**Prova - API de consulta de usuários:**

Inicialmente foi criado as classes principais para a funcionalidade da consulta de usuários:

**Entidade:** User.java

foi criada para apresentar a tabela user com os dados id, nome e cpf

**Repositório:** UserRepository.java

foi criado o repository para manipular os dados da tabela, sem os comando sql

**Dto:** UserDTO.java

foi criado o DTO para proteger a estrutura dos dados do banco de dados ao retornar

**Serviço:** UserService.java

Foi criado a função findUsers, utilizando o specification para a busca dinâmica com os parâmetros se passados, e aí passa para o repository buscar os dados, ele devolve uma Page<User> que depois passa a ser uma lista de UserDTO.

**Controller:** UserController.java

Foi criado o endpoint: “/api/users”.

**Dados:** data.sql para popular o banco.

foi criado para adicionar uns dados iniciais ao banco de dados para teste.

**DESAFIOS**

O primeiro desafio foi um erro de incompatibilidade que estava dando erro ao iniciar o projeto, estava setado o Java 24 e tive que configurar para o Java 21.

O segundo foi alguns imports que estava me deixando confuso na montagem do service mas acabei achando o problema e resolvendo.

Também tive alguns problemas em relação ao banco de dados pois o data.sql estava sendo executado antes do hibernate criar a tabela user, então foi adicionar no arquivo application.properties a linha:  
*spring.jpa.defer-datasource-initialization=true*

e foi resolvido.

**Estrutura Inicial do Projeto**

* Criação das classes principais da funcionalidade:
  + Entidade @Entity **User** (name, cpf, phone).
  + Repositório **UserRepository** estendendo JpaRepository e JpaSpecificationExecutor.
  + Classe **UserService** para a lógica de negócio, usando Specification para filtros dinâmicos.
  + Classe **UserController** para expor o endpoint GET /api/users.
  + Arquivo data.sql para popular o banco com dados de teste.