Introdução

As APIs RESTful em C# são amplamente utilizadas devido à sua integração com o framework ASP.NET Core, que fornece ferramentas robustas para construir serviços web. Para garantir a eficiência e a manutenibilidade da API, é fundamental seguir boas práticas de desenvolvimento. Este texto discute essas práticas e demonstra como aplicá-las a um projeto com as classes **Pedido** e **Fornecedor**.

Boas Práticas no Desenvolvimento de APIs RESTful

Estruturação de Endpoints

Os endpoints devem ser simples, descritivos e refletir os recursos que representam. A convenção REST sugere o uso de substantivos no plural para representar recursos.

Exemplo de endpoint no C#:

```
[ApiController]

[Route("pedidos")]

public class PedidosController : ControllerBase

{

    private readonly IPedidoRepository _repository;



    public PedidosController(IPedidoRepository repository)

    {

        _repository = repository;

    }



    [HttpGet]

    public ActionResult<List<Pedido>> GetAll()

    {
```

```
        return Ok(_repository.GetAll());

    }




    [HttpGet("{id}")]

    public ActionResult<Pedido> GetById(int id)

    {

        var pedidoEncontrado = _repository.GetById(id);



        if(pedidoEncontrado == null)

            return NotFound();



        return Ok(pedidoEncontrado);

    }
```

Uso de Verbos HTTP

Os verbos HTTP devem ser usados de maneira consistente:

- **GET**: Para leitura de dados.
- **POST**: Para criação de novos recursos.
- **PUT**: Para atualização completa de recursos existentes.
- **DELETE**: Para exclusão de recursos.

Exemplo de mapeamento de métodos:

```
[HttpPost]

    public ActionResult Post(Pedido pedido)

    {

        _repository.Post(pedido);
```

```
        return Created();
    }



    [HttpPut("{id}")]
    public ActionResult Put(int id, Pedido pedidoAtualizado)
    {
        var pedidoEncontrado = _repository.Put(id, pedidoAtualizado);

        if(pedidoEncontrado == null)
            return NotFound();

        return Ok(pedidoEncontrado);
    }

    [HttpDelete("{id}")]
    public ActionResult Delete(int id)
    {
        var pedidoEncontrado = _repository.Delete(id);

        if(pedidoEncontrado == null)
            return NotFound();
        return NoContent();
    }
}
```

Conclusão

O desenvolvimento de APIs RESTful em C# utilizando o framework ASP.NET Core permite a criação de serviços web eficientes, escaláveis e aderentes às melhores práticas da arquitetura REST. A aplicação de boas práticas, como a definição clara de endpoints, uso correto de verbos HTTP, retorno de status adequados e encapsulamento de dados com DTOs, não apenas melhora a experiência do desenvolvedor, mas também promove uma comunicação padronizada e previsível entre sistemas.

Neste texto, apresentamos como essas práticas podem ser implementadas em um projeto fictício com as classes **Pedido** e **Fornecedor**, ilustrando cada etapa com exemplos em código. A estruturação adequada de endpoints e métodos, o tratamento de erros com respostas apropriadas e a documentação dos recursos são essenciais para garantir a manutenibilidade e escalabilidade do sistema.