirán una pequeña cantidad de

dinero del juego y una cantidad de puntos de ranking determinada por lo bien (o mal) que les hayas ido la batalla.

7.1.9. Enfermería y cementerio

Cuando una tropa que está participando en una batalla huye del combate o sus puntos de vida se reducen a cero durante el mismo puede sufrir heridas permanentes o efectos negativos de algún tipo. Para comprobar las consecuencias de la batalla, se realizarán tiradas aleatorias para determinar qué le ha pasado a cada tropa. Como consecuencia de la batalla las cartas podrán ver disminuidos sus atributos, podrán resultar heridas (irán a la enfermería y se perderán algunos combates) e incluso podrán morir para siempre (irán al cementerio).

7.2. Apéndice B: Tutorial

7.2.1. ¿Cómo empezar?

Para poder disfrutar de *Rebel War* lo primero que tienes que hacer es registrarte como usuario. Para ello entra en la página del juego con tu navegador http://shiva.fdi.ucm.es:8095/RebelWar/RebelWar.html y sigue los siguientes pasos:

Figura 7.B.1.: Pantalla de home público

1. Haz clic en el botón “¡Regístrate!”.
2. Rellena todos los campos obligatorios (los marcados con un *). Pon especial atención al escribir tu email y la contraseña y su repetición.
3. Selecciona una de las razas disponibles: Humanos, Elfos, Orcos y Enanos. Si vas seleccionándolas de una en una podrás ver a la derecha de la pantalla una breve descripción que puede facilitar tu elección.
4. Haz clic en el botón “Registro”.

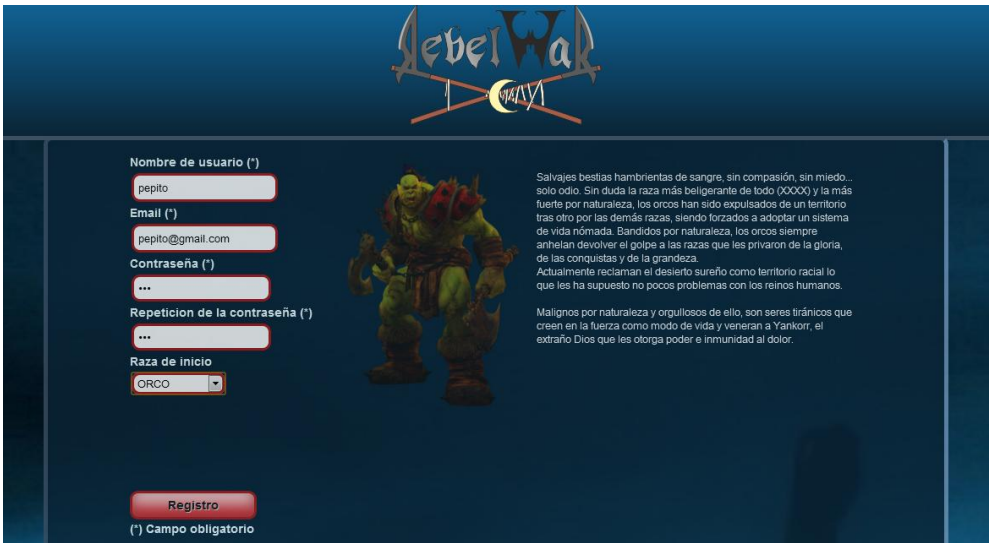


Figura 7.B.2.: Pantalla de registro de un nuevo usuario

¡Enhorabuena! Ya eres un usuario de *Rebel War*. Para poder empezar a jugar nada más registrarte, *Rebel War* te regala un sobre de veinte cartas de la raza que hayas seleccionado. Podrás verlas en cuanto pulses el botón “Registro” y se complete tu registro en el juego. Esas veinte cartas se añaden a tu colección y además se crea un mazo llamado “Inicial” con ellas, para que puedas empezar a jugar sin necesidad de más preparativos por tu parte. Si quieres saber más sobre tus cartas pasa el ratón por encima de ellas y podrás ver un zoom de las mismas a la izquierda de la pantalla.

