Patrones de Diseño
MediCareDesk

Presentado por:

Yamid Alfonso Gonzalez Torres

Jenny Catherine Herrera Garzon

Edwin Andres Marin Vanegas

Diego Steven Pinzon Yossa

Profesor:

Oscar Eduardo Alvarez Rodriguez

Universidad Nacional de Colombia

Facultad de Ingeniería

Ingeniería de software

2025

I. Introducción

Este documento tiene como propósito presentar y justificar la selección de dos patrones de diseño orientados a objetos contemplados para el sistema **MediCareDesk**. Aunque su implementación completa aún no se ha consolidado, se documenta su intención de uso, beneficios y el rol que cumplirán dentro de la arquitectura monolítica desarrollada con **Python, Tkinter y SQLite**.

Los patrones seleccionados, **Observer** (comportamiento) y **Builder** (creacional), responden a la necesidad de mantener bajo acoplamiento, claridad en la construcción de objetos, y una estructura escalable y mantenible.

II. Patrón Observer

A. Definición y propósito

El patrón **Observer** permite establecer una relación de notificación automática entre un objeto central ("sujeto") y uno o varios "observadores". Cuando el sujeto cambia de estado (por ejemplo, se programa o verifica una toma), notifica a todos los observadores para que reaccionen.

Este patrón es ideal cuando se desea separar la lógica que produce los eventos de aquella que reacciona a ellos, lo que mejora la **modularidad**, reduce el **acoplamiento** y facilita la **escalabilidad**.

B. Intención de uso en MediCareDesk

En MediCareDesk, este patrón se aplicará para notificar de forma automática a distintos módulos del sistema cuando ocurran eventos importantes, como por ejemplo:

- Una **toma de medicamento programada** activa una notificación visual o sonora (alerta).
- Una **toma verificada** se registra automáticamente en el historial.
- Una **toma omitida** actualiza estadísticas de adherencia al tratamiento.

Donde:

- **Sujeto:** Módulo de programación de tomas
- **Observadores posibles:** Módulo de alertas, historial, evaluador de adherencia.

Esto permite que el sistema reaccione sin que los módulos estén acoplados entre sí.

C. Justificación

El patrón Observer es útil para:

- Automatizar respuestas a eventos clínicos (ej. alertas).
- Evitar el entrelazamiento del código entre módulos.
- Favorecer la incorporación de nuevas funcionalidades sin alterar la lógica existente.
- Facilitar pruebas por separado (los observadores pueden ser simulados).

Su uso se alinea con el enfoque modular del sistema y su ejecución centralizada (arquitectura monolítica).

III. Patrón Builder

A. Definición y propósito

El patrón **Builder** permite construir objetos complejos paso a paso, mediante una interfaz clara y flexible, separando la construcción del uso.

En MediCareDesk, entidades como **Paciente**, **Medicamento**, **Tratamiento** o **Toma** requieren muchos atributos. Usar constructores tradicionales con múltiples parámetros puede ser confuso, propenso a errores y difícil de mantener.

B. Intención de uso en MediCareDesk

Por ejemplo, al crear un paciente, en lugar de:

```python
paciente = Paciente("1098347621", "Claudia Martínez", 75, "F", "3001234567",
"Diabetes tipo II")
```

Se propone usar:

```python
paciente = PacienteBuilder()\
    .set_id("1098347621")\
    .set_nombre("Claudia Martínez")\
    .set_edad(75)\
    .set_genero("F")\
```

```
    .set_contacto_emergencia("3001234567")\
    .set_observaciones("Diabetes tipo II")\
    .build()
```

Esto permite:

- Validar los datos durante la construcción.
- Asegurar que no falte ningún campo esencial.
- Hacer el código más **legible y mantenible**.

Este patrón será clave para entidades que se crean desde formularios, donde los datos se ingresan de forma **interactiva y progresiva**.

C. Justificación

El patrón Builder aporta:

- Claridad en la creación de entidades con múltiples atributos.
- Validaciones previas a la construcción definitiva del objeto.
- Reducción de errores y mejoras en la mantenibilidad.
- Facilidad para extender entidades sin romper código existente.

Además, facilita pruebas, generación de datos de ejemplo, e integración con formularios Tkinter.

IV. Conclusión

Los patrones **Observer** y **Builder** fueron seleccionados para estructurar el sistema MediCareDesk de forma escalable, clara y profesional. Su uso contribuye a una mejor organización del código, separando responsabilidades y permitiendo evolución futura sin comprometer la arquitectura.

Ambos patrones se integran perfectamente en la arquitectura monolítica con SQLite, y serán implementados progresivamente a medida que avance el desarrollo del sistema.