

Borrador introducción

- **Planteamiento del problema:**

Desarrollar una aplicación que simule el sistema de batallas de Pokémon. La aplicación debe gestionar un combate por turnos entre usuario y rival, determinando el orden de ataque con base en la velocidad y finalizando el encuentro cuando la salud de alguno llegue a cero. Asimismo, el sistema debe implementar la lógica de efectividad de tipos de acuerdo con la tabla de tipos y ofrecer una interfaz gráfica interactiva que permita seleccionar ataques y visualizar el estado actual del combate.

- **Motivación:**

La simulación de un sistema de batalla por turnos representa un escenario ideal para la implementación avanzada de POO, ya que requiere la interacción constante entre múltiples objetos con estados complejos. Utilizar Flutter permite aplicar lógica de backend y comprender cómo los objetos se vinculan con una interfaz gráfica reactiva, usando el patrón de diseño MVC y acercando el desarrollo a un entorno de producción de software moderno y funcional.

- **Objetivos:**

Implementar una jerarquía de clases que modele correctamente las entidades del juego aplicando conceptos fundamentales como herencia, polimorfismo, clases abstractas y el patrón de diseño MVC, así como integrar la lógica con la interfaz de usuario mediante el manejo de estados en Flutter, asegurando que las reglas del juego se ejecuten de manera consistente y transparente para el usuario.

Borrador conclusión

El desarrollo de este proyecto permitió integrar de manera efectiva el paradigma de la Programación Orientada a Objetos con la lógica de desarrollo de aplicaciones en Flutter. Se comprobó que el uso de clases abstractas y herencia es esencial para gestionar la diversidad de los elementos de un programa sin duplicar código, permitiendo que el sistema sea escalable y sostenible.

La implementación de la tabla de tipos y el cálculo de daño evidenció la importancia de centralizar las reglas en valores finales para evitar inconsistencias en el estado de la batalla. Además, la gestión de turnos basada en la velocidad reforzó la comprensión del flujo de control y la manipulación de objetos en tiempo de ejecución. Finalmente, el uso de Flutter y el patrón de diseño MVC demostró cómo la programación orientada a objetos sirve como base sólida para construir interfaces gráficas dinámicas, donde cada elemento visual responde a los cambios en los modelos de datos.