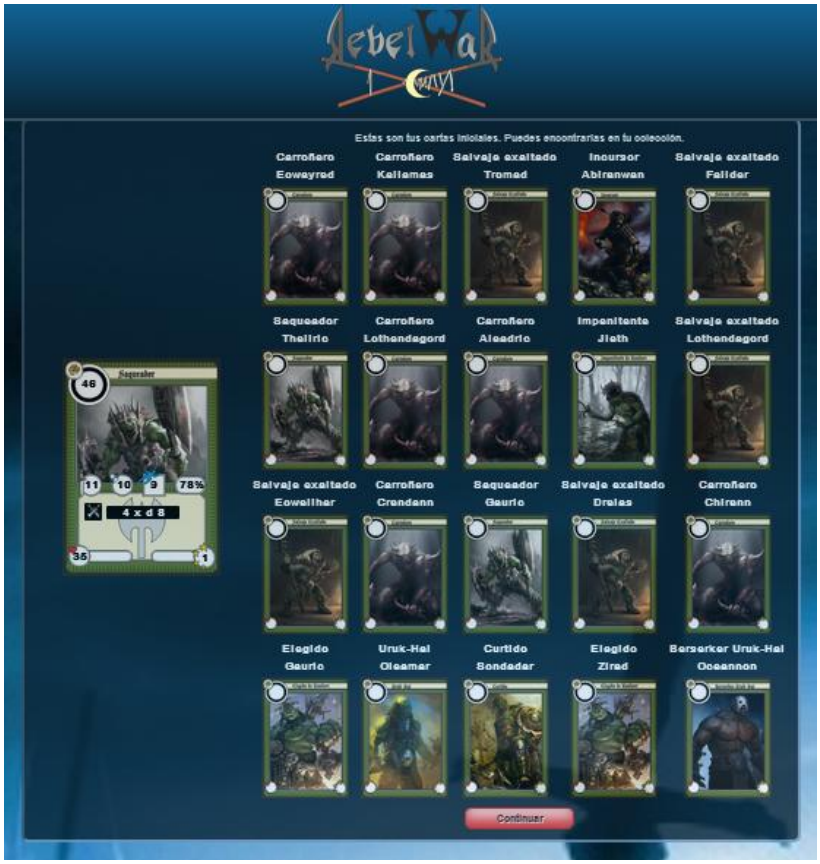


Figura 7.B.3.: Pantalla de sobre inicial

¿Ya has terminado de ver tu sobre de inicio? Entonces haz clic en el botón “Continuar” y podrás empezar a disfrutar del juego.

Aunque ya tienes cartas suficientes y un mazo preparado para combatir, es conveniente que antes de empezar a combatir te familiarices un poco con la interfaz de usuario y con las otras secciones del juego.

Figura 7.B.4.: Pantalla de home de usuario

Primero vamos con la interfaz de usuario. En la parte izquierda de la pantalla puedes ver algunos datos sobre tu usuario y su estatus en el juego: tu nombre de usuario, tu experiencia acumulada, tu nivel y las monedas de oro de las que dispones. La experiencia la irás consiguiendo combatiendo contra otros usuarios y con ella irás subiendo de nivel. Las monedas de oro también las conseguirás combatiendo, pero, además, puedes conseguir más dinero vendiendo cartas. La parte derecha de la pantalla está a su vez dividida en dos partes: superior e inferior. En la superior se encuentran los botones que dan acceso a las secciones principales de *Rebel War* y en la inferior irán apareciendo los contenidos de dichas secciones según navegues por la página.

Ahora, vamos con las secciones. Como puedes ver en la pantalla “Home”, el juego tiene 4 secciones principales: Combates, Colección, Mercado y Perfil.

7.2.2. Ver mi colección

Figura 7.B.5.: Pantalla de colección

Al hacer clic en el botón “Colección” entrarás en la sección con el mismo nombre. Allí podrás ver las cartas de tu colección y de tus mazos; crear, editar y eliminar mazos y ver las cartas que tienes en la enfermería y el cementerio.

