Grupo 15 - Tech Challenge 1

Criado em	@August 8, 2024 11:06 PM

Desenvolvimento do Projeto

O projeto foi desenvolvido através de etapas visando respeitar os processos do desenvolvimento de software e garantir o entendimento dos requisitos por todos do grupo antes de evoluir para o desenvolvimento.

Discovery

Para o entendimento completo dos requisitos passados para o projeto, foram realizadas reuniões com todos os integrantes da equipe para realizar a leitura e entendimento em conjunto do documento de requisitos passados para o projeto(*Plataforma Postech Fiap > Tech Challenge Fase 1*), envolvendo:

- **Brainstorm:** Após a leitura dos requisitos foi realizada uma sessão de brainstorm para levantar todas as ideias de cada integrante, visando reunir, filtrar e estruturar as ideias
- **Desenho de Estrutura/Arquitetura:** Foi realizada uma sessão de escolha de arquitetura do projeto visando definir qual a estrutura do projeto e como serão atendidos os requisitos
- **Levantamento das atividades:** Através do Trello, foi realizada a separação dos itens a serem desenvolvidos em atividades(cards), atribuindo dependências e prazos para execução
- **Atribuição de atividades:** De acordo com o conhecimento e facilidade de cada membro da equipe com cada tema/atividade, foram atribuídas as atividades para cada um

Desenho de Arquitetura

Após o entendimento dos requisitos, foi elaborada a estrutura da aplicação no seguinte formato:

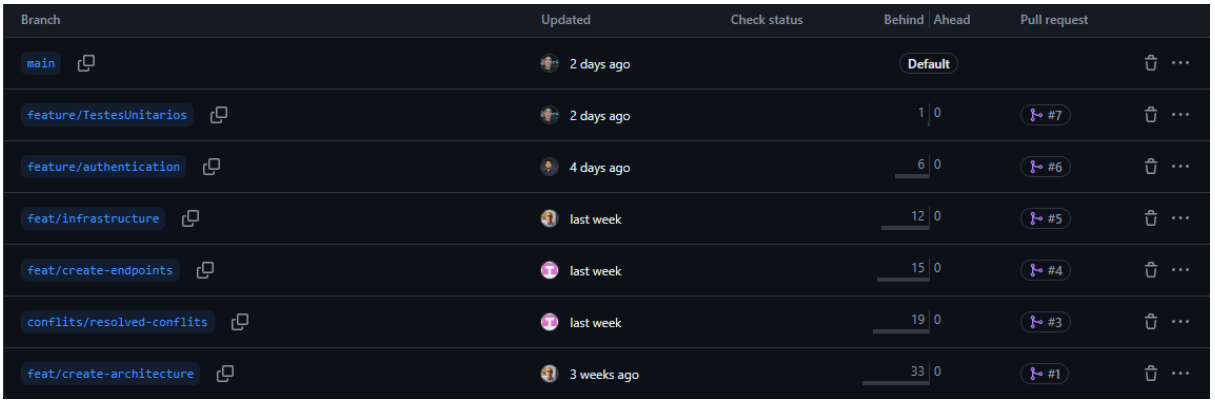
- **Api:** Camada de apresentação que contém os controllers da API, sendo o ponto de contato com o usuário/aplicações externas
- **Application:** A camada que contém os handlers da aplicação, que são as classes responsáveis pelo processamento das requisições que chegam nas controllers. Neste contexto a controller chama o seu respectivo handle que vai realizar o processamento da requisição chamando os devidos métodos de outras camadas e orquestrando esse fluxo de processamento
- **Domain:** Camada onde estão localizados os principais componentes da aplicação voltados ao domínio(Entidades e Repositórios)
- **Infrastructure:** Camada de suporte às demais camadas, responsável por fornecer recursos úteis para as demais áreas do projeto(Criptografia, Conexão ao banco de dados, Middlewares)
- **TestesUnitarios:** Projeto de testes unitários da aplicação
- **Crypto.Test:** Projeto de teste da criptografia, onde podem ser executadas operações de criptografia e decriptografia via console(um projeto que não faz parte diretamente da solução, mas é um utilitário importante para gerar e ler valores criptografados)

