Planejamento das Escolhas Tecnológicas

Paulo Vinícius Moreira Dutra

14 de abril de 2025

1 Definição da Arquitetura do Sistema: Escolher a Estrutura Tecnológica, Frameworks e Banco de Dados

1.1 Backend: ASP.NET

A escolha do **ASP.NET** é baseada na robustez e confiabilidade da plataforma para o desenvolvimento de APIs. Com o **ASP.NET Core**, obtemos uma solução multiplataforma, eficiente e escalável para o desenvolvimento de sistemas back-end. A plataforma permite criar APIs RESTful ou GraphQL com alta performance, o que é essencial para a comunicação entre o front-end e o banco de dados.

1.1.1 Frameworks adicionais

- **ASP.NET Core**: Utilizado para o desenvolvimento de APIs modernas e escaláveis.
- **Swagger**: Para documentação automatizada das APIs, facilitando a comunicação entre o desenvolvimento e a equipe de testes.

1.2 Frontend: Flutter

O **Flutter** foi escolhido para o desenvolvimento do front-end, devido à sua capacidade de criar interfaces de usuário nativas com uma base de código única para Android e iOS. O uso de **Flutter** permite uma rápida prototipagem e excelente desempenho, essencial para a criação de interfaces interativas e dinâmicas.

1.2.1 Flutter Web

Se necessário, o **Flutter Web** pode ser explorado para criar uma versão web do sistema utilizando a mesma base de código, garantindo maior escalabilidade e flexibilidade.

1.3 Banco de Dados: MySQL

O **MySQL** foi escolhido devido à sua confiabilidade, robustez e ampla adoção em sistemas de grande escala. Sua integração com o **ASP.NET** é bem suportada, e ele oferece desempenho adequado para aplicações que exigem consistência de dados.

1.3.1 Considerações Técnicas

- Utilização do **MySQL Workbench** para administração e visualização do banco de dados.
- O **Entity Framework Core** será utilizado para simplificar a integração entre o banco de dados e o **ASP.NET Core**.

2 Refinamento dos Requisitos com Casos de Uso

Os casos de uso descrevem as funcionalidades do sistema, detalhando os fluxos de interação entre os usuários e o sistema.

2.1 Frontend: Flutter

O front-end irá se comunicar com o **backend** através de requisições API. Exemplos de casos de uso incluem:

- **Cadastro de Usuário**: O usuário preenche seus dados, que são enviados ao servidor para criação de um novo registro.
- **Tela Principal**: Exibe os dados armazenados no banco de dados, com interações dinâmicas baseadas na API.

2.1.1 Arquitetura em Camadas

Para garantir um código limpo, testável e com boa manutenção, será adotada uma arquitetura em camadas, organizada da seguinte forma:

- **Apresentação (UI)**: Camada responsável por exibir os dados e capturar as interações do usuário.
- **Service**: Camada intermediária que trata regras de negócio e coordena as chamadas ao repositório.
- **Repository**: Responsável pela comunicação com fontes de dados externas, como APIs e banco de dados local.

2.1.2 Padrões Utilizados

Serão utilizados os seguintes padrões arquiteturais:

- **Service Locator**: Facilita a injeção e o gerenciamento de dependências, centralizando o acesso aos serviços da aplicação.
- **Repository**: Abstrai as fontes de dados, permitindo que o código da camada de serviço não dependa diretamente das implementações de API.
- **Service**: Centraliza as regras de negócio, mantendo a lógica da aplicação organizada e separada da interface.

2.1.3 Flutter Web

Caso haja necessidade, o **Flutter Web** poderá ser explorado para expandir a aplicação para navegadores, aproveitando a base de código existente.

2.2 Backend: ASP.NET

O back-end será responsável por fornecer as APIs para autenticação e CRUD de dados. Exemplos de casos de uso incluem:

- **Autenticação de Usuário**: O sistema valida as credenciais do usuário e retorna um token de autenticação.
- **CRUD de Dados**: O sistema realiza operações de criação, leitura, atualização e exclusão de registros no banco de dados.

3 Modelagem UML

A modelagem UML será utilizada para representar visualmente a estrutura e funcionamento do sistema. Os diagramas principais incluirão:

3.1 Diagramas de Classe

Serão representadas as principais entidades do sistema, como **Usuário**, **Produto**, **Pedido**, e suas interações com o banco de dados.

3.2 Diagramas de Casos de Uso

Descreverão as interações do usuário com o sistema, como o processo de login, visualização de dados e criação de pedidos.

4 Planejamento das Iterações de Desenvolvimento

O desenvolvimento será dividido em ciclos iterativos , cada um focado em funcionalidades específicas.

Para isso ver documento de plano de liberações

4.1 Priorização

A prioridade será dada às funcionalidades essenciais, como autenticação de usuários e operações CRUD. Funcionalidades adicionais serão implementadas conforme o andamento do projeto.

5 Análise de Viabilidade do Projeto

Antes de iniciar o desenvolvimento, será realizada uma análise de viabilidade técnica e econômica.

5.1 Viabilidade Técnica

A escolha do **ASP.NET Core** para o back-end, **MySQL** como banco de dados e **Flutter** para o front-end garante uma integração eficiente e a escalabilidade necessária para o projeto. Essas tecnologias são amplamente suportadas e bem documentadas.

5.2 Viabilidade Econômica

O uso do **Flutter** para o desenvolvimento multiplataforma reduz significativamente os custos de desenvolvimento e manutenção. O **ASP.NET Core** também oferece uma solução econômica para o desenvolvimento de APIs de alta performance.