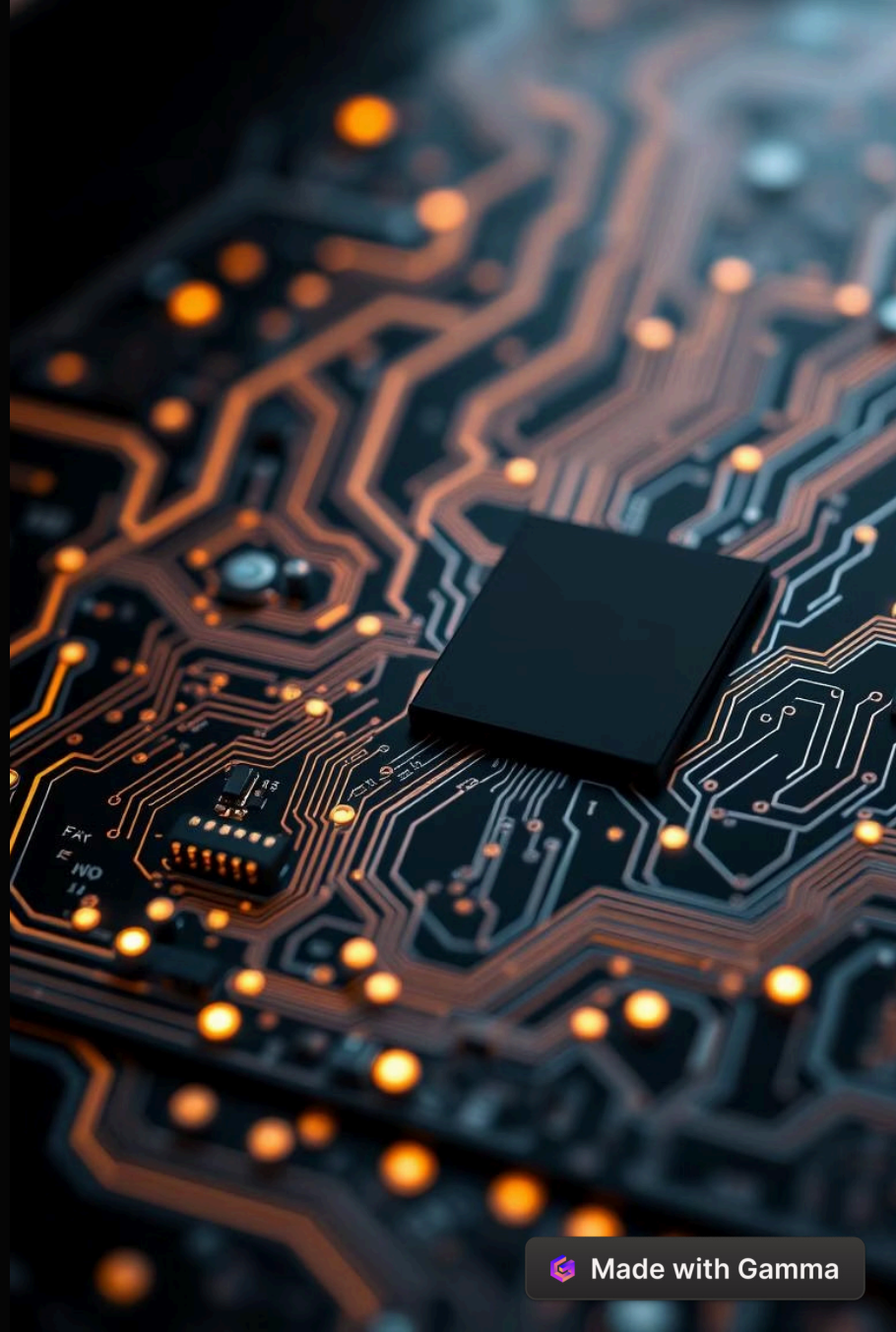
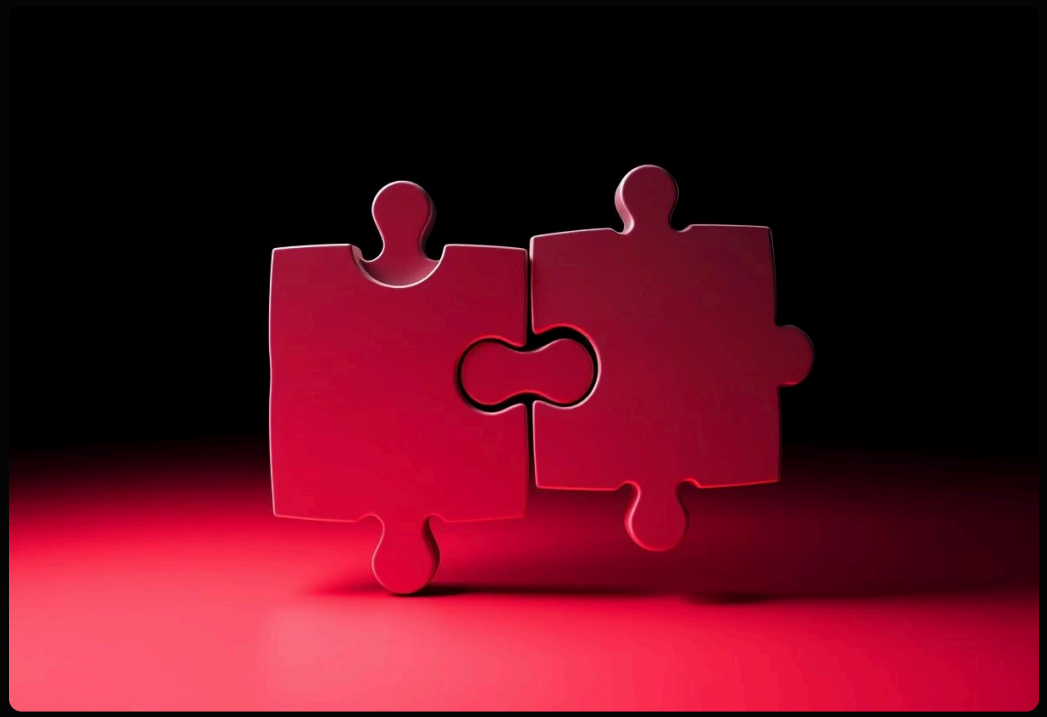
Introdução ao Padrão Adapter

