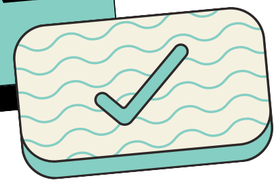
UNIPAR

Frameworks de Desenvolvimento Web

CONCEITOS INICIAIS

Prof. Juliane Correia

AGENDA



01 Padrões

02 Anti-padrões

03 Cliente-Servidor

04 Arquitetura MVC

05 RESTful APIs

06 WebSockets

PADRÕES

Padrões de projeto (Design Patterns) são soluções comprovadas para problemas recorrentes no desenvolvimento de software. Eles fornecem um modelo reutilizável para resolver problemas comuns, facilitando a manutenção e evolução do software. Esses padrões são como “receitas” que ajudam os desenvolvedores a tomar decisões informadas durante o processo de desenvolvimento.

PADRÕES

CRIACIONAIS

Propósito: Enfocar no processo de criação de objetos, abstraindo e ocultando a lógica de criação.

Singleton: Garante que uma classe tenha apenas uma instância e fornece um ponto global de acesso a ela.
Factory Method: Define uma interface para criar objetos, mas permite que as subclasses alterem o tipo de objeto que será criado.

ESTRUTURAIS

Propósito: Facilitar o design de classes e objetos através da composição, agregação e simplificação das relações entre objetos.

Adapter: Permite que classes com interfaces incompatíveis trabalhem juntas.
Composite: Compõe objetos em estruturas de árvore para representar hierarquias parte/todo.

COMPORTAMENTAIS

Propósito: Se concentram em algoritmos e na atribuição de responsabilidades entre objetos.

Observer: Define uma dependência um-para-muitos entre objetos, de modo que quando um objeto muda de estado, todos os seus dependentes são notificados.
Strategy: Define uma família de algoritmos, encapsula cada um deles e os torna intercambiáveis.

