LABORATÓRIO DA PROGRAMAÇÃO

Nome: Eduarda dos Santos Matos
Turma:3N
Data: 03/12/2024

Avaliação Semestral

A avaliação semestral se trate de uma aplicação web API que contemplam todos os ensinamentos passados atem o então momento, utilizando dos conceitos de boas praticas, pensando em um fácil entendimento e facilitando possíveis manutenções futuras.

A tarefa dada aos alunos era criar uma web api para gerenciar Pedidos e Fornecedores, na qual deveria ter as funções de criar, atualizar, listar por ID, listar todos e deletar.

No desenvolvimento desta aplicação, nosso objetivo não é apenas cumprir os requisitos do exercício, mas também criar uma base sólida para a construção de APIs RESTful que sejam fáceis de manter. Estamos aplicando conceitos importantes que são essenciais para no mercado de trabalho, garantindo que a solução final seja eficiente.

Boas Praticas

Uso Correto dos Verbos HTTP

- **GET**: Utilizado para buscar dados de um recurso específico. Tanto para buscar uma lista completa quanto para um dado específico de uma lista.

```
[HttpGet]
0 referências
public async Task <ActionResult<List<Pedido>>> Get()
{
    var pedidos = await _repository.GetTodosPedidos();
    return Ok(pedidos);
}
```

```
[HttpGet("{id}")]
0 referências
public async Task< ActionResult<Pedido>> Get(int id)
{
    var pedido = await _repository.GetPedido(id);
        if (pedido == null)
        return NotFound();
    return Ok(pedido);
}
```

POST: Usado para criar um novo recurso.

```
[HttpPost]
0 referências
public async Task <ActionResult> Post(Pedido pedido)
{
    await _repository.AdicionarPedido(pedido);
    return Created();
}
```

PUT: Empregado para atualizar um recurso existente.

```
[HttpPut("{id}")]
0 referências
public async Task <ActionResult<Pedido>> Put(int id, Pedido pedidoAtualizado)
{
   var pedido = await _repository.AtualizarPedido(id,pedidoAtualizado);
    if(pedido==null)
    return NotFound();


    return Ok(pedido);
}
```

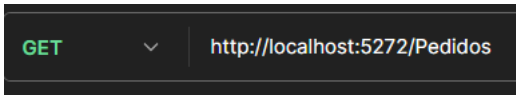
DELETE: Serve para deletar um recurso.

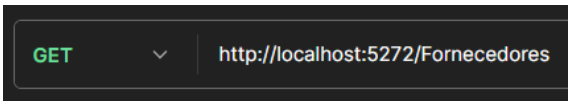
```
[HttpDelete("{id}")]
0 referências
public async Task< ActionResult> Delete(int id)
{
    var pedidoParaRemover = await _repository.DeletarPedido(id);
    if (pedidoParaRemover == null)
        return NotFound();


    return NoContent();
```

URLs intuitivas e fáceis

As URLs devem ser descritivas e fáceis de entender. As utilizadas nesta avaliação são um exemplo disto.





Uso Correto dos Códigos de Status HTTP

Os códigos de status HTTP informam o cliente sobre o resultado da solicitação. Exemplos utilizados nesta aplicação:

200 OK: Requisição bem-sucedida.

404 Not Found: O recurso solicitado não foi encontrado.

204 No Content: A requisição foi processada com sucesso.

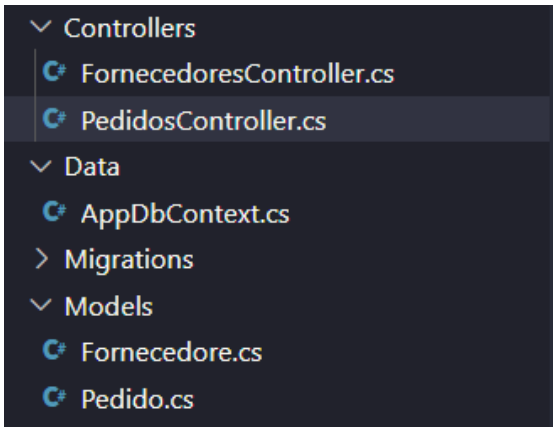
Respostar Conscientes

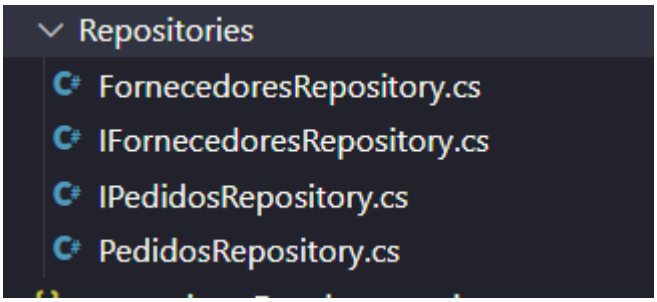
A resposta da API deve ser consistente e sempre retornar um formato padrão.


```
"id": 6,
"data": "0001-01-01T00:00:00",
"valorTotal": 12121.0,
"status": "pendente",
"descricao": "celular"
```


Separação entre lógica de controle e acesso a dados

Usar o padrão Repository para separar a lógica de acesso ao banco de dados da lógica de controle. Isso torna o código mais organizado e facilita a manutenção e testes.





A criação de uma Web API para gerenciar o cadastro de Pedidos e Fornecedores ofereceu uma ótima oportunidade para aplicar os conceitos de desenvolvimento de APIs RESTful, como o uso adequado de métodos HTTP, a persistência de dados com Entity Framework e a implementação de boas práticas, como o padrão Repository e a injeção de dependência.

Por meio deste projeto foi possível entender como esses conceitos de APIs são importantes para resolver problemas do mundo real, permitindo criar sistemas mais eficientes e fáceis de modificar quando necessário.

Foi um aprendizado de forma prática que auxiliou na fixação de um tema foi tratado em aula, possibilitando o melhor entendimento e compreensão.