**Criando uma solução em branco:**

BLANK SOLUTION

* Iremos criar uma pasta de trabalho vazia para incluir os projetos.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Escolha a opção: **Solução em branco**

Solução vazia sem projetos

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Crie as seguintes pastas na solution:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

1 – Presentation Layer

Camada de apresentação.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

/Program.cs

Classe responsável por executar o projeto. No caso de aplicações web, ele faz com que o projeto seja executado no servidor web selecionado (em caso de desenvolvimento local este servidor é o IIS – INTERNET INFORMATION SERVICE).

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ProjetoMVC01.Presentation

{

**public class Program**

**{**

**public static void Main(string[] args)**

**{**

**CreateHostBuilder(args).Build().Run();**

**}**

**public static IHostBuilder CreateHostBuilder(string[] args) =>**

**Host.CreateDefaultBuilder(args)**

**.ConfigureWebHostDefaults(webBuilder =>**

**{**

**webBuilder.UseStartup<Startup>();**

**});**

**}**

}

/appsettings.json

Arquivo de configuração que iremos utilizar no projeto .NET para mapear parâmetros de configuração do sistema, como por exemplo:

* Mapeamento da string de conexão do banco de dados.
* Mapeamento de parâmetros de envio de email
* Mapeamento de parâmetros para geração de logs do sistema
* Etc.

**{**

**"Logging": {**

**"LogLevel": {**

**"Default": "Information",**

**"Microsoft": "Warning",**

**"Microsoft.Hosting.Lifetime": "Information"**

**}**

**},**

**"AllowedHosts": "\*"**

**}**

/Startup.cs

Classe onde iremos configurar as preferencias de inicialização do projeto NET CORE. Aqui iremos mapear como o projeto devera funcionar em tempo de execução.

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ProjetoMVC01.Presentation

{

**public class Startup**

**{**

**// This method gets called by the runtime.**

**// Use this method to add services to the container.**

**// For more information on how to configure your application,**

**// visit https://go.microsoft.com/fwlink/?LinkID=398940**

**public void ConfigureServices(IServiceCollection services)**

**{**

**}**

**// This method gets called by the runtime.**

**// Use this method to configure the HTTP request pipeline.**

**public void Configure(IApplicationBuilder app, IWebHostEnvironment env)**

**{**

**if (env.IsDevelopment())**

**{**

**app.UseDeveloperExceptionPage();**

**}**

**app.UseRouting();**

**app.UseEndpoints(endpoints =>**

**{**

**endpoints.MapGet("/", async context =>**

**{**

**await context.Response.WriteAsync("Hello World!");**

**});**

**});**

**}**

**}**

}

**Executando:**

Padrão do plano de fundo, Tabela

Descrição gerada automaticamente com confiança média

Padrão MVC – MODEL, VIEW e CONTROLLER

Consiste em um padrão de desenvolvimento   
de aplicações web baseado em 3 camadas:

* **MODEL**

Camada responsável pelos dados de entrada / saída do sistema web.

* **VIEW**

Camada que representa as páginas web do projeto, composta de **HTML**, **CSS** e **JAVASCRIPT** (**FRONTEND**)

* **CONTROLLER**

Camada que gerencia as páginas (VIEWS) do projeto, recebendo as requisições e retornando as respostas.

No NET CORE, para que possamos utilizar o padrão MVC, precisamos criar as pastas abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Em NET CORE MVC, o padrão para criarmos a página inicial   
do projeto é colocá-la com o endereço: /Home/Index

**Onde:**

/Home/Index

**Nome do controlador Nome da página**

(Classe de controle (Arquivo de extensão .cshtml

Que gerencia a página) que será aberto no navegador)

Primeiro, iremos criar o controlador:

/Controllers/HomeController.cs

\*\* **Regra**: O nome das classes de controle sempre   
deve terminar com a palavra Controller.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ProjetoMVC01.Presentation.Controllers

{

**public class HomeController : Controller**

**{**

**//método para abrir a página /Home/Index**

**public IActionResult Index()**

**{**

**return View();**

**}**

**}**

}

Criando a página **Index**:

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Páginas Razor

É o nome dado a tecnologia utilizado em NET CORE MVC para criação e desenvolvimento das páginas web. São páginas que possuem a extensão **.cshtml (junção de C# + HTML)**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Página criada:**

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

**Organização dos controllers e views:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**@{**

**Layout = null;**

**}**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta name="viewport" content="width=device-width" />**

**<title>Index</title>**

**</head>**

**<body>**

**<h1>Bem vindo ao projeto!</h1>**

**</body>**

**</html>**

Precisamos configurar o projeto para, ao ser executado,   
abrir no navegador a página **/Home/Index** como página inicial.