O Adapter é um padrão de projeto estrutural que é essencial para resolver incompatibilidades entre interfaces. Ele promove a reutilização e flexibilidade do código.



O que é o Padrão Adapter?



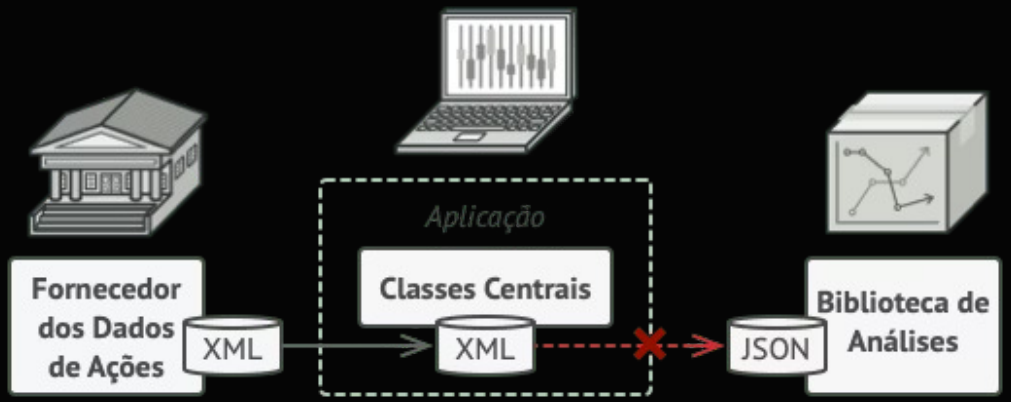
Adaptador de Interfaces

O Adapter atua como um intermediário entre interfaces incompatíveis. Permite que classes colaborem sem modificar seus códigos fonte.

Abstração

O padrão fornece uma abstração, conectando diferentes estruturas. Isso facilita a integração e a manutenção do sistema.

