**Desenvolvimento - Projeto Fullstack - Varejo360**

* Para iniciar o projeto fiz uma organização no github para deixar os repositórios do back e do front juntos;

BACKEND

* Comecei pelo back-end, criando um projeto Java conforme as especificações em https://start.spring.io/.
* Estruturei as pastas do projeto, incluindo controller, model, repository e service.
* Criei o arquivo index.html na pasta static.
* Construí os modelos Product e User.
* Desenvolvi os repositórios ProductRepository e UserRepository.
* Implementei as rotas GET e POST para User, desenvolvendo os respectivos services e controllers, e testei o código a cada etapa.
* Configurei alguns parâmetros no arquivo application.properties, na pasta resource.
* Criei um arquivo data.sql para inserção automáticas no banco de dados ao iniciar a aplicação. (Obs: adicionei configurações no application.properties para isso.)
* Completei o CRUD de User.
* Criei a pasta Dto e o UserDto para validações, fazendo as substituições necessárias nos services e controllers.
* Com as rotas de User funcionando, repeti o processo para Product. Já estabeleci a relação entre Product e User, passando o user\_id no corpo da requisição.
* Preparei as rotas para Product, implementando AppException e GlobalException.
* Implementei regras de negócio nos services, como email e código de produto únicos.
* Realizei a conexão com o PostgreSQL, adicionando configurações no application.properties.
* Com o back-end estável, iniciei a implementação da autenticação e da rota de login.
* Instalei as dependências necessárias (spring-boot-starter-security, spring-security-test).
* Implementei UserDetail em User e adicionei um novo método no UserRepository para retornar o campo de login.
* Criei o AuthService e o SecurityConfig para ajustar temporariamente as configurações do Spring Security, além de criar novas autenticações.
* Desenvolvi a rota de login no AuthController, com criptografia de senha e geração de token.
* Refatorei os controllers e services de Product para aceitar o token, eliminando a necessidade de passar o user\_id na requisição.
* Existe uma pasta chamada “workspace” dentro da pasta “backend”, com um arquivo exportado do insomnia com todas as rotas da API.

FRONTEND

* Iniciei o front-end com Vite, criando um projeto TypeScript.
* Instalei todas as dependências que planejei utilizar.
* Limpei o projeto inicial do Vite e comecei a estruturá-lo, construindo pastas e arquivos necessários.
* Desenvolvi cada arquivo com uma estrutura mínima.
* Normalmente, começo pelos contexts (providers) e, com o back-end pronto, testo cada funcionalidade desenvolvida.
* Configurei o CORS da API, o que demandou algum tempo para encontrar a solução.
* Implementei proteção de rotas, além das páginas de login e cadastro, que já estão funcionais.
* Comecei a desenvolver o dashboard, implementando primeiro o cabeçalho com informações do usuário e o logout.
* Renderizei a tabela de produtos e implementei suas funcionalidades (filtro, ordenação, paginação), testando cada uma delas.
* Adicionei o formulário para criar produtos.
* Apliquei um CSS mínimo para organizar visualmente o projeto, mas ainda falta implementar o modal de edição de produto.
* Alterei o tipo do campo 'code' para String na API, facilitando o processo de atualização.
* Agora, resta finalizar o CSS.
* Após concluir o CSS, iniciei o processo de dockerização do projeto. Como ainda não havia estudado Docker, precisei aprender desde o início. Criei imagens separadas para os projetos, mas enfrentei dificuldades para conectar o back-end ao SQL.
* Decidi fazer o deploy do projeto usando o GitHub da organização criada no início.
* Com o deploy concluído, retomei a dockerização e resolvi o problema de conexão com o banco de dados. Contudo, encontrei um problema com o front-end, que solucionei atualizando a versão do Node e alterando a porta.

OBS: Como comecei o projeto na organização, os commits ficaram lá, só quando o projeto já estava quase pronto eu o copiei para o “fork”, então te adicionei também na organização para fins de avaliação dos commits.

* Lembrando que a API está em um servidor gratuito e precisa de alguns segundos (talvez minutos) para “acordar”.

GIT\_ORGANIZAÇÃO: <https://github.com/vaga-fullstack-varejo360>

DEPLOY\_URL\_FRONT: <https://frontend-iota-eight-62.vercel.app/>

DEPLOY\_URL\_BACK: <https://varejodb.onrender.com>