Vitoria Amaro Soares da Silva

24/03/2025

Situação-Problema: Sistema de gerenciamento de uma Biblioteca.

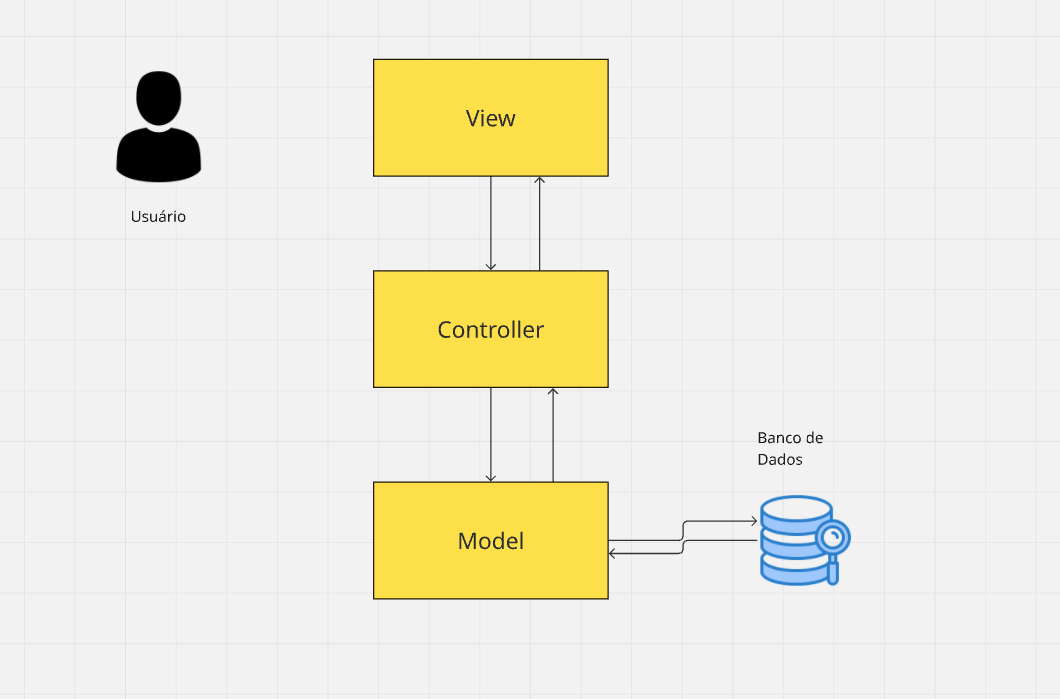
**Características que tornam essa aplicação um software back-end web**

Uma aplicação **back-end web** é a parte de um sistema que lida com o processamento de dados e a lógica de negócios **nos bastidores**. Aqui estão suas principais características:

1. **Processamento de Requisições**: Recebe e processa dados enviados pelo front-end (usuário).
2. **Lógica de Negócios**: Aplica regras para manipular dados (como cálculos e decisões).
3. **Interação com Banco de Dados**: Armazena e recupera dados (ex: informações do usuário).
4. **Autenticação e Autorização**: Garante que apenas usuários autorizados acessem certas funcionalidades.
5. **APIs**: Oferece interfaces para que o front-end interaja com o back-end.

Em resumo, o back-end é responsável pela parte **invisível** de uma aplicação web, garantindo que tudo funcione corretamente e de forma segura.

* **Diagrama MVC:**

****

**MVC** é a divisão de aplicação em três camadas: interação do usuário(View), a camada de manipulação(Model) e a camada de controle (Controller).

Neste diagrama, mostro a estrutura dessa camada. O usuário, por sua vez, interage com o View.

**Model** - Lida com todos os dados, lógica e regras. Ele representa um objeto ou um conjunto de objetos, podendo acessar dados, manipulá-los e enviar para a visão.

**View -** Responsável pela representação gráfica dos dados, ou seja, como os dados são exibidos aos usuários. Contém apenas a lógica de apresentação.

**Controller** - Processa todas as solicitações e manipula os dados usando o componente do modelo e exibe os dados na visão.

**Elementos de entrada (endpoints, dados enviados) , processamento e saída para a programação da aplicação e adicionar evidências dessa interação (screenshots).**

**EndPoints** são URLs que o cliente acessa para interagir com o servidor. Essas URLs correspondem a métodos como:

**GET:** Retorna os dados do usuário (ex.: pegar informações de um usuário).

**POST:** Envia dados para criar algo novo (ex.: criar um novo usuário).

**PUT:** Atualiza dados existentes (ex.: mudar o nome de um usuário).

**DELETE:** Deleta dados (ex.: excluir um usuário).

**Processamento:** O servidor processa os dados recebidos, valida, interage com o banco de dados e aplica a lógica de negócios.

**Saída:**

O servidor envia uma resposta de volta ao cliente:

* **GET /livros**: Retorna um JSON com os dados dos livros.
* **Padrões de Design:**

Padrão de Design ou Padrões de Projetos são conjuntos de ferramentas para soluções de problemas comuns em design de software, onde você tem a chance de alterar e melhorar o desempenho. Há 3 padrões, são eles:

**Padrão Criacional:** Oferecem muitas alternativas de criação de objetos, aumentando a flexibilidade e a reutilização do código. Exemplos:

* Factory Method
* Builder
* Abstract Factory
* Singleton

**Padrão Estrutural**: Explica como montar objetos e classes em estruturas maiores, mantendo a flexibilidade. Exemplos:

* Facade
* Proxy
* Adapter
* Bridge

**Padrão Comportamental:** Trabalham com algoritmos e designação de responsabilidade entre os objetos.Exemplos:

* Strategy
* Visitor
* Command

**Quais padrões poderiam ajudar nesse projeto de sistema de gerenciamento de uma Biblioteca?**

O Padrão Singleton pode ser ideal para o sistema, pois garante que exista apenas uma instância de algumas classes essenciais para o sistema, como uma conexão única.

O Padrão Facade também pode ser útil, pois em um sistema de biblioteca,ao fornecer uma interface de uso mais simples para os usuários acessarem diferentes funcionalidades como empréstimo, sem precisar entender os detalhes internos de como o sistema é implementado.