Esta sección tiene 3 subsecciones: Colección, Enfermería y Cementerio. En la primera de ellas podrás ver tus cartas, para ello selecciona el mazo (también puedes seleccionar la colección completa) que quieras ver en el combobox de la esquina superior derecha. Las cartas pertenecientes al mazo seleccionado se mostrarán en una tabla en el centro de la pantalla. A la izquierda de los atributos de cada carta aparece una pequeña imagen de la misma. Si quieres ver la carta en grande con todos sus detalles pasa el ratón por encima y la carta en cuestión aparecerá a la izquierda de la pantalla, debajo de tus datos.

Figura 7.B.6.: Pantalla de colección mostrando la colección completa

A la derecha de las cartas hay unos botones que te permiten gestionar tu colección. Primero aparecen las acciones que puedes realizar sobre las cartas y después las que puedes realizar sobre los mazos. A continuación tienes una pequeña explicación sobre cada una de ellas, aunque algunas se explicarán con más detenimiento más adelante.

Añadir a un mazo: Añade las cartas seleccionadas a un mazo.

Quitar del mazo: Quita las cartas seleccionadas del mazo que estés visualizando. Recuerda que no puedes quitar cartas de la colección completa.

Subir de nivel: Sube de nivel la carta seleccionada. Las cartas que pueden subir aparecerán en color verde como se indica en la leyenda de la esquina inferior derecha. Para subir de nivel las cartas debes seleccionarlas de una en una.

Vender: Pone en venta la carta o las cartas seleccionadas. Al poner alguna carta en venta no podrás usarla en tus mazos para combatir, así que es posible que tengas que retocar los mazos que contenían esas cartas antes de poder volver a jugar.

Cambiar mazo activo: El mazo activo es el mazo que utilizas para combatir. Haciendo clic en este botón cambias el mazo que tenías activo por otro de tu colección.

Eliminar mazo: Elimina uno de tus mazos. Recuerda que no puedes eliminar el mazo que tienes activo.

Crear nuevo mazo: Crea un mazo vacío con el nombre que tú quieras. Recuerda que es un mazo vacío, para poder combatir con él tendrás que añadirle cartas.

Comprobar mazos: Comprueba si tus mazos son válidos para combatir. Para saber más sobre la composición de los mazos consulta las reglas del juego o haz clic en el botón “Restricciones mazos”.

Restricciones mazos: Te da información sobre las características de un mazo válido para combatir.

En la segunda subsección, “Enfermería”, podrás ver las cartas que han resultado heridas en combate y, cuando cumplan su convalecencia, curarlas.

Para curar una carta solo tienes que hacer clic en el botón “Curar” y la carta volverá a estar disponible para combatir. Las cartas heridas no pueden pertenecer a tu mazo activo, así que cuando cures una carta recuerda volver a incluirla en tu mazo si quieres usarla.

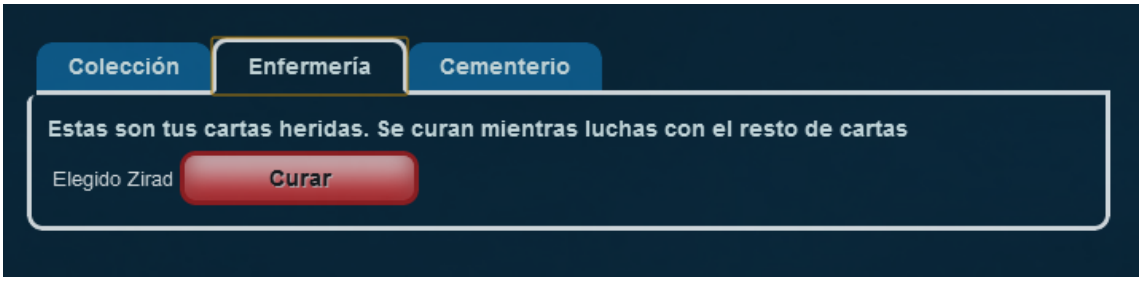


Figura 7.B.7.: Pantalla de enfermería

En la tercera y última subsección, “Cementerio”, podrás ver las cartas que han muerto en combate. Es muy raro que una carta muera en combate, pero puede pasar y si pasa la verás aparecer en el cementerio. Las cartas muertas no pueden recuperarse, pero puedes eliminarlas para siempre de tu colección si quieres haciendo clic en el botón “Eliminar”.

