Curso: C# COMPLETO - Programação Orientada a Objetos + Projetos http://educandoweb.com.br **Prof. Dr. Nelio Alves**

Projeto: GUI Web com ASP.NET Core

Objetivo geral:

• Introduzir o aluno ao desenvolvimento de aplicações web com ASP.NET Core MVC • Permitir que o aluno conheça os fundamentos e a utilização do framework, de modo que ele depois possa estudar as especificidades que desejou

PROJETO NO GITHUB:

https://github.com/acenelio/workshop-asp-net-core-mvc

Visão geral do ASP.NET Core MVC

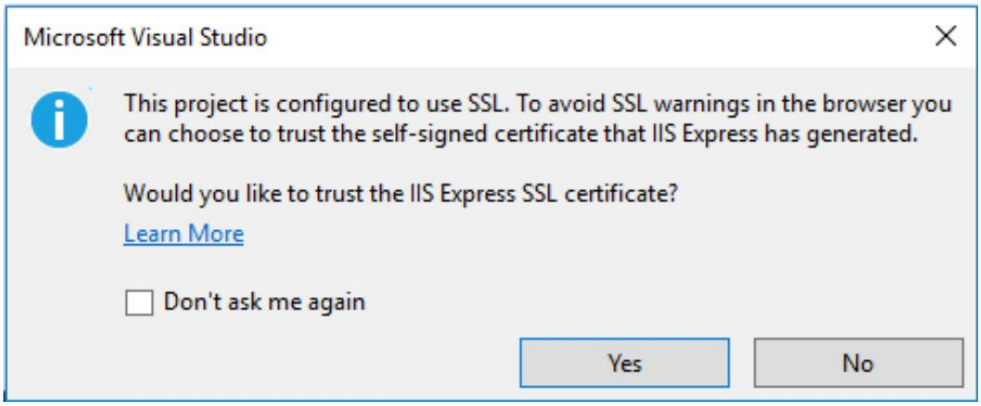
o É um framework para criação de aplicações web o Criado pela Microsoft e comunidade o Open source o Roda tanto no .NET Framework quanto no .NET Core o O framework trabalha com uma estrutura bem definida, incluindo:
- o Controladores
- o Visualizações
  - o Modelos ÿ Modelos de visualização o https://docs.microsoft.com/en-us/aspnet/core/mvc/overview

