



Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Verificação, Validação e Teste de Software (CK0241)

TB03 - Relatório de Análise Estática

E-Commerce

Grimberg Cryzan - 362964
Beatriz Monte - 371781
José João Silva - 408847
Levir César Ribeiro - 400555

Histórico de versões

Versão	Data	Autor	Descrição
1.0	31/01/2021	José João	Criação do documento com base no template
2.0	31/01/2021	José João	Criação da introdução, aplicação e código fonte, descrição da ferramenta. referências
3.0	31/01/2021	José João	Criação da lista de problemas analisados: I01 ao I10
4.0	31/01/2021	José João	Criação dos resultados gerais, discussão.
5.0	03/02/2021	José João	Acréscimo da ferramenta codacy no tópico de descrição da ferramenta. Edição dos resultados gerais;
6.0	03/02/2021	José João	Criação da lista de problemas analisados: I11 ao I15
7.0	03/02/2021	José João	Edição do tópico de discussão
8.0	03/02/2021	Grimberg Cryzan	Edição do tópico de conclusão
9.0	03/02/2021	Grimberg Cryzan	Edição do índice

Sumário

Introdução	4
Aplicação e código fonte	4
Descrição das ferramentas de Análise Estática	4
Resultados gerais	6
Lista de problemas analisados	6
Discussão	17
Conclusão	19
Referências	20
Glossário	20

1. Introdução

No desenvolvimento de software é comum que encontremos falhas por diversos motivos. Pequenas falhas podem gerar pequenos problemas na execução de um sistema assim como grandes falhas podem gerar bugs ou vulnerabilidades no sistema. Para diminuir essas falhas, podemos aplicar diversas estratégias de teste. Uma dessas estratégias é a análise estática de software.

A análise estática de software trabalha diretamente com o código de um projeto. Por meio de uma ferramenta, o analisador estático, os componentes de software de um projeto são verificados sem que o software seja executado. O objetivo principal é encontrar erros de programação, desde más práticas até falhas de segurança.

O relatório aqui apresentado possui como objetivo apresentar os resultados gerais da análise estática realizada no projeto ecommerce e apresentando possíveis soluções para alguns dos problemas encontrados.

1.1. Aplicação e código fonte

O E-commerce é uma aplicação open source, desenvolvida para fins educacionais, mas podendo ser estendida para uso comercial. O objetivo é que ele consiga fornecer, através de uma interface amigável, maneiras de gerenciar todos dados básicos presentes em um e-commerce, tais como funcionários, clientes, produtos, categoria de produtos, vendas e dados pessoais. Todos os códigos fonte do projeto estão disponíveis em: <https://github.com/jjoaosilva/ecommerce>.

É possível observar que a aplicação trabalha com a arquitetura MVC(model-view-controller) em que temos:

- View: desenvolvida em JavaScript com a biblioteca React JS;
- Model e Controller: desenvolvida em Python com o micro framework Flask e configurado para o SGBD PostgreSQL.

1.2. Descrição das ferramentas de Análise Estática

Foram utilizadas duas ferramentas de análise estática: SonarQube e Codacy. A ferramenta **SonarQube**, que é uma ferramenta automática de revisão de código capaz de detectar bugs, vulnerabilidades e code smells (chamados de maus cheiros de código como más práticas, por exemplo) e que suporta 26 linguagens diferentes.

O SonarQube se utiliza de regras e indicações de severidade dos erros para indicar o panorama de cada problema encontrado. As seguintes são: **Code Smell**, **Bug**, **Vulnerability** e **Security Hotspots**. para **Code smells** e **Bugs** não são esperados falsos positivos com o objetivo do desenvolvedor não se perguntar se aquela alteração é realmente necessária. Para **Vulnerabilities** são esperados 80% deles sejam verdadeiros positivos e em **Security Hotspots** espera-se que 80% dos indicados sejam marcados como revisados.

