FIAP GRADUAÇÃO

FIAP

ENTERPRISE APPLICATION DEVELOPMENT

Prof. Me. Thiago T. I. Yamamoto

#08 – WEB API

 *thiagoyama@gmail.com*  *thiagoyama*

TRAJETÓRIA

- Plataforma .NET
- Linguagem C# e Orientação a Objetos
- ASP.NET Core – Rotas e Controller
- ASP.NET Core – Razor e Tag Helpers
- ASP.NET Core – Layout e Partial Views
- Entity Framework Core
- Entity Framework Core - Relacionamentos
- ASP.NET Core – WEB API

#09 - AGENDA

- Arquitetura REST
- Métodos HTTP
- Respostas HTTP
- Projeto
- Controller
  - GET
  - POST
  - PUT
  - DELETE

WEB COMO MEIO DE COMUNICAÇÃO

MODELO ARQUITETURAL REST

- Atualmente, uma nova abordagem de construção de webservices vem sendo utilizada:
- **Web Services RESTFul (REpresentational State Transfer)**
  - Simples, leve, fácil de desenvolver e evoluir;
  - Tudo é um recurso (Resource);
  - Cada recurso possui um identificador (URI);
  - Recursos podem ser de vários formados: html, xml, json;
  - Protocolo HTTP;
  - Os métodos HTTP: GET, POST, PUT, DELETE são utilizados na arquitetura REST.

MÉTODOS HTTP

- O protocolo HTTP define oito métodos que indicam a ação a ser realizada: GET, POST, PUT, DELETE, TRACE, HEAD, OPTIONS, CONNECT.

- Os 4 métodos principais que serão utilizados nos serviços REST:
  - **GET**: recupera um recurso;
  - **POST**: cria um novo recurso;
  - **PUT**: atualiza um recurso existente;
  - **DELETE**: remove um recurso;

RESPOSTAS HTTP

- Existem 5 **tipos** de código de resposta do HTTP:
  - **1xx:** Informação – o pedido foi recebido e está sendo processada;
  - **2xx:** Sucesso – indica que a requisição foi bem sucedida;
  - **3xx:** Redirecionamento – indica a ação que deve ser tomada para completar a requisição;
  - **4xx:** Erro no Cliente – indica que foi realizada uma requisição que não pode ser atendida;
  - **5xx:** Erro no servidor – ocorreu um erro no servidor;

FORMATOS

- Recursos podem estar em vários formados: html, xml, json..
- Vamos trabalhar com **Json – JavaScript Object Notation.**
  - Formato simples e leve para transferência de dados.
  - Uma alternativa para o XML

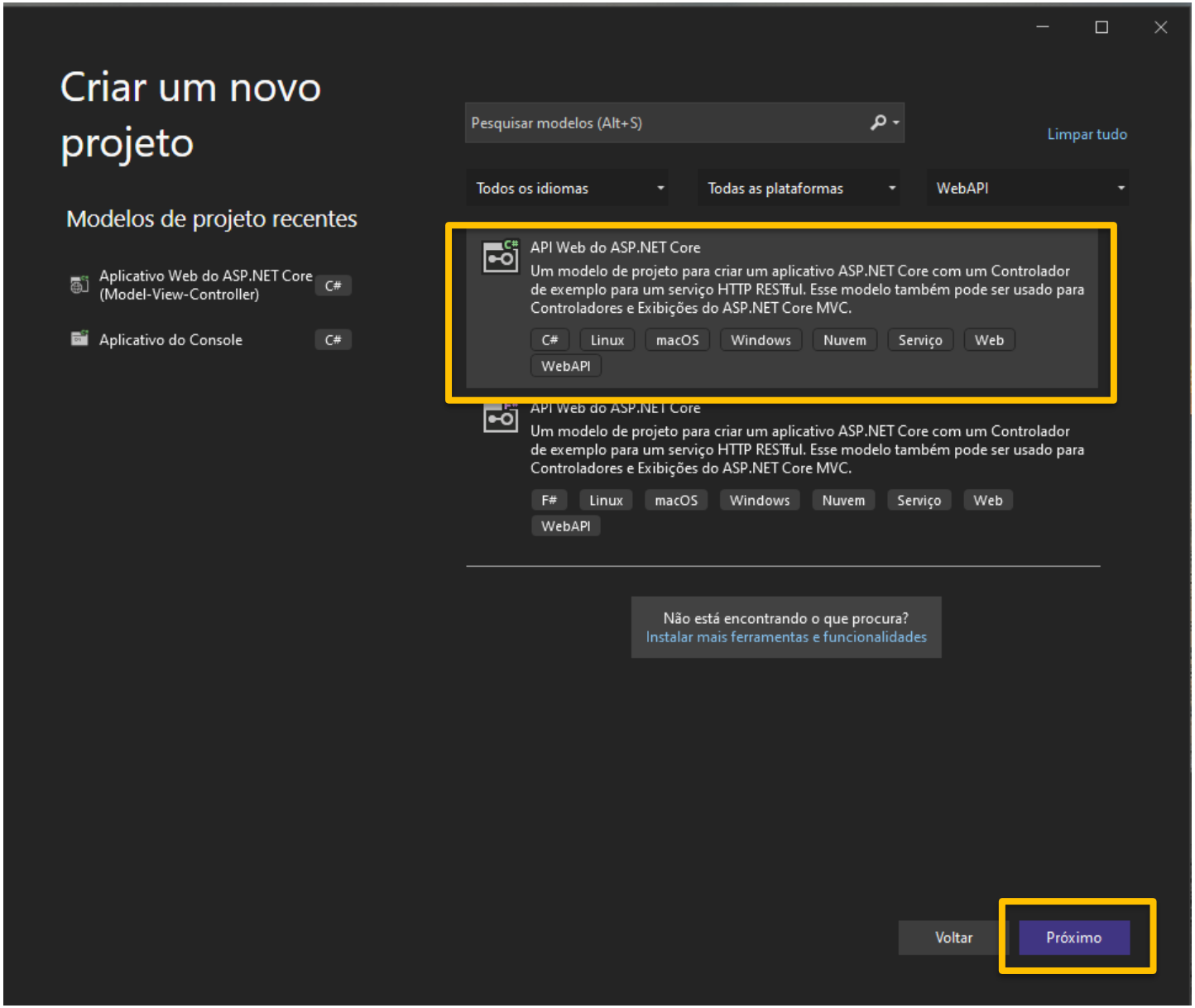
Exemplo:

```
{
   "show": "Oasis",
   "preco": 150,
   "local": "São Paulo"
}
```

Validador de formato Json: http://jsonlint.com/

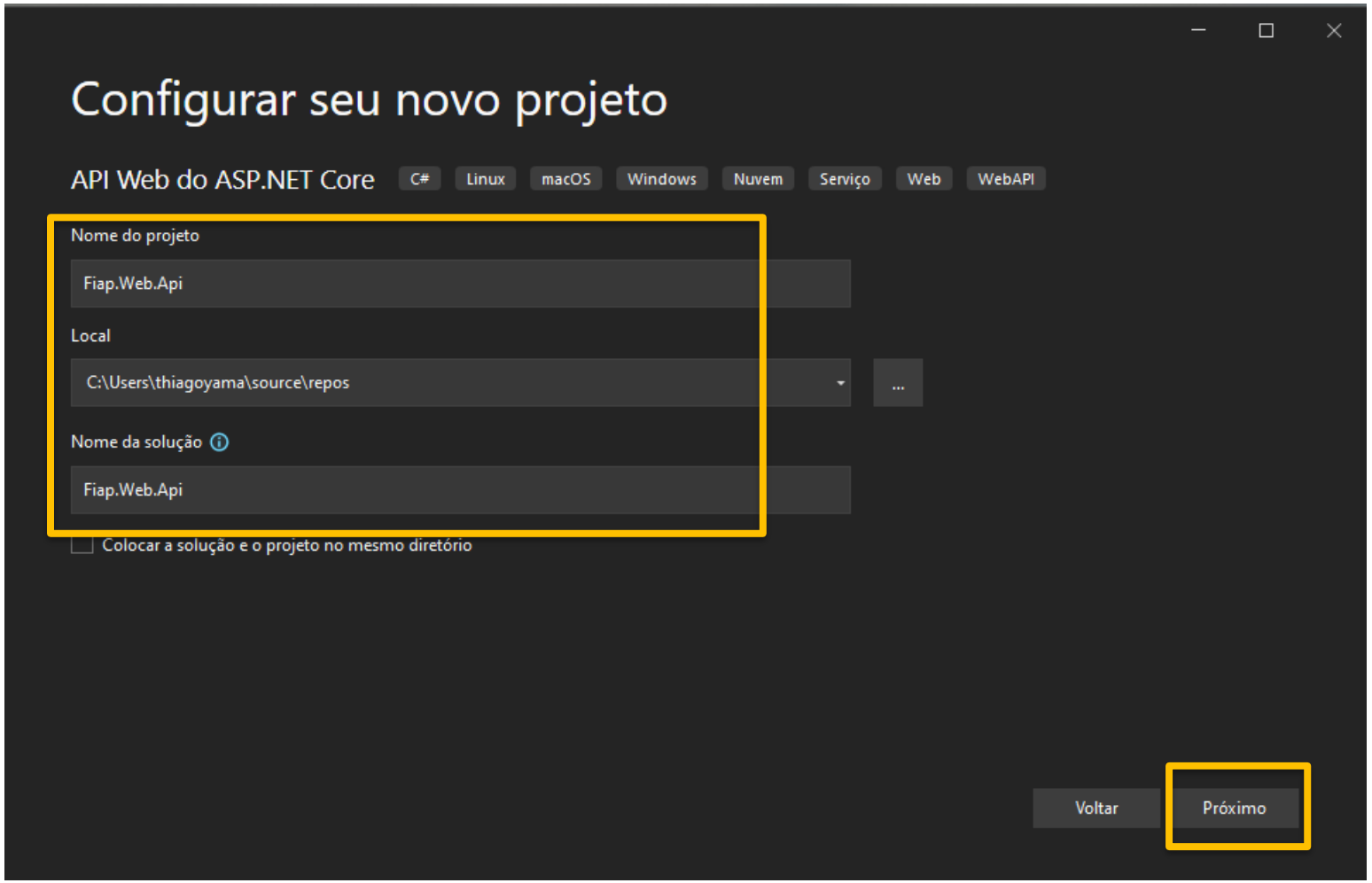
PROJETO

CRIANDO O PROJETO



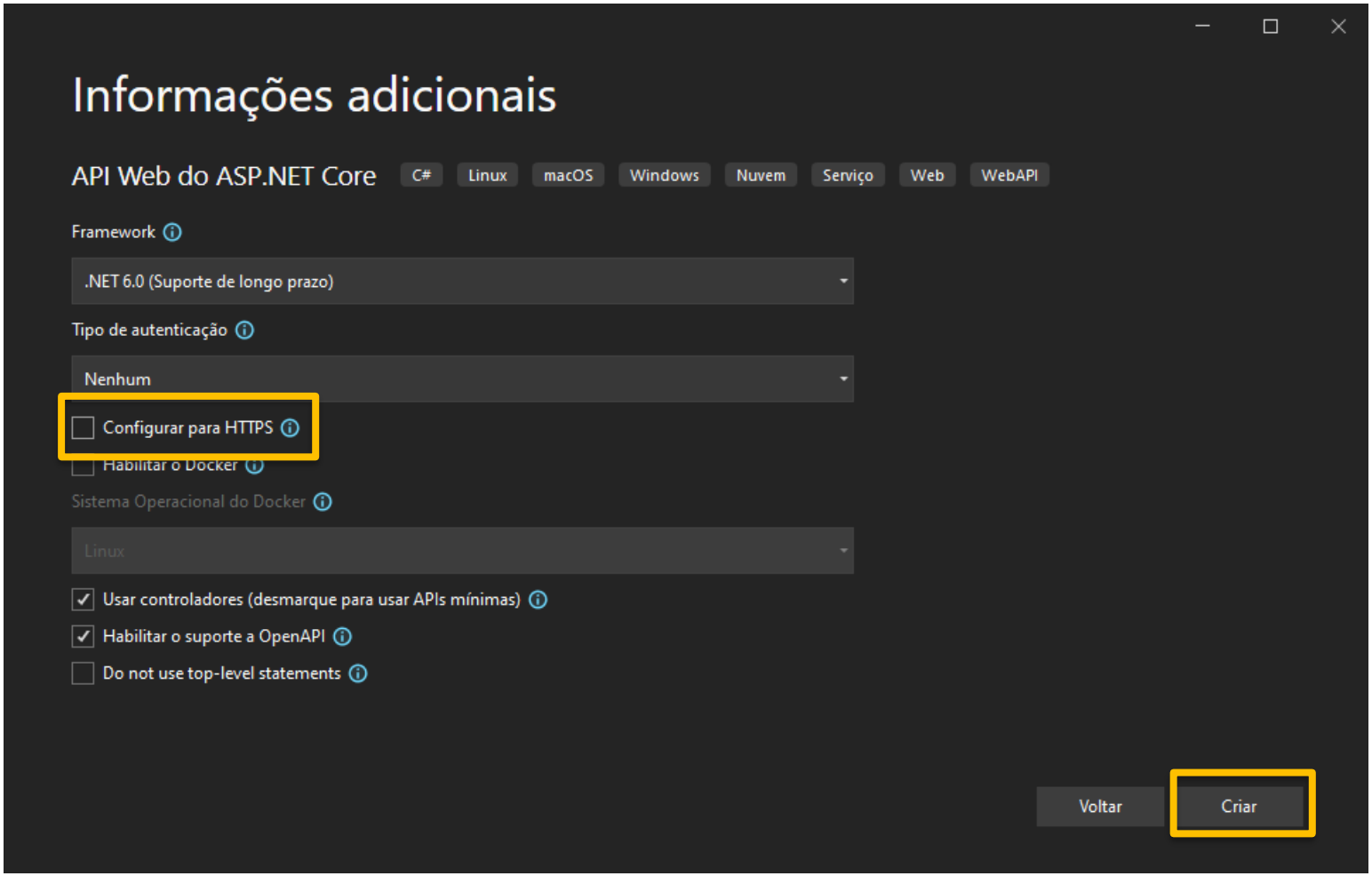
CRIANDO O PROJETO

- Configure o **nome do projeto**, o **diretório** onde será gravado o projeto e o **nome da Solução;**



CRIANDO O PROJETO

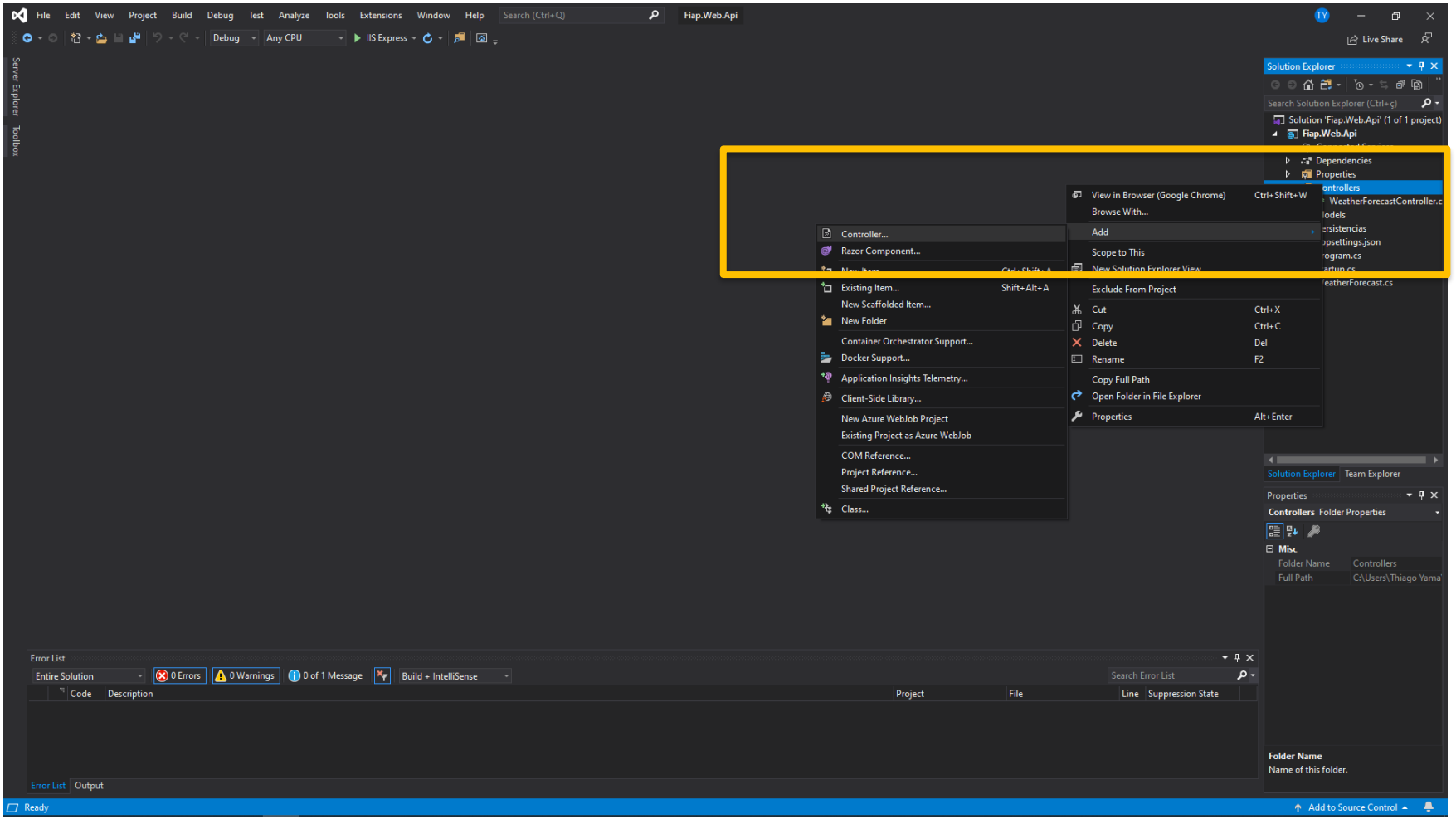
- Desmarque o Https e finalize o processo!



IMPLEMENTAÇÃO

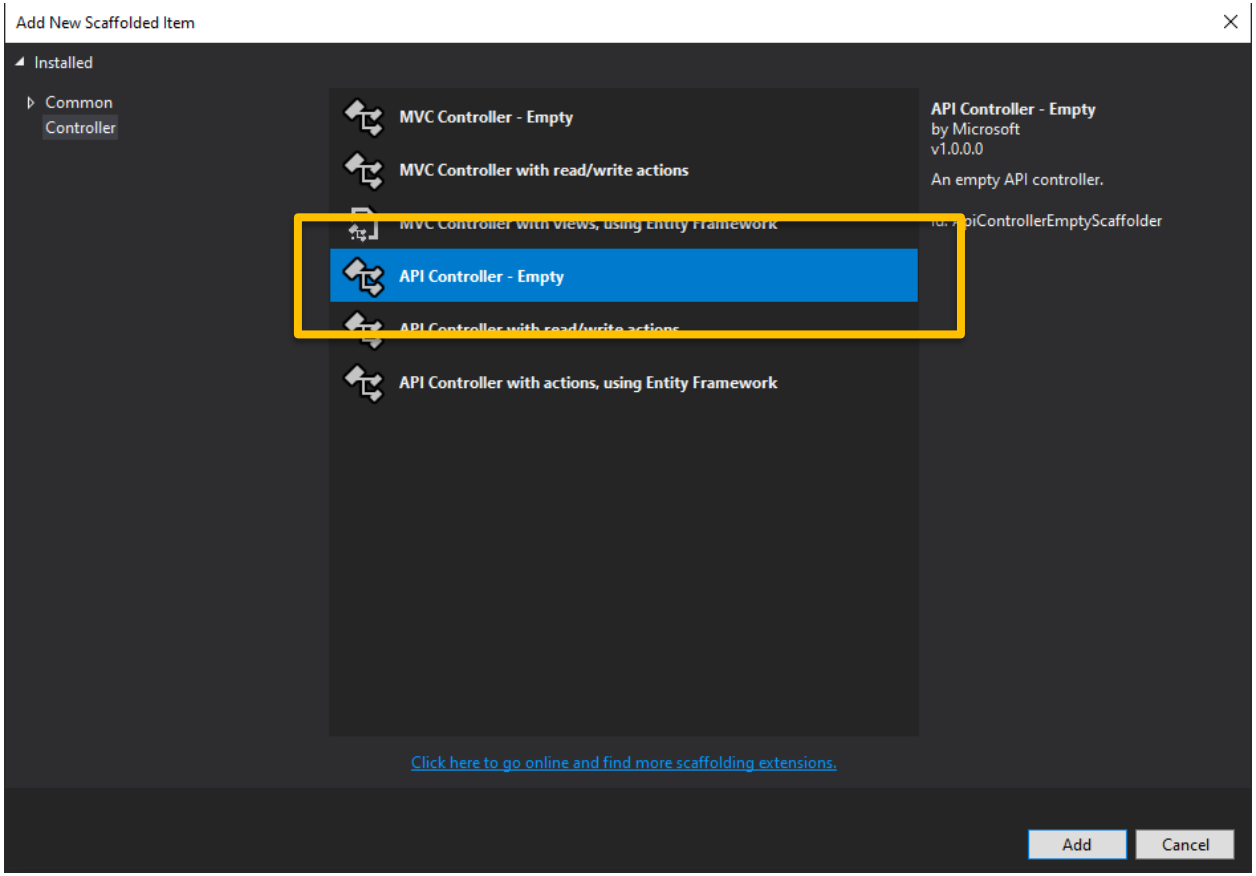
CONTROLLER

- Crie um novo controller:



CONTROLLER

- Escolha a opção **API Controller – Empty**:



CONTROLLER

Herança para controllers de API

```
[Route("api/[controller]")]
[ApiController]
public class ProdutoController : ControllerBase
{
        private LojaContext _context;

        public ProdutoController(LojaContext context)
        {
                _context = context;
        }

}
```

Contexto para operações no banco, pode ser o **repository** também.

GET

```
[HttpGet]
public ActionResult<IList<Produto>> Get()
{
        return _context.Produtos.ToList();
}

[HttpGet("{id}")]
public ActionResult<Produto> Get(int id)
{
        var produto = _context.Produtos.Find(id);
        if (produto == null)
                return NotFound();
        return produto;
}
```

POST

```
[HttpPost]
public ActionResult<Produto> Create(Produto produto)
{
        _context.Produtos.Add(produto);
        _context.SaveChanges();
        return CreatedAtAction("Get",
                new { id = produto.ProdutoId }, produto);
}
```

PUT

```
[HttpPut("{id}")]
public ActionResult Put(Produto produto, int id)
{
        var produto = _context.Produtos.Find(id);
        if (_context.Produtos.Find(id) == null)
                return NotFound();
        produto.Id = id;
        _context.Produtos.Update(produto);
        _context.SaveChanges();
        return NoContent();
}
```

DELETE

```
[HttpDelete("{id}")]
public ActionResult Delete(int id)
{
        var produto = _context.Produtos.Find(id);
        if (_context.Produtos.Find(id) == null)
                return NotFound();

        _context.Produtos.Remove(produto);
        _context.SaveChanges();
        return NoContent();
}
```

VOCÊ APRENDEU..

- Sobre o **Arquitetura Rest;**
- Como implementar **uma API Rest;**
- Desenvolver as operações básicas com GET, POST, PUT e DELETE;

"Muda tuas ideias e mudarás teu mundo". (Norman Vincent Peale)