Centro universitário de Brasília

Filipe Soares e Illana Nogueira

Desenvolvimento de Sistema para Reserva de Restaurantes

Documento de arquitetura e tecnologias

Brasília, DF

2025

1. Visão geral da arquitetura

O sistema ReservaApi foi desenvolvido seguindo o padrão de Arquitetura em Camadas (N-Tier), com separação física e lógica entre o Front-end (Interface) e o Back-end (Regras de Negócio).

1.1.Motivo da escolha

Essa abordagem foi escolhida para garantir:

- **Desacoplamento:** O Front-end pode ser substituído (por um App Mobile, por exemplo) sem afetar a API.
- **Escalabilidade:** As camadas podem ser hospedadas em servidores diferentes.
- **Manutenibilidade:** Regras de negócio centralizadas na API.

1.2. Componentes do Sistema

A solução é composta por três projetos distintos dentro da solução:

1.2.1 Front-end (ReservaFront):

- Padrão: MVC (Model-View-Controller).
- Responsabilidade: Apresentação das telas ao usuário e consumo da API.
- Comunicação: Utiliza HttpClient para enviar requisições JSON para o Back-end. Não possui acesso direto ao banco de dados.

1.2.2 Back-end (ReservaApi / swagger):

- Padrão: RESTful API.
- Responsabilidade: Processamento de dados, validação de regras de negócio (ex: não permitir conflito de horários) e persistência.
- Segurança: Implementa políticas de CORS para controlar quem pode acessar os recursos.

1.2.3 Banco de Dados:

- Tipo: Relacional (SQLite).
- Acesso: Gerenciado via ORM (Entity Framework Core), eliminando a necessidade de SQL manual.

2. Tecnologias Utilizadas

Tecnologia	Versão	Função no Projeto	Justificativa
.NET	8.0 (LTS)	Plataforma Base	Versão mais recente e estável (Long Term Support), oferecendo alta performance e segurança.
C#	12	Linguagem	Linguagem fortemente tipada, padrão da plataforma .NET.
ASP.NET Core Web API	8.0	Framework Back-end	Facilita a criação de endpoints REST, injeção de dependência e tratamento de requisições HTTP.
ASP.NET Core MVC	8.0	Framework Front-end	Permite o desenvolvimento rápido de interfaces web utilizando Razor Views.
Entity Framework Core	8.0	ORM (Object-Relational Mapping)	Abstrai o acesso ao banco de dados, permitindo manipular tabelas como se fossem classes C#.
SQLite	3	Banco de Dados	Banco leve, serverless e portátil. Ideal para projetos acadêmicos pois não exige instalação de servidor SGBD.
xUnit	2.5	Testes Automatizados	Framework de testes unitários utilizado para garantir a qualidade e funcionamento das regras de negócio.
Swagger / OpenAPI	-	Documentação	Gera automaticamente uma interface interativa para testar os endpoints da API.

3. Diagramas de Entidade-Relacionamento

O banco de dados foi modelado para atender às regras de negócio de reservas. A estrutura principal consiste em:

- **Tabela Clientes:** Armazena dados pessoais e credenciais de acesso.
- **Tabela Mesas:** Representa a capacidade física do restaurante. Possui status de disponibilidade.
- **Tabela Reservas:** Entidade associativa que liga quem (Cliente) reservou Onde (Mesa) e quando (Data).

4. Estrutura de pastas e código

O código-fonte está organizado da seguinte forma para facilitar a navegação:

- 📂 **Projeto-Final-main** (Raiz da Solução)
  - 📂 **swagger** *(Projeto da API)*
    - 📁 Controllers/ - Pontos de entrada da API (Clientes, Mesas, Auth).
    - 📁 Data/ - Contexto do Banco de Dados (DbContext).
    - 📁 Models/ - As classes (Cliente, Mesa, Reserva).
  - 📂 **ReservaFront** *(Projeto do Site)*
    - 📁 Controllers/ - Lógica de navegação das telas.
    - 📁 Views/ - Arquivos .cshtml (HTML + C#).
    - 📁 wwwroot/ - CSS, Imagens e JavaScript.
  - 📂 **ReservaApi.Tests** *(Projeto de Testes)*
    - 📄 TestesDeClientes.cs - Testes unitários do CRUD de clientes.

5. Fluxo de Execução

Para ilustrar o funcionamento da arquitetura, descrevemos o fluxo de "Criar uma Reserva":

1. Usuário acessa a tela de "Nova Reserva" no navegador (Front-end).

2. Usuário preenche data e mesa e clica em "Salvar".

3. O Controller do Front-end recebe os dados e faz um POST para http://localhost:5271/api/reservas.

4. A API recebe a requisição:
    - Verifica se o usuário está autenticado.
    - Verifica no banco se a mesa está livre naquela data (Regra de Negócio).

5. Se estiver tudo certo, a API salva no SQLite e retorna 201 Created.

6. O Front-end recebe a confirmação e exibe a mensagem de sucesso na tela.