



ASP.NET MVC

Objetivo do Curso

Capacitar você a criar aplicações web modernas e escaláveis utilizando o ASP.NET MVC, abordando desde os conceitos básicos até tópicos avançados. Ao final do curso, você estará preparado(a) para desenvolver aplicações completas, com interfaces dinâmicas, integração com banco de dados e recursos de segurança.

Pré-requisitos

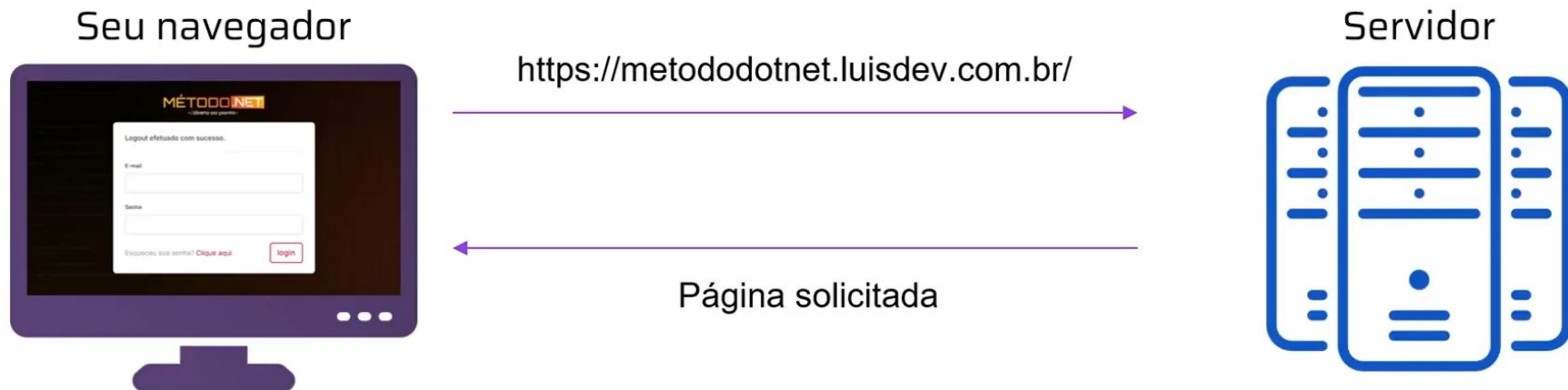
- Conhecimento básico de programação em C#.
- Noções de HTML, CSS e JavaScript são desejáveis, mas não obrigatórias.

Protocolo HTTP

O que é o HTTP?

- HTTP (HyperText Transfer Protocol) é um protocolo de comunicação baseado em texto que segue o modelo cliente-servidor.
- Facilita a troca de informações entre dispositivos na web.

Como o HTTP funciona?



Métodos HTTP

- GET

Recupera informações do servidor.
Exemplo: acessar uma página web.

- PUT

Atualiza dados no servidor.
Exemplo: editar perfil de usuário.

- POST

Envia dados ao servidor para criação ou processamento.
Exemplo: envio de formulários.

- DELETE

Remove dados do servidor.
Exemplo: deletar um comentário em uma postagem.

*Existem outros métodos menos comuns, como PATCH, OPTIONS e HEAD.

Métodos HTTP

HTTP Status Codes		
200	400	500
200: Sucesso	400: Bad Request	500: Internal Server
201: Created	401: Unauthorized	503: Service Unavailable
204: No Content	403: Forbidden	504: Geteway Timeout
	404: Not Found	

O Padrão MVC no ASP.NET

O que é o um padrão arquitetural?

Um padrão arquitetural é uma forma de organizar e estruturar sistemas de software para resolver problemas comuns de maneira eficiente. Ele define como as partes do sistema devem ser organizadas e trabalhar juntas, ajudando a tornar o software mais fácil de entender, manter e expandir.

