Patrones de Comportamiento

Definición

Se centran en la asignación de responsabilidades entre los objetos y cómo interactúan para lograr un **comportamiento deseado en el sistema.** Son herramientas valiosas para abordar problemas **relacionados con la interacción** y el comportamiento de los objetos en un sistema, a la vez que mejoran la calidad, la mantenibilidad y la **escalabilidad del software.**

En el contexto del desarrollo móvil, se refiere a como las clases y objetos interactuan entres si para lograr una funcionalidas especifica

Propósito

El propósito de los patrones de comportamiento es mejorar la comunicación y colaboración entre objetos en una aplicación, promoviendo un diseño mas modular y mantenible.

Esto facilita la implementacion de nuevas funcionalidades y la resolucion de problemas de manera mas eficiente

Ejemplos en el desarrollo móvil

Imaginemos una aplicación de mensajería. El sujeto podría ser un objeto que representa un mensaje, y los observadores serían los usuarios suscritos a recibir notificaciones sobre ese mensaje. Cuando llega un nuevo mensaje, el sujeto notifica a todos sus observadores para que actualicen su interfaz y muestren el nuevo contenido.

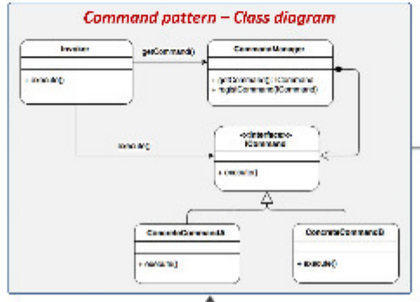
Beneficios

Los patrones de comportamiento en software encapsulan procesos que deben ejecutarse dentro de la funcionalidad de una aplicación. Los patrones de diseño ayudan a los desarrolladores de software a cambiar la forma en que funciona el sistema sin un rediseño completo.

Herramientas

Command

Nos permite ejecutar operaciones sin conocer los detalles de la implementación de la misma. Las operaciones son conocidas como comandos y cada operación es implementada como una clase independiente que realiza una acción muy concreta



ICommand: Interface que describe la estructura de los comandos, la cual define el método de ejecución genérico para todos los comandos.
Concrete Command: Representan los comandos concretos, éstos deberán heredar de ICommand. Cada una de estas clases representa un comando que podrá ser ejecutado de forma independiente.
Command Manager: Este componente nos servirá para administrar los comandos que tenemos disponibles en tiempo de ejecución, desde aquí podemos solicitar los comandos o registrar nuevos.
Invoker: El invoker representa la acción que dispara alguno de los comandos.

BIBLIOGRAFIA

https://reactiveprogramming.io/blog/es/patrones-de-diseno/command

https://www.perplexity.ai/