Proyecto 2: Juego de Cartas Online Inspirado en Hearthstone

Modelado y Programación
profesora: Rosa Victoria Villa Padilla

Emilio Durán Tapia, 320301867
Oscar David Hernández Rodríguez, 420002945

28 de octubre, 2024

Problemática

Los juegos de cartas en línea han captado la atención de jugadores en todo el mundo por su capacidad para ofrecer experiencias de estrategia profunda, con interacciones que requieren habilidad, planificación y adaptabilidad en cada enfrentamiento. Sin embargo, diseñar e implementar un juego de este tipo conlleva varios retos técnicos. Esto exige un sistema robusto de sincronización que mantenga la coherencia entre dispositivos y asegura una experiencia sin interrupciones, independientemente de la ubicación de los jugadores.

Objetivo: El objetivo principal es diseñar e implementar un juego de cartas en línea que combine las estrategias de Hearthstone con el universo de personajes y habilidades icónicas de animes como Hunter x Hunter y Dragon Ball. Este juego permitirá a los jugadores sumergirse en el rol de personajes legendarios como Gon, Killua, Hisoka y Goku, utilizando cartas que capturan fielmente sus habilidades y estilos de combate. Cada carta y combinación de habilidades está diseñada para recrear las dinámicas de estos personajes en intensos enfrentamientos estratégicos, brindando una experiencia única.

Alcance del Proyecto

El sistema permitirá:

- Crear cartas predefinidas como criaturas, hechizos y trampas mediante los patrones **Abstract Factory** y **Builder**.
- Administrar partidas en tiempo real utilizando **websockets** para la comunicación entre jugadores.
- Definir un sistema de turnos claro y bien estructurado, inspirado en Hearthstone.
- Implementar estrategias dinámicas mediante el patrón **Strategy**.
- Permitir personalización y ampliación de habilidades usando el patrón **Decorator**.

- Mantener una arquitectura que soporte futuras expansiones del juego sin complicaciones técnicas.

Reglas y Estados del Juego

Cada partida comienza con dos jugadores que controlan a **Hunters**, cada uno con 30 puntos de vida. El objetivo es reducir los puntos de vida del oponente a 0 utilizando cartas que representan criaturas, hechizos y trampas. La partida se organiza en turnos alternos y se rige por las siguientes reglas:

1. Fase Inicial

- El jugador que comienza recibe 3 cartas, con la opción de descartar y robar nuevas cartas una vez.
- El segundo jugador recibe 4 cartas y tiene la misma posibilidad de descartar hasta 3 cartas.

2. Turnos

Cada jugador tiene varias opciones durante su turno:

- Jugar una carta desde su mano, pagando su costo correspondiente.
- Atacar con las criaturas en juego.
- Pasar el turno al oponente utilizando un botón o comando para indicar el fin de su turno.

El turno finaliza cuando el jugador decide no realizar más acciones o cuando agota las acciones disponibles.

3. Daño y Puntos de Vida

- Cada criatura tiene puntos de ataque y vida. El daño infligido se resta de la vida del oponente o de la criatura objetivo.
- Si los puntos de vida del **Hunter** llegan a 0, su oponente gana la partida.

4. Aura y Costos

- Cada jugador comienza con un espacio de aura limitado, que aumenta con cada turno. El aura es necesaria para jugar cartas más poderosas.
- Las cartas tienen un costo de aura específico que debe ser pagado al jugarlas.
- **Card**: Clase abstracta que representa una carta genérica. Contiene atributos como descripción, costo de nen y rareza.
- **CardDecorator**: Clase abstracta que extiende **Card** y actúa como base para los decoradores de cartas.

- **BasicMinionCard**: Decorador concreto que extiende **CardDecorator** y añade atributos de ataque y defensa a una carta.
- **SpellCard**: Clase concreta que extiende **Card** y representa una carta de hechizo.
- **WeaponCard**: Clase concreta que extiende **Card** y representa una carta de arma.
- **TauntDecorator**: Decorador concreto que extiende **CardDecorator** y añade la habilidad de provocación a una carta.
- **BattlecryDecorator**: Decorador concreto que extiende **CardDecorator** y añade la habilidad de grito de batalla a una carta.
- **ChargeDecorator**: Decorador concreto que extiende **CardDecorator** y añade la habilidad de carga a una carta.

La clase **Deck** se encarga de cargar las cartas desde un archivo JSON y aplicar los decoradores correspondientes según los efectos especificados en el archivo. Esto permite una gran flexibilidad y extensibilidad en la creación de nuevas cartas y habilidades.

Implementación de Websockets

La comunicación en tiempo real entre los jugadores se gestiona mediante **websockets**. Esto asegura que las acciones de un jugador se reflejen inmediatamente en la interfaz del otro jugador, proporcionando una experiencia de juego fluida y dinámica. La implementación de websockets se realiza utilizando Spring Boot, que facilita la configuración y gestión de las conexiones.

- **WebSocketConfig**: Clase de configuración que habilita y configura los websockets en la aplicación.
- **GameController**: Controlador que maneja las conexiones websocket y las interacciones en tiempo real entre los jugadores.
- **GameService**: Servicio que contiene la lógica del juego y se comunica con el controlador para actualizar el estado del juego y enviar mensajes a los jugadores.

Conclusión

Este proyecto busca desarrollar un sistema de juego de cartas en línea que combine estrategias complejas con una experiencia amigable para los jugadores. La integración de patrones de diseño proporcionará una arquitectura clara, extensible y fácil de mantener. El uso de **websockets** asegurará partidas fluidas y dinámicas, ofreciendo una experiencia atractiva y competitiva.

Pasos de ejecución

1. **Compilar el Servidor**:

```
mvn package -Pservidor
```

2. **Renombrar el Archivo JAR del Servidor**:

```
mv target/Proyecto02_SpaceCraft-1.0-SNAPSHOT.jar

target/ServidorJuego-1.0-SNAPSHOT.jar
```

3. **Compilar el Cliente**:

```
mvn package -Pcliente
```

4. **Renombrar el Archivo JAR del Cliente**:

```
mv target/Proyecto02_SpaceCraft-1.0-SNAPSHOT.jar

target/ClienteJuego-1.0-SNAPSHOT.jar
```

5. **Verificar la Ubicación de los Archivos JAR**:

```
ls target
```

Deberías ver algo como esto:

```
ClienteJuego-1.0-SNAPSHOT.jar
ServidorJuego-1.0-SNAPSHOT.jar
```

6. **Ejecutar el Servidor**:

```
java -jar target/ServidorJuego-1.0-SNAPSHOT.jar
```

7. **Ejecutar el Cliente**:

```
java -jar target/ClienteJuego-1.0-SNAPSHOT.jar
```

Scripts de Automatización

Script para Windows (build_and_run.bat)

```
@echo off
REM Compilar el Servidor

mvn package -Pservidor
move target\Proyecto02_SpaceCraft-1.0-SNAPSHOT.jar

target\ServidorJuego-1.0-SNAPSHOT.jar

REM Compilar el Cliente

mvn package -Pcliente
move target\Proyecto02_SpaceCraft-1.0-SNAPSHOT.jar

target\ClienteJuego-1.0-SNAPSHOT.jar
```

```
REM Verificar la Ubicación de los Archivos JAR

dir target

REM Ejecutar el Servidor

start cmd /k "java -jar target\ServidorJuego-1.0-SNAPSHOT.jar"

REM Ejecutar el Cliente

start cmd /k "java -jar target\ClienteJuego-1.0-SNAPSHOT.jar"
```