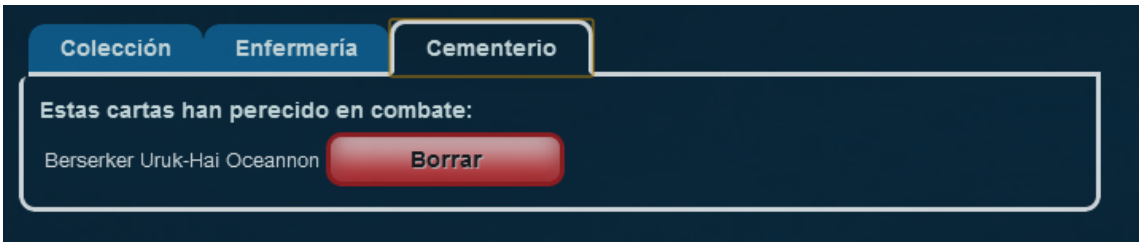


Figura 7.B.8.: Pantalla de cementerio

7.2.3. Mi primera batalla

Ahora que has visto tus cartas y te has familiarizado un poco con *Rebel War* es hora de jugar tu primera batalla. No tienes que hacer nada más que hacer clic en el botón “Combates”, recuerda que al registrarte se creó un mazo activo válido para combatir.

Cuando entras en la sección “Combates” ves la lista de salas disponibles para combatir. En todas ellas lucharás contra otro usuario que el juego seleccionará aleatoriamente.

Según tu nivel podrás combatir en unas salas, para que la lucha entre dos jugadores sea equilibrada. La primera vez solo podrás jugar en “Sala Tutorial”, así que haz clic en el botón con las espadas cruzadas que se encuentra a la derecha de la descripción de dicha sala y prepárate para tu primera batalla.

A continuación se muestran los jugadores que están jugando actualmente en la sala. No te preocupes por ellos, haz clic en el botón “Jugar” y el juego te asignará un contrincante. Cuando esto pase se te informará con un popup, pulsa “Aceptar” y comenzará el combate.

Figura 7.B.9.: Pantalla de selección de cartas

Como se explica detenidamente en las reglas del juego, los combates tienen 4 fases. Las 3 primeras son de preparación, tendrás un minuto para llevar a cabo cada una de ellas, y la última es el combate en sí. En la primera fase el juego seleccionará unas cartas de tu mazo activo y del mazo activo de tu contrincante, éstas serán las cartas que participen en la batalla. En la pantalla de “Selección de cartas” puedes ver tus cartas seleccionadas, en la parte de abajo, y los datos y una muestra de las cartas de tu contrincante, en la parte de arriba. Haz clic en el botón “Continuar” para pasar a la siguiente fase.

