Universidad Tecnológica de Tijuana

Tema:

Selection of Design Patterns

Materia:

Desarrollo movil Integral

Grupo:

10-B

Nombre del Alumno:

Sanchez Zamudio Guadalupe

Docente:

Ray Brunette Parra Galaviz

Fecha de Elaboración:

10/01/2024

Selección de Patrones de Diseño

Los patrones de diseño son soluciones probadas para problemas comunes en el desarrollo de software. Elegir el patrón adecuado para las tecnologías utilizadas no solo mejora la calidad del código, sino que también facilita su mantenimiento y escalabilidad. A continuación, describo los patrones de diseño que considero más relevantes para mi caso, utilizando React, React Native y Django.

Patrón Componente (Component Pattern)

Tecnologías aplicables: React y React Native

El patrón de componente es uno de los pilares fundamentales en React y React Native. Consiste en dividir la interfaz de usuario en pequeñas piezas reutilizables llamadas componentes. Cada componente encapsula su lógica, estado y presentación, lo que permite una construcción modular y eficiente de aplicaciones.

Por qué este patrón es adecuado:

Reutilización: Los componentes se pueden usar en múltiples partes de la aplicación, reduciendo el código duplicado.

Mantenimiento: Al tener piezas pequeñas y bien definidas, el código es más fácil de actualizar.

Modularidad: Los cambios en un componente no afectan al resto de la aplicación, siempre que las interfaces estén correctamente definidas.

Patrón Observador (Observer Pattern)

Tecnologías aplicables: React y React Native

El patrón observador es útil para manejar eventos y actualizar el estado en la aplicación. En React, esto se implementa mediante el uso de state y props, así como bibliotecas como Redux o Context API para manejar estados globales.

Por qué este patrón es adecuado:

Sincronización automática: Permite que los componentes "observadores" reaccionen automáticamente cuando hay cambios en el estado.

Eficiencia: Evita actualizaciones manuales, ya que el flujo de datos reactivo gestiona las modificaciones por sí mismo.

Patrón de Fachada (Facade Pattern)

Tecnologías aplicables: Django y React

El patrón de fachada simplifica las interacciones entre sistemas complejos al proporcionar una interfaz unificada. En este caso, se puede implementar en el backend (Django) para manejar las solicitudes de la API, encapsulando la lógica y exponiendo solo lo necesario al frontend.

Por qué este patrón es adecuado:

Abstracción: Oculta la complejidad del backend al frontend.

Simplicidad: Reduce el número de puntos de interacción entre el cliente y el servidor.

Patrón MVC (Model-View-Controller)

Tecnologías aplicables: Django

El patrón MVC organiza la aplicación en tres capas principales:

Model (Modelo): Maneja la lógica de datos y las interacciones con la base de datos.

View (Vista): Define cómo se presenta la información al usuario.

Controller (Controlador): Gestiona las solicitudes del usuario y coordina la interacción entre el modelo y la vista.

Django adopta una variación de este patrón, conocida como MTV (Model-Template-View), donde:

Model: Es similar al modelo en MVC y maneja los datos.

Template: Define la capa de presentación.

View: Combina la lógica del controlador y la vista.

Por qué este patrón es adecuado:

Separación de responsabilidades: Cada capa tiene una función bien definida, lo que facilita la gestión del código.

Escalabilidad: Es más fácil extender la funcionalidad de una aplicación bien estructurada.

Soporte integrado en Django: La estructura de Django está diseñada para trabajar de manera natural con MTV.