Relatório de Avaliação da Qualidade do Código

Disciplina: Engenharia de Software II

Semana 14 – Avaliação de Qualidade do Código

Tema do Projeto: Plataforma para troca e venda responsável de cavalos

1. Identificação do Projeto

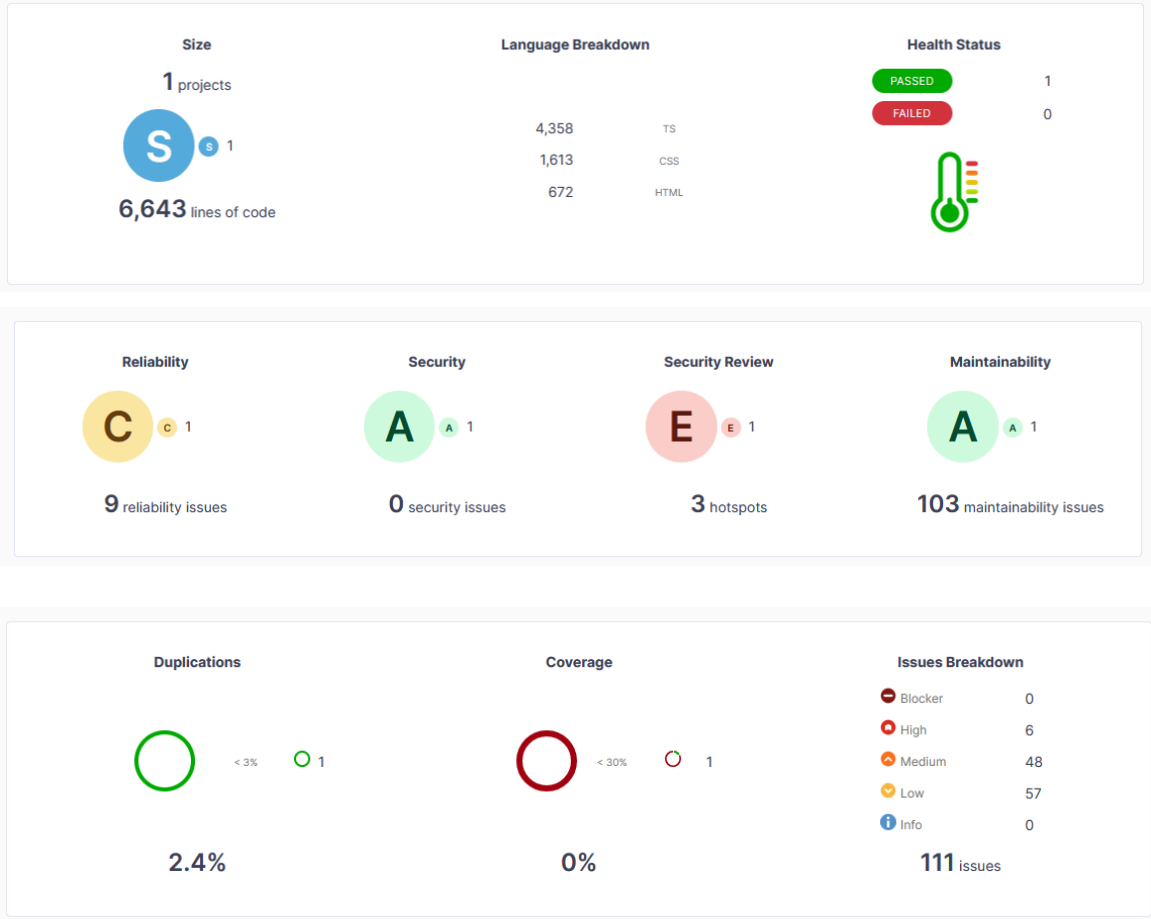
Nome do grupo/alunos: EquiTrade - João Avila, Priscila, Iuri Saad, Pedro Rosemberg