criação do projeto

Lista de

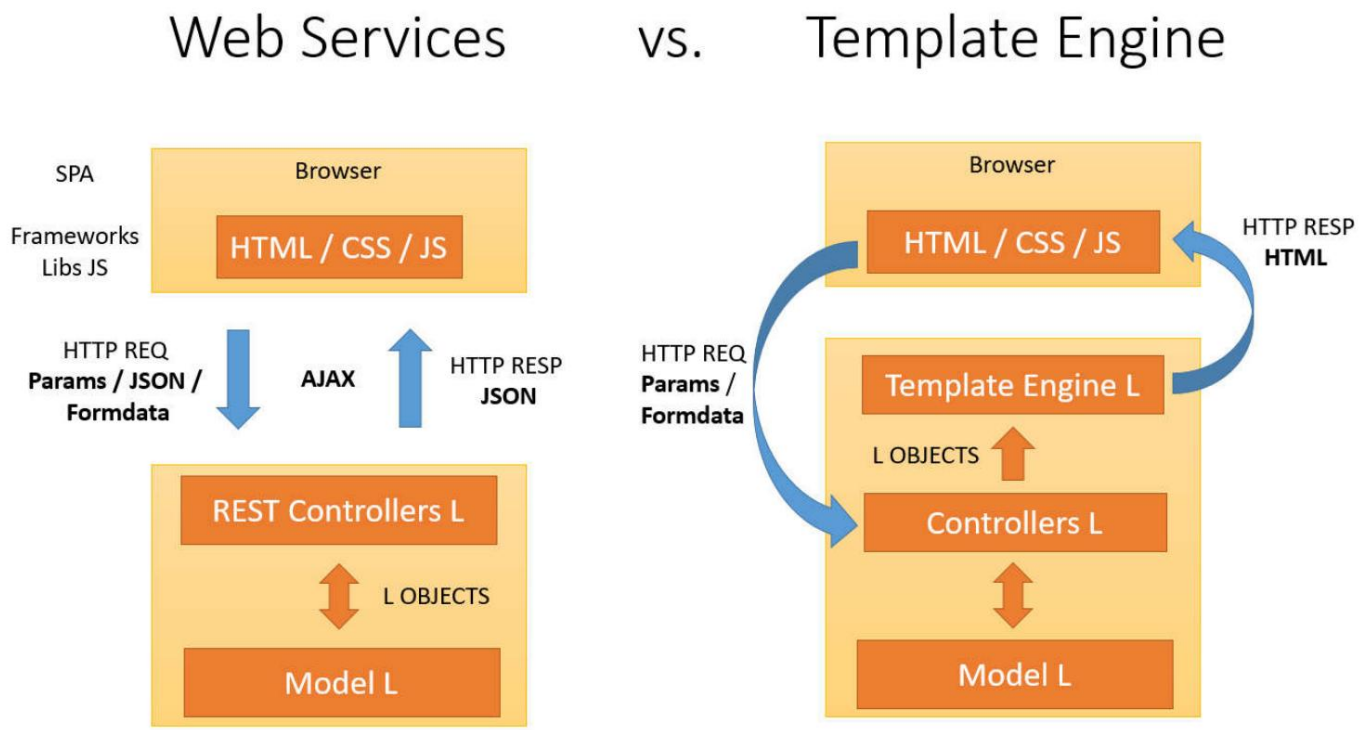
- **verificação:** Arquivo -> Novo -> Projeto -> Visual C# -> Web -> ASP.NET Core Web Application
  - o Criar diretório para solução o Criar novo repositório Git o Aplicativo Web (Model-View-Controller) o (NÃO) autenticação o (NÃO) Habilitar suporte ao Docker o Configurar para HTTPS • Observar pasta do projeto e confirmações • Executar projeto o Com depuração: F5 o Sem depuração: CTRL+F5 ÿ Recarregamento ao vivo É possível parar
    - ÿ o IIS manualmente • Criar repositório Git remoto e enviar projeto

Se esta caixa de diálogo aparecer:



Sim -> Instalar certificado -> Sim

Atualização: aplicativos Web MVC com mecanismo de modelo



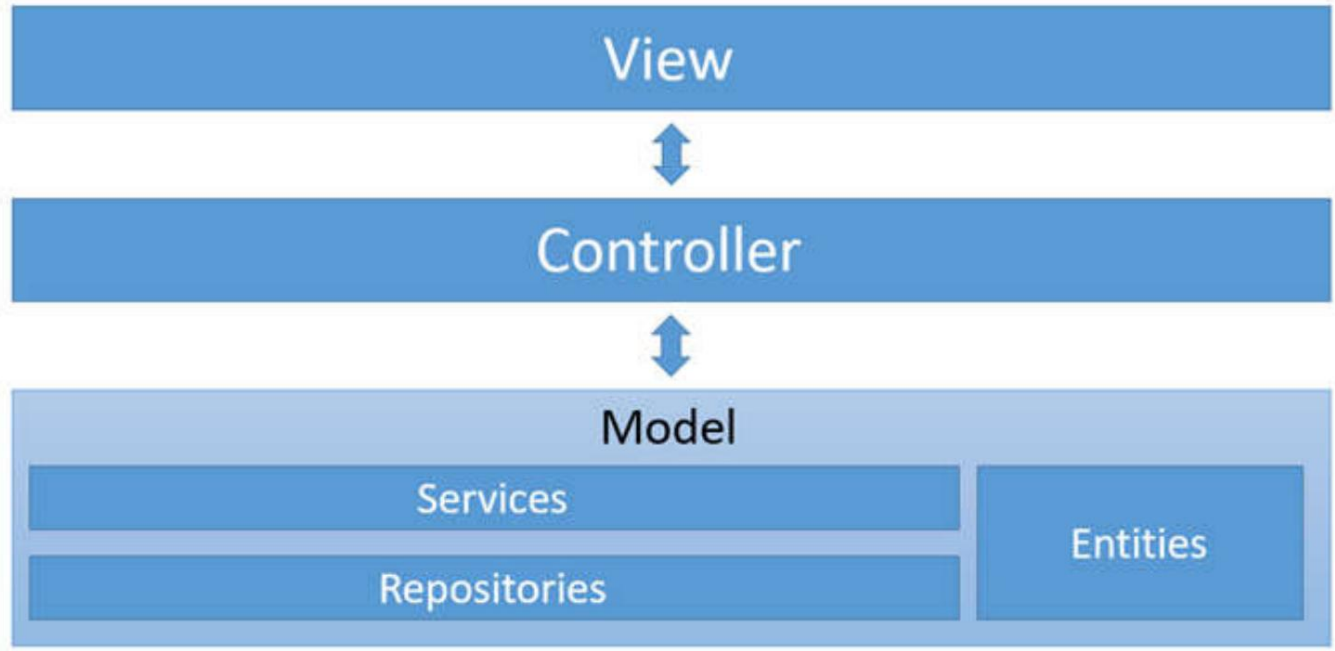
Responsabilidade de cada parte do MVC:

- **Modelo:** estrutura de entidades de domínio e suas transformações (modelo de domínio) o Entidades
  o Serviços
  (relacionados a entidades) ÿ Repositórios
  (acesso persistente a dados)

- **Controladores:** recebe as interações do usuário e as trata

- **Visualizações:** define a estrutura e o comportamento da interface do usuário

Arquitetura geral:



Estrutura do projeto

Lista de controle:

- wwwroot: recursos da aplicação (css, imagens, etc.) • Controllers:
controladores MVC da aplicação • Models: entidades e
"modelos de visão" • Views: páginas (observe
convenções de nomenclatura contra controladores)
  o Compartilhado: visualizações usadas por mais de um controlador
- appsettings.json: configuração de recursos externos (logging, strings de conexão, etc.) • Program.cs: ponto de
entrada • Startup.cs: configuração
do aplicativo

Testes do primeiro controlador e das páginas do Razor

Lista de controle:

- Padrão de rota: Controller / Ação / Id o Cada método do controller é mapeado para uma ação
- Natural Templates • Bloco C# na Razor Page: @{ } • ViewData Dictionary • Tag Helpers in Razor Pages. Exemplos: asp-controller e asp-action • IActionResult

Tipo	Construtor de métodos
VerResultado	Visualizar
PartialViewResult	PartialView
ContentResult	Contente
RedirectResult	redirecionar
RedirectToRouteResult	RedirectToAction Ex: RedirectToAction("Index", "Home", new { page = 1, sortBy = price}))
JsonResult	json
ArquivoResultado	Arquivo
HttpNotFoundResult	HttpNotFound
ResultadoVazio	-

Primeiro Model-Controller-View - Departamento

Lista de controle:

- Crie uma nova pasta ViewModels e mova ErrorViewModel (incluindo namespace)
  - o CTRL+SHIFT+B para corrigir referências
- Criar classe Modelos/Departamento • Criar controlador: botão direito Controladores -> Adicionar -> Controlador -> Controlador MVC Vazio
  - o Name: DepartmentsController **(PLURAL)** o Instanciar uma List<Department> e retorná-la como parâmetro do método View

- Criar nova pasta Vistas/Departamentos **(PLURAL)** • Criar vista: botão direito Vistas/Departamentos -> Adicionar -> Vista
  - o View name: Index o Template: List o Model class: Department o Change Title para Departments o Notice: ÿ @definição de modelo ÿ
    - intellisense para modelo ÿ Métodos auxiliares ÿ @foreach bloco

Excluindo visualização e controlador de departamento

Lista de controle:

• Excluir controlador •

Excluir pasta Vistas/Departamentos

CRUD Andaime

Lista de controle:

• Controladores do botão direito -> Adicionar -> Novo Item Scaffolded
  o Controladores MVC com visualizações, usando Entity Framework o
  Classe de modelo: Departamento o
  Classe de contexto de dados: + e aceite o nome o
  Visualizações (opções): todos os
  três o Nome do controlador: DepartmentsController

