ANÁLISIS ESTÁTICO– BAD DOPO CREAM

POOB

13 DE DICIEMBRE DE 2025

ANDRES FELIPE PINEDA GAITAN
HEIDY ALEJANDRA ORJUELA RAMIREZ

1. Resultado inicial del análisis estático

El análisis estático del proyecto Bad Dopo Cream, realizado antes del proceso de refactorización, permitió evaluar la estructura del código sin ejecutar la aplicación, centrándose en la organización de paquetes, relaciones entre clases, uso de principios de diseño y posibles problemas.

Inicialmente, el sistema presentaba una separación básica entre las capas de Dominio y Presentación, lo cual es positivo desde el punto de vista arquitectónico. Sin embargo, se identificaron varios aspectos a mejorar:

- Alto acoplamiento entre clases de presentación y lógica de control, especialmente en ControladorJuego, el cual asumía múltiples responsabilidades.
- Algunas clases del dominio concentraban demasiada lógica, afectando el principio de responsabilidad única.
- Existía duplicación de decisiones de creación de objetos (por ejemplo, frutas, enemigos y obstáculos) sin un mecanismo centralizado.
- No estaba claramente definida una jerarquía completa para entidades como frutas, enemigos, obstáculos y baldosas.
- La gestión de recursos gráficos estaba parcialmente acoplada a la lógica de renderizado.
- No se evidenciaba una estrategia clara para la extensión del sistema (nuevos niveles, nuevas IAs, nuevos tipos de entidades).

A nivel de diseño, aunque el sistema funcionaba correctamente, el análisis estático reveló que la extensibilidad podían verse comprometidas a largo plazo.

2. Decisiones tomadas tras el análisis estático

Luego de estudiar el resultado inicial del análisis estático, se tomaron las siguientes decisiones de diseño y refactorización:

1. Refuerzo de la arquitectura en capas, manteniendo una separación estricta entre:
    - Dominio (lógica del juego)
    - Presentación (interfaz gráfica y renderizado)
2. Reorganización de la jerarquía de clases del dominio, estableciendo una superclase Entidad y especializaciones claras:
    - Helado
    - Fruta
    - Enemigo
    - Obstaculo (con Pared, BloqueHielo y Fogata)
    - Baldosa (BaldosaNormal y BaldosaCaliente)

3. Introducción de fábricas (FactoryEntidad) para centralizar la creación de entidades, reduciendo el acoplamiento y facilitando la inclusión de nuevos tipos.
4. Separación de responsabilidades en la presentación, dejando a Renderizador la lógica de dibujo y a RecursosGraficos la carga y gestión de imágenes.
5. Preparación del sistema para extensión, permitiendo agregar nuevos niveles, enemigos, frutas o tipos de IA sin modificar clases existentes, respetando el principio Open/Closed.

Como resultado final del análisis estático, el sistema quedó con una estructura más clara, modular y preparada para crecimiento, mejorando notablemente su mantenibilidad.

3. Resultado final del análisis estático

Tras aplicar las decisiones anteriores, el análisis estático final mostró:

- Bajo acoplamiento entre capas.
- Jerarquías bien definidas y coherentes.
- Clases con responsabilidades claras.
- Facilidad para extender el sistema sin modificar código existente.
- Mejor legibilidad y organización del proyecto.

Esto evidencia una mejora significativa en la calidad del diseño orientado a objetos.