Cada problema possui um tipo, **Code Smell**, **Bug** ou **Vulnerability**, e uma severidade, **Bloquer**, **Critical**, **Major**, **Minor** ou **Info**. O tipo e a severidade ajudam o

desenvolvedor a avaliar qual o tipo do problema e qual problema deve ter mais prioridade. Ainda é possível trabalhar em cima do ciclo de vida de um problema com: **Open, Confirmed, Resolved, Reopened e Closed**.

Vale a pena salientar que todos esses recursos são gerenciados de maneira visual. O SonarQube executa um servidor local e disponibiliza uma interface web alocada na porta 9000 na máquina do usuário. Dessa maneira, o uso da ferramenta se torna mais fácil e agradável.

A ferramenta **Codacy** possui o mesmo objetivo porém suporta mais 30 linguagens de programação diferentes. Possui versões pagas possuindo integrações com outras ferramentas como Github, Slack, Jira, Gitlab e Gitbucket porém possui versão de graça com integração com Github e Gitbucket , por exemplo.

O Codacy se utiliza de seis categorias de problemas: **Security, Error Prone, Code Style, Compatibility, Unused Code e Performance**. Por categorias, temos os objetivos:

- **Segurança:** problemas de segurança, vulnerabilidades potenciais, dependências inseguras.
- **Propenso a erros:** práticas / padrões inadequados que fazem com que o código falhe / propenso a bugs.
- **Estilo do código:** relacionado ao estilo do código, comprimento da linha, tabulações e espaços.
- **Compatibilidade:** identifica o código que tem problemas com sistemas mais antigos ou suporte de plataforma cruzada.
- **Código não utilizado:** código desnecessário não em uso.
- **Desempenho:** código escrito de forma ineficiente.

Cada um desses problemas possuem “levels” que indicam a severidade do problema em questão. As três são: **Info**, visualmente indicado em azul, **Warning**, visualmente indicado por amarelo e **Error**, visualmente indicado por vermelho. Ainda é possível trabalhar em cima do ciclo de vida de um problema com: **Ignore Issue, Ignore Pattern e Ignore file**.

Vale a pena salientar que todos esses recursos são disponíveis por uma plataforma web, possuindo interface web amigável e documentação atualizada

2. Resultados gerais

As duas ferramentas foram utilizadas em dois escopos do projeto: o back-end (Python) e o front-end(Javascript). A ideia é conseguir analisar de maneira individual cada um desses escopos e seus diferentes tipos de problemas. É esperado uma grande quantidade de problemas relacionados a Code Smells e Security já que o projeto foi desenvolvido em um ambiente de universidade para trabalho final de uma disciplina.

2.1. Lista de problemas analisados

As tabelas abaixo indicarão os problemas encontrados com as seguintes características: identificador(aqui será indicado pela I de *Issue*), linguagem de programação, componente(back-end ou front-end) ferramentas envolvidas, categoria, localização, mensagem, trecho de código, proposta de solução e comentários.

Identificador	I01
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	SonarQube
Categoria	Code Smell / Critical
Localização	app/endpoints/categoria/view.py - L1
Mensagem	Import only needed names or import the module and then use its members.
Trecho do código	<pre>from app.models.categoria.utils.CategoriaDAO import *</pre>
Proposta de solução	<pre>from app.models.categoria.utils.CategoriaDAO import CategoriaDAO</pre>
Comentários	O arquivo view.py referente aos endpoints da classe Categoria, estava importando todo o módulo <i>CategoriaDAO</i> ao invés de apenas importar a classe <i>CategoriaDAO</i> que é de fato o que realmente precisa ser importado.

Identificador	I02
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	SonarQube
Categoria	Code Smell / Minor
Localização	app/endpoints/funcionario/view.py - L10
Mensagem	Rename this local variable "fDAO" to match the regular expression <code>^[_a-z][a-z0-9_]*\$</code> .
Trecho do código	<pre>fDAO = FuncionarioDAO()</pre>
Proposta de solução	<pre>f_dao = FuncionarioDAO()</pre>
Comentários	A regra utilizada pelo SonarQube em relação a nomes de variáveis para Python não inclui letras maiúsculas.

