Cronograma e Planejamento

Nome do projeto: Microsserviço de cadastro/gerenciamento de usuários.
Autor: Leonardo Santos

Objetivo do projeto:
Aprimorar habilidades de desenvolvimento de software a partir de uma demanda realista: a criação de uma API REST para cadastro e gerenciamento seguro de um usuário. O projeto incluirá levantamento de requisitos, documentação detalhista e implementação seguindo as boas práticas de segurança e arquitetura.

Tecnologias e padrões implementados:
Java, Spring Boot, Spring JPA, Spring Security, AWS, Docker, PostgreSQL, Maven, JUnit, Mockito, Spring Actuator, arquitetura em camadas (layered), microsserviços escaláveis, API REST, SOLID e ACID.

Requisitos:
Deve permitir o cadastro de um usuário, com os seguintes dados:

- Nome (mínimo de 3 caracteres)
- Data de nascimento (válida e com idade superior a 16 anos)
- E-mail (válido e único)
- Senha de acesso (mínimo 8 caracteres, letras e números, maiúscula e símbolo)
  Senhas serão armazenadas de forma segura com o algoritmo de hash Bcrypt.

Deve permitir o login de usuários já cadastrados a partir de:
A autenticação será feita via JWT.

- E-mail
- Senha

Deve permitir que usuário com acesso administrador acesse a lista de todos os usuários cadastrados, retornando:

- ID
- Nome
- Data de nascimento
- E-mail

Deve permitir que usuário com acesso administrador delete um usuário cadastrado, a partir do:

- ID

Persistência:
Dados serão armazenados em um banco de dados relacional PostgreSQL.

Testes:
Serão implementados testes automatizados para garantir confiabilidade e qualidade. Será utilizado Junit e Mockito.

I. **Testes Unitários:**
   - Criar usuários com dados válidos;
   - Criar usuário com e-mail já cadastrado (deve falhar);
   - Criar usuário com senha que não atende aos requisitos (deve falhar);
   - Tentativa de login com dados corretos;
   - Tentativa de login com dados incorretos;

II. **Testes de Integração:**
- Autênticação JWT funcionando corretamente;
- Validação de permissões do administrador;

III. **Testes de API:**
- Endpoints principais;
- Simular grandes quantidades de requisições simultâneas;

IV. **Testes de Segurança:**
- Injeção SQL;
- Força bruta no Login;
- JWT inválidos;

Integração:
Uma imagem e um conteiner serão criados utilizando Docker, e será feito o deploy na nuvem AWS EC2, onde deverá ficar disponível para ser utilizada como um microsserviço para outros sistemas.

Cronograma:

1. **Planejamento e Setup:**
   1. Criar repositório GIT;
   2. Criar o banco de dados que armazenará os dados da aplicação;
   3. Configurar o ambiente de desenvolvimento Spring e Maven;
   4. Criar as entidades e os endpoints básicos;

Entrega: Projeto configurado e estrutura básica já estabelecida.

2. **Processamento dos usuários:**
   1. Criar os diferentes tipos de acesso: Usuário e administrador;
   2. Criar a validação dos dados inseridos;
   3. Criar testes unitários com JUnit e mockito;

Entrega: API armazenando dados corretamente.

3. **Segurança:**
   1. Configurar o sistema de autenticação com Spring security e JWT;
   2. Adicionar logs com actuator;

Entrega: API com segurança aprimorada.

4. **Deploy e documentação:**
   1. Criar documentação Swagger;
   2. Criar docker compose;
   3. Configurar CI/CD com github actions;
   4. Fazer deploy na nuvem AWS;

Entrega: API documentada e rodando na nuvem.