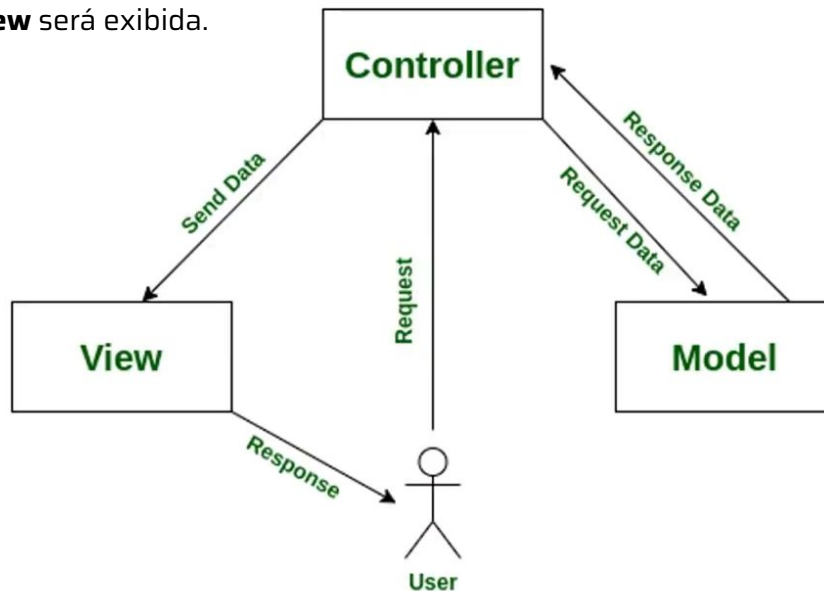
O que é o padrão MVC?

O padrão MVC (Model-View-Controller) é uma arquitetura de software amplamente adotada, especialmente no desenvolvimento de aplicações web, devido à sua capacidade de organizar o código de maneira eficiente e modular.

A principal característica do MVC é a separação da aplicação em três componentes principais, cada um com uma responsabilidade bem definida:

Estrutura do MVC no ASP.NET

O Controller é o componente intermediário entre a **View** e o **Model**. Ele é responsável por processar as requisições feitas pelo usuário (como acessar uma página ou enviar um formulário), interagir com o **Model** para obter ou modificar dados, e finalmente escolher qual **View** será exibida.



A **View** é a camada responsável por exibir os dados ao usuário. Ela define a interface com o usuário, ou seja, o que o usuário vê na tela.

O **Model** é o componente responsável por gerenciar a **lógica de negócios** e os **dados** da aplicação. Ele representa os dados que a aplicação manipula e a lógica que os acompanha.

Por que usar o padrão MVC?

Separação de responsabilidades: Cada componente tem um papel bem definido, o que facilita a manutenção e a evolução do código.

Testabilidade: A lógica de negócios é separada da interface, tornando os testes unitários mais simples.

Reutilização de código: O modelo pode ser reutilizado em diferentes partes da aplicação.

Facilidade de colaboração: Times podem trabalhar simultaneamente em diferentes camadas da aplicação (front-end e back-end).

Rotas no ASP.NET MVC

O que são rotas?

As rotas no ASP.NET Core MVC são responsáveis por mapear URLs para ações específicas nos controladores. Elas determinam como as solicitações são direcionadas para o controlador e a ação correspondente, facilitando a execução da lógica necessária e retornar a resposta ao cliente.

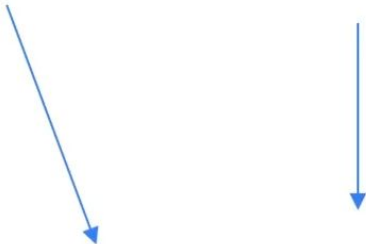
Passagem de dados: Model Binding

O que é o Model Binding?

O Model Binding é o processo pelo qual o ASP.NET Core MVC converte dados de uma requisição HTTP (query strings, formulários, headers, corpos de requisição, etc.) em objetos de modelo definidos em sua aplicação. Ele facilita a manipulação de dados enviados pelos usuários sem a necessidade de parsing manual.

Como funciona o Model Binding?

`https://localhost:7063/Usuario/ BuscaPorUrl?nome=Joao&sobrenome=Santos`



`[HttpGet]`
`public IActionResult BuscaPorUrl(string nome, string sobrenome)`

Introdução ao Bootstrap



O que é o Bootstrap?

Bootstrap é um framework front-end de código aberto desenvolvido para facilitar o processo de criação de sites e aplicativos web modernos, responsivos e visualmente consistentes.

Ele é amplamente utilizado devido à sua facilidade de uso, documentação rica e ampla compatibilidade com navegadores. Ele acelera o desenvolvimento, reduzindo a necessidade de escrever estilos do zero.

Porque usar o Bootstrap?

- Acelera o desenvolvimento front-end.
- Design responsivo embutido.
- Ampla compatibilidade com navegadores.
- Extensa documentação e comunidade ativa.

