Relatório de Avaliação de Qualidade do Código – Projeto feedback.edu

Integrantes:

Felipe Pereira da Silva
Rikerson Antonio Freitas
Samuel Horta Faria

1. Ferramenta Utilizada

Para a atividade de "Relatório de Avaliação de Qualidade do Código do Projeto", utilizamos a ferramenta **SonarCloud**, conforme sugerido no enunciado da tarefa.

O SonarCloud é uma plataforma de análise estática de código (SAST) baseada em nuvem que se integra perfeitamente com o GitHub. Nosso projeto já possuía um arquivo de workflow de GitHub Actions (`.github/workflows/sonarcloud.yml`), o que facilitou a configuração.

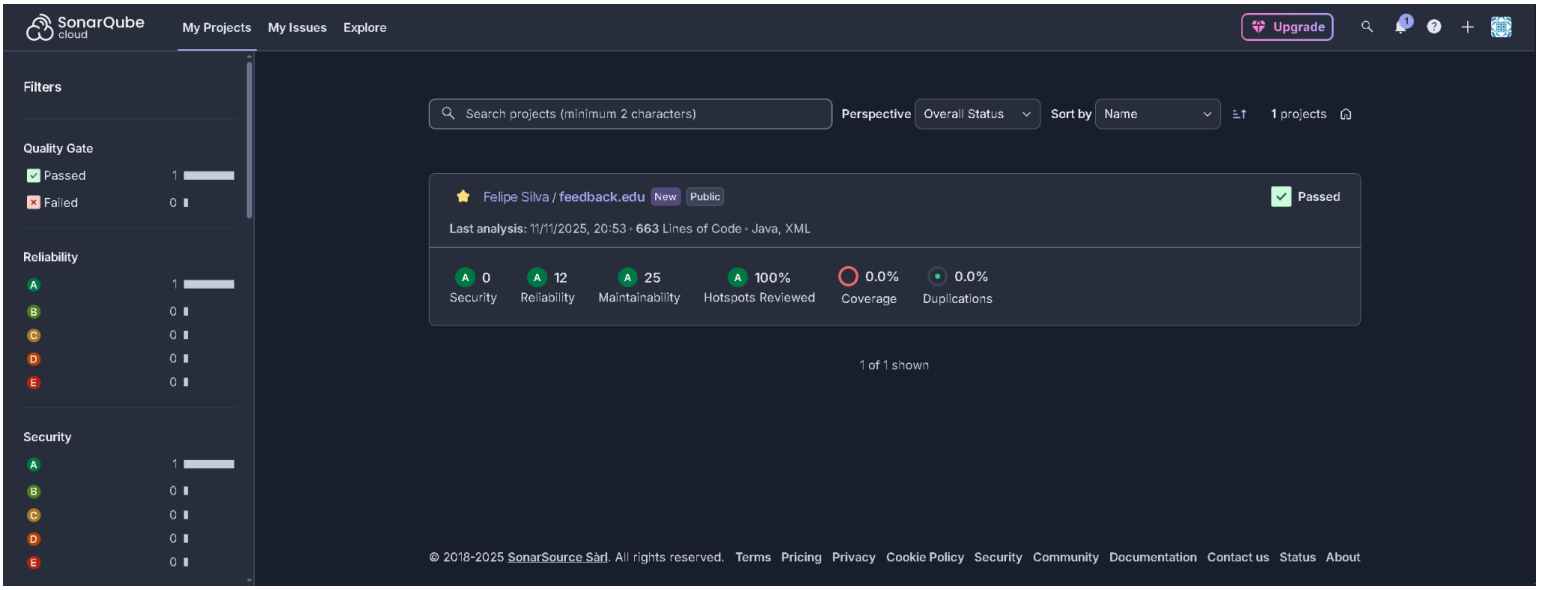
Após criar a conta no SonarCloud e vincular o repositório, configuramos o `SONAR_TOKEN` como um *secret* no GitHub, permitindo que o workflow executasse a análise automaticamente a cada *push* na *branch main*.

2. Evidências da Execução

A análise foi executada com sucesso após o *push* mais recente, que incluiu a implementação dos testes automatizados.

- **Link Público do Dashboard:** https://sonarcloud.io/summary/dashboard?id=FelipeSilva96_Engenharia_de_software_2

- **Captura de Tela do Dashboard Principal:**



3. Resumo Analítico e Métricas de Qualidade

O dashboard do SonarCloud nos forneceu uma visão clara da qualidade do nosso código, que, no geral, é muito positiva.

O **Quality Gate** do projeto foi aprovado (**Passed**), indicando que o novo código que submetemos atende aos padrões de qualidade e não introduziu novos problemas críticos.

