TEST DESARROLLADOR BACKEND

SIMULACIÓN TORNEO DE TENIS

Valdivia Ignacio Ariel

Febrero 2025

Contenido

Consigna del Test

Se desea modelar el comportamiento de un torneo de tenis.

- La modalidad del torneo es por **eliminación directa**.
- Se puede asumir que la cantidad de jugadores es una **potencia de 2**.
- El torneo puede ser **Femenino o Masculino**.
- Cada jugador tiene un **nombre** y un **nivel de habilidad** (valor entre 0 y 100).
- En un enfrentamiento entre dos jugadores influyen el **nivel de habilidad** y la **suerte** para decidir al ganador. La suerte se define como el usuario lo desee en su diseño.
- En el **torneo masculino**, se deben considerar la **fuerza** y la **velocidad de desplazamiento** como parámetros adicionales para calcular el ganador.
- En el **torneo femenino**, se debe considerar el **tiempo de reacción** como parámetro adicional para calcular el ganador.
- **No existen empates**.
- Se requiere que, a partir de una lista de jugadores, se **simule el torneo** y se obtenga como resultado **el ganador del mismo**.
- Se recomienda realizar la solución en su IDE preferido.
- Se valorarán las **buenas prácticas de Programación Orientada a Objetos (POO)**.
- Se puede definir por parte del usuario cualquier **cuestión adicional** que considere no aclarada en la consigna.
- Se pueden agregar las **aclaraciones** que se consideren necesarias en la entrega del ejercicio.
- **Cualquier extra que aporte será bienvenido.**
- Se prefiere el **modelado en capas** o el uso de **arquitecturas limpias (Clean Architecture)**.
- Se prefiere la entrega del código en un **sistema de versionado** como GitHub, GitLab o Bitbucket.

Nota sobre eliminación directa:
El sistema de eliminación directa implica que el perdedor de cada enfrentamiento es eliminado del torneo, mientras que el ganador avanza a la siguiente fase. El proceso continúa hasta que solo queda un **campeón**.

Puntos extra (Opcionales)

1. **Testing de la solución** (Unit Test).
2. **API REST** (Swagger + Integration Testing):
    - Dado una lista de jugadores, retorna el resultado del torneo.
    - Permite consultar el resultado de torneos finalizados exitosamente, con algunos filtros como:
        - Torneo Masculino y/o Femenino.
        - Otros criterios a definir.
3. **Uso de una base de datos** (en lugar de datos embebidos).
4. **Subir el código a un repositorio en GitHub/GitLab**.
5. **Desplegar la solución en un servicio como AWS/Azure**, usando **Docker o Kubernetes**.

1. Introducción

Este proyecto implementa una API para la simulación de torneos de tenis bajo la modalidad de eliminación directa. Se desarrolló utilizando **Laravel 11**, con una arquitectura basada en capas, implementando buenas prácticas de programación orientada a objetos (POO) y aplicando los principios de **SOLID**.

La solución permite registrar jugadores, iniciar torneos, simular enfrentamientos y determinar un ganador final. Se desarrolló una **API REST** documentada con **Swagger**, soportando el almacenamiento en base de datos mediante **MySQL**, con despliegue en **Digital Ocean utilizando Docker**.

IgnaValdivia/TestTorneoTenis: Simulación del comportamiento de un torneo de tenis

2. Planteamiento del Problema

El objetivo es modelar un torneo de tenis con las siguientes características:

- Modalidad de **eliminación directa**.
- **Jugadores en potencias de 2**.
- **Diferencias entre torneos masculinos y femeninos**:
  - Masculino: habilidad, fuerza y velocidad.
  - Femenino: habilidad y tiempo de reacción.
- **Sistema de puntuación** basado en habilidades y un factor aleatorio de suerte.
- **No se permiten empates**.

Se busca desarrollar una **API REST** que permita gestionar los torneos y jugadores, realizar simulaciones de partidas, consultar torneos finalizados, entre otros.

3. Modelo de Objetos

IgnaValdivia/TestTorneoTenis at modelos

El sistema sigue una arquitectura basada en el principio de responsabilidad única y separación de preocupaciones. Los modelos principales son:

3.1. Jugador

Representa a un participante en el torneo. Los jugadores pueden ser de género Masculino o Femenino y tienen atributos generales como:

- **nombre**
- **dni**
- **genero**
- **habilidad**

Los jugadores pueden participar en torneos y jugar partidas.

3.2. JugadorMasculino y JugadorFemenino

Para respetar la normalización de la base de datos, se decidió separar los atributos específicos de cada género:

- **JugadorMasculino** tiene atributos adicionales: **fuerza**, **velocidad**.
- **JugadorFemenino** tiene **reaccion** como atributo adicional.

Cada instancia de Jugador tiene una relación uno a uno con su tipo correspondiente (Masculino o Femenino).

3.3. Torneo

Un torneo agrupa jugadores bajo una categoría (Masculino o Femenino) y maneja su estado:

- **nombre**
- **tipo**
- **fecha**
- **estado (Pendiente, Finalizado)**
- **ganador_id (cuando finaliza el torneo)**

3.4. Partida

Cada enfrentamiento entre jugadores se modela como una Partida, que almacena:

- **torneo_id**
- **jugador1_id**
- **puntaje1**
- **jugador2_id**
- **puntaje2**
- **ganador_id**
- **ronda**

4. Relaciones entre Entidades

- Un **Torneo** puede tener muchos **Jugadores** mediante una relación de muchos a muchos (torneo_jugador).
- Un **Torneo** tiene muchas **Partidas** (uno a muchos).
- Cada **Partida** involucra dos **Jugadores**.
- Un **Jugador** pertenece a múltiples **torneos** y participa en múltiples **partidas**.