O Desafio da Incompatibilidade

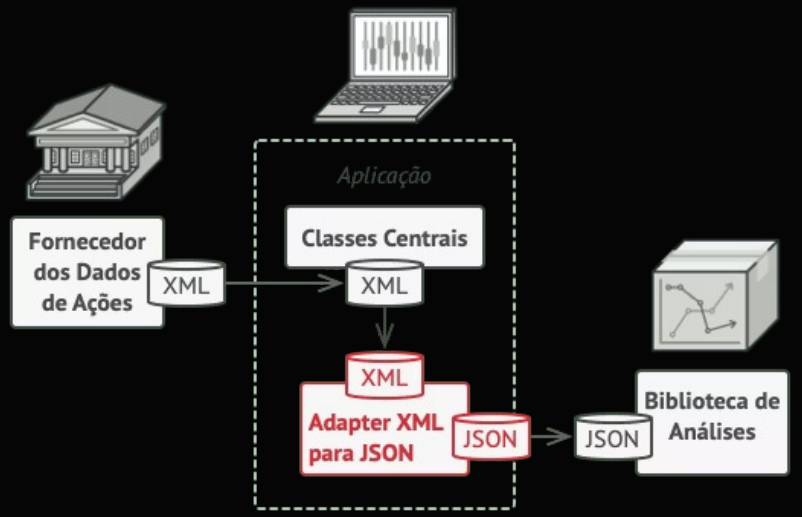


Cenário Típico

Imagine uma aplicação de monitoramento do mercado de ações. Ela recebe dados em formato XML, mas sua biblioteca externa só processa JSON. Como resolver?

O Problema

É preciso integrar dados de diferentes formatos. Sem o Adapter, seria necessário modificar a biblioteca ou a fonte de dados.



A Solução: Adaptando Interfaces

XML – JSON

Você cria adaptadores XML-para-JSON para cada classe da biblioteca de análise que seu código trabalha diretamente.

Chamada Segura

Adaptador recebe uma chamada, ele traduz os dados entrantes XML em uma estrutura JSON e passa a chamada para os métodos apropriados de um objeto de análise encoberto.

Vantagens do Padrão Adapter

Reutilização

Reutilize código existente sem modificações.

Integração

Facilite a integração de sistemas legados.

Manutenção

Simplifique a manutenção, minimizando impacto.

Suporte

Adicione novos tipos de interfaces facilmente.

Desvantagens do Padrão Adapter

Complexidade Adicional

A introdução de adaptadores pode aumentar a complexidade do projeto.

Dependências

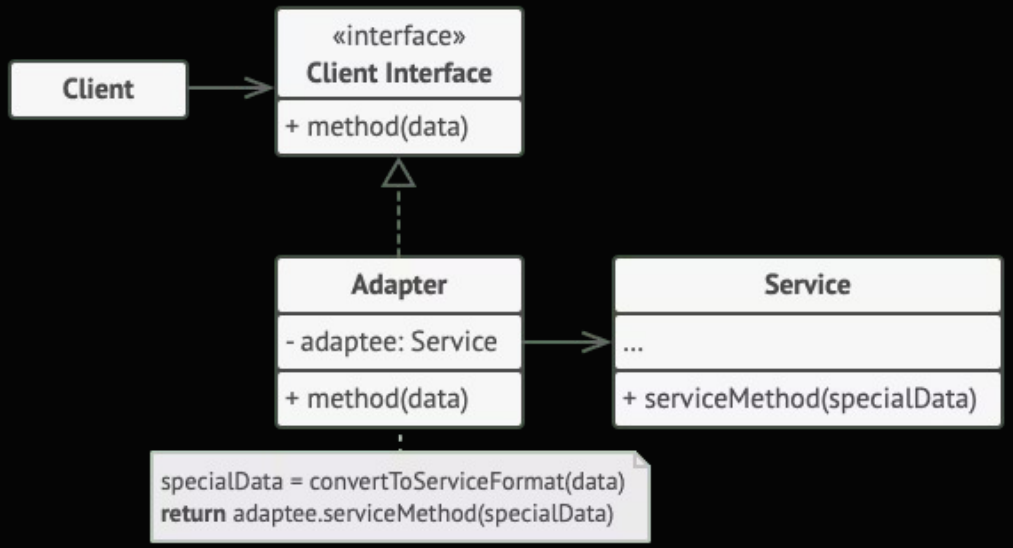
Pode aumentar as dependências entre classes.