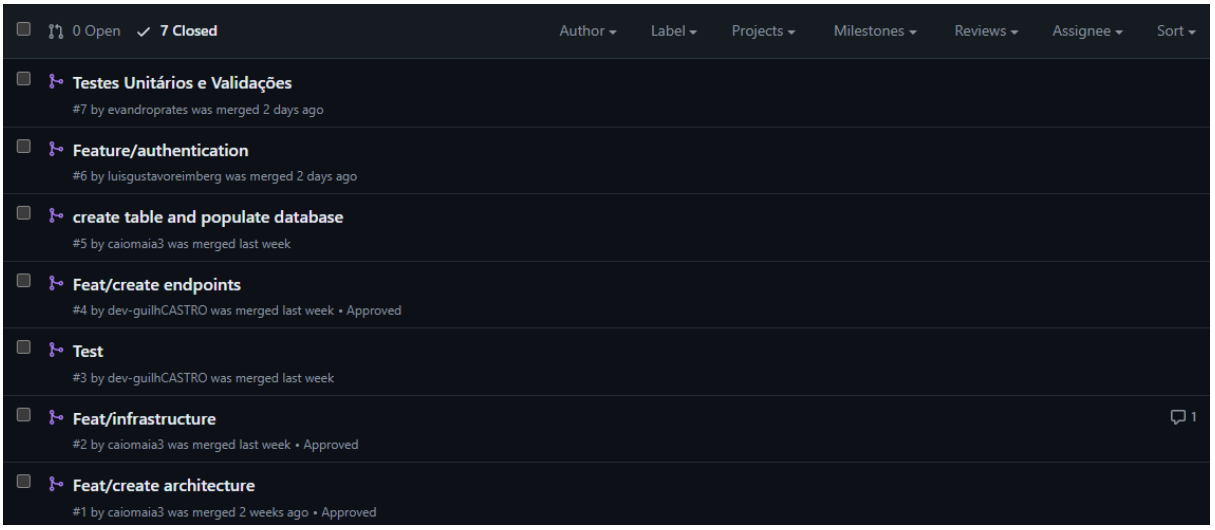
A estrutura de arquitetura apresentada foi escolhida para atender as principais necessidades do projeto visando organizar o código da melhor forma e atribuir cada responsabilidade específica por camada e visando atender as boas práticas de:

- Reutilização de código: Os códigos que podem ser reutilizados por diversas camadas da aplicação estão em uma camada que suporta das demais, ou seja, que pode ser referenciada por todas as outras camadas
- Exposição limitada dos recursos da aplicação: Apenas os recursos necessários(controllers) estão expostos para consumo por outras aplicações/usuários
- Facilidade de entendimento do código: Com métodos menos verbosos e classes com responsabilidades bem definidas, o código se destaca por ser de fácil entendimento mesmo para quem está o vendo pela primeira vez
- Divisão de responsabilidades: Camadas, classes e métodos contam com responsabilidades delimitadas e bem definidas

Desenvolvimento e Versionamento de Código

O código foi desenvolvido paralelamente pelos integrantes da equipe e versionado através do git/github, o que possibilitou a junção segura e controlada dos diferentes trechos de código e projeto. Neste quesito, cada integrante atuou em uma branch separada e, após a finalização da tarefa, foram realizados os processos de pull request, code review e merge





Recursos do projeto

Para atender os recursos e requisitos, o projeto conta com diversos recursos desenvolvidos utilizando boas práticas

Testes unitários

- A aplicação possui testes, desenvolvidos com XUnit, responsáveis por validar todos os pontos de entrada e operações que podem ser realizados, garantindo e testando tanto o "caminho feliz"/sucesso nas operações, quanto as validações de erro como, por exemplo, os parâmetros obrigatório faltantes

Validação de valores obrigatórios

- A API conta com validações nas DTOs que garantem o preenchimento dos parâmetros obrigatórios para cada operação, através do Fluent Validation

Segurança

- Com um formato de autenticação via JWT, a API possui um processo de autenticação utilizando a validação das senhas dos usuários através dos hashs, ou seja, a senha original não é acessível e não é utilizada para validação, todo o processo de validação é feito pelo AspNetCore.Identity que cria os hash das senhas que são armazenados no banco de dados e compara esses hashs para login/autenticação.
- O processo de autorização da aplicação é definido através do tipo de usuário(role), que define quais endpoints/operações são acessíveis por quais tipos de usuários, impedindo que funções administrativas e mais críticas sejam realizadas/acessadas por usuários comuns
- A senha dos usuários não é exposta de forma alguma na aplicação, garantindo maior segurança

Criptografia

- A chave JWT para geração de tokens e a string de conexão do banco de dados estão armazenadas de forma criptografada, utilizando o algoritmo AES que tem suas chaves de criptografia armazenadas em variáveis de ambiente do sistema, mantendo maior segurança contra acesso indevido, visto que nenhuma informação pertinente a criptografia está armazenada em código ou arquivo de configuração

Docker

- A solução utiliza o Docker para executar o banco de dados(SQL Server), facilitando a compatibilidade e configuração inicial da solução, assim como deixando a solução menos acoplada/dependente de um ambiente específico

Middleware de erro

- Qualquer erro(exception) que aconteça na aplicação durante o processamento será captura de por um middleware de exception,

implementado para capturar a exceção, logar a informação de erro de forma segura(em arquivo de log, console ou banco de dados) e expor apenas uma mensagem genérica de erro, evitando que informações da aplicação sejam expostas no response da aplicação, como por exemplo o trace/caminho da exceção