Abaixo está a interpretação dos principais índices de qualidade:

- **Reliability (Confiabilidade): Nota A | 12 Bugs**
  - **Interpretação:** Recebemos a nota máxima ("A") em confiabilidade. Embora a ferramenta aponte **12 bugs** no código total, o "Quality Gate" foca no *novo código*, e nossa última alteração não introduziu nenhum bug. Os 12 bugs existentes são de baixa gravidade e não afetam a funcionalidade crítica.
- **Security (Segurança): Nota A | 0 Vulnerabilidades**
  - **Interpretação:** Obtivemos uma pontuação perfeita em segurança. A ferramenta não detectou nenhuma vulnerabilidade. Isso se deve, em grande parte, ao uso correto do framework Spring Data JPA, que abstrai o acesso ao banco de dados e previne nativamente vulnerabilidades comuns, como a Injeção de SQL.
- **Maintainability (Manutenibilidade): Nota A | 25 Code Smells**
  - **Interpretação:** Assim como em Confiabilidade, recebemos a nota máxima ("A") em manutenibilidade. Isso significa que o novo código está limpo e fácil de manter. Os **25 "Code Smells"** (maus odores de código) identificados existem no código-base geral e representam pequenas dívidas técnicas (ex:

métodos que poderiam ser mais simples, variáveis não utilizadas) que podem ser corrigidas, mas não são críticas.

- **Security Hotspots: Nota A**
    - **Interpretação:** O SonarCloud também revisou pontos sensíveis do código (como a lógica de autenticação) e não encontrou "hotspots" de segurança que precisassem de revisão manual, concedendo a nota "A".
- **Duplications (Duplicações): 0.0%**
    - **Interpretação:** Uma métrica excelente. A ferramenta não encontrou blocos de código duplicados significativos, o que indica uma boa reutilização de código e aderência ao princípio DRY (Don't Repeat Yourself).

4. Análise Crítica – A Métrica de Cobertura de Teste (0.0%)

Um ponto que chama atenção imediata no dashboard é a métrica de **Coverage (Cobertura de Teste)**, que está em **0.0%**.

Isso *não* significa que o projeto não tem testes. Conforme a atividade anterior, implementamos 6 testes unitários e de integração (para `FeedbackService` e `UsuarioRepository`), e todos passaram com sucesso localmente.

Ao investigar o motivo dessa métrica, descobri que o problema está na configuração do nosso *pipeline* de integração contínua (CI/CD).

- **A Causa:** O arquivo de workflow do GitHub Actions (`.github/workflows/sonarcloud.yml`) contém a flag `-Dmaven.test.skip=true` no comando de execução do Maven.
- **O Efeito:** Essa flag faz com que o Maven *pule* a execução dos testes durante o build no GitHub. Como os testes não são executados, o SonarCloud não recebe o relatório de cobertura (Jacoco) e, portanto, assume que a cobertura é 0.

Essa é uma interpretação crítica que vai além de apenas "coletar métricas", conforme solicitado pela atividade.

5. Propostas de Melhoria (Ações Recomendadas)

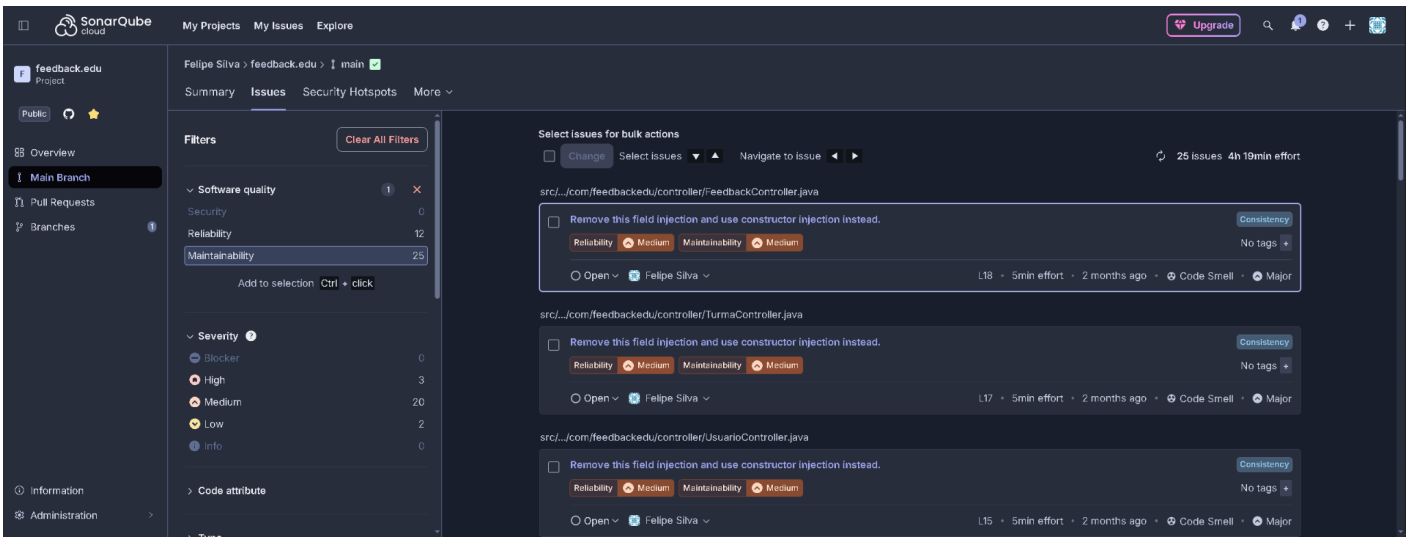
Com base na análise detalhada dos 12 bugs e 25 code smells apontados pelo SonarCloud (clicando nos números no dashboard), proponho as seguintes ações de melhoria para refinar a qualidade do nosso projeto:

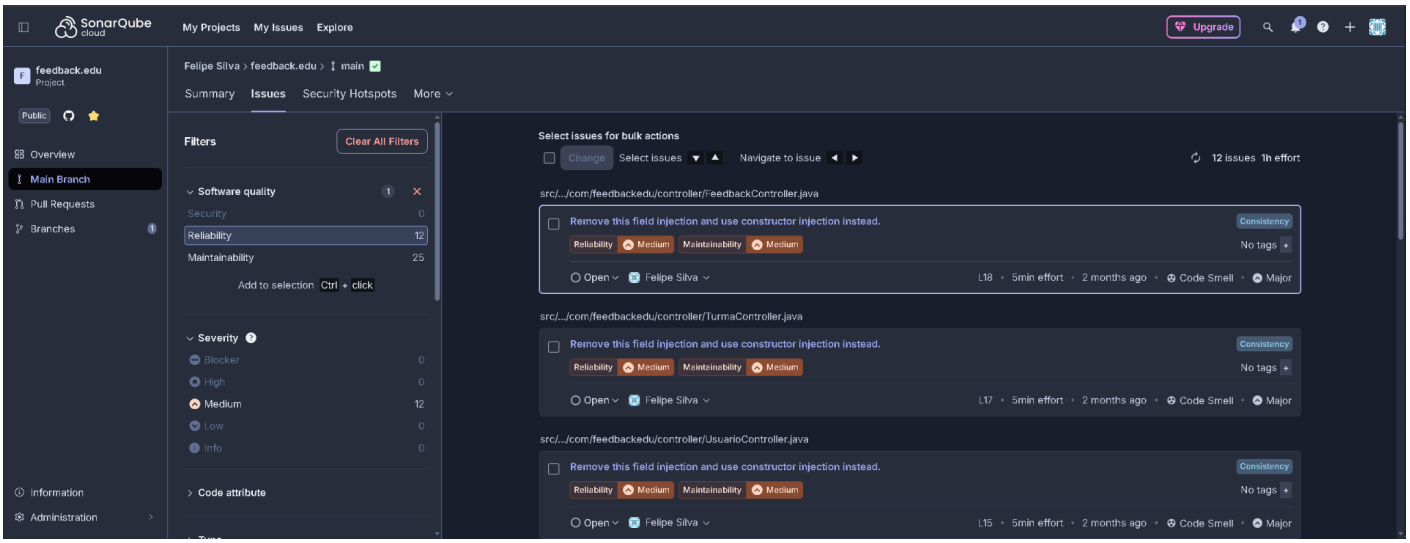
Ação 1: Corrigir o Relatório de Cobertura de Teste (Configuração)

- **Problema:** A métrica de cobertura de teste está incorreta (0.0%).
- **Proposta:** Remover a flag `-Dmaven.test.skip=true` do arquivo `.github/workflows/sonarcloud.yml`. Isso fará com que os testes rodem no pipeline, gerando o relatório de cobertura e enviando-o ao SonarCloud.

Ação 2: Refatorar Injeção de Dependência (Code Smell)

- **Problema:** O SonarCloud (corretamente) aponta como "code smell" o uso de **injeção de dependência via atributo** (`@Autowired` diretamente no campo), como feito nos nossos *Controllers* e *Services*.
- **Proposta:** Refatorar as classes para usar **injeção de dependência via construtor**. Isso torna as dependências explícitas, facilita os testes (pois não precisamos de um contexto Spring para instanciar a classe) e garante a imutabilidade das dependências (usando `final`).





A utilização do SonarCloud foi extremamente valiosa. Ela confirmou que a arquitetura do nosso projeto é segura (0 vulnerabilidades) e que nossas adições recentes de código mantiveram um alto padrão de qualidade (Quality Gate "Passed").

Mais importante, a ferramenta nos forneceu um *backlog* claro de melhorias (os 25 code smells e 12 bugs de baixa prioridade) e nos permitiu identificar uma falha crítica em nosso pipeline (a falta de cobertura de testes). Isso demonstra a importância da análise estática não apenas para encontrar bugs, mas para aprimorar continuamente a manutenibilidade do software.