Utilização Bootstrap

```
<button>Sem Bootstrap</button>
```

Sem Bootstrap

```
<button class="btn btn-primary">Com Bootstrap</button>
```

Com Bootstrap

```
<button style="
display: inline-block;
font-weight: 400;
text-align: center;
white-space: nowrap;
vertical-align: middle;
user-select: none;
background-color: #0d6efd;
border: 1px solid #0d6efd;
padding: 0.375rem 0.75rem;
font-size: 1rem;
line-height: 1.5;
border-radius: 0.375rem;
color: #fff;
text-decoration: none;
transition: color 0.15s ease-in-out, background-color 0.15s ease-in-out, border-color 0.15s ease-in-out, box-shadow 0.15s ease-in-out;
cursor: pointer;
">
  Sem Bootstrap
</button>
```

Entendendo a responsividade do Bootstrap



1

2

3

4

5

6

7

8

9

10

11

12

Nome 4 colunas	Sobrenome 4 colunas	DataNascimento 4 colunas
---------------------------------	--------------------------------------	---

Nome:				Sobrenome:				Data de Nascimento:			
<input type="text"/>				<input type="text"/>				<input type="text" value="dd/mm/aaaa"/>			
1	2	3	4	1	2	3	4	1	2	3	4
© 2024 - Gerenciamento De Pessoas - Privacy											

Extra small	Small	Medium	Large	Extra large
<576px	≥576px	≥768px	≥992px	≥1200px
.col	.col-sm	.col-md	.col-lg	.col-xl

Introdução ao Entity Framework

O que é o Entity Framework (EF)?

O Entity Framework (EF) é uma biblioteca de mapeamento objeto-relacional (ORM, do inglês Object-Relational Mapper) que faz parte do ecossistema .NET. Ele atua como um intermediário entre a aplicação e o banco de dados, permitindo que os desenvolvedores trabalhem com dados de maneira intuitiva e orientada a objetos, sem a necessidade de escrever diretamente comandos SQL para as operações mais comuns.

Benefícios do Entity Framework

Produtividade Aumentada

Consulta Baseada em LINQ

Independência de Banco de Dados

Gerenciamento de Migrações

Manutenção Facilitada

APLICAÇÃO

Pessoa

Id = 1

Nome = José

Sobrenome = Santos

DataNascimento = 25/06/1995



Entity Framework.



BANCO DE DADOS

ID	NOME	SOBRENOME	DATANASCIMENTO
1	José	Santos	25/06/1995

Separação de Responsabilidade entre as Camadas de Controller, Service e Repository

Services

A camada de serviço é responsável por centralizar e gerenciar a lógica de negócio. Ela atua como um intermediário entre a camada de apresentação (UI) e a camada de repositório, abstraindo regras complexas e simplificando o consumo pelos controladores ou APIs

Por que é importante?

Evita que controladores ou componentes de UI precisem lidar com regras complexas.

Facilita alterações na lógica de negócio sem impactar outras partes do sistema.

Repository

A camada de repositório é responsável por abstrair o acesso ao banco de dados. Ela fornece uma interface para realizar operações de CRUD (Create, Read, Update, Delete) e outras consultas, de forma que a lógica de acesso aos dados fique desacoplada das demais partes do sistema.

Por que é importante?

A aplicação não precisa saber detalhes sobre como os dados são armazenados ou recuperados.

Permite trocar o banco de dados (por exemplo, de SQL Server para PostgreSQL) sem grandes alterações no restante do sistema.

