
Plano de Documentação das Decisões do Projeto
Mayardes Oliveira

Histórico de Alterações

Data	Versão	Descrição	Autor
18/01/2020	1.0	Elaboração do Front-End.	Mayardes
19/01/2020	1.1	Elaboração do Back-End.	Mayardes
19/01/2020	1.2	Elaboração da Documentação	Mayardes

Conteúdo

1	INTRODUÇÃO	4
2	OBJETIVOS DO PROJETO	4
2.1	PRINCIPAIS OBJETIVOS DO PROJETO	4
2.2	OBJETIVOS DO GERENCIAMENTO DO PROJETO.....	4
3	METODOLOGIA DE DESENVOLVIMENTO DO SISTEMA (MDS).....	4
3.1	FRONTEND DO PROJETO.....	4
3.2	BACKEND DO PROJETO	5
3.3	PRINCIPAIS PRODUTOS DO DESENVOLVIMENTO	7
4	PROCESSO DE TESTE	7
4.1	CONFIGURAÇÃO DA ROTA	7
4.2	INICIAR FRONTEND.....	7
4.3	INICIAR BACKEND	8
4.4	PRONTO PARA REALIZAR AS OPERAÇÕES.....	8

1 Introdução

Este documento fornece uma visão abrangente das decisões que foram definidas ao longo do processo de desenvolvimento.

Nele são apresentados: a metodologia de desenvolvimento, a descrição e os objetivos do projeto, as métricas aplicadas, os objetivos das iterações, o cronograma e os produtos do desenvolvimento e o processo de homologação

2 Objetivos do Projeto

2.1 Principais Objetivos do Projeto

O projeto tem objetivo em sua primeira versão de realizar o cadastro de produtos para uma empresa. Onde o estoquista será o usuário desta aplicação.

O estoquista deve realizar as seguintes operações:

- Cadastro
- Edição
- Busca
- Exclusão

A aplicação deve facilitar o gerenciamento de produtos em estoque.

2.2 Objetivos do Gerenciamento do Projeto

- Entregar os produtos deste projeto com qualidade e dentro dos prazos negociados.
- Estabelecer comunicação eficiente entre os stakeholders do projeto (ver lista dos stakeholders na seção correspondente deste documento), permitindo que todos recebam as informações que necessitam sobre o projeto.
- Acompanhar tempestivamente os riscos e dependências entre as equipes envolvidas no projeto eliminando gargalos que afetem o cronograma das atividades.

3 Metodologia de desenvolvimento do sistema (MDS)

3.1 Frontend do Projeto

- a) Node.js 12.14.1
- b) Angular 8

- Esta aplicação possui uma interface para representar nosso modelo de dados, chamada **produtos**, possuindo atributos: *Id*, *Nome*, *QuantidadeItens*, *ValorUnitario*.
- O serviço para realizar as chamadas no Backend foi definido usando o nome *Apiservices*. Este único serviço possui todas as chamadas a serem realizadas usando o protocolo HTTP.
- Este projeto possui duas telas que estão presentes na pasta estoque. Tela de cadastro/edição e listagem. Conforme solicitado pelo cliente.
- A tela de listagem possui três botões: Inclusão, Deletar e Editar. Onde Deletar/Editar estão de acordo com o item da lista selecionado. Nesta mesma tela também teremos a

funcionalidade de pesquisa, onde o estoquista poderá pesquisar pelo código do produto gerado de acordo com o seu cadastro.

- A decisão por ter escolhido o Id para busca é justamente para encontrar a chave única do produto, considerando que não houve outro atributo único informado pelo cliente.

3.2 Backend do Projeto

- a) Linguagem C#
- b) Entity Framework
- c) Implementação do Design Pattern Repository e Unit Of Work
- d) Padrão MVC.