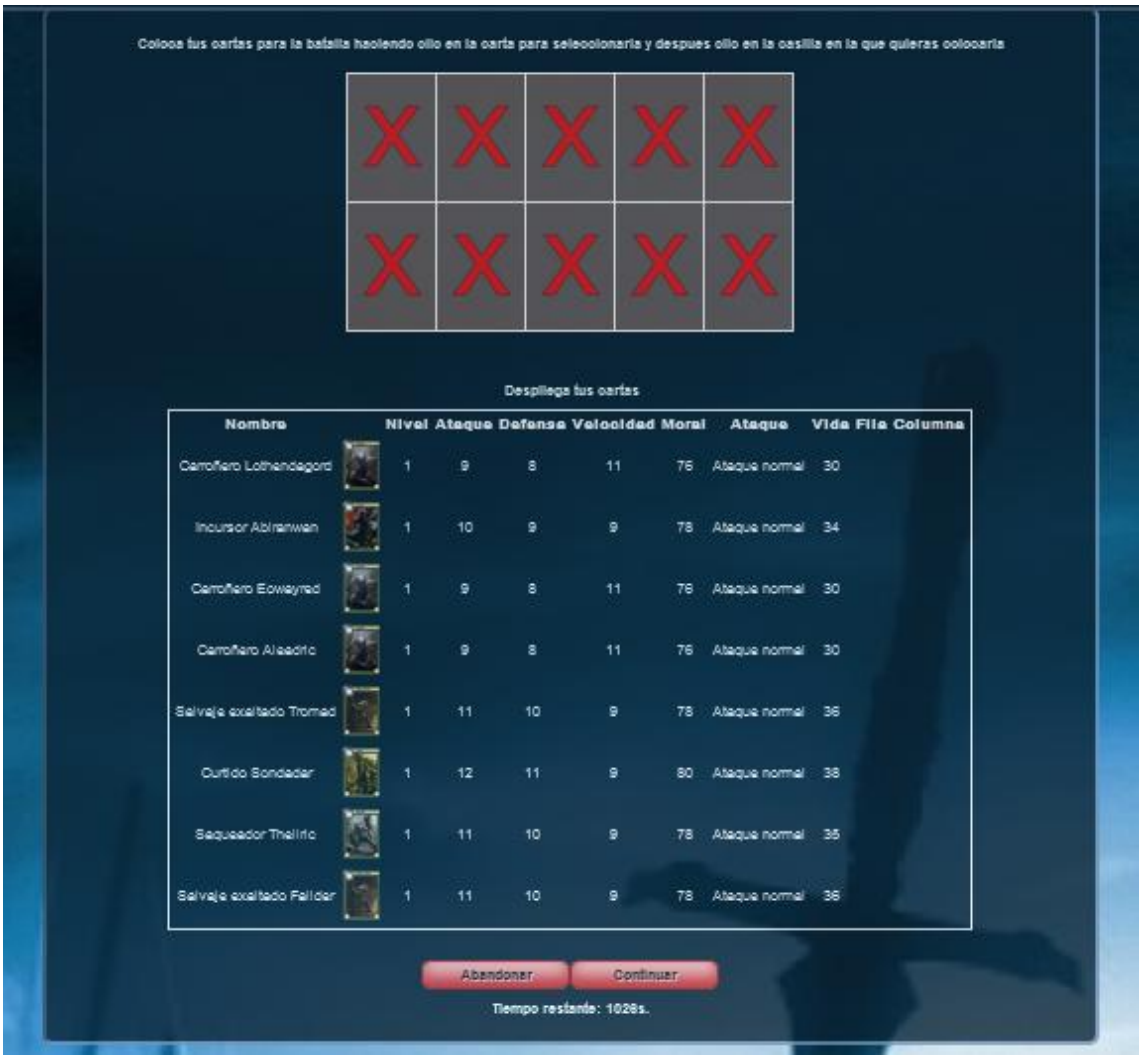


Figura 7.B.10.: Pantalla de despliegue de tropas

La siguiente pantalla se corresponde con la segunda fase de juego, “Despliegue de tropas”, aquí tendrás que colocar tus cartas seleccionadas en el tablero. En la parte superior de la pantalla aparece el tablero y en la inferior una lista de tus cartas acompañadas por una pequeña imagen de cada una de ellas. Para colocar una carta en el tablero haz clic sobre su imagen y después haz clic sobre la casilla donde quieras colocarla. No puedes colocar la misma carta en más de una casilla diferente. Si te equivocas y quieres quitar una carta, solo tienes que hacer clic en ella en el tablero. Cuando acabes haz clic en el botón “Continuar”.