Adaptação do MySQL e primeira migração

Observação: estamos usando o fluxo de trabalho CODE-FIRST

Lista de controle:

• Em appsettings.json, defina a string de conexão:
  o "server=localhost;userid=developer;password=1234567;database=saleswebmvcappdb" • Em Startup.cs, corrija a definição de DbContext para o sistema de injeção de dependência:

```
services.AddDbContext<SalesWebMvcAppContext>(options =>
        options.UseMySql(Configuration.GetConnectionString("SalesWebMvcContext"), construtor =>
builder.MigrationsAssembly("SalesWebMvc")));
```

• Instale o provedor MySQL:
  o Abra o console do gerenciador de pacotes NuGet o
  Install-Package Pomelo.EntityFrameworkCore.MySql

• Parar o IIS •
CTRL+SHIFT+B

• Inicie o servidor MySQL:
  o Painel de Controle -> Ferramentas Administrativas -> Serviços •

Inicie o MySQL Workbench

• Package Manager Console -> criar a primeira migração: o Add-
  Migration Initial o Update-
  Database

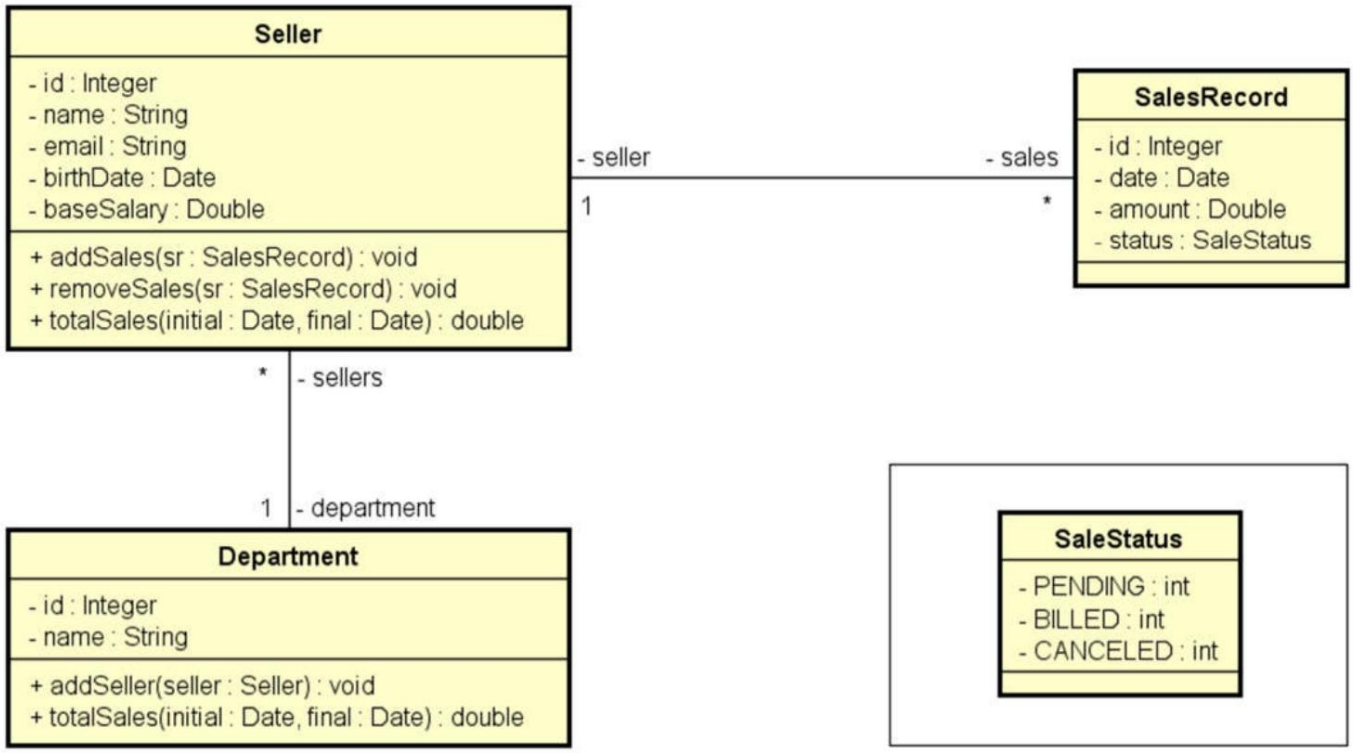
• Verifique o banco de dados no MySQL Workbench
• Aplicativo de teste: CTRL+F5

