SERVICIO NACIONAL DE APRENDIZAJE SENA

Patrones arquitectónicos orientados a la realización de la interfaz de usuario

Título: investigación de patrones arquitectónicos orientado a la realización de la interfaz de usuario

Juliana Tique
David Solano
Jaider Moreno

09/05/2025

TABLA DE CONTENIDOS

Introducción

Los patrones arquitectónicos para el desarrollo de la interfaz de usuario son soluciones reutilizables que resuelven problemas comunes en la realización del software, los patrones arquitectónicos para el desarrollo de infraestructura UX, son especializados en la gestión de estructura, organización y gestión de los elementos visuales.

Conceptos fundamentales

¿Qué es un patrón de arquitectura?

Según (Renato, 2023)

En términos generales, un patrón de arquitectura es la conceptualización de una solución genérica y reutilizable, aplicable a un problema de diseño de software en un contexto determinado, satisfaciendo las necesidades del negocio.

Los patrones arquitectónicos son la representación de las buenas prácticas y estructuras de diseño probadas, de modo que puedan reutilizarse. El objetivo es reutilizar las experiencias y conocimientos arquitectónicos que han dado buenos resultados en el pasado.

¿Por qué son importantes estos patrones?

Diversificación de tareas:

Al trabajar con estos patrones, se permite la fácil lectura del código, lo cual es útil en grupos de trabajo de gran tamaño y su cooperación.

Reutilización de código:

Permite que el software sea escalable y reutilizable, ahorra tiempo.

Mantenimiento y testing:

Facilita el mantenimiento del código ya que la lógica de presentación y de negocio están separadas.

Descripción de los patrones de diseño más comunes:

Creacionales

Se centran en la creación de objetos para el sistema, con independencia de cómo se hayan creado, su composición o su representación. Por ejemplo, Singleton, que da una sola instancia, pero con acceso global, o Factory Method, que define una interfaz para el objeto, pero sin decidir las instancias, que se eligen en las subclases. (Arquitectura de Software: 5 Patrones Principales, 2024)

Estructurales

Otro de los patrones de diseño de software son los estructurales. Se encargan de la composición de objetos o clases y de que las entidades se combinen correctamente. Algunos ejemplos son Adapter, que hace que clases con interfaces incompatibles puedan trabajar juntas, o Decorator, que añade responsabilidades adicionales a un objeto de forma dinámica para extender funcionalidades. (Arquitectura de Software: 5 Patrones Principales, 2024)

Comportamiento

Estos patrones se centran en la comunicación y en la asignación de responsabilidades entre objetos. Destacan Observer, que define la dependencia entre objetos para que cuando uno cambia, sus dependientes se actualicen automáticamente, o Strategy, que define una familia de algoritmos, encapsula cada uno y los hace intercambiables, para que aquellos puedan variar sin importar quién lo utiliza.

Con esta explicación, ya deberías comprender la diferencia entre arquitectura de software y su diseño. Pero si quieres seguir aclarando dudas y descubriendo nuevos conceptos, puedes hacerlo con nuestros másteres. (Arquitectura de Software: 5 Patrones Principales, 2024)

Descripción de patrones arquitectónicos mas comunes

Modelo-Vista-Controlador (MVC):

La arquitectura Model-View-Controller (MVC) es un patrón de diseño que separa una aplicación en tres componentes principales, que son:

- Modelo (Model). Este representa los datos y la lógica de negocio del sistema. Se encarga del acceso a los datos almacenados, ya sea en una base de datos o en otros medios. También define las reglas de negocio que se deben cumplir.

- Vista (View). Esta capa se encarga de la presentación de los datos. Por tanto, interactúa con el usuario, muestra la información y recibe la entrada. Se diseña para que sea independiente del modelo. Es decir, que, como en la hexagonal, se puede cambiar sin afectar a los datos subyacentes.

