Documentação das decisões de projeto

Problema 1
Não consegui fazê-lo a tempo.

Problema 2
Não utilizei o Dapper para persistência de dados, devido a falta de conhecimento da tecnologia e seria necessário um pouco mais de tempo para implementação. Então utilizei o EntityFramework e code firsr migrations que é um dos mais utilizados e que tinha pelo menos conhecimento básico. O que tenho mais conhecimento é o Nhibernate.

Sobre os diagramas e desenho arquitetural também não consegui fazer a tempo.

Documentação sobre decisão de estrutura do projeto back-end

A solução foi divida em 4 projetos:

- Domínio: projeto responsável pelas entidades (classes referentes ao banco de dados).
- Data: responsável pela conexão com banco de dados, mapeamento das entidades e consultas.
- Controller: responsável pelas "regras de negócios" do projeto, como validações e outros tratamentos.
- Api: Responsável por pelas rotas, requisições e qualquer controle relacionado à conexão com front-end;

Projeto Dominio:
Somente classe de estoque.

Projeto Data:
Foi criado a classe Interface IRepositorioBase que contem a assinatura das principais funcionalidade de um crud e que pode ser usado para todas a entidade que estão ligadas ao banco de dados.
A classe RepositorioBase contem a implementação dos métodos que existe na interface IrepositorioBase e que pode ser usado para todas a entidade que estão ligadas ao banco de dados.
Criado a Classe IRespositorioEstoque e RespositorioEstoque herdando das demais classes citadas acima que corresponde a ações específicas da entidade estoque.

Projeto Controller:
A classe ControllerBase faz a comunicação(chamada) com os métodos que está no projeto Data através da interface e que pode ser usado para todas a entidade que estão ligadas ao banco de dados.

A classe ControllerEstoque que herda da classe acima que corresponde a ações específicas da entidade estoque.

Projeto API:
Contem duas classes:
ValuesController, nesta classe contem uma rota de execução inicial, somente para teste e uma rota para pegar o token de validação de autorização(autenticação) das rotas. Rotas sem autenticação.

EstoqueController: Contem as rotas específicas referente ao estoque. Todas as rotas estão são necessário de autenticação.

Documentação sobre decisão de estrutura do projeto front-end

Criado classe axios.js que faz a comunicação com o back-end e responsável pela autenticação das rotas. Está buscando o token a toda requisição, não é a melhor forma, pois pode ser verificado a expiração do token, mas pelo curto tempo está desta forma.

A classe stockService é referente as requisições específicas das rotas de estoque.

Dentro da pasta componentes existem duas classes:
Index, corresponde a tela inicial que exibe a listagem de produtos.
StockNewEdit, referente a tela de cadastro e de edição de produtos.

Para interface foi utilizado componentes material-ui