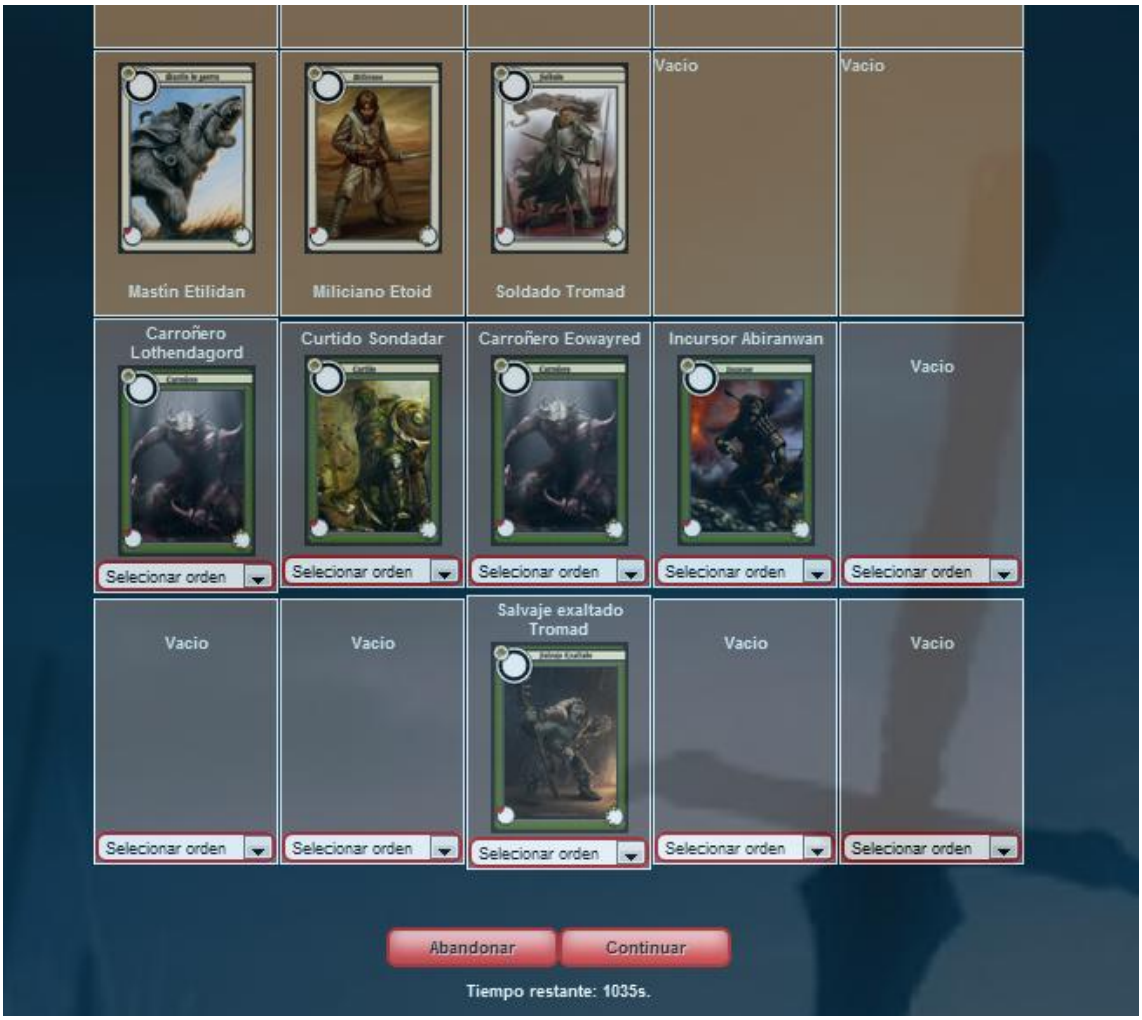


Figura 7.B.11.: Pantalla de adjudicación de órdenes

La última fase es la de “Adjudicación de órdenes”. En la siguiente pantalla podrás ver el tablero completo, en la parte de arriba aparecen las cartas de tu contrincante tal y como las ha colocado y en la parte de abajo aparecen las tuyas también colocadas. Debajo de tus cartas hay un combobox en que tendrás que seleccionar la orden que quieras darle a cada carta. Cuando hayas dado todas las órdenes pulsa “Continuar”.

Cuando tu contrincante y tú hayáis terminado de adjudicar las órdenes aparecerá un popup para informarte de que la batalla va a dar comienzo. Pulsa “¡A luchar!” y en unos segundos podrás ver el resultado del combate.

Figura 7.B.12.: Pantalla de resultado

En la pantalla de resultado aparece el resultado final de la batalla y los puntos conseguidos por cada jugador. Si quieres más datos sobre cómo ha ido el combate haz clic en el botón “Informe de batalla”.