Identificador	I03
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	SonarQube
Categoria	Code Smell / Critical
Localização	app/models/categoria/utils/CategoriaDAO.py - L8
Mensagem	Add a nested comment explaining why this method is empty, or complete the implementation.
Trecho do código	<pre>class CategoriaDAO: def __init__(self): pass</pre>

Proposta de solução	<pre>class CategoriaDAO: def __init__(self): pass # this class has no attributes because it only handles CRUD from another class</pre>
Comentários	É importante informar que a omissão de inicializador em uma classe é feita de forma proposital para que não haja conflitos entre desenvolvedores.

Identificador	I04
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	SonarQube
Categoria	Code Smell / Critical / Duplication
Localização	app/endpoints/produto/view.py - L56
Mensagem	Define a constant instead of duplicating this literal "Problema na procura de Produtos!!!" 3 times.
Trecho do código	<pre>return jsonify({ "status": False, "mensagem": "Problema na procura de Produtos!!!", "payload": error.args</pre>
Proposta de solução	<pre>error_in_creating_product = "Problema na criacao do Produto!!!" return jsonify({ "status": False, "mensagem": erro_in_creating_product, "payload": error.args</pre>
Comentários	O módulo possuía mais de um local com a mesma frase de retorno em casos de erro. Portanto foi criada uma constante com o texto e a constante utilizada em todos os casos, dessa forma removendo as duplicações.

Identificador	I05
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	SonarQube
Categoria	Code Smell / Minor
Localização	app/models/categoria/utils/CategoriaDAO.py - L34
Mensagem	Remove the unused local variable "c"
Trecho do código	<pre>c = db.session.delete(categoria)</pre>
Proposta de solução	<pre>_ = db.session.delete(categoria)</pre>
Comentários	No módulo indicado há a presença de variáveis para retorno de funções específicas, porém elas não são utilizadas. Neste caso podemos suprimir o retorno com o uso de _.

Identificador	I06
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	SonarQube
Categoria	Security Hotspots / Cross-Site Request Forgery (CSRF) / High priority
Localização	app/__init__.py
Mensagem	Make sure disabling CSRF protection is safe here.

Trecho do código	<pre> from flask_script import Manager from flask_migrate import Migrate, MigrateCommand from flask_cors import CORS import os app = Flask(__name__) app.config.from_object('config') </pre>
Proposta de solução	<pre> app = Flask(__name__) csrf = CSRFProtect() csrf.init_app(app) # Compliant </pre>
Comentários	<p>As vulnerabilidades CSRF ocorrem quando os invasores podem enganar um usuário para executar operações autenticadas confidenciais em um aplicativo da web sem seu consentimento. Como está sendo usado o micro framework Flask é recomendado o uso do módulo CSRFProtect.</p>

Identificador	I07
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	SonarQube
Categoria	Security Hotspots / Insecure Configuration / Low priority
Localização	app/__init__.py
Mensagem	Make sure this permissive CORS policy is safe here.
Trecho do código	<pre> import os app = Flask(__name__) app.config.from_object('config') CORS(app) </pre>
Proposta de solução	<pre> app = Flask(__name__) CORS(app, resources={r"/*": {"origins": "*", "send_wildcard": "False"}}) # Compliant </pre>

Comentários	A política de mesma origem em navegadores impede, por padrão e por razões de segurança, que um frontend javascript execute uma solicitação HTTP de origem cruzada para um recurso que possui uma origem diferente (domínio, protocolo ou porta) da sua. O destino solicitado pode anexar cabeçalhos HTTP adicionais em resposta, chamados CORS, que agem como diretivas para o navegador e alteram a política de controle de acesso / relaxam a política de mesma origem. Como alternativa de solução, O cabeçalho Access-Control-Allow-Origin deve ser definido apenas para uma origem confiável e para recursos específicos.
--------------------	--

Identificador	I08
Linguagem de programação	JavaScript
Componente	Front-end
Ferramentas envolvidas	SonarQube
Categoria	Code Smell / Minor
Localização	src/App.js - 2
Mensagem	Remove this unused import of 'Redirect'.
Trecho do código	<pre>import React, { Component } from 'react'; import { HashRouter, Route, Switch, Redirect } from 'react-router-dom';</pre>
Proposta de solução	<pre>import React, { Component } from 'react'; import { HashRouter, Route, Switch } from 'react-router-dom';</pre>
Comentários	Não há razões para importação de módulos que não são usados. Em aplicações web é necessário ter esse cuidado ja que isso vai influenciar no load da página de um usuário.

