




Ministério da Justiça

Projeto: SIMAP

Nota Técnica

MJ	SIMAP – Nota Técnica	
-----------	-----------------------------	--

Revisão	Descrição	Autor	Data
1.0	Construção do documento	Israel Branco	31/01/2020

1 Sumário

2	Introdução.....	4
3	Apresentação do cenário atual.....	5
3.1	Front-End.....	6
3.2	Simap-Integration.....	7
3.3	Simap-Util.....	7
3.4	Simap-Entity.....	7
3.4	Simap-Presentation.....	7
3.2	Tecnologias utilizadas.....	8
3.3	Modelagem de dados.....	9
4	Análise técnica.....	10
4.1	SonarQube.....	10
4.2	OWASP Dependency Check.....	12
4.3	NPM Audit.....	13
4.4	Análise sobre os resultados.....	15
4.4.1	Manutenibilidade de código.....	15
4.4.2	Confiabilidade.....	16
4.4.3	Performance e estabilidade.....	16
4.4.3	Escalabilidade.....	16
5	Recomendações.....	17
6	Conclusão.....	18

2 Introdução

Este documento visa reportar o resultado da análise efetuada na aplicação SIMAP. Para este estudo foram desconsiderados todo o contexto negocial ao qual a ferramenta está inserida, também foram desconsideradas o ambiente ao qual a ferramenta esta operando sendo analisado puramente questões que tangem a qualidade de código, padrões de codificação, vulnerabilidades de dependências e concepção arquitetural.



3 Apresentação do cenário atual

Esta sessão ira descrever a arquitetura, tecnologias, frameworks e dependências que compõe a base da aplicação.

O SIMAP está construído para funcionar em ambiente WEB com uma segregação entre as camadas de front-end e back-end, sua arquitetura esta projetada para trabalhar de forma desacoplada e distribuída.

O backend da aplicação está construído sobre a stack Java Enterprise Edition, já a aplicação front-end está construída para trabalhar como uma SPA - Single Page Application com o framework Angular.

O diagrama a seguir representa o modelo de componentes ao qual a aplicação está construída, suas dependências e seu modelo de comunicação.

