Lidando com as alterações interruptivas durante a migração de uma aplicação para o .NET 8

Nagib Sabbag Filho

Leaders.Tec.Br, 1(1), ISSN: 2966-263X, 2024.

e-mail: profnagib.filho@fiap.com.br

DOI: https://doi.org/10.5281/zenodo.13264815

PermaLink: https://leaders.tec.br/artigo/lidando-com-as-alteracoes-interruptivas-durante-a-migracao-de-uma-aplicacao-para-o-net-8

Received: 04 Jul 2024 / Accepted: 06 Jul 2024 / Published online: 08 Jul 2024

Abstract:

Lidando com alterações interruptivas durante a migração para o .NET 8: estratégias para garantir uma transição suave.

Key words:

migração, alterações interruptivas, .net 8, compatibilidade, apis, testes.

Entendendo as alterações interruptivas

Conhecidas como breaking changes, são modificações no código que podem causar incompatibilidades com o funcionamento anterior da aplicação. Durante a migração para o .NET 8, é importante identificar e lidar com essas alterações de forma eficiente para garantir que a aplicação continue funcionando corretamente. Elas podem incluir:

• 1. Mudanças de API: Alterações em assinaturas de métodos, parâmetros ou propriedades que impactam o uso existente. Por exemplo, a remoção de um método que era amplamente utilizado ou a mudança na assinatura de um método que exige novos parâmetros.

• 2. Comportamento diferente: Modificações no comportamento de funções ou classes que podem quebrar fluxos de trabalho existentes. Por exemplo, uma função que anteriormente retornava um valor padrão pode passar a lançar uma exceção em determinados cenários.

• 3. Depreciação de recursos: Remoção ou descontinuação de recursos ou funcionalidades obsoletas. Isso pode incluir métodos, classes ou até bibliotecas inteiras que não são mais suportadas na nova versão.

Principais desafios durante a migração para o .NET 8

Um dos principais desafios durante a migração para o .NET 8 é a necessidade de atualização de bibliotecas e frameworks utilizados pela aplicação. É comum que novas versões do .NET tragam mudanças significativas que podem impactar diretamente no funcionamento do código existente. Além disso, a documentação pode não estar completa ou atualizada, o que pode dificultar a identificação de todas as mudanças necessárias.

Outro desafio é garantir que todas as dependências de terceiros sejam compatíveis com a nova versão. Isso pode exigir a atualização ou substituição de pacotes NuGet e outras bibliotecas externas.

Estratégias para lidar com as alterações interruptivas

Uma das estratégias mais eficazes para lidar com as alterações interruptivas durante a migração para o .NET 8 é realizar testes de regressão abrangentes. Isso inclui a execução de testes automatizados e manuais para identificar possíveis problemas causados pelas alterações no código. É importante ter uma boa cobertura de testes para garantir que todas as partes críticas da aplicação sejam verificadas.

Outra estratégia é utilizar ferramentas de análise de código para identificar alterações interruptivas. Ferramentas como o .NET Portability Analyzer podem ajudar a detectar incompatibilidades entre diferentes versões do .NET.

Além disso, é recomendável criar um plano de migração detalhado, que inclua todas as etapas necessárias para atualizar a aplicação. Isso pode incluir a atualização de bibliotecas, a modificação do código para compatibilidade com o .NET 8, e a realização de testes extensivos para garantir que tudo funcione corretamente.

Exemplo prático de testes unitários atuando como testes de regressão

Suponha que durante a migração para o .NET 8, uma alteração no comportamento de uma função de criptografia cause incompatibilidades com o restante da aplicação. Nesse caso, é importante revisar o código da função afetada e ajustá-la de acordo com as novas diretrizes do .NET 8. Por exemplo, se um método de criptografia foi atualizado para usar um novo algoritmo por padrão, você pode precisar ajustar o código para especificar o algoritmo antigo, se necessário.