/Startup.cs

É a classe de configuração do projeto NET CORE onde iremos definir o modo de navegação do projeto e a página inicial que deverá ser exibida quando entrarmos ou executarmos o projeto.

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ProjetoMVC01.Presentation

{

public class Startup

{

// This method gets called by the runtime.

// Use this method to add services to the container.

// For more information on how to configure your application,

// visit https://go.microsoft.com/fwlink/?LinkID=398940

public void ConfigureServices(IServiceCollection services)

{

//definir que o modo de navegação do projeto

//web é MVC (Controllers e Views)

**services.AddControllersWithViews();**

}

// This method gets called by the runtime.

// Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseRouting();

**//mapeando a página inicial do projeto**

**app.UseEndpoints(endpoints =>**

**{**

**endpoints.MapControllerRoute(**

**name: "default",**

**pattern: "{controller=Home}/{action=Index}"**

**);**

**});**

}

}

}

**Executando:**

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

**Desenvolvimento FRONTEND WEB:**

Utilização das linguagens: HTML, CSS e JAVASCRIPT.

**HTML**

Utilizado para criação do conteúdo das páginas web.

**CSS**

Utilizado para folhas de estilo (formatação, estilização, responsividade).

**JAVASCRIPT**

Utilizado para implementar os eventos e comportamentos da página de acordo com as interações do usuário.

Para a parte de CSS iremos utilizar um toolkit   
(biblioteca de estilos) chamada Bootstrap.

<https://getbootstrap.com/>

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

<https://getbootstrap.com/docs/5.1/getting-started/download/>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

No projeto NET CORE, a pasta onde deverão ficar

os arquivos CSS e JS deve ter o nome de /wwwroot

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Cole as pastas CSS e JS do bootstrap dentro de wwwroot:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Para que o projeto NET CORE MVC possa reconhecer o uso da pasta **wwwroot**, precisamos adicionar a seguinte linha de código na classe Startup:

/Startup.cs

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ProjetoMVC01.Presentation

{

public class Startup

{

// This method gets called by the runtime.

// Use this method to add services to the container.

// For more information on how to configure your application,

// visit https://go.microsoft.com/fwlink/?LinkID=398940

public void ConfigureServices(IServiceCollection services)

{

//definir que o modo de navegação do projeto

//web é MVC (Controllers e Views)

services.AddControllersWithViews();

}

// This method gets called by the runtime.

// Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

**//habilitar a pasta /wwwroot**

**app.UseStaticFiles();**

app.UseRouting();

//mapeando a página inicial do projeto

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}"

);

});

}

}

}

/Views/Home/Index.cshtml

Adicionando os arquivos CSS e JS do bootstrap.

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

**<!-- folha de estilo CSS -->**

**<link href="~/css/bootstrap.css" rel="stylesheet" />**

</head>

<body>

<h1>Bem vindo ao projeto!</h1>

**<!-- arquivos javascript -->**

**<script src="~/js/bootstrap.js"></script>**

</body>

</html>

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Página de Layout mestre

Em NET CORE MVC podemos criar páginas de layout que sejam   
utilizadas como templates para outras páginas do projeto.

No NET CORE MVC, em geral, páginas de layout são criadas

dentro do seguinte caminho: /Views/Shared

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

Dentro de **/Views/Shared** iremos criar uma página que será utilizada como layout do projeto (template para as demais páginas do projeto).

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

/Views/Shared/Layout.cshtml

Desenhando a página de layout do sistema.

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informática</title>

<!-- folhas de estilo CSS -->

<link href="~/css/bootstrap.css" rel="stylesheet" />

</head>

<body>

<!-- menu do sistema -->

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Sistema de Controle de Tarefas</a>

<button class="navbar-toggler" type="button" data-bs-

toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page"

href="#">Página inicial</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#"

id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

Gerenciar minhas tarefas

</a>

<ul class="dropdown-menu" aria-

labelledby="navbarDropdown">

<li><a class="dropdown-item" href="#">

Cadastrar tarefas</a></li>

<li><a class="dropdown-item" href="#">

Consultar tarefas</a></li>

<li><hr class="dropdown-divider"></li>

<li><a class="dropdown-item" href="#">

Relatório de tarefas</a></li>

</ul>

</li>

</ul>

</div>

</div>

</nav>

<!-- definir o local onde será inserido o conteudo das demais páginas -->

**@RenderBody()**

<!-- arquivos javascript -->

<script src="~/js/bootstrap.js"></script>

</body>

</html>

/Views/Home/Index.cshtml

Fazendo a página Index possa HERDAR o layout.