Mudando de tema

Lista de controle:

- Vá para: http://bootswatch.com/3 (verifique a versão Bootstrap) •
  Escolha um tema •
  Baixe bootstrap.css o Sugestão:
  renomeie para bootstrap-name.css o Salve o arquivo
  em wwwroot/lib/bootstrap/dist/css (cole dentro do Visual Studio)
- Abra _Layout.cshtml o
  Atualize a referência de bootstrap

Outras entidades e segunda migração



Lista de controle:

- Implementar modelo de domínio
  o Atributos básicos o
  Associação (vamos usar **ICollection,** que corresponde a List, HashSet, etc. - **INSTANTIATE!)** o
  Construtores **(padrão** e **com argumentos)** o Métodos
  personalizados
- Adicionar DbSet's em DbContext
- Add-Migration OtherEntities o
  Update-Database

Serviço de semeadura

Lista de

verificação: •

- Pare o IIS em Data, crie SeedingService
- Em Startup.cs, registre SeedingService para o sistema de injeção de dependência Em
- Startup.cs, adicione SeedingService como parâmetro do método Configure. Semente de chamada para perfil de desenvolvimento

VendedoresControlador

Lista de controle:

• Criar links de Departamentos e Vendedores na barra de navegação

• Controlador -> Adicionar -> Controlador -> Controlador MVC - Vazio -> **SellersController** • Criar
pasta Views/Sellers • Views/Sellers ->
Add -> View o Exibir nome: Índice o
Alterar título

SellerService e FindAll básico

Lista de

verificação: • Criar serviços de
pasta • Criar SellerService

- Em Startup.cs, registre SellerService no sistema de injeção de dependência Em
- SellerService, implemente FindAll, retornando List<Seller> Em
- SellersController, implemente o método Index, que deve chamar SellerService.FindAll Em Views/Sellers/
- Index, escreva o código de modelo para mostrar Sellers

• Sugestão: classes de usuário "table-striped table-hover" para tabela • Nota:
vamos aplicar formatação em classes posteriores

Formulário de criação simples

Lista de controle:

- Em Visualizações/Vendedores/Índice, crie um link para "Criar"
- No controlador, implemente a ação GET "Criar"
- Em Visualizações/Vendedores, crie a visualização "Criar"
- Em Services/SellerService, crie o método Insert
- No controlador, implemente a ação POST "Criar"

Referência:

https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery

Chave estrangeira não nula (integridade referencial)

Checklist:

- Em Seller, adicione DepartmentId
- Eliminar banco de dados • Criar nova migração, atualizar banco de dados
- Atualizar SellerService.Insert por enquanto: obj.Department = _context.Department.First();

SellerFormViewModel e componente de seleção de departamento

Lista de controle:

- Criar DepartmentService com o método FindAll
- Em Startup.cs, registre DepartmentService no sistema de injeção de dependência
- Criar SellerFormViewModel
- No controlador:
  - Nova dependência: DepartmentService o Atualize a ação GET "Criar"
- Em Visualizações/Vendedores/Criar:
  - Atualizar tipo de modelo para SellerFormViewModel o Atualizar campos de formulário o Adicionar componente selecionado para DepartmentId

```
<div class="form-group">
    <label asp-for="Seller.DepartmentId" class="control-label"></label> <select asp-
    for="Seller.DepartmentId" asp-items="@(new SelectList(Model.Departments,"Id ",
"Nome"))" class="form-control"></select> </div>
```

- No controlador, atualize a ação POST "Criar" **-> NÃO NECESSÁRIO! :)**
- Em SellerService.Insert, exclua a "Primeira" chamada