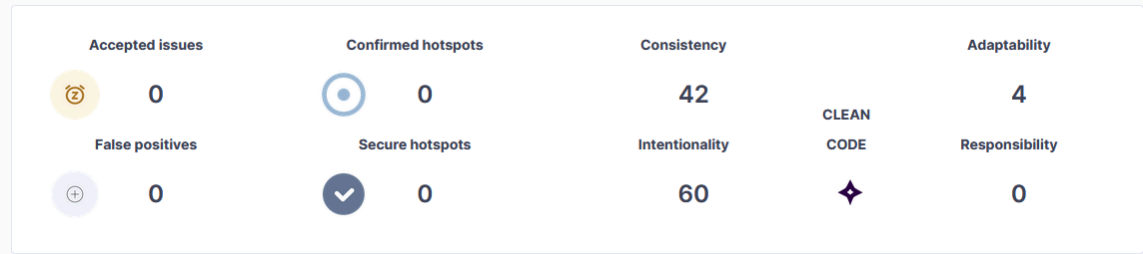
Ferramenta utilizada: SonarQube

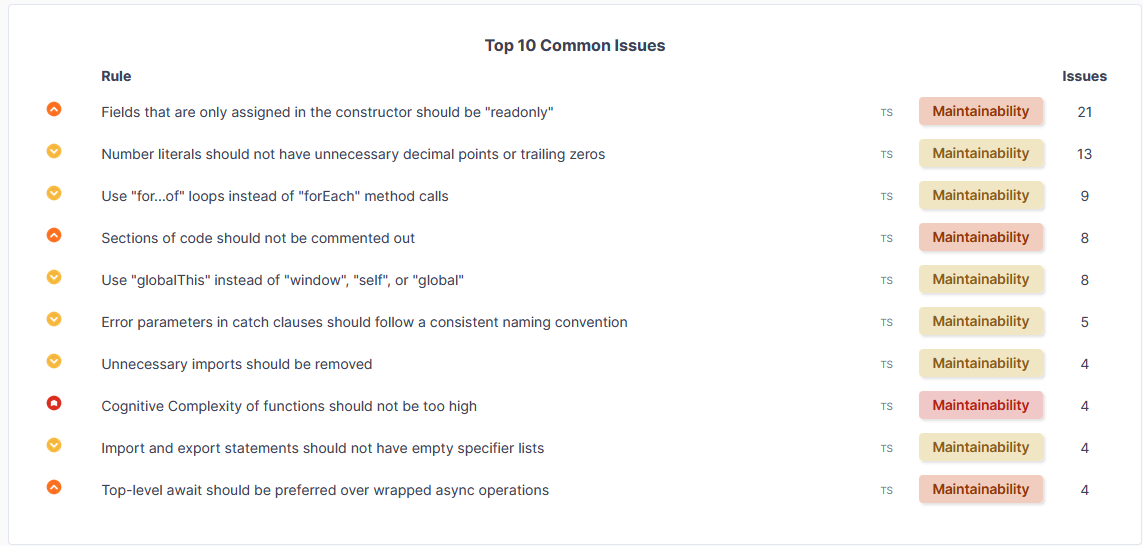
Data da análise: 20/11/2025

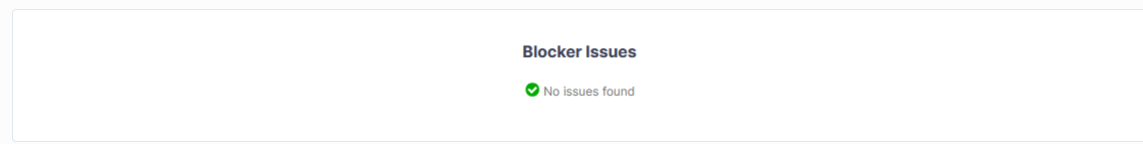
2. Evidência da Análise

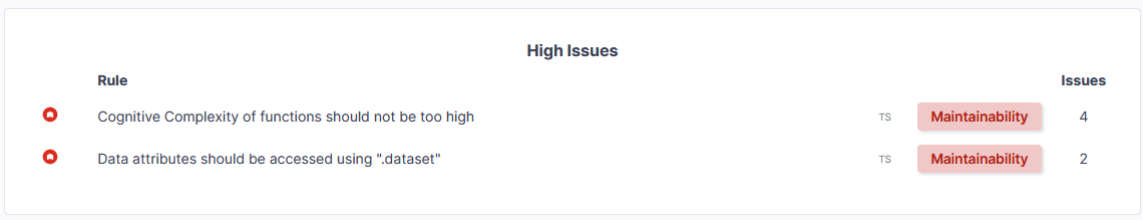
Anexos:











3. Resultados Obtidos

Métrica	Valor Obtido	Status/Nível	Observações

Cobertura de testes (%)	0%	Muito Baixa	1.419 linhas não cobertas; nenhum teste contribuindo para cobertura
Maintainability (manutenibilidade)	A(103 issues)	Bom	103 code smells; foco em readonly, complexidade e imports
Reliability (confiabilidade)	C (9 issues)	Médio	Problemas com promises, headings HTML e métodos globais
Security (segurança)	A (0 issues)	Ótimo	Nenhuma vulnerabilidade; existem 3 security hotspots
Duplicação de código (%)	2.4%	Aceitável	228 linhas duplicadas; 10 blocos; 4 arquivos
Code Smells detectados	103	Alto	Principal fonte de issues; maioria ligada a maintainability

4. Principais Problemas Detectados

4.1 Problemas de Manutenibilidade

- **Uso inadequado de imports**

  Exemplos:

  Preferir node: url ao invés de url

  Preferir node: path ao invés de path

- **Campos não reassinalados deveriam ser readonly**

  Exemplos:

  userService e authService no UserController.ts

  cavaloService no CavaloController.ts

- **Funções com complexidade cognitiva excessiva**

  Exemplos:

  src/controllers/CavaloController.ts: função updateCavalo() com complexidade 16, acima do permitido (15);

  src/controllers/UserController.ts: função com complexidade de 18, também acima do limite.

- **Uso de forEach que deveria ser substituído por for...of**

  Exemplo:

  CavaloController.ts

4.2 Problemas de Confiabilidade (Reliability)

Total: 9 issues, com as seguintes causas:

3 casos de heading elements should have accessible content

2 casos de Promises should not be misused

2 casos de Number static methods should be preferred over global equivalents

1 caso de Label elements missing associated control

1 caso de replaceAll ao invés de replace com regex global

4.3 Problemas de Segurança (Security Hotspots)

1. **Denial of Service (DoS) via Regex**

Local:

src/resources/scripts/ts/cadastroUsuario.ts

Regex vulnerável a super-linear runtime:

```
const emailRegex = /^[^\5@]+@ [^\s@]+\.[^\5@]+$/;
```

2. **Insecure Configuration (CORS)**

O SonarQube indica risco de habilitar CORS sem validação adequada

3. **Exposição de versão de framework**

O relatório indica que o framework pode estar expondo versão no servidor

5. Ações Recomendadas para Melhoria

- Reduzir complexidade das funções

Refatorar updateCavalo() e funções do UserController quebrando em funções auxiliares.

Aplicar early return para reduzir aninhamento.

Extrair validações repetidas para utilitários.

- Marcar propriedades como readonly

Aplicar em:

userService, authService,

cavaloService

Todos os campos mostrados como "never reassigned".

- Ajustar imports

Trocar url por node:url

Trocar path por node:path

- Remover código comentado

Remover seções de código inutilizadas apontadas pelo SonarQube.

- Substituir forEach por for...of

Aumenta performance e reduz criação de closures desnecessárias

6. Conclusão

A análise realizada pelo SonarQube indica que o EquiTrade apresenta boa qualidade geral, especialmente em segurança e manutenibilidade, ambas classificadas com rating A.

Entretanto, há alguns pontos importantes a destacar:

- 103 issues de manutenibilidade, principalmente campos que deveriam ser readonly, funções com complexidade elevada e uso inadequado de imports;
- Duplicação de 2,4%, indicando oportunidade de refatorar trechos repetidos;
- 3 Security Hotspots, incluindo um risco real de DoS via regex.

A análise permitiu identificar trechos complexos, práticas de código inconsistentes e pontos de vulnerabilidade. Como aprendizado, o relatório reforça a importância de manter padrões consistentes, reduzir complexidade cognitiva e estruturar a base de código pensando em escalabilidade e manutenção contínua.