5. Arquitectura del Proyecto

El proyecto sigue una **arquitectura basada en capas**, con una organización que separa la lógica de negocio de la infraestructura. Se implementaron **Repositories**, **Services**, **DTOs**, y **Controllers** para estructurar el código de manera modular y reutilizable.

IgnaValdivia/TestTorneoTenis at controllers

IgnaValdivia/TestTorneoTenis at service

IgnaValdivia/TestTorneoTenis at repositories

5.1. Capas del Proyecto

- **Controladores (Controllers):**
  Manejan las solicitudes HTTP y delegan la lógica de negocio a los servicios.
  - **JugadorController**
  - **PartidaController**
  - **TorneoController**

- **Servicios (Services):**
  Contienen la lógica de negocio y reglas específicas del dominio.
  - **JugadorMasculinoService**
  - **JugadorFemeninoService**
  - **JugadorService**
  - **PartidaService**
  - **TorneoService**

- **Repositorios (Repositories):**
  Gestionan la persistencia de datos a través de **Eloquent ORM**.
  - **JugadorMasculinoRepository**
  - **JugadorFemeninoRepository**
  - **JugadorRepository**
  - **PartidaRepository**
  - **TorneoRepository**

- **Modelos (Models):**
  Representan las entidades de la base de datos con relaciones definidas en **Eloquent**.

- **DTOs (Data Transfer Objects):**
  Definen estructuras de datos para mejorar la separación entre las capas de la aplicación y formatean la salida de datos de manera estructurada para evitar exponer directamente los modelos.
  - **JugadorMasculinoDTO**
  - **JugadorFemeninoDTO**
  - **JugadorDTO**
  - **PartidaDTO**
  - **TorneoDTO**

5.2. Relaciones de los Modelos

- **Jugador**: Modelo base para los jugadores.
- **JugadorMasculino y JugadorFemenino**: Modelos específicos con atributos adicionales.
- **Torneo**: Relacionado con múltiples jugadores y múltiples partidas.
- **Partida**: Contiene dos jugadores y un ganador, pertenece a un torneo.

6. Estilo de Código

Se siguieron los principios de **Clean Code**:

- ✓ Aplicación de **SOLID**.
- ✓ **Separación de responsabilidades** en capas.
- ✓ **Uso de interfaces** para definir contratos en los servicios.
- ✓ Documentación con **Swagger** para describir la **API REST**.

7. Base de Datos

Se utilizó **MySQL** con el ORM **Eloquent** para la gestión de datos. La base de datos está normalizada, separando las entidades de jugadores según su género y creando relaciones adecuadas:

- **Torneos** → **Jugadores** (relación *muchos a muchos*).
- **Torneos** → **Partidas** (relación *uno a muchos*).
- **Partidas** → **Jugadores** (relación *uno a muchos* con el ganador).

8. API REST y Documentación

Se utilizó Swagger (L5-Swagger) para generar documentación interactiva de los endpoints. Los endpoints permiten:

- ✓ **Administrar (crear, listar, actualizar y eliminar) torneos**.
- ✓ **Administrar (crear, listar, actualizar y eliminar) jugadores y consultar sus partidas o torneos**.
- ✓ **Consultar partidas**
- ✓ **Simular el torneo y obtener al ganador**.

La documentación está disponible a través de /api/documentation. (L5 Swagger UI)

9. Testing

Se desarrollaron **pruebas unitarias y de integración** para garantizar el correcto funcionamiento del sistema.

IgnaValdivia/TestTorneoTenis at test_repositories

IgnaValdivia/TestTorneoTenis at test_service

IgnaValdivia/TestTorneoTenis at test_controllers

Se implementaron a distintos niveles:

9.1. Pruebas Unitarias

Se enfocaron en los **Repositories** para verificar que la manipulación de datos en la base de datos sea correcta.

9.2. Pruebas de Integración

Validaron el comportamiento de los **Services**, asegurando la correcta interacción entre los repositorios y la lógica de negocio.

9.3. Pruebas de API (End-to-End)

Se escribieron pruebas para cada endpoint de la **API**, asegurando que las respuestas sean correctas.

10. Despliegue

IgnaValdivia/TestTorneoTenis at deploy

El proyecto fue desplegado en un droplet de **Digital Ocean** usando **Docker y Nginx**. Se utilizaron tres contenedores en **Docker Compose**:

- ✓ **Aplicación Laravel (PHP-FPM)**.
- ✓ **Base de datos MySQL**.
- ✓ **Servidor web Nginx**.

10.1. Configuración del Droplet

- Sistema Operativo: Ubuntu 22.04
- Dirección IP asignada: 164.90.153.140
- Base de datos: MySQL 8
- Web server: Nginx

Este enfoque permite escalabilidad y portabilidad, asegurando que la API pueda ser ejecutada en cualquier entorno con Docker.

Conclusión

El proyecto cumple con los requisitos del test, aplicando buenas prácticas de arquitectura de software, pruebas y despliegue. Se priorizó la separación de responsabilidades mediante capas y la implementación de una API REST documentada. El uso de Docker facilitó la portabilidad y el despliegue en DigitalOcean. Además, las pruebas aseguran la confiabilidad del sistema y la correcta ejecución de los torneos simulados.