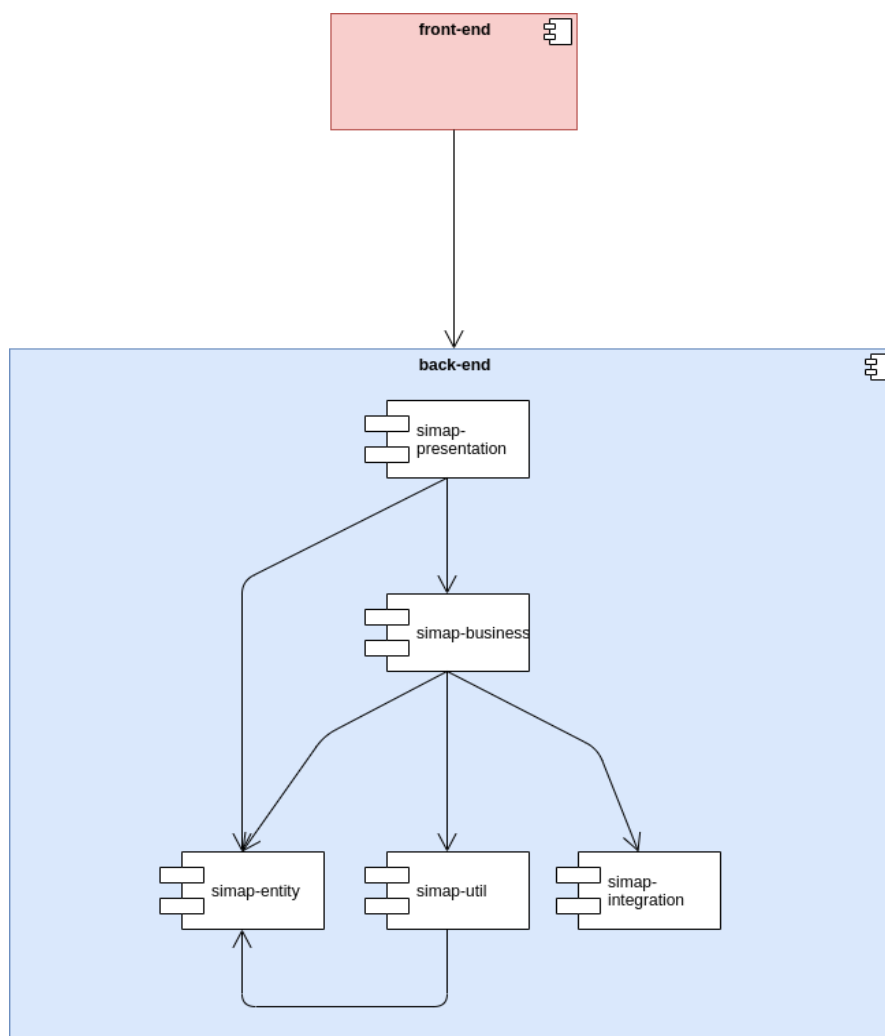


Figura 1: Diagrama de componentes

3.1 Front-End

Esta camada representa a interface com o usuário da aplicação e está organizada de forma lógica por segregação de módulos/funcionalidades, proporciona boa capacidade produtiva para manutenções corretivas e evolutivas.

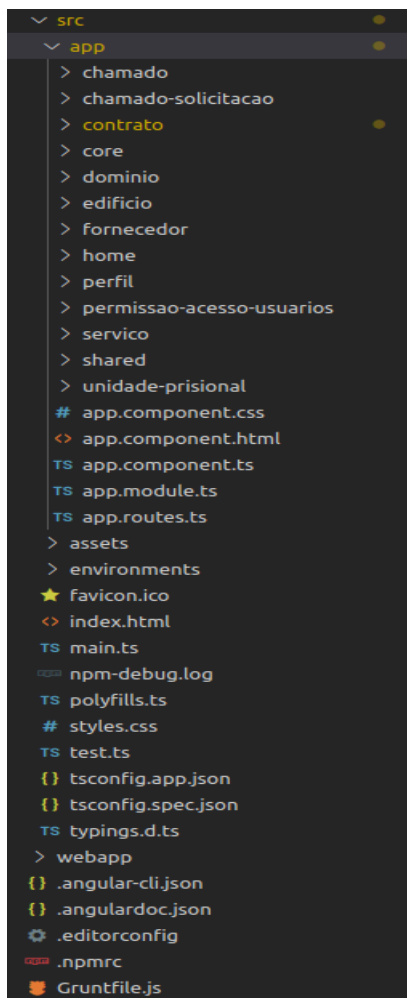


Figura 1: Organização do projeto front-end

3.2 Simap-Integration

Componente oferece classes utilitárias para iteração com SGBD (Sistema Gerenciador de Banco de Dados).

3.3 Simap-Util

Embora este venha a compor a estrutura back-end do projeto o mesmo não contempla implementação de código, existindo somente referências de dependência com a arquitetura de referência JEE7 do Ministério da Justiça.

3.4 Simap-Entity

Componente contempla entidades POJO (Plain Old Java Object) que possuem objetivo de trafegar estado entre as camadas da aplicação. Embora haja neste projeto a presença de classes utilitárias, sua concepção adota padrões do tipo VO (Value Object) e DTO (Data Transfer Object).

3.4 Simap-Presentation

Representa a camada da aplicação back-end que disponibiliza API Rest (Representation Transfer State) para consumo da aplicação front-end com a utilização do design pattern Facade.