Figura 7.B.13.: Parte superior de la pantalla de informe de batalla

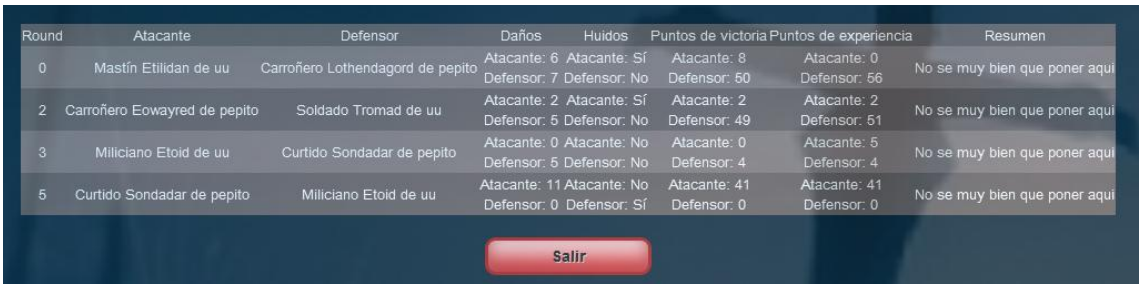


Figura 7.B.14.: Parte inferior de la pantalla de informe de batalla

En la pantalla de “Informe de batalla” podrás ver el tablero completo y, debajo de éste, una lista con los datos de todas las luchas que han compuesto la batalla. Para más información sobre cómo interpretar los datos puedes consultar las reglas del juego.

Ya has completado tu primera batalla y, con ella, habrás ganado experiencia y monedas de oro. Para subir de nivel y poder ganar más monedas de oro sigue combatiendo.

7.2.4. Crear y editar un mazo

Para crear un nuevo mazo:

1. Haz clic en el botón “Colección” para ir a la sección de la colección.
2. Haz clic en el botón “Crear nuevo mazo”.
3. Aparecerá un popup en el que tendrás que poner nombre al nuevo mazo que se va a crear. Cuando lo tengas haz clic en el botón “Aceptar” y se habrá creado un mazo vacío con ese nombre.

Ahora tienes que añadirle cartas, para saber cómo crear un mazo válido haz clic en “Restricciones mazos”:

1. Selecciona “Colección completa” en el combobox de la esquina superior derecha de la pantalla para que se muestren todas las cartas de tu colección
2. Selecciona las cartas que quieras añadir al nuevo mazo y haz clic en “Añadir a un mazo”.
3. En el popup emergente selecciona el mazo recién creado y haz clic en “Aceptar”. Si las cartas que has elegido forman un mazo válido, tendrás un mazo nuevo con el que podrás combatir.

7.2.5. Cambiar el mazo activo

Para cambiar el mazo con el que quieres combatir:

1. Ve a la Colección haciendo clic en el botón “Colección”.
2. Haz clic en “Cambiar mazo activo”.
3. Selecciona el mazo que quieras poner como activo en el popup emergente y pulsa “Aceptar”.

7.2.6. Eliminar un mazo

Si te has cansado de uno de tus mazos puedes eliminarlo para siempre. Solo se eliminará el mazo, las cartas seguirán formando parte de tu colección.

1. Haz clic en el botón “Colección” para ir a la colección.
2. Haz clic en el botón “Eliminar mazo”.
3. Selecciona el mazo que quieres eliminar en el popup emergente y pulsa “Aceptar”.

7.2.7. Vender mis cartas

Para conseguir más dinero puedes poner tus cartas en venta, pero recuerda que si las pones en venta no podrás usarlas para combatir. Cuando otro usuario compre tu carta, el coste de la misma se sumará al total de tus monedas de oro.

1. Haz en el botón “Colección” para ir a tu colección.
2. Despliegua la colección completa seleccionando “Colección completa” en el combobox de la esquina superior derecha.
3. Selecciona las cartas que quieras vender.
4. Haz clic en el botón “Vender” y pulsa “Aceptar”.

7.2.8. Mercado

Figura 7.B.15.: Pantalla de mercado de cartas

En el mercado puedes comprar cartas de otros usuarios o sobres. También puedes recuperar una carta que hayas puesto en venta si cambias de opinión. Para ir al mercado haz clic en el botón “Mercado”.

Comprar cartas de otro usuario o recuperar tus cartas puestas en venta: En la pestaña “Mercado de cartas” puedes comprar las cartas que han puesto en venta otros usuarios o recuperar las que pusiste tú en venta. Solo tienes que hacer clic en el botón “Comprar” o “Recuperar” respectivamente.

