Análisis de Diseño - Cumplimiento de MVC y SOLID

Arquitectura MVC

Modelo (Paquete: modelo):

- Define las entidades del dominio y contiene la lógica de negocio.
- Ej: Producto.java, TiendaLocal.java, FeriaItinerante.java.

Vista (Paquete: vista):

- Muestra los menús al usuario y recoge datos desde consola.
- Ej: ConsolaUI.java.

Controlador (Paquete: controlador):

- Media entre modelo y vista, maneja la lógica de aplicación.
- Ej: ComercioController.java, ProductoController.java, ArchivoController.java.

Principios SOLID

S - Single Responsibility:

- Cada clase tiene una única responsabilidad.
- Ej: Producto.java representa productos; ArchivoController.java maneja archivos.

O - Open/Closed:

- Las clases están abiertas para extensión pero cerradas para modificación.
- Ej: ActividadEconomica permite agregar nuevos tipos de comercio sin cambiar el resto.

L - Liskov Substitution:

- Las subclases pueden reemplazar a su clase padre sin errores.
- Ej: TiendaLocal y FeriaItinerante implementan métodos de ActividadEconomica.

I - Interface Segregation:

- Clases con métodos necesarios y enfocados.

Análisis de Diseño - Cumplimiento de MVC y SOLID

- Diseño permite separar fácilmente en interfaces específicas si se requiere.

D - Dependency Inversion:

- La vista depende de abstracciones (controladores), no directamente del modelo.

- Ej: ConsolaUI usa ComercioController, no accede directamente a PuntoComercial.

Resumen Tabular

| Principio | Ejemplo Concreto |

|----------|------------------|

| Modelo | Producto, ActividadEconomica, PuntoComercial |

| Vista | ConsolaUI |

| Controlador | ComercioController, ArchivoController |

| SRP | Cada clase tiene una única responsabilidad |

| OCP | Nuevos tipos de comercio no rompen código existente |

| LSP | Las subclases se usan como la clase padre |

| ISP | Diseño preparado para interfaces enfocadas |

| DIP | ConsolaUI depende de controladores |