Exercício Aula 11

Prof: Henrique Augusto Maltauro

Codificar Back-end de Aplicações Web

Exercício

Criar uma API para cadastro e consulta de produtos e categorias.

- Pasta Domain
  - Produto e Categoria
- Pasta Handlers
  - ProdutoHandler e CategoriaHandler
- Pasta Controllers
  - ProdutoController e CategoriaController

Exercício

- Classe Categoria

Deve possuir os seguintes atributos:

- public long Id
- public string Descricao

Exercício

- Classe Categoria

Não precisa do método construtor, e todos os atributos devem estar marcados com [JsonPropertyName].

Exercício

- Classe CategoriaHandler

Deve possuir uma herança de ControllerBase.

Deve possuir os seguintes atributos:

- private static List<Categoria> _categorias

Exercício

- Classe CategoriaHandler

Deve possuir os seguintes métodos:

- public IActionResult GetById(long id)
- public IActionResult Post(Categoria categoria)

Exercício

- Classe CategoriaHandler

Não precisa do método construtor.

O método GetById deve receber um id e consultar as categorias na lista _categorias usando LINQ.

Se a categoria não existir, retornar um IActionResult NotFound, informando que a categoria não existe. Se tudo der certo, retornar um IActionResult Ok com a categoria consultada.

Exercício

- Classe CategoriaHandler

O método Post deve receber um Categoria e salvar esse Categoria na lista _categorias.

Se já existir uma categoria com aquele Id, retornar um IActionResult Conflict, informando que a categoria já existe. Se tudo der certo, retornar um IActionResult Created.

Exercício

- Controller CategoriaController

Deve possuir os seguintes atributos:

- private CategoriaHandler _handler

Exercício

- Controller CategoriaController

Deve possuir os seguintes métodos:

- public CategoriaController (Método construtor)

Exercício

- Controller CategoriaController

Deve possuir as seguintes rotas:

- [HttpGet]
  [Route("{id}")]
  public async Task<IActionResult> GetById(long id)

Exercício

- Controller CategoriaController

Deve possuir as seguintes rotas:

- [HttpPost]
  [Route("novo")]
  public async Task<IActionResult> Post([FromBody]
  Categoria categoria)

Exercício

- Controller CategoriaController

O método construtor deve inicializar o _handler.

A rota GetById deve receber um id e usá-lo para chamar o método GetById do _handler.

A rota Post deve receber a Categoria por body params (JSON), e usá-lo para chamar o método Post do _handler.

Exercício

- Classe Produto

Deve possuir os seguintes atributos:

- public long Id
- public string Nome
- public DateTime Validade
- public long IdCategoria
- public Categoria? Categoria

Exercício

- Classe Produto

Não precisa do método construtor, e todos os atributos devem estar marcados com [JsonPropertyName].

Exercício

- Classe ProdutoHandler

Deve possuir uma herança de ControllerBase.

Deve possuir os seguintes atributos:

- private static List<Produto> _produtos

Exercício

- Classe ProdutoHandler

Deve possuir os seguintes métodos:

- public IActionResult GetById(long id)
- public IActionResult Post(Produto produto)
- public IActionResult GetNaoVencidos()

Exercício

- Classe ProdutoHandler

Não precisa do método construtor.

O método GetById deve receber um id do produto e consultar os produtos na lista _produtos usando LINQ.

Se o produto não existir, retornar um IActionResult NotFound, informando que o produto não existe. Se tudo der certo, retornar um IActionResult Ok com o produto consultado.

Exercício

- Classe ProdutoHandler

O método Post deve receber um Produto e salvar esse Produto na lista _produtos.

Nesse ponto, o produto vai estar com a categoria vazia, e com o Id da categoria ele deve consultar a categoria, e adicionar esta categoria no produto antes de salvar.

Exercício

- Classe ProdutoHandler

Ainda no método Post, se a categoria não existir, retornar um IActionResult NotFound, informando que a categoria informada não existe.

Se já existir um produto com aquele Id, retornar um IActionResult Conflict, informando que o produto já existe. Se tudo der certo, retornar um IActionResult Created.

Exercício

- Classe ProdutoHandler

O método GetNaoVencidos deve consultar os produtos não vencidos na lista _produtos usando LINQ, tendo como base o dia atual.

Se não existir produtos não vencidos, retornar um IActionResult NoContent. Se tudo der certo, retornar um IActionResult Ok com os produtos consultados.

Exercício

- Controller ProdutoController

Deve possuir os seguintes atributos:

- private ProdutoHandler _handler

Exercício

- Controller ProdutoController

Deve possuir os seguintes métodos:

- public ProdutoController (Método construtor)

Exercício

- Controller ProdutoController

Deve possuir as seguintes rotas:

- [HttpGet]
  [Route("{id}")]
  public async Task<IActionResult> GetById(long id)

Exercício

- Controller ProdutoController

Deve possuir as seguintes rotas:

- [HttpPost]
  [Route("novo")]
  public async Task<IActionResult> Post([FromBody]
  Produto produto)

Exercício

- Controller ProdutoController

Deve possuir as seguintes rotas:

- [HttpGet]
  [Route(“nao_vencidos”)]
  public async Task<IActionResult> GetNaoVencidos()

Exercício

- Controller ProdutoController

O método construtor deve inicializar o _handler.

A rota GetById deve receber um id e usá-lo para chamar o método GetById do _handler.

Exercício

- Controller ProdutoController

A rota Post deve receber o Produto por body params (JSON), e usá-lo para chamar o método Post do _handler. **A categoria do JSON do produto não deve ser informada.**

A rota GetNaoVencidos deve chamar o método GetNaoVencidos do _handler.