





Fiap.Challenge.Wtc

Projeto desenvolvido seguindo os princípios da Arquitetura Hexagonal (Clean Architecture) para o desafio da FIAP.

Arquitetura

Este projeto implementa a **Arquitetura Hexagonal**, garantindo:

-  Separação clara de responsabilidades
-  Alta testabilidade
-  Baixo acoplamento
-  Independência de frameworks e tecnologias

Estrutura do Projeto

```
├── src/
│   ├── Fiap.Challenge.Wtc.API/           # 🌐 Camada de Apresentação (Web API)
│   ├── Fiap.Challenge.Wtc.Application/   # ⚙️ Camada de Aplicação (Use Cases)
│   ├── Fiap.Challenge.Wtc.Domain/        # 🏠 Camada de Domínio (Regras de Negócio)
│   └── Fiap.Challenge.Wtc.Infrastructure/ # 🔧 Camada de Infraestrutura
├── tests/
│   ├── Fiap.Challenge.Wtc.Tests.Unit/    # 🖋️ Testes Unitários
│   └── Fiap.Challenge.Wtc.Tests.Integration/ # 🔗 Testes de Integração
└── docs/                                # 📖 Documentação
```

Como Executar

Pré-requisitos

- .NET 8.0 ou superior
- IDE de sua preferência (Visual Studio, VS Code, Rider)

Passos

1. Clone o repositório

```
git clone <url-do-repositorio>
cd Fiap.Challenge.Wtc
```

2. Restaurar dependências

```
dotnet restore
```

3. Compilar a solução

```
dotnet build
```

4. Executar a API

```
dotnet run --project src/Fiap.Challenge.Wtc.API
```

5. Executar testes

```
# Todos os testes
dotnet test

# Apenas testes unitários
dotnet test tests/Fiap.Challenge.Wtc.Tests.Unit

# Apenas testes de integração
dotnet test tests/Fiap.Challenge.Wtc.Tests.Integration
```

Testes

O projeto possui duas suítes de testes:

- **Testes Unitários:** Testam componentes isoladamente
- **Testes de Integração:** Testam a integração entre componentes

Documentação

Para informações detalhadas sobre a arquitetura, consulte:

- [Documentação da Arquitetura](#)

Tecnologias Utilizadas

- **.NET 8**
- **ASP.NET Core Web API**
- **xUnit** (Testes)
- **Swagger/OpenAPI** (Documentação da API)

Princípios da Arquitetura Hexagonal

Domain (Núcleo)

- Entidades e Value Objects
- Regras de negócio puras
- Interfaces de repositório (Ports)

Application (Casos de Uso)

- Orquestração de operações
- DTOs e contratos
- Use Cases específicos

Infrastructure (Adapters)

- Implementações de repositórios
- Serviços externos
- Configurações de tecnologia

API (Interface)

- Controllers REST
- Middlewares
- Tratamento de exceções

Contribuição

1. Faça um fork do projeto
2. Crie uma branch para sua feature (`git checkout -b feature/AmazingFeature`)
3. Commit suas mudanças (`git commit -m 'Add some AmazingFeature'`)
4. Push para a branch (`git push origin feature/AmazingFeature`)
5. Abra um Pull Request

Licença

Este projeto está sob a licença MIT. Veja o arquivo [LICENSE](#) para mais detalhes.

Desenvolvido como parte do desafio da FIAP seguindo as melhores práticas de Clean Architecture e desenvolvimento .NET.