Identificador	I09
Linguagem de programação	JavaScript
Componente	Front-end
Ferramentas envolvidas	SonarQube
Categoria	Code Smell / Major
Localização	src/containers/app/componentes/dataTable.js - L65
Mensagem	'index' is already declared in the upper scope.
Trecho do código	<pre> {this.state.data.map((item, index) => <tr key={index}> {this.props.columnData.map((column, index) => </pre>
Proposta de solução	<pre> {this.state.data.map((item, location) => <tr key={location}> {this.props.columnData.map((column, index) => </pre>
Comentários	<p>Neste caso, temos um map aninhado em outro map e ambos usam o mesmo nome de variável "index". Por uma questão de escopo, o código funciona, porém esse formato dificulta a manutenção do código.</p>

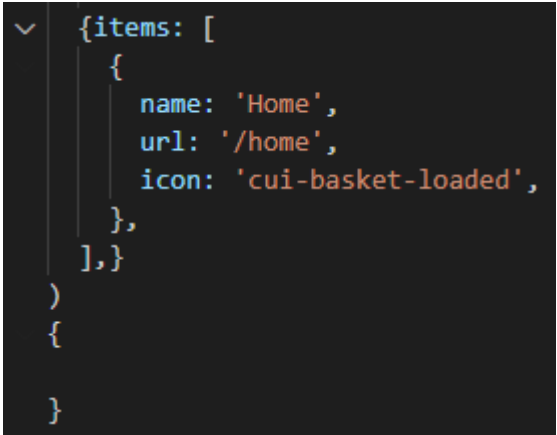
Identificador	I10
Linguagem de programação	JavaScript
Componente	Front-end
Ferramentas envolvidas	SonarQube
Categoria	Code Smell / Major
Localização	src/store/ducks/carrinho.js - L14
Mensagem	Default parameters should be last.

Trecho do código	<pre>const add = (state = INITIAL_STATE, action) =>{</pre>
Proposta de solução	<pre>const add = (action, state = INITIAL_STATE) =>{</pre>
Comentários	Neste caso, deve-se fazer o uso de parâmetros com valores default no final de assinaturas para facilitar o seu uso.

Identificador	I11
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	Codacy
Categoria	Code Style / Level: Error
Localização	services/ecommerce/app/endpoints/cliente/view.py - L28
Mensagem	function already defined line 7
Trecho do código	<pre>@app.endpoint('/procurar-clientes') def index(): @app.endpoint('/deletar-cliente') def index():</pre>
Proposta de solução	<pre>@app.endpoint('/procurar-clientes') def search_cliente(): @app.endpoint('/deletar-cliente') def delete_cliente():</pre>
Comentários	Neste módulo existem duas funções com o mesmo nome “index”. O código funciona bem pois o framework acessa as funções com base no endpoint registrado. A melhor solução é renomear as funções para nomes mais significativos e únicos.

Identificador	I12
Linguagem de programação	Python
Componente	Back-end
Ferramentas envolvidas	Codacy
Categoria	Security / Level: Warning
Localização	services/ecommerce/app/models/Item/Utils/ItemDAO.py - L26
Mensagem	Possible SQL injection vector through string-based query construction.
Trecho do código	<pre>def read(self, venda_id): sql = """SELECT itens.quantidade, itens.valor_unidade, produtos.nome FROM itens INNER JOIN produtos on itens.produto_id=produtos.id WHERE venda_id = {venda_id} """.format(venda_id=venda_id)</pre>
Proposta de solução	<pre>def read(self, venda_id): if type(venda_id) is int: sql = """SELECT itens.quantidade, itens.valor_unidade, produtos.nome FROM itens INNER JOIN produtos on itens.produto_id=produtos.id WHERE venda_id = {venda_id} """.format(venda_id=venda_id)</pre>
Comentários	Neste caso, como não há nenhuma tratativa da variável venda_id, é possível que haja injeção de query indevidas. Para contornar isso, podemos verificar o valor de venda_id antes de realizar a consulta.