**@{**

**Layout = "~/Views/Shared/Layout.cshtml";**

**}**

**<div class="container mt-4">**

**<h5>Seja bem vindo ao projeto</h5>**

**<p>Sistema desenvolvido em NET CORE MVC com DAPPER e SqlServer</p>**

**</div>**

Executando:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Criando as demais páginas do projeto:**

* /Tarefas/Cadastro
* /Tarefas/Consulta
* /Tarefas/Relatorio

Segundo o padrão MVC, o caminho das páginas é composto do nome do controlador e em seguida o nome da página (VIEW), por exemplo:

/Tarefas/Cadastro

/Tarefas/Consulta

/Tarefas/Relatorio

**Nome do controlador Nome das páginas**

(TarefasController) (Arquivos .cshtml)

Primeiro, iremos criar o controlador:

/Controllers/TarefasController.cs

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ProjetoMVC01.Presentation.Controllers

{

**public class TarefasController : Controller**

**{**

**//método para abrir a página /Tarefas/Cadastro**

**public IActionResult Cadastro()**

**{**

**return View();**

**}**

**//método para abrir a página /Tarefas/Consulta**

**public IActionResult Consulta()**

**{**

**return View();**

**}**

**//método para abrir a página /Tarefas/Relatorio**

**public IActionResult Relatorio()**

**{**

**return View();**

**}**

**}**

}

**Criando as páginas:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**@{**

**Layout = "~/Views/Shared/Layout.cshtml";**

**}**

**<div class="container mt-4">**

**<h5>Cadastro de tarefas</h5>**

**<p>Preencha o formulário abaixo para incluir uma tarefa.</p>**

**</div>**

**@{**

**Layout = "~/Views/Shared/Layout.cshtml";**

**}**

**<div class="container mt-4">**

**<h5>Consulta de tarefas</h5>**

**<p>Informe o período de datas para consultar suas tarefas.</p>**

**</div>**

**@{**

**Layout = "~/Views/Shared/Layout.cshtml";**

**}**

**<div class="container mt-4">**

**<h5>Relatório de tarefas</h5>**

**<p>Informe o período de datas e o tipo de relatório desejado.</p>**

**</div>**

Aplicativo

Descrição gerada automaticamente com confiança baixa

Voltando na página de layout do projeto:

/Views/Shared/Layout.cshtml

* Adicionando nos links do menu o caminho   
  para abrir as páginas do sistema:

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>COTI Informática</title>

<!-- folhas de estilo CSS -->

<link href="~/css/bootstrap.css" rel="stylesheet" />

</head>

<body>

<!-- menu do sistema -->

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Sistema de Controle de Tarefas</a>

<button class="navbar-toggler" type="button" data-bs-

toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page"

**href="/Home/Index"**>Página inicial</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#"

id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

Gerenciar minhas tarefas

</a>

<ul class="dropdown-menu" aria-

labelledby="navbarDropdown">

<li><a class="dropdown-item"

**href="/Tarefas/Cadastro"**>

Cadastrar tarefas</a></li>

<li><a class="dropdown-item"

**href="/Tarefas/Consulta"**>

Consultar tarefas</a></li>

<li><hr class="dropdown-divider"></li>

<li><a class="dropdown-item"

**href="/Tarefas/Relatorio"**>

Relatório de tarefas</a></li>

</ul>

</li>

</ul>

</div>

</div>

</nav>

<!-- definir o local onde será inserido o conteudo das demais páginas -->

@RenderBody()

<!-- arquivos javascript -->

<script src="~/js/bootstrap.js"></script>

</body>

</html>

**Executando:**

Abrindo as páginas:

<http://localhost:35152/Home/Index>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

<http://localhost:35152/Tarefas/Cadastro>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

<http://localhost:35152/Tarefas/Consulta>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

<http://localhost:35152/Tarefas/Relatorio>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Criando a camada de acesso a banco de dados:

Repository Layer

Interface gráfica do usuário, Aplicativo, PowerPoint

Descrição gerada automaticamente

Esta camada de repositório de dados (acesso a banco de dados) será criada como um projeto do tipo **Biblioteca de Classes (Class Library)**

Este tipo de projeto é apenas um componente que irá conter classes para serem utilizadas por outras camadas do projeto (Biblioteca)

2 – Repository Layer

Camada de repositório.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Biblioteca de Classes**

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Projeto criado:**

Adicione as pastas abaixo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

/Entities/Tarefa.cs

Criando a classe de entidade inicial do projeto.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ProjetoMVC01.Repository.Entities

{

**/// <summary>**

**/// Classe de entidade para modelagem de Tarefa**

**/// </summary>**

**public class Tarefa**

**{**

**#region Propriedades**

**public Guid IdTarefa { get; set; }**

**public string Nome { get; set; }**

**public string Descricao { get; set; }**

**public DateTime Data { get; set; }**

**public TimeSpan Hora { get; set; }**

**#endregion**

**}**

}

Iremos categorizar as tarefas por Prioridade, teremos tarefas de   
**BAIXA**, **MÉDIA** ou **ALTA** prioridade. Para isso vamos criar um **ENUM**.