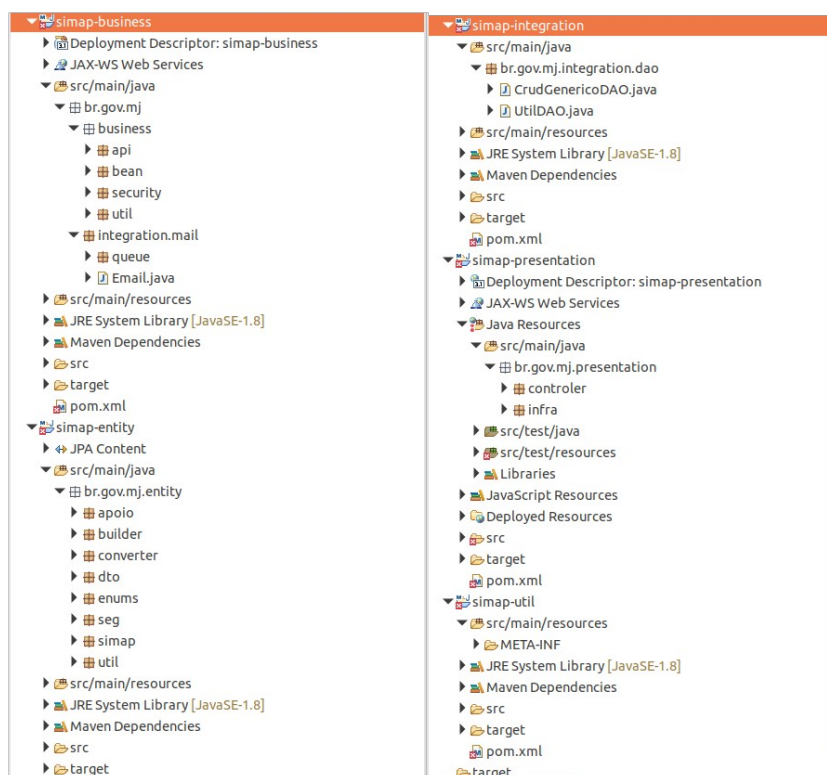


Figura 2: Organização do projeto back-end

3.2 Tecnologias utilizadas

Esta sessão descreve as tecnologias, frameworks e principais bibliotecas utilizadas na construção do projeto, descrevendo versões e propósitos de utilização.

Nome	Versão	Utilização	Observação
Java	1.8	Linguagem de programação.	
Javaee-api	7.0	Stack Java Enterprise Edition	
Hibernate	4.3.7	Framework ORM	
Wildfly	8.x	Servidor de aplicação JEE.	Utiliza containers EJB, CDI e Servlet.
Jackson-Annotations	2.8.5	Biblioteca para serializar/deserializar quando se trabalha com Pojos/Json	
Junit	4.12	Utilizado para a criação de testes automatizados.	
Jboss Arquillian	1.1.5	Framework para realização testes de integração.	Divergência de versões entre conare-web e parent
Apache POI	3.10	Componente utilizado para manipular documentos baseado em formato MS Office.	
PostgreSQL		Banco de dados relacional	



3.3 Modelagem de dados

A estrutura de banco de dados esta composta pela utilização de 3 schemas (apoio, seg e simap), estas estruturas não demonstram ausência de normatização em sua composição.

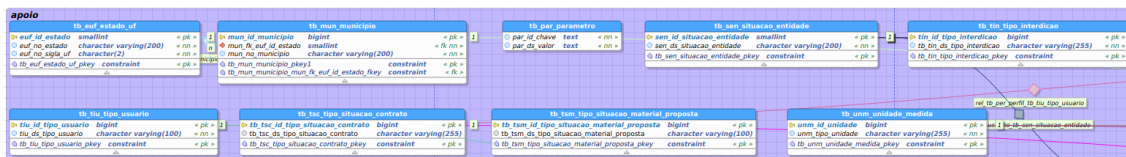


Figura 3: schema apoio