Identificador	I13
Linguagem de programação	JavaScript
Componente	Front-end
Ferramentas envolvidas	Codacy
Categoria	Error Prone / Level: Error

Localização	front/src/_nav.js - L62
Mensagem	Empty block statement.
Trecho do código	 <pre> {items: [{ name: 'Home', url: '/home', icon: 'cui-basket-loaded', },],}) { } </pre>
Proposta de solução	Neste caso as soluções são: fazer um comentário dentro do bloco vazio ou remover um bloco vazio.
Comentários	Bloco vazios podem indicar erros de refatoração e confusão na leitura do código.

Identificador	I14
Linguagem de programação	JavaScript
Componente	Front-end
Ferramentas envolvidas	Codacy
Categoria	Error Prone / Level: Error
Localização	front/src/containers/app/componentes/productCard.js - L60
Mensagem	Unexpected console statement.

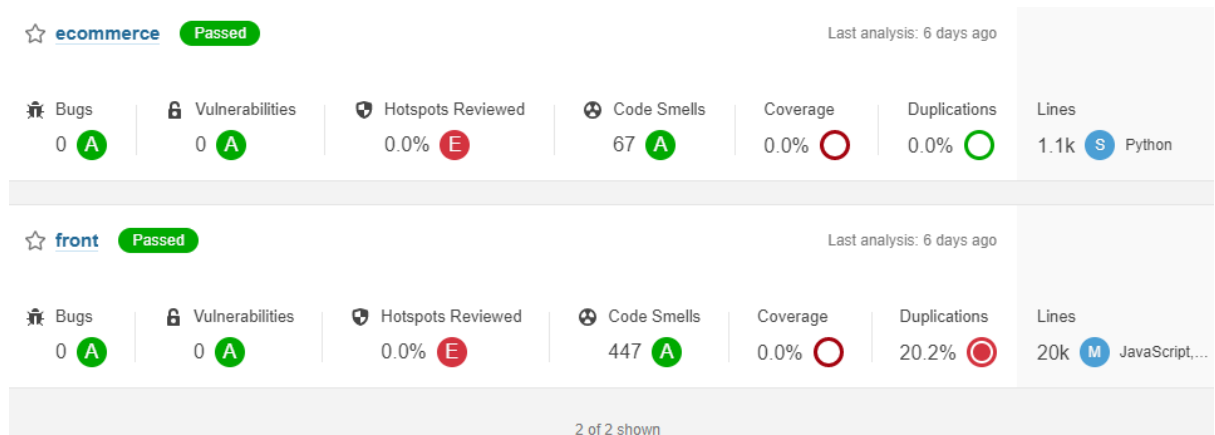
Trecho do código	<pre>componentWillMount(){ console.log("Product card: ", this.props) }</pre>
Proposta de solução	Neste caso todo o bloco do ciclo de vida “componentWillMount” pode ser removido sem nenhum impacto.
Comentários	Ao ser executado no browser do usuário, o “console.log” irá lançar “prints” no console do usuário. Além disso, dependendo do que há nas mensagens pode assim vaziar informações úteis.

Identificador	I15
Linguagem de programação	JavaScript
Componente	Front-end
Ferramentas envolvidas	Codacy
Categoria	Security / Level: Info
Localização	front/src/serviceWorker.js - L17
Mensagem	Generic Object Injection Sink
Trecho do código	<pre>const add = (action, state = INITIAL_STATE) =>{ if(state.length !== 0){ for(var i = 0; i < state.length; i++){ if(state[i].product_id === action.product.product_id){</pre>
Proposta de solução	<pre>const add = (action, state = INITIAL_STATE) =>{ if(state.length !== 0){ for(var i = 0; i < state.length; i++){ if(state[parseInt(i)].product_id === action.product.product_id){</pre>
Comentários	Em casos de acessos a array, em JS, é indicado o uso do parses para o acesso por index. Pois é possível a injeção de funções ou prototypes por hackers.