- A linguagem C# preferida pelo candidato e decidida para o cargo pretendido.
- Entity framework Code First oferece funcionalidades para a criação de um banco de dados a partir das classes (e respectivas configurações) que representam diferentes entidades dentro de um sistema. Este recurso pode ser extremamente útil em aplicações que funcionem como provas de conceito, em que as diferentes funcionalidades foram concebidas com o objetivo de demonstrar uma arquitetura elaborada com um fim específico.
- A implementação dos padrões Repository e Unit Of Work ajudam a insolar a camada de negócio da camada de armazenamento de dados para facilitar as implementações de Unit Test ou Test Driven Development (TDD), como mostra a figura abaixo:

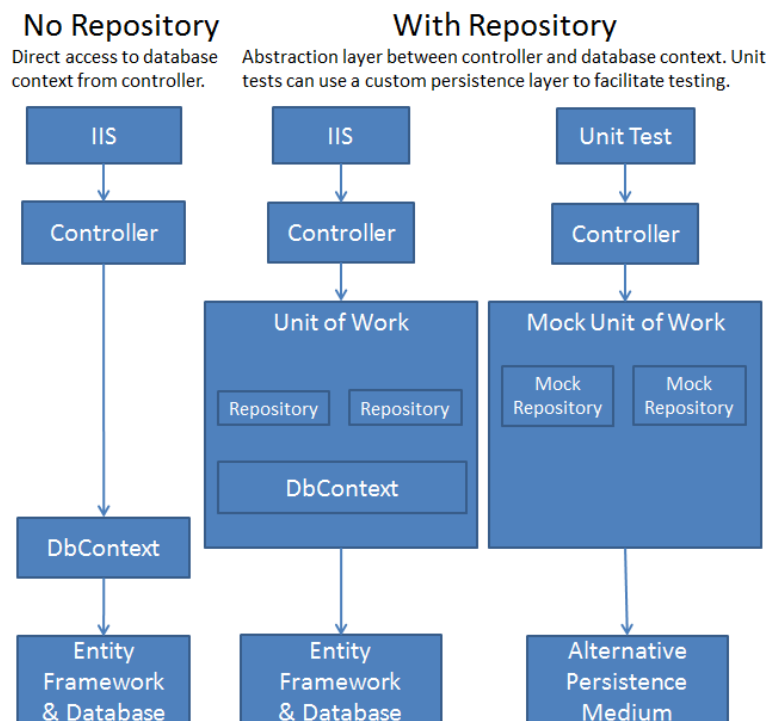


Figura 1- Padrão Repository e Unit of Work

- Conceitualmente a classe que implementa o padrão Repository encapsula o conjunto de objetos e as operações executadas sobre eles, proporcionando uma visão mais orientada a objeto da camada de acesso a dados.
- O padrão Unit Of Work controla o que fazemos durante uma transação geralmente executada sobre uma regra de negócio e coordenando essas alterações para a camada de acesso a dados.
- É um dos padrões de design mais comuns nas empresas de desenvolvimento de software. De acordo com Martin Fowler, o padrão de unidade de trabalho "mantém uma lista de objetos afetados por uma transação e coordena a escrita de mudanças e trata possíveis problemas de concorrência".
- O padrão MVC possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o **Model**, o **Controller** e a **View**, executa o que lhe é definido e nada mais do que isso.