Proliferação

Há risco de criar muitos adaptadores em sistemas complexos.

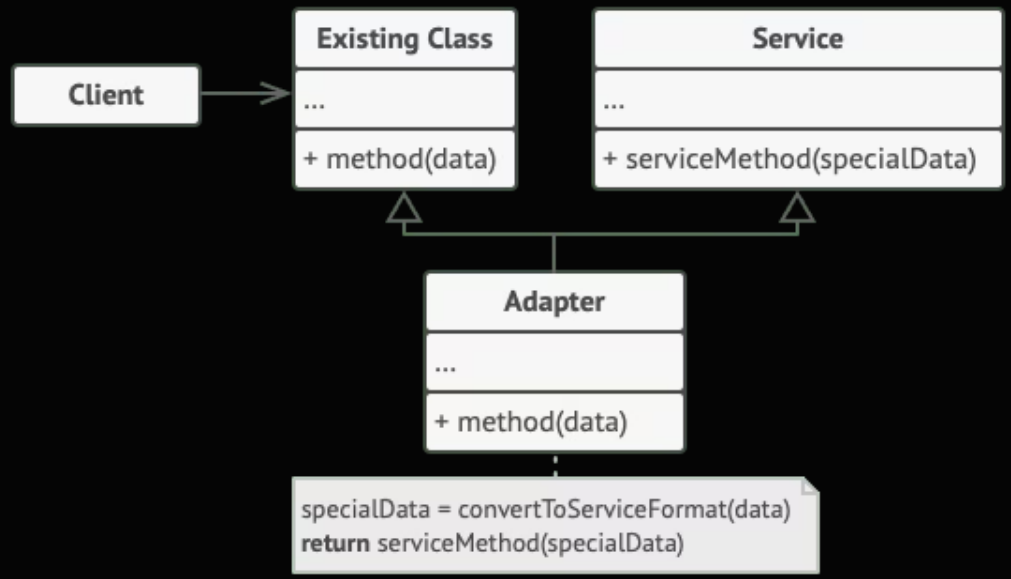
Tipos de Adapter

Object Adapter



Usa composição para adaptar interfaces. É mais flexível e desacoplado.

Class Adapter



Usa herança múltipla (onde suportado). É mais restritivo e acoplado.

A escolha depende dos requisitos do projeto e das linguagens utilizadas.

Conclusão: Adaptando para o Futuro

Escalabilidade

Beneficia a escalabilidade do sistema.

Compatibilidade

O Adapter resolve problemas de incompatibilidade.

Integração

Facilita a integração entre sistemas.

A escolha do padrão deve estar alinhada com as necessidades do projeto. Avalie cuidadosamente antes de implementar.

Exemplo Prático em C#

```
public interface INotificacao
{
    void Enviar(string destinatario, string mensagem);
}

public class SmsLegado
{
    public void EnviarSms(string numeroTelefone, string conteudo)
    {
        Console.WriteLine($"SMS enviado para {numeroTelefone}: {conteudo}");
    }
}

public class SmsParaNotificacaoAdapter : INotificacao
{
    private readonly SmsLegado _smsLegado;

    public SmsParaNotificacaoAdapter(SmsLegado smsLegado)
    {
        _smsLegado = smsLegado;
    }

    public void Enviar(string destinatario, string mensagem)
    {
        _smsLegado.EnviarSms(numeroTelefone: destinatario, conteudo: mensagem);
    }
}


public class ClienteNotificacao
{
    public void DispararNotificacao(INotificacao notificacao)
    {
        notificacao.Enviar("+5511999999999", "Sua conta foi atualizada!");
    }
}
```

```
class Program
{
    static void Main()
    {
        var cliente = new ClienteNotificacao();

        var smsLegado = new SmsLegado();

        var adapter = new SmsParaNotificacaoAdapter(smsLegado);

        cliente.DispararNotificacao(adapter);
    }
}
```