3. Discussão

Como já dito, as ferramentas foram utilizadas para a análise em dois escopos diferentes. Chamaremos o escopo de back-end de ecommerce e chamaremos o escopo de front-end de front.

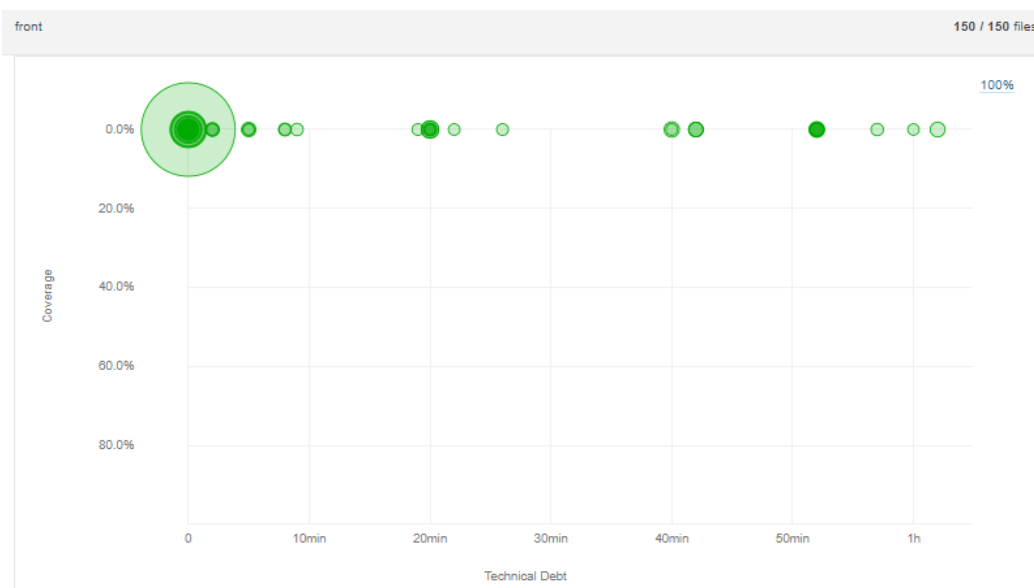
Vamos analisar os resultados gerados pelo SonarQube:

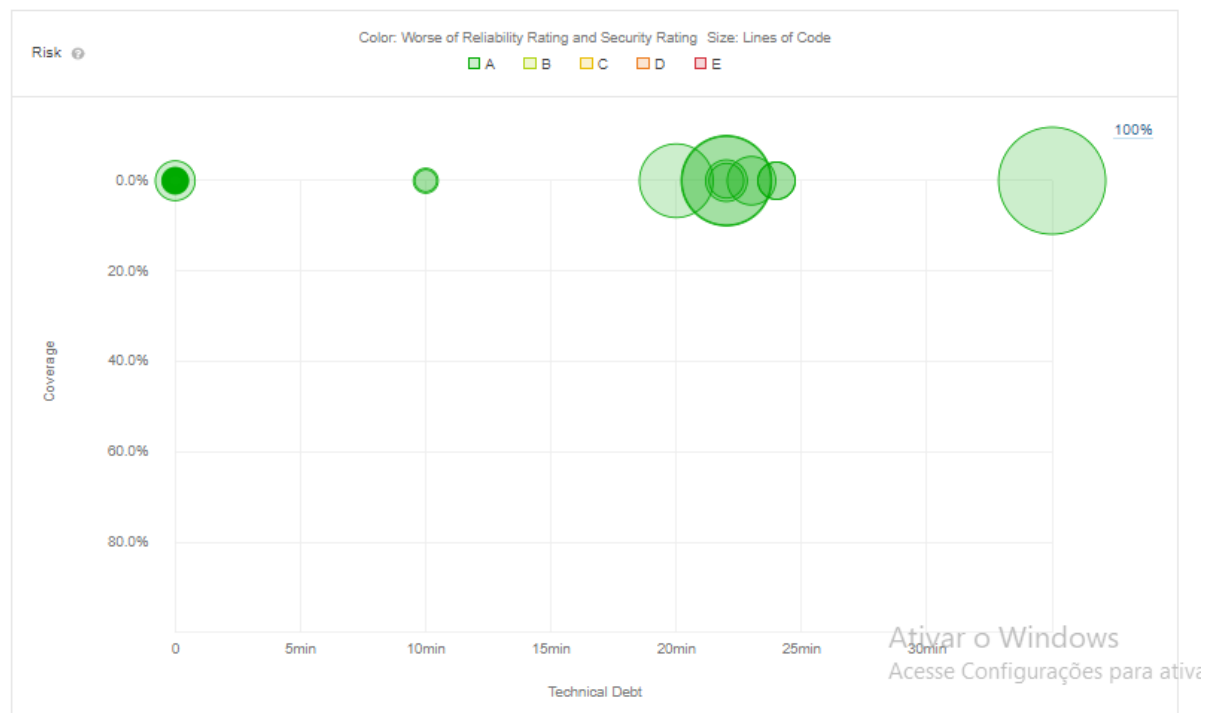


A imagem acima descreve os resultados gerais dos dois escopos. Podemos observar que em métricas gerais os dois possuem status Passed, porém podemos observar uma nota baixa em Hotspots, coverage (em relação a testes unitários) e uma baixa porcentagem de Duplications.

Indo mais afundo, podemos observar que o back-end possui 67 issues de Code Smells e 3 em relação a Security Hotspots. Já o front-end possui 447 issues de Code Smells, 2 em relação a Security Hotspots e 20.3% de Duplications. Vale salientar que essa grande diferença se dá também pelo número de linhas do back-end e do front-end: o back-end possui apenas 5.5% do número de linhas em comparação com o front-end. Porém, apesar de serem projetos relativamente grandes, ambos possuem classificação A em relação a manutenção.

O SonarQube também nos fornece gráficos que mostram uma visão mais ampla do projeto, relacionando as métricas:

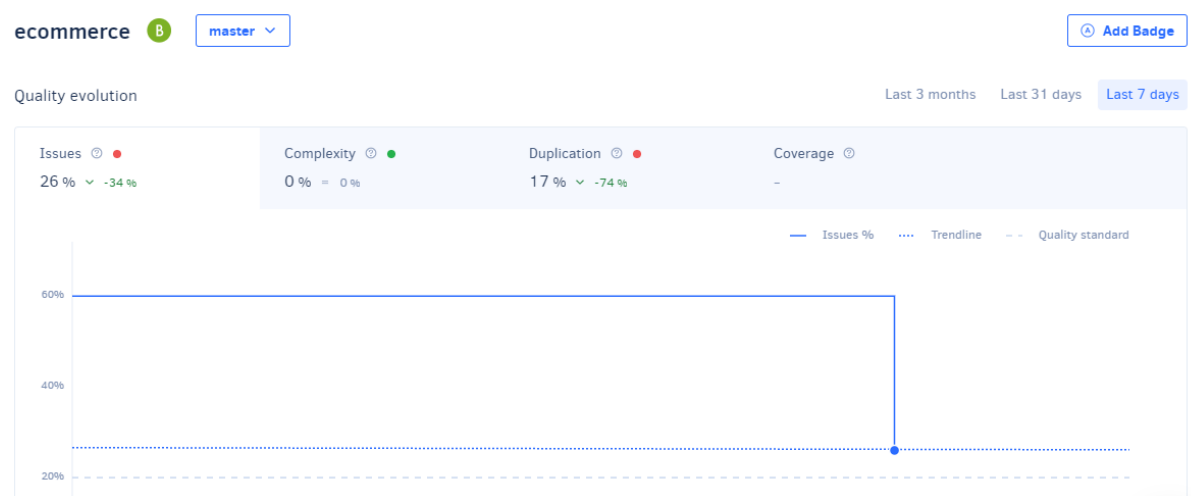




Os dois gráficos mostram a classificação A em relação às métricas gerais, na cor verde, e a relação entre coverage e a dívida técnica, que possivelmente diz respeito ao tempo necessário para realização dos testes unitários naquele módulo em si.