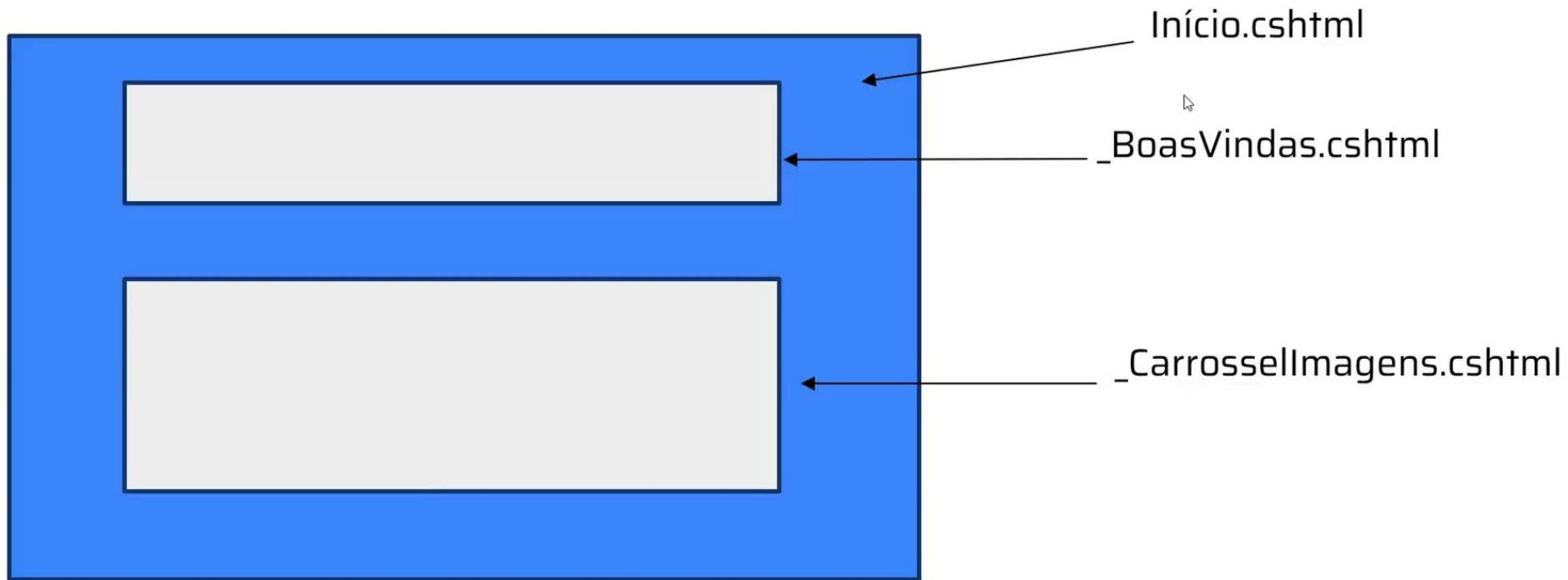
Fluxo de requisição



Conceito e benefícios das Views Parciais

O Que São Views Parciais?

As Views Parciais são componentes reutilizáveis que representam partes de uma interface de usuário (UI) em uma aplicação ASP.NET MVC. Elas são semelhantes às Views tradicionais, mas são projetadas para renderizar apenas uma parte da página, em vez de toda a estrutura da página.



Características principais

São chamadas a partir de uma View principal ou diretamente do Controller.

Não possuem layout próprio (por padrão, não renderizam a `_Layout.cshtml`).

Recomendadas para partes de UI que precisam ser reutilizadas em várias páginas ou componentes.

Quando Usar Views Parciais?

Reuso de Código: Você tem seções comuns da UI, como menus, rodapés ou cabeçalhos.

Modularidade: Quer dividir páginas grandes em pequenos componentes para facilitar a organização e manutenção do código.

Desempenho: Renderizar apenas partes específicas da página ao invés de recarregá-la completamente (usando técnicas como AJAX).

O que é o JQuery e sua utilidade em projetos web?

O que é jQuery?

O jQuery é uma biblioteca de JavaScript rápida, pequena e rica em funcionalidades que simplifica tarefas comuns de desenvolvimento web, como manipulação do DOM, gerenciamento de eventos, animações e chamadas AJAX.

Principais características do jQuery

Compatibilidade: Funciona de forma consistente em diferentes navegadores, resolvendo problemas de compatibilidade.

Facilidade de Uso: Fornece uma sintaxe simples e concisa para realizar tarefas que exigiriam mais código em JavaScript puro.

Comunidade Ampla: Possui uma vasta comunidade e muitos recursos, como plugins prontos e documentação.

Por que usar jQuery em projetos web?

Embora frameworks modernos como React, Angular e Vue.js sejam amplamente usados, o jQuery ainda é útil em muitas situações, especialmente em projetos que:

- Precisam de soluções rápidas para manipular o DOM.
- Incluem scripts leves para funcionalidades específicas.
- Têm dependências em bibliotecas legadas que utilizam jQuery.

O que é o Identity e sua importância no ASP.NET Core MVC

Principais Recursos do Identity

Autenticação e autorização integradas: Suporte para autenticação de usuários e controle de acesso baseado em permissões.

Armazenamento em banco de dados: Ele usa o Entity Framework Core para armazenar informações de usuários em um banco de dados relacional.

Criptografia de senha: Senhas dos usuários são armazenadas de forma segura utilizando hash.

Customização: Possibilidade de personalizar os modelos, tabelas e regras de autenticação para atender às necessidades do projeto.

Integração com provedores externos: Suporte a login com contas de terceiros, como Google, Facebook, Microsoft, etc.

O que é Identity?

O ASP.NET Core Identity é um framework de identidade fornecido pela Microsoft que permite aos desenvolvedores adicionar funcionalidades de autenticação e autorização em aplicações web. Ele fornece um conjunto robusto de ferramentas e APIs para gerenciar:

- **Usuários:** Criação, edição e exclusão.
- **Senhas:** Gerenciamento seguro e criptografado.
- **Funções (Roles):** Para controle de acesso baseado em perfis (admin, usuário, etc.).
- **Claims e Tokens:** Suporte a autenticação baseada em tokens e permissões específicas.