Roteiro de prática para criação de uma aplicação web MVC com .Net 6

criação da aplicação

```
dotnet new mvc -o NomeApp
```

Confiar no certificado HTTPS antes de rodar a aplicação

```
dotnet dev-certs https -trust
```

Rodar aplicação

```
dotnet watch run
```

Criar um modelo
Na pasta “Models” adicionar um novo arquivo com o nome do modelo desejado (NomeModelo.cs), nesse arquivo codificar um modelo padrão

Ex. Cliente.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace HotelApp.Models
{
    public class Cliente
    {
        public int Id {get; set;}
        public string Nome {get; set;}
        public string Cpf {get; set;}
        public string Rg {get; set;}
        public string Endereco {get; set;}
        public string Telefone {get; set;}
        public string Email {get; set;}

        [Display(Name="Data de Nascimento")]
        [DataType(DataType.Date)]
        public DateTime DtNascimento  {get; set;}
    }
}
```

Ex.: Quarto.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace HotelApp.Models
{
    public class Quarto
    {
        public int Id {get; set;}
        public int Numero  {get; set;}
        public string Descricao  {get; set;}
        public String TipoQuarto  {get; set;}
        public double PrecoQuarto  {get; set;}
        public string Localizacao {get; set;}
        public string StatusQuarto {get; set;}

    }
}
```

Ex.: Reserva.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace HotelApp.Models
{
    public class Reserva
    {
        public int Id {get; set;}
        public int ClienteID {get; set;}
        public int QuartoID {get; set;}
        [DataType(DataType.Date)]
        public DateTime DataCheckIn {get; set;}
        [DataType(DataType.Date)]
        public DateTime DataCheckOut {get; set;}
        public string Status {get; set;}

        //Relacionamentos
        public virtual Cliente Cliente {get; set;}
        public virtual Quarto Quarto {get; set;}


    }
}
```

Adicionar pacotes necessários

```
dotnet tool uninstall --global dotnet-aspnet-codegenerator
dotnet tool install --global dotnet-aspnet-codegenerator
dotnet tool uninstall --global dotnet-ef
dotnet tool install --global dotnet-ef
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet add package Microsoft.EntityFrameworkCore.SQLite
dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
```

Para MAC e Linux exportar o path das ferramentas

```
export PATH=$HOME/.dotnet/tools:$PATH
```

Fazer o Scaffold para cada modelo

```
dotnet-aspnet-codegenerator controller -name MoviesController -m Movie -
dc MvcMovieContext --relativeFolderPath Controllers --useDefaultLayout --
referenceScriptLibraries -sqlite
```

Remover a seguinte linha do arquivo MvcMovie.csproj

```
<Nullable>enable</Nullable>
```

Criar a primeira migration

```
dotnet ef migrations add InitialCreate
dotnet ef database update
```

Injeção de Dependência: Injetar o context da base da dado na inicialização da aplicação
Arquivo Program.cs

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<MvcMovieContext>(options =>

options.UseSqlite(builder.Configuration.GetConnectionString("MvcMovieCont
ext")));
```

Arquivo appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "MvcMovieContext": "Data Source=MvcMovie.db"
  }
}}
```

Injetar o context no construtor do controlador Controllers/MoviesController.cs

```
public class MoviesController : Controller
{
    private readonly MvcMovieContext _context;

    public MoviesController(MvcMovieContext context)
    {
        _context = context;
    }
```

Alteração no View para uma lista de objetos seguindo modelo

```
@model IEnumerable<MvcMovie.Models.Movie>

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Title)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ReleaseDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Genre)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Price)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.ReleaseDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Genre)
```

```
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Price)
            </td>
            <td>
                <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
                <a asp-action="Details" asp-route-
id="@item.Id">Details</a> |
                <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
            </td>
        </tr>
}
    </tbody>
</table>
```

Decorators e restrições – Arquivo Models/Movie.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace MvcMovie.Models
{
    public class Movie
    {
        public int Id { get; set; }
        public string? Title { get; set; }

        [Display(Name = "Release Date")]
        [DataType(DataType.Date)]
        public DateTime ReleaseDate { get; set; }
        public string? Genre { get; set; }

        [Column(TypeName = "decimal(18, 2)")]
        public decimal Price { get; set; }
    }
}
```

No Controller. Obter dados relacionados ao objeto

Método details

```
 var reserva = await _context.Reserva
                .Include(r => r.Cliente)
                .Include(r => r.Quarto)
                .FirstOrDefaultAsync(m => m.Id == id);
```

Método Create - Get

```
public IActionResult Create()
        {
            ViewData["ClienteID"] = new SelectList(_context.Cliente,
"Id", "Nome");
            ViewData["QuartoID"] = new SelectList(_context.Quarto, "Id",
"Numero");
            return View();
        }
```

Método Create – Post

```
 public async Task<IActionResult>
Create([Bind("Id,ClienteID,QuartoID,DataCheckIn,DataCheckOut,Status")]
Reserva reserva)
        {
            if (ModelState.IsValid)
            {
                _context.Add(reserva);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["ClienteID"] = new SelectList(_context.Cliente,
"Id", "Nome", reserva.ClienteID);
            ViewData["QuartoID"] = new SelectList(_context.Quarto, "Id",
"Numero", reserva.QuartoID);
            return View(reserva);
        }
```

Método Edit – Post

```
public async Task<IActionResult> Edit(int id,
[Bind("Id,ClienteID,QuartoID,DataCheckIn,DataCheckOut,Status")] Reserva
reserva)
        {
            if (id != reserva.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(reserva);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!ReservaExists(reserva.Id))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["ClienteID"] = new SelectList(_context.Cliente,
"Id", "Nome", reserva.ClienteID);
            ViewData["QuartoID"] = new SelectList(_context.Quarto, "Id",
"Numero", reserva.QuartoID);
            return View(reserva);
        }
```

No View, arquivo Edit.cshtml

```
<select asp-for="ClienteID" class="form-control" asp-
items="ViewBag.ClienteID"></select>
                <span asp-validation-for="ClienteID" class="text-
danger"></span>

No View, arquivo Index.cshtml
<td>
                @Html.DisplayFor(modelItem => item.Cliente.Nome)
</td>
```