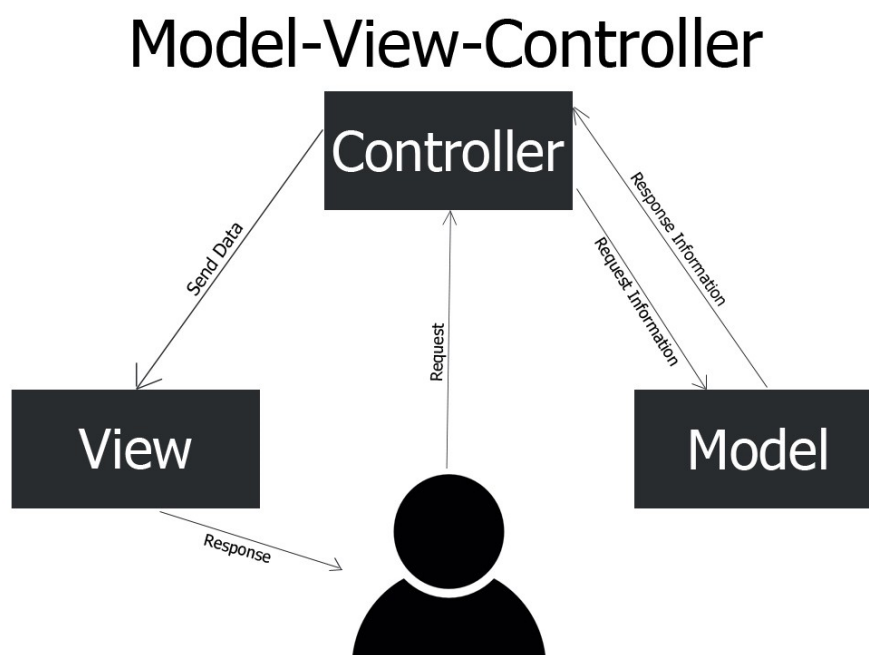


Figura 2 - Etapas de execução do padrão MVC

- A utilização do padrão MVC trás como benefício isolar as regras de negócios da lógica de apresentação, a interface com o usuário. Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, proporcionando assim muito mais flexibilidade e oportunidades de reuso das classes.
- Uma das características de um padrão de projeto é poder aplicá-lo em sistemas distintos. O padrão MVC pode ser utilizado em vários tipos de projetos como, por exemplo, desktop, web e mobile.

3.3 Principais Produtos do Desenvolvimento

A tabela a seguir apresenta as funcionalidades que o este produto gera para o cliente.

Módulo	Funcionalidades
Tela-Listagem	Listar todos os produtos cadastrados. Listar um único produto. Excluir um único produto listado. Editar um único produto listado.
Tela-Cadastro	Cadastrar um produto. Editar um produto selecionado na tela de listagem.

4 Processo de testes produto

4.1 Configuração da rota.

- Para realizar os testes em homologação será necessário verificar a configuração da rota:

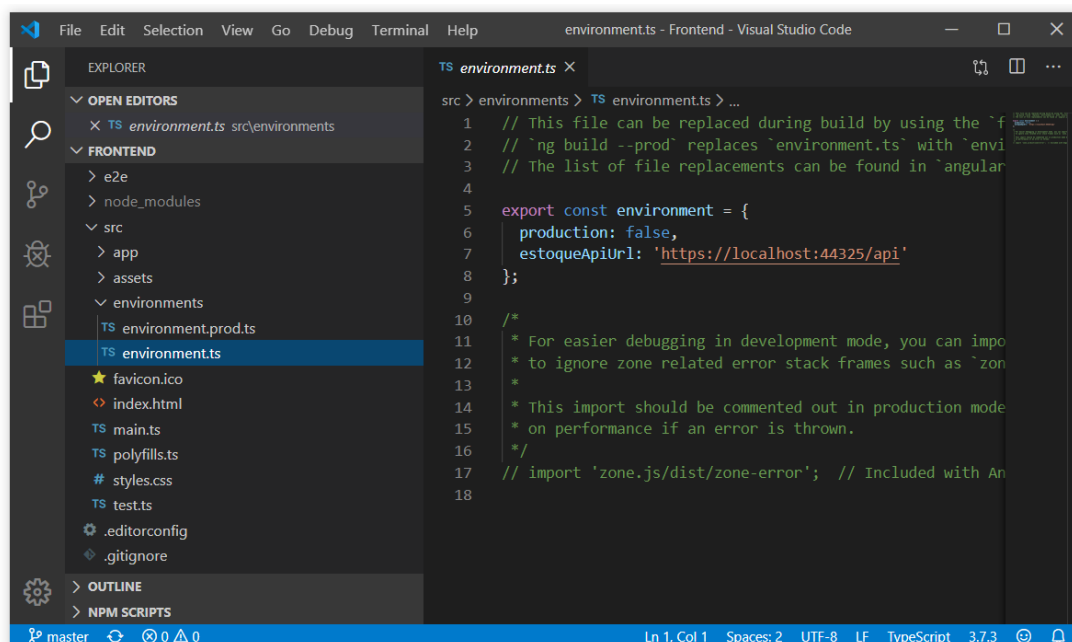
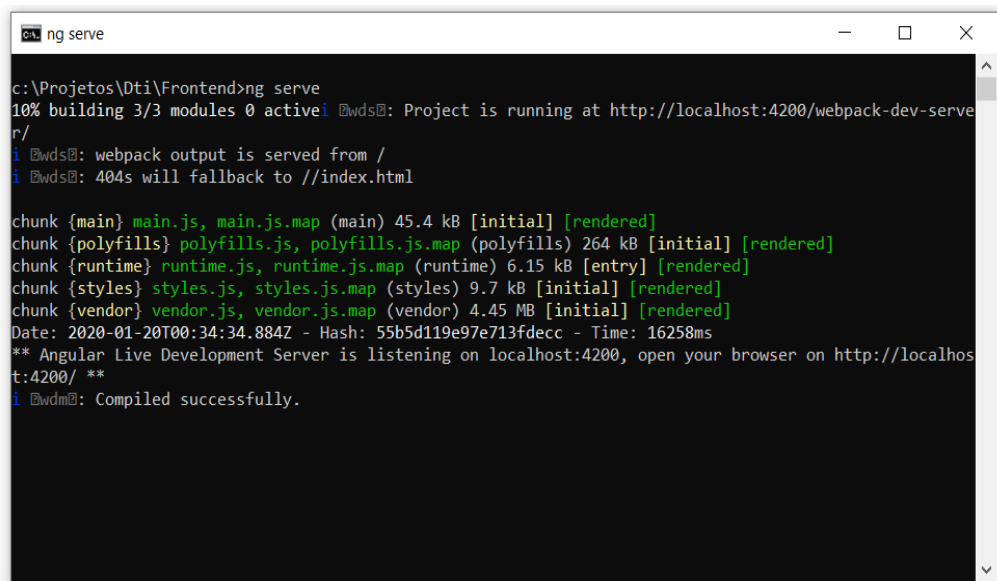