Por exemplo... Abaixo temos um código em csharp disponibilizando uma solução de criptografia

```
using System;
using System.Security.Cryptography;
using System.Text;

public class CryptographyService
{
    private readonly string key = "example_key_1234";

    public string Encrypt(string plainText)
    {
        using (Aes aes = Aes.Create())
        {
            byte[] keyBytes = Encoding.UTF8.GetBytes(key);
            aes.Key = keyBytes;

            ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
            using (var ms = new MemoryStream())
            {
                ms.Write(aes.IV, 0, aes.IV.Length);
                using (var cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write)
)
                using (var sw = new StreamWriter(cs))
                {
                    sw.Write(plainText);
                }
                return Convert.ToBase64String(ms.ToArray());
            }
        }
    }

    public string Decrypt(string cipherText)
```

```
    {
        byte[] fullCipher = Convert.FromBase64String(cipherText);
        using (Aes aes = Aes.Create())
        {
            byte[] iv = new byte[aes.BlockSize / 8];
            byte[] cipher = new byte[fullCipher.Length - iv.Length];

            Array.Copy(fullCipher, iv, iv.Length);
            Array.Copy(fullCipher, iv.Length, cipher, 0, cipher.Length);

            aes.Key = Encoding.UTF8.GetBytes(key);
            aes.IV = iv;

            ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key, aes.IV);
            using (var ms = new MemoryStream(cipher))
            using (var cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Read))
            using (var sr = new StreamReader(cs))
            {
                return sr.ReadToEnd();
            }
        }
    }
}
```

Abaixo um exemplo de teste de regressão para o método de criptografia usando o framework XUnit:

```
using System;
using Xunit;

public class CryptographyServiceTests
{
    private readonly CryptographyService _cryptographyService;

    public CryptographyServiceTests()
    {
        _cryptographyService = new CryptographyService();
    }

    [Fact]
    public void EncryptDecrypt_ShouldReturnOriginalString()
    {
        // Arrange
        string originalText = "Hello, World!";

        // Act
        string encryptedText = _cryptographyService.Encrypt(originalText);
        string decryptedText = _cryptographyService.Decrypt(encryptedText);

        // Assert
        Assert.Equal(originalText, decryptedText);
    }

    [Fact]
    public void Encrypt_ShouldReturnDifferentString()
    {
        // Arrange
```

```
        string originalText = "Hello, World!";

        // Act
        string encryptedText = _cryptographyService.Encrypt(originalText);

        // Assert
        Assert.NotEqual(originalText, encryptedText);
    }

    [Fact]
    public void Decrypt_InvalidData_ShouldThrowException()
    {
        // Arrange
        string invalidCipherText = "invalid_cipher_text";

        // Act & Assert
        Assert.Throws(() => _cryptographyService.Decrypt(invalidCipherText));
    }
}
```

Outro ponto bastante útil seria consultar as páginas oficiais. Por exemplo, a página Alterações interruptivas no .NET 8 nos ajuda a entender melhor esse cenário, apresentando todos os pontos que precisam ser tratados, em tópicos:

• Asp.NET Core: Mudanças em controladores, views e middleware que podem afetar o comportamento da aplicação web.

• Containers: Atualizações nas configurações e suporte para contêineres Docker e Kubernetes.

• Principais bibliotecas do .NET: Alterações em bibliotecas comuns como System.IO, System.Net, e outras que podem impactar diversas partes do aplicativo.

• Criptografia: Mudanças nos algoritmos e métodos de criptografia que podem afetar a segurança e a compatibilidade de dados criptografados.

• Implantação: Novas práticas e ferramentas para implantação que podem exigir ajustes no pipeline de CI/CD.

• Entity Framework Core: Atualizações no ORM que podem afetar o mapeamento de entidades e consultas de banco de dados.

• Outros tópicos consideravelmente relevantes: Inclui melhorias de desempenho, segurança e novos recursos que podem exigir adaptações no código.

Conclusão

Lidar com as alterações interruptivas durante a migração para o .NET 8 pode ser desafiador, mas com as estratégias certas e um planejamento adequado, é possível minimizar os impactos e garantir uma transição suave. Mantenha-se atualizado com as novas versões do .NET e esteja sempre atento às alterações que podem afetar sua aplicação. Além disso, considere participar de comunidades e fóruns de desenvolvedores para compartilhar experiências e obter suporte durante o processo de migração.

Referências

• Alterações interruptivas no .NET 7. Disponível em https://learn.microsoft.com/pt-br/dotnet/core/compatibility/7.0;

• Alterações interruptivas no .NET 8. Disponível em https://learn.microsoft.com/pt-br/dotnet/core/compatibility/8.0.

Nagib é Professor Universitário e Tech Manager.
Possui uma trajetória de conquistas em certificações técnicas e ágeis,
incluindo MCSD, MCSA e PSM1.
PG em Gestão de TI pelo SENAC e MBA em Tecnologia de Software pela USP,
Nagib cursou programas de extensão do MIT e Universidade de Chicago.
Outras conquistas incluem a autoria de um artigo sobre chatbots, revisado por pares e apresentado na Universidade de Barcelona.