/Enums/PrioridadeTarefa.cs

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace ProjetoMVC01.Repository.Enums**

**{**

**/// <summary>**

**/// Tipo multivalorado para definir as prioridades de uma tarefa**

**/// </summary>**

**public enum PrioridadeTarefa**

**{**

**BAIXA = 1,**

**MEDIA = 2,**

**ALTA = 3**

**}**

**}**

**Relacionando a entidade Tarefa ao ENUM:**

using ProjetoMVC01.Repository.Enums;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ProjetoMVC01.Repository.Entities

{

**/// <summary>**

**/// Classe de entidade para modelagem de Tarefa**

**/// </summary>**

**public class Tarefa**

**{**

**#region Propriedades**

**public Guid IdTarefa { get; set; }**

**public string Nome { get; set; }**

**public string Descricao { get; set; }**

**public DateTime Data { get; set; }**

**public TimeSpan Hora { get; set; }**

**public PrioridadeTarefa Prioridade { get; set; }**

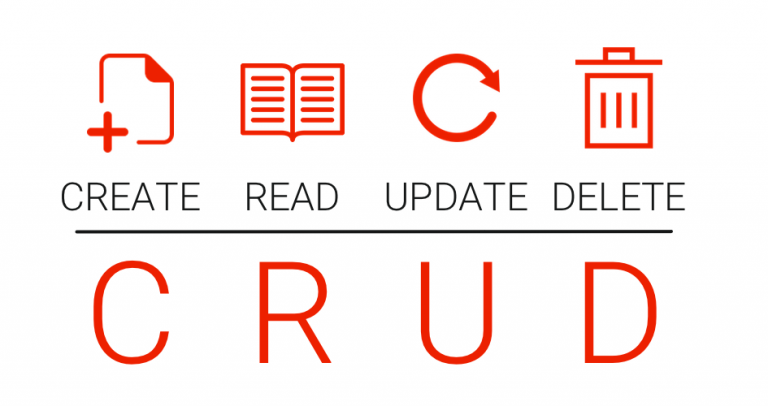
**#endregion**

**}**

}

**Criando as interfaces para construir o repositório:**

Primeiro, iremos criar uma interface genérica para definir os métodos   
base do repositório (CRUD – CREATE, READ, UPDATE e DELETE)



/Interfaces/IBaseRepository.cs

Criando a interface genérica.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace ProjetoMVC01.Repository.Interfaces**

**{**

**/// <summary>**

**/// Interface genérica para definir os métodos base do repositorio**

**/// </summary>**

**/// <typeparam name="T">Tipo generico para a entidade</typeparam>**

**public interface IBaseRepository<T>**

**{**

**#region Métodos abstratos**

**void Create(T obj);**

**void Update(T obj);**

**void Delete(T obj);**

**List<T> GetAll();**

**T GetById(Guid id);**

**#endregion**

**}**

**}**

Em seguida, iremos criar uma interface específica para **TarefaRepository**

/Interfaces/ITarefaRepository.cs

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

using ProjetoMVC01.Repository.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ProjetoMVC01.Repository.Interfaces

{

**/// <summary>**

**/// Interface de repositorio especifica para Tarefa**

**/// </summary>**

**public interface ITarefaRepository : IBaseRepository<Tarefa>**

**{**

**#region Métodos abstratos**

**List<Tarefa> GetByDatas(DateTime dataMin, DateTime dataMax);**

**#endregion**

**}**

}

Criando a classe de repositório para Tarefa

/repositories/TarefaRepository.cs

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**using ProjetoMVC01.Repository.Entities;**

**using ProjetoMVC01.Repository.Interfaces;**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace ProjetoMVC01.Repository.Repositories**

**{**

**/// <summary>**

**/// Classe de repositorio de dados para Tarefa**

**/// </summary>**

**public class TarefaRepository : ITarefaRepository**

**{**

**public void Create(Tarefa obj)**

**{**

**throw new NotImplementedException();**

**}**

**public void Update(Tarefa obj)**

**{**

**throw new NotImplementedException();**

**}**

**public void Delete(Tarefa obj)**

**{**

**throw new NotImplementedException();**

**}**

**public List<Tarefa> GetAll()**

**{**

**throw new NotImplementedException();**

**}**

**public Tarefa GetById(Guid id)**

**{**

**throw new NotImplementedException();**

**}**

**public List<Tarefa> GetByDatas(DateTime dataMin, DateTime dataMax)**

**{**

**throw new NotImplementedException();**

**}**

**}**

**}**

**Estrutura do projeto Repository:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Continua…