Especificación de Requisitos de Software (ERS) y Descripción del Diseño

Proyecto: Oh Hell! - Juego de Cartas Multijugador

Equipo: Caos controlado **Versión del Documento:** 1.0 **Fecha:**12/12/2025

---

1. Introducción

1.1 Propósito

El propósito de este documento es describir la arquitectura técnica y el diseño detallado del sistema “Oh Hell!”. Este documento actúa como el “anteproyecto” que unifica la visión estructural y de comportamiento del software implementado, sirviendo de guía para el mantenimiento y la evolución futura del sistema.

1.2 Alcance

El sistema implementa una versión web multijugador del juego de cartas, utilizando una arquitectura Cliente-Servidor basada en estándares empresariales de Java.

1.3 Referencias a Requisitos

Este documento de diseño satisface las especificaciones definidas en los siguientes documentos del repositorio del proyecto:

- **Especificación de Requisitos Funcionales (ERF):** Docs/Requerimientos_Funcionales.md
- **Especificación de Requisitos Técnicos (ERT):** Docs/Requerimientos_Técnicos.md

---

2. Diseño Arquitectónico

2.1 Visión general

El sistema sigue un patrón arquitectónico en capas sobre el modelo MVC desacoplado, donde el Frontend actúa como Vista y el Backend expone una API REST que actúa como Controlador.

2.2 Descomposición en Subsistemas

Siguiendo el principio de diseño arquitectónico, el sistema se organiza en:

1. **Capa de presentación (Frontend):**

- Responsable de la interacción con el usuario y la visualización del estado del juego.
- Tecnologías: HTML5, CSS3, JavaScript (Vanilla).

2. **Capa de servicios (Backend - API):**

- Expone los recursos del sistema a través de endpoints HTTP.
- Tecnologías: Jakarta EE 10 (JAX-RS).

3. **Capa de Lógica de Negocio (Backend - Core):**

- Implementa las reglas del juego (validación de cartas, cálculo de puntuación, gestión de turnos).
- Tecnologías: CDI (Context and Dependency Injection).

4. **Capa de Persistencia (Data):**

- Gestiona el almacenamiento y recuperación de datos.
- Tecnologías: JPA / JDBC, PostgreSQL.

3. Diseño Estructural (Vista Estática)

Esta sección describe la estructura estática del sistema utilizando diagramas de clases y paquetes.

3.1 Diagrama de Paquetes

La organización del código fuente en el backend sigue la siguiente estructura jerárquica.

```
com.ohhell.ohhellapi
├── resources       (Capa API: Controladores REST JAX-RS)
├── services        (Capa Lógica: Reglas del juego y gestión de estado)
├── models          (Capa de Datos: Entidades JPA y DTOs)
│   ├── entities    (Mapeo directo a BBDD: Player, Game, Round)
│   └── dtos        (Objetos de transferencia de datos)
├── dao             (Capa de Acceso a Datos: Repositorios)
└── utils           (Utilidades: Algoritmos de barajado, constantes)
```

3.2 Modelo de Clases (Entidades principales)

Las clases principales y sus relaciones (asociaciones y composiciones) son:

- **Game:** Entidad raíz que agrega **Players** y **Rounds.**
  - Relación: Un “Game” se compone de muchas “Round”.

- **Player:** Representa al usuario.
  - Relación: Un “Player” realiza muchos “Bid” (Apuestas) y juega muchas “Card” en los “Tricks”.

- **Round:** Representa una mano de cartas.
  - Relación: Una “Round” contiene 13 “Tricks” (Bazas) como máximo.

- **Card:** Objeto valor que representa una carta física (Palo y Valor).

4. Diseño de Comportamiento (Vista Dinámica)

Esta sección describe cómo interactúan los objetos y cómo cambia el sistema a lo largo del tiempo.

4.1 Máquina de Estados de la Partida (Game Lifecycle)

El objeto “Game” posee un ciclo de vida complejo modelado mediante una máquina de estado:

1. **WAITING (Estado Inicial):** La partida ha sido creada, esperando jugadores.
    - Transición: startGame() [numPlayers >= 3] -> pasa a IN_PROGRESS
2. **IN_PROGRESS:** La partida está activa.
    - Sub-estados (Rondas):
        - **BETTING:** Los jugadores realizan sus apuestas.
        - **PLAYING:** Los jugadores juegan sus cartas (bazas).
        - **SCORING:** Se calculan los puntos al final de la ronda.
3. **FINISHED (Estado Final):** Se han jugado todas las rondas, se declara el ganador.

---

5. Diseño de Despliegue (Infraestructura)

Este apartado describe la disposición física del software en el hardware.

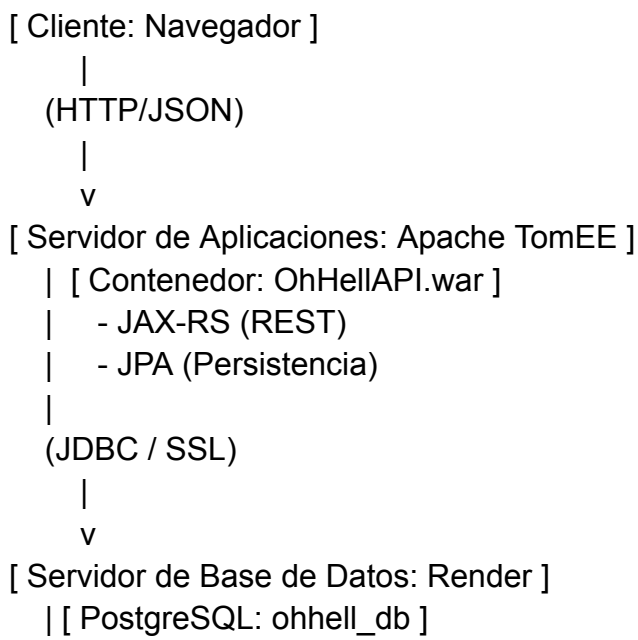
5.1 Nodos y Artefactos

El diagrama de despliegue se compone de los siguientes nodos:

- **Nodo Cliente:** Navegador Web (Chrome/Firefox).
    - Artefactos: Archivos estáticos HTML/JS/CSS.
- **Nodo Servidor de Aplicaciones:** Apache TomEE WebProfile.
    - Entorno de Ejecución: JVM (Java 21).
    - Artefacto Desplegado: OhHellAPI.wa (Web Archive).

- **Nodo Servidor de Datos:** Instancia de PostgreSQL alojada en render.
    - Protocolo: Conexión JDBC sobre TCP/IP (SSL habilitado).

5.2 Esquema de Despliegue



---

6. Consideraciones Finales de Diseño

6.1 patrones Utilizados

- **DTO (Data Transfer Object):** Para desacoplar las entidades internas de la base de datos de la respuesta JSON enviada al cliente.

- **DAO (Data Access Object):** Para centralizar las consultas SQL/JPQL y separar la lógica de acceso a datos de la lógica de negocio.

- **Singleton:** Utilizado en la gestión de configuración global del jeugo.

6.2 Seguridad

El diseño implementa seguridad mediante:

- Validación de entradas en el Backend (evitando confiar solo en el Frontend)
- Gestión de sesiones mediante cookies/tokens para identificar al jugador en cada petición REST.