Outro ponto que vale a pena ressaltar é todo o suporte que a ferramenta SonarQube fornece para a resolução das issues. Em cada uma é possível verificar a prioridade, o motivo daquela issue existir e como você pode realizar o conserto da issue. Outro ponto é o fornecimento de um ciclo de vida bem robusto, podendo configurar se foi já concluído ou até mesmo se está sendo refeito, além de suportar usuários em um projeto a partir do histórico Git.

Vamos analisar os resultados gerados pelo Codacy:



A imagem acima mostra a classificação geral das duas aplicações, que no caso é B. Possuindo, somando o back-end e o front-end, 26% dos projetos em issues e 17% em duplications. Dessas issues temos: 1630 issues para o front-end, onde 1602 dizem respeito a code style, 15 a error prone e 13 a security, e 230 para o back-end, onde 224 dizem respeito a code style e 6 a security.

O uso das duas ferramentas se deu devido às poucas variações de categorias encontradas nas análises feitas pelas duas ferramentas. Ambas as ferramentas possuem interface web, o que facilita muito o uso das funcionalidades e leitura de todos os resultados e métricas fornecidas. O Codacy possui uma interface mais bonita do que o SonarQube, porém no Sonarqube é mais fácil encontrar os dados e realizar novas inspeções do mesmo código.

O Codacy faz uma nova análise a partir de qualquer commit na branch escolhida para ser analisada, o que facilita muito. Porém, como foi usada a partir da leitura do repositório no github e nesse repositório temos o código do back-end e do front-end, foi mais difícil a realização da avaliação desses componentes de maneira separada, assim como foi feito no SonarQube.

4. Conclusão

Assim como foi exposto acima, o documento trata da análise estática do software E-commerce, feita a partir de duas ferramentas, o Sonarqube, em que se pode observar mais detalhadamente os erros provenientes das duas partes do código, o back e o front, já que a ferramenta faz a distinção entre essas duas partes. Já diferentemente da ferramenta anterior, o Codacy, não faz essa distinção, porém, faz distinção entre as diferentes linguagens usadas no projeto, também se pode escolher qual pasta será analisada ou não e é mais conveniente pois faz essa inspeção de código assim que se faz o commit para o github, além de ter uma interface mais amigável.

E também como foi citado, o uso das duas ferramentas se fez necessário pela falta de variações de categorias encontradas, exigindo que buscássemos mais categorias que uma única ferramenta poderia oferecer, para podermos ter uma visão mais completa dos problemas do código.

No documento também foram expostos trechos de códigos com os diversos problemas encontrados em ambas as ferramentas, assim como ideias de soluções de como resolvê-los. Tais problemas sendo predominantemente do tipo estilo, e no geral, o código teve boa performance em ambas avaliações, sendo outro motivo da procura de duas ferramentas ao invés de apenas uma. A análise detalhada dos erros apresentados se encontra no tópico de discussão.

Com isso, após a análise dos resultados concluímos que felizmente o código possui poucos erros e vulnerabilidades, mesmo sendo usado ferramentas com parâmetros diferentes, sendo a maior parte dos erros apontados de estilo, podendo ser facilmente corrigidos.

5. Referências

Teste estático e dinâmico: entenda as características. Disponível em:
<<https://blog.convisoappsec.com/teste-estatico-e-dinamico-entenda-diferencas/>>.
Acesso em: 31, janeiro de 2021.

SonarQube documentation. Disponível em: <<https://docs.sonarqube.org/latest/>>.
Acesso em: 31 janeiro de 2021.

Codacy documentation. Disponível em:<<https://docs.codacy.com>>. Acesso em: 03 de fevereiro de 2021.

Glossário

Termo	Definição