ANTI PADRÕES

SPAGHETTI CODE

GOD OBJECT

GOLDEN HAMMER

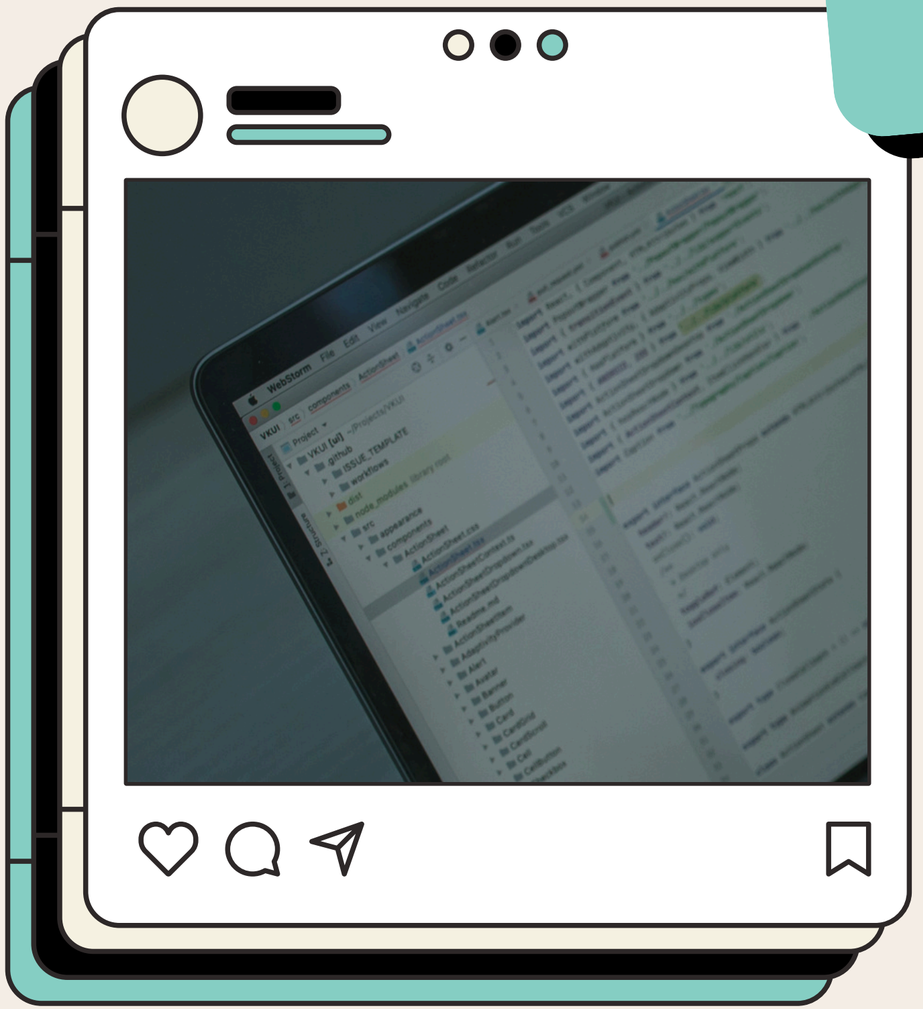
COPY-PASTE PROGRAMMING

Anti-padrões são práticas de design ou implementação que parecem boas ideias inicialmente, mas que acabam por ser prejudiciais ou contraproducentes a longo prazo. Identificar e evitar anti-padrões é crucial para o desenvolvimento de software de qualidade.

ARQUITETURA WEB

- A arquitetura **cliente-servidor** é um modelo de interação entre duas entidades: o cliente (que faz solicitações) e o servidor (que processa essas solicitações e responde).
- **Cliente**: O cliente é a parte que solicita recursos ou serviços do servidor. Em uma aplicação web, o cliente geralmente é um navegador web que faz uma solicitação HTTP a um servidor web.
- **Servidor**: O servidor é a parte que recebe a solicitação do cliente, processa-a (possivelmente interagindo com um banco de dados ou executando lógica de negócios) e envia uma resposta de volta ao cliente.

FLUXO

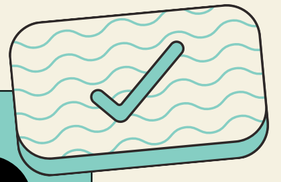


01 O cliente faz uma solicitação ao servidor

02 O servidor processa a solicitação

03 O servidor envia uma resposta de volta ao cliente

ARQUITETURA MVC



M

Model (Modelo): Representa os dados e a lógica de negócios. É responsável por acessar e manipular os dados, geralmente interagindo com um banco de dados.

V

View (Visão): A parte da aplicação responsável por apresentar os dados ao usuário. Ela é o que o usuário vê na interface.

C

Controller (Controlador): Atua como um intermediário entre o Model e a View. Ele captura as interações do usuário e determina como responder a essas interações, invocando métodos do Model e selecionando a View apropriada.

RESTFUL

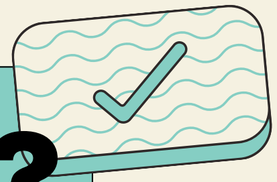
- **REST (Representational State Transfer) é um estilo de arquitetura para criar serviços web. Uma API que adere aos princípios REST é chamada de RESTful API.**

- **RESTful API: Permite que diferentes sistemas interajam entre si através de HTTP, utilizando operações padrão como GET, POST, PUT, DELETE.**

WEBSOCKETS

WebSocket é um protocolo de comunicação bidirecional em tempo real, que permite a troca de dados entre um cliente (geralmente um navegador) e um servidor através de uma única conexão TCP. Diferente do protocolo HTTP tradicional, que é baseado em solicitações (request-response), WebSocket permite que o servidor envie dados para o cliente sem a necessidade de o cliente fazer uma nova solicitação. Isso cria um canal de comunicação persistente entre cliente e servidor, ideal para aplicações que necessitam de atualizações em tempo real, como chats, jogos online, ou atualizações de mercado financeiro.

COMO FUNCIONA?



ESTABELECE A CONEXÃO

A comunicação começa com um "handshake" HTTP, onde o cliente solicita a atualização da conexão para WebSocket. Se o servidor aceita, a conexão é estabelecida, e o protocolo é atualizado para WebSocket.

COMUNICAÇÃO BIDIRECIONAL

Após o handshake, o canal de comunicação é mantido aberto, permitindo que o cliente e o servidor enviem mensagens entre si a qualquer momento, sem a necessidade de reestabelecer a conexão.

ENCERRAMENTO DA CONEXÃO

A conexão WebSocket permanece aberta até que o cliente ou o servidor envie um sinal para fechá-la.


ATIVIDADE!