- Controlador (Controller). Es el intermediario entre los dos componentes anteriores. Procesa las solicitudes o entradas del usuario; las procesa y realiza las operaciones necesarias en el modelo, como por ejemplo, actualizarlo, y decide qué vista es la que se mostrará después con los resultados, (Arquitectura de Software: 5 Patrones Principales, 2024).

Este es uno de los patrones más conocidos. Separa la lógica de negocio (modelo), la interfaz de usuario (vista) y la lógica de control (controlador). Esto facilita la gestión de cambios y mejora la escalabilidad.

Modelo-Vista-ViewModel (MVVM):

Muy usado en aplicaciones basadas en frameworks como WPF, Xamarin, o Angular. Aquí se añade el ViewModel, que sirve como un intermediario entre la vista y el modelo. Este patrón permite una separación aún más clara entre las responsabilidades.

Modelo-Vista-Presentador (MVP):

Similar al MVC, pero el presentador es el que maneja la lógica de presentación y puede ser más adecuado para ciertas aplicaciones con interfaces más complejas o interacciones más detalladas.

Flux/Redux:

Este patrón es muy popular en aplicaciones de una sola página (SPA) y en frameworks como React. Redux, que es una implementación de Flux, maneja el estado de la aplicación de manera centralizada, lo que facilita la depuración y la previsibilidad.

Component-based Architecture:

Con este patrón, la interfaz de usuario se divide en componentes reutilizables y modulares, lo que facilita el mantenimiento y la escalabilidad de la UI. Este es el enfoque que se utiliza en frameworks modernos como React, Vue, Angular.

Single Page Application (SPA):

En este patrón, la interfaz de usuario se carga completamente en una sola página, y las interacciones con el usuario no requieren recargar la página completa. Se utilizan herramientas y frameworks como React, Angular o Vue.js para crear aplicaciones web de una sola página.

Retos y desafíos de la implementación

Complejidad Inicial de Implementación

Descripción: Algunos patrones arquitectónicos, como MVVM o Flux/Redux, requieren una mayor inversión inicial de tiempo y esfuerzo para implementarlos correctamente. La cantidad de código estructural necesario al principio puede hacer que el desarrollo sea más complejo y lento, sobre todo para equipos o desarrolladores sin mucha experiencia con el patrón.

Curva de Aprendizaje

Descripción: Los patrones como Redux o MVVM suelen tener una curva de aprendizaje empinada, especialmente para quienes no están familiarizados con la gestión del estado o el manejo de la data binding (vinculación de datos).

Rendimiento y Sobrecarga de Recursos

Descripción: Algunos patrones arquitectónicos, especialmente aquellos que implican la separación estricta de las responsabilidades, como MVC o MVP, pueden generar una sobrecarga adicional en el rendimiento de la aplicación debido a la necesidad de múltiples capas y objetos para manejar las interacciones.

Gestión de Estado Complejo

Descripción: En aplicaciones que requieren manejar múltiples estados complejos o datos de usuario en tiempo real (como en una aplicación de redes sociales o de comercio electrónico), patrones como Redux o Flux pueden hacer que la gestión del estado sea difícil de manejar.

Conclusiones

Los patrones de diseño son estándares para el buen manejo y desarrollo del software, permiten la comprensión , estructuración, cooperación y mantenimiento del software para que sea escalable.

Bibliografía

Arquitectura de software: 5 patrones principales. (2024). Inesdi. https://www.inesdi.com/blog/arquitectura-de-software-5-patrones-principales/

Renato, E. (2023, February 28). ¿Se han preguntado alguna vez como se diseñan los sistemas a gran escala en una organización? Antes de comenzar el desarrollo de algún software o solución tecnológica, conviene elegir la arquitectura adecuada que proporcione la funcionalidad y atributos de calidad deseados. En ese particular, cobra. Linkedin.com. https://es.linkedin.com/pulse/patrones-en-la-arquitectura-de-software-elmo-renato-castro-ramirez