Figura 7.B.16.: Pantalla de compra de sobres

Comprar un sobre de cartas: Otra forma de comprar cartas es comprar un sobre. Cada sobre contiene 3 cartas de la raza que elijas y cuesta 100 monedas de oro. Para comprar un sobre entra en la pestaña "Comprar sobres" y haz clic sobre la imagen del sobre de la raza que quieras. Las cartas que te toquen en el sobre se añadirán directamente a tu colección.

En el mercado también puedes ver un histórico de tus compras y ventas de cartas (pestaña "Tus compras y ventas") y un histórico de todas las ventas y compras de cartas de todos los usuarios (pestaña "Historial"), que puede darte una idea de las cartas más compradas.

7.2.9. Perfil: Visualizar y actualizar tus datos

Haciendo clic en el botón "Perfil" entrarás en la sección con el mismo nombre. En está sección podrás consultar tus estadísticas y tus datos y cambiar algunos datos de tu perfil, como por ejemplo la contraseña.

Estadísticas: Al entrar en la sección "Perfil" o hacer clic en la pestaña "Estadísticas" podrás ver tus estadísticas de combate: número total de combates ganados, perdidos, empatados y abandonados y una información más detallada sobre cada uno de ellos (datos de tu contrincante, tus puntos y los suyos y la experiencia y el oro que ganaste).

Datos: Al hacer clic en la pestaña "Datos" podrás ver tus datos de usuario en el juego: nombre, nivel, experiencia acumulada y email.

Configuración: Al hacer clic en la pestaña "Configuración" podrás actualizar tus datos de usuario en el juego. Rellena los campos de los datos que quieras modificar y pulsa el botón "Guardar cambios". Si lo que quieres es cambiar tu contraseña es necesario que rellenes los campos "Contraseña actual", "Nueva contraseña" y "Repetición de la nueva contraseña".

7.2.10. Más ayuda

Figura 7.B.17.: Panel del perfil de usuario

Si tienes más dudas puedes consultar la ayuda de *Rebel War* haciendo clic en el botón con la interrogación que se encuentra a la izquierda de la pantalla encima de tu nombre.

Referencias

Adobe Flash. (2012). Obtenido de http://www.adobe.com/products/flash.html

Apache Struts. (22 de Junio de 2012). Obtenido de http://struts.apache.org/

Google Developers Google Web toolkit Tutorial Overview. (23 de Marzo de 2012). Obtenido de https://developers.google.com/web-toolkit/doc/latest/tutorial/index

Google Web Toolkit. (23 de Marzo de 2012). Obtenido de https://developers.google.com/web-toolkit/

Magic: The Gathering. (2012). Obtenido de http://www.wizards.com/Magic/Summoner/

MySQL. (2012). Obtenido de http://dev.mysql.com/downloads/

Oracle How to write Doc Comments for the Javadoc Tool. (2012). Obtenido de http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

PHP. (26 de Mayo de 2012). Obtenido de http://www.php.net/

Schwaber, K. a. (2001). *Agile software development with Scrum.* Prentice Hall.

Wikipedia Google Web Toolkit. (29 de Mayo de 2012). Obtenido de http://es.wikipedia.org/wiki/Google_Web_Toolkit

Wikipedia HTML. (23 de Junio de 2012). Obtenido de http://es.wikipedia.org/wiki/HTML

Wikipedia JavaEE. (30 de Abril de 2012). Obtenido de http://es.wikipedia.org/wiki/Java_EE

Wikipedia JavaScript. (2012). Obtenido de http://es.wikipedia.org/wiki/JavaScript

Wikipedia MySQL. (18 de Mayo de 2012). Obtenido de http://es.wikipedia.org/wiki/Mysql

Wikipedia Scrum. (8 de Junio de 2012). Obtenido de http://es.wikipedia.org/wiki/Scrum

Wikipedia Subversión (Software). (25 de Mayo de 2012). Obtenido de http://es.wikipedia.org/wiki/Subversion

Wikipedia Tomcat. (6 de Mayo de 2012). Obtenido de http://es.wikipedia.org/wiki/Tomcat

Wikipedia Wizards of the Coast. (22 de Abril de 2012). Obtenido de http://en.wikipedia.org/wiki/Wizards_of_the_Coast