Referência: https://stackoverflow.com/questions/34624034/select-tag-helper-in-asp-net-core-mvc

Excluir vendedor

Lista de controle:

- Em SellerService, crie as operações FindById e Remove
- No controlador, crie a ação GET "Excluir"
- Em Exibir/Vendedores/Índice, verifique o link para a ação "Excluir"
- Criar exibição de confirmação de exclusão: Exibir/Vendedores/Excluir • Aplicativo de teste
- No controlador, crie a ação POST "Excluir" • Aplicativo de teste

Detalhes do vendedor e carregamento ansioso

Lista de controle:

- https://docs.microsoft.com/en-us/ef/core/query/related-data

Lista de controle:

- Em Exibir/Vendedores/Índice, verifique o link para a ação "Detalhes"
- No controlador, crie a ação GET "Detalhes"
- Create view: View/Sellers/Details • Include in FindAll: Include(obj => obj.Department) (namespace: Microsoft.EntityFrameworkCore)

Atualizar vendedor e exceção de serviço personalizado

Lista de controle:

- Criar pasta Serviços/Exceções • Criar NotFoundException e DbConcurrencyException
- Em SellerService, crie o método Update
- Em Exibir/Vendedores/Índice, verifique o link para a ação "Editar"
- No controlador, crie a ação GET "Editar"
- Create view: View/Sellers/Edit (similar a Create, plus hidden id) • Test app
- No controlador, crie a ação POST "Editar" • Aplicativo de teste
- **Aviso:** ASP.NET Core seleciona a opção com base em DepartmentId

Retornando a página de erro personalizada

Lista de verificação:

- Atualizar ErrorViewModel • Atualizar Error.cshtml Em SellerController: o Criar ação de erro com parâmetro de mensagem o Atualizar chamadas de método

Localidade do aplicativo, formatação de número e data

Lista de verificação:

- Em Startup.cs, defina opções de localização
  - No vendedor: o Defina rótulos personalizados [Display] o Defina semântica para data [DataType] o Defina formatos de exibição [DisplayFormat]

Validação

Lista de controle:

- Em Vendedor, adicione anotações de validação

```
[Required(ErrorMessage = "{0} obrigatório")]

[EmailAddress(ErrorMessage = "Digite um e-mail válido")]

[Intervalo(100.0, 50000.0, ErrorMessage = "{0} deve ser de {1} a {2}")]
```

- Atualize o HTML para criar e editar a exibição

```
Resumo:
<div asp-validation-summary="All" class="text-danger"></div>


Campo: <span asp-validation-for="Name" class="text-danger">

Validação do lado do cliente:
@section Scripts
    { @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

- Atualizar SellersController

Operações assíncronas usando tarefas (async, await)

Lista de controle:

- Atualizar DepartmentService • Atualizar SellerService • Atualizar SellersController

Tratamento de exceção para exclusão (integridade referencial)

Lista de controle:

- Criar exceção personalizada IntegrityException Em
- SellerService.RemoveAsync, capture DbUpdateException e lance IntegrityException Em SellersController, atualize
- a ação Delete POST

Preparando exibições de navegação de pesquisa de vendas

Lista de controle:

• Criar SalesRecordsController com ação Index, SimpleSearch e GroupingSearch • Criar pastas Views/SalesRecords • Criar visualização Index

com formulários de pesquisa • Criar link

"Sales" na barra de navegação principal •

Criar visualizações SimpleSearch e GroupingSearch

Implementação de pesquisa simples

Lista de controle:

• Crie SalesRecordService com operação FindByDate Em

- Startup.cs, registre SalesRecordService no sistema de injeção de dependência Em
- SalesRecordsController, atualize a ação SimpleSearch •

Atualize a exibição SimpleSearch •

Opcional: formate a data e o número do SalesRecord

Implementando a pesquisa de agrupamento

Lista de controle:

- Em SalesRecordService, crie a operação FindByDateGrouping
- Em SalesRecordsController, atualize a ação GroupingSearch •

Atualize a visualização GroupingSearch