DECISÕES DO PROJETO DE SOFTWARE

Revisões[1] do Documento

[1] Revisões são melhoramentos na estrutura do documento e também no seu conteúdo. O objetivo primário desta tabela é a fácil identificação da versão do documento. Toda modificação no documento deve constar nesta tabela.

Revisão	Data	Autores	Observações
A	18/08/2020	Renan Barbosa	Inclusão da decisão de usar o axios para requisições HTTP.

Sumário

1. Objetivo

Este documento tem por objetivo apresentar as decisões de construção do software em atendimento aos requisitos fornecidos para um sistema de controle de estoque.

2. Decisões de projeto

Projeto de front-end
Contexto: O front-end da aplicação deve ser feito utilizando o React.

Decisão: Criar um projeto utilizando o template ASP.NET Core Web Application e React.JS na versão 3.1 do ASP.NET Core.

Consequências: Agilidade no desenvolvimento do projeto, o template traz uma estrutura completa para o projeto. O React permite utilizar o JavaScript, ES6 e TypeScript, além do reaproveitamento de código e manutenção através da criação de componentes.

Chamada ao serviço de Back-end
Contexto: Integrar o front-end com uma WebApi enviando requisições HTTP.

Decisão: Utilizar a biblioteca axios.

Consequências: O axios fornece uma API simples para enviar as requisições HTTP e gerenciar as repostas no formato JSON. As requisições feitas pela biblioteca retornam uma promise que é compatível com o JavaScript ES6.

Projeto de back-end
Contexto: O back-end da aplicação deve ser feito utilizando REST WebApi em .NET Core.

Decisão: Criar uma WebApi em um projeto separado do projeto de front-end, utilizando o Template ASP.NET Core Web Application e API, na versão 3.1 do ASP.NET Core. Além da WebApi, o projeto de back-end deve ser composto por mais outros 2 projetos, cada projeto deverá ser desenvolvido com o seguinte propósito:

- **WebApi**: Camada que deve aceitar a entrada de solicitações HTTP na rede como GET, POST, PUT e DELETE. O retorno de todos os seus métodos deve se no formato JSON.
- **Core**: Esta camada deve conter os objetos de domínio, não deve permitir conexões de rede ou acesso a banco de dados. A camada deve permitir futuras implementações de interfaces para representar suas dependências.
- **Infra**: Camada de infraestrutura que contêm os interesses do banco de dados. Deve permitir acesso ao banco de dados, deve conter a implementação física para interfaces de acesso a dados.

Consequências: O projeto de back-end fica totalmente desacoplado do projeto de front-end, tornando a arquitetura mais limpa, de fácil manutenção e podendo ser integrada com outras aplicações que necessitem consultar o estoque de produtos, como exemplo uma aplicação mobile. Além disso o ASP.NET Core fornece uma API robusta que permite diversas configurações.

Persistência de dados
Contexto: A persistência deve ser feita utilizando o Dapper.

Decisão: A persistência com o Dapper deve ser feita na camada de infraestrutura do back-end.

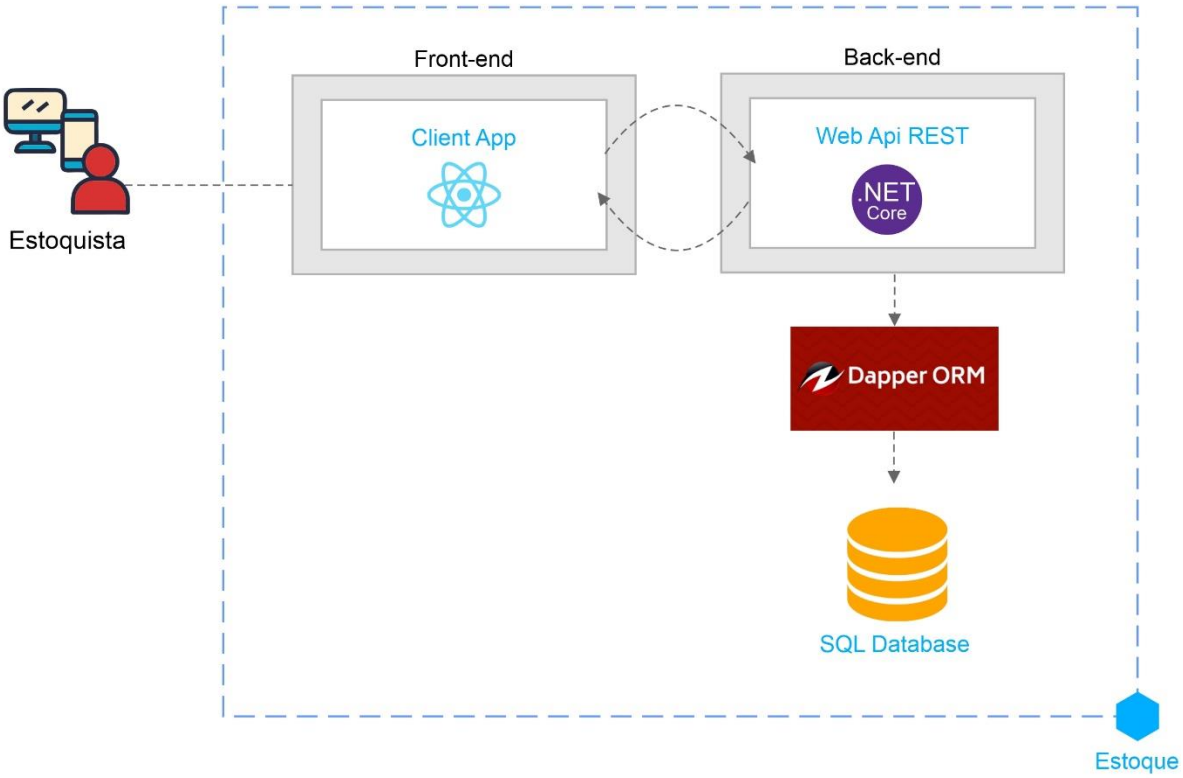
Consequências: O Dapper facilita as questões relacionadas ao banco de dados, tornado o projeto independente de uma tecnologia de banco de dados específica, ou seja, é possível mudar de banco de dados sem causar impactos nas implementações do sistema, modificando somente a string de conexão.

Banco de dados
Contexto: Utilizar um sistema de banco de dados (SGBD) relacional.

Decisão: Utilizar o SQL Server.

Consequências: O SQL Server possui uma integração simplificada com o Visual Studio ou VS Code, porém exige uma capacidade de processamento maior da máquina onde irá rodar o servidor do banco.

3. Desenho de solução



4. Desenho arquitetural