Figura 3 - Configuração da rota para API

4.2 Iniciar o Frontend

- Iniciar o Frontend com o comando `ng serve`:



```
c:\Projetos\Dti\Frontend>ng serve
10% building 3/3 modules 0 active i @wds: Project is running at http://localhost:4200/webpack-dev-server/
i @wds: webpack output is served from /
i @wds: 404s will fallback to //index.html

chunk {main} main.js, main.js.map (main) 45.4 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 264 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 9.7 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 4.45 MB [initial] [rendered]
Date: 2020-01-20T00:34:34.884Z - Hash: 55b5d119e97e713fdecc - Time: 16258ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
i @wdm: Compiled successfully.
```

Figura 4- Iniciando frontend

4.3 Iniciar o Backend

- Abrir a solução do visual Studio e alterar a `ConnectionString` no `WebConfig`, localizado na raiz da aplicação.

```
<connectionStrings>
  <add name="ProductConnection" connectionString="Data Source=(localdb)\MSSQLLocalDB;
    Initial Catalog=DatabaseProdutos; Integrated Security=true" providerName="System.data.SqlClient" />
</connectionStrings>
```

4.4 Pronto

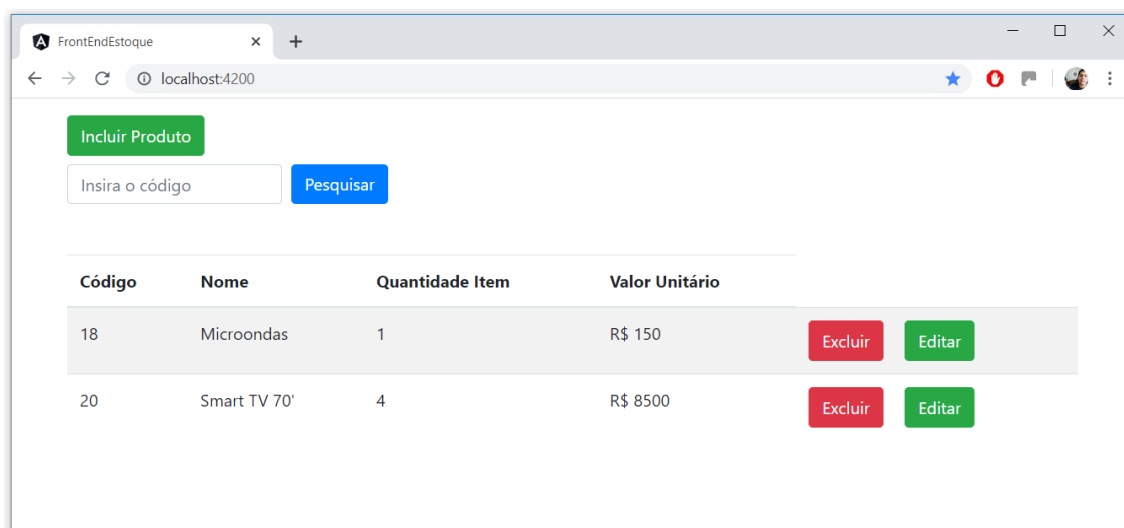


Figura 5- Pronto