**Parte 2: Questionário Técnico**

**Responda às seguintes perguntas:**

**1. Explique a diferença entre IEnumerable e IQueryable em .NET.**

**R:** IEnumerable**:** Está no namespace System.Collections.Generic. É a interface base para todas as coleções genéricas no .NET.

IQueryable: Está no namespace System.Linq. Herda de IEnumerable<T>, mas adiciona funcionalidades específicas para consultas.

**2. O que são migrations no Entity Framework e como elas são usadas?**

**R:** migrations são um mecanismo que permite gerenciar e aplicar alterações no esquema do banco de dados de forma incremental, mantendo-o sincronizado com o modelo de dados da aplicação.

Quando você altera o modelo de dados (por exemplo, adiciona uma nova propriedade a uma classe ou cria uma nova entidade), o Entity Framework pode gerar uma migration para refletir essas alterações no banco de dados.

**3. Descreva uma situação em que você usaria um Repository Pattern e os benefícios dessa abordagem.**

**R:** O Repository Pattern é um padrão de design que abstrai a camada de acesso a dados, fornecendo uma interface limpa e consistente para interagir com os dados, independentemente da fonte (banco de dados, API, arquivos, etc.). Ele é comumente usado em aplicações que seguem princípios como SOLID e Domain-Driven Design (DDD).

**4. Como você lidaria com a concorrência em um ambiente multi-thread em .NET?**

**R:** Lidar com concorrência em .NET exige uma combinação de técnicas e boas práticas para garantir que o código seja seguro e eficiente.

O uso de **Task** e **async/await** pode ajudar a gerenciar operações assíncronas de forma eficiente, evitando bloqueios desnecessários.

**Parte 3: Desafio de Otimização**

Código melhorado.

public async Task<List<Product>> GetProductsAsync()

{

using (var context = new ApplicationDbContext())

{

var products = await context.Products.ToListAsync();

var tasks = products.Select(async product =>

{

product.Description = await LoadDescriptionAsync(product.Id);

return product;

});

var updatedProducts = await Task.WhenAll(tasks);

return updatedProducts.ToList();

}

}

private async Task<string> LoadDescriptionAsync(int productId)

{

// Simulando uma operação cara de forma assíncrona

await Task.Delay(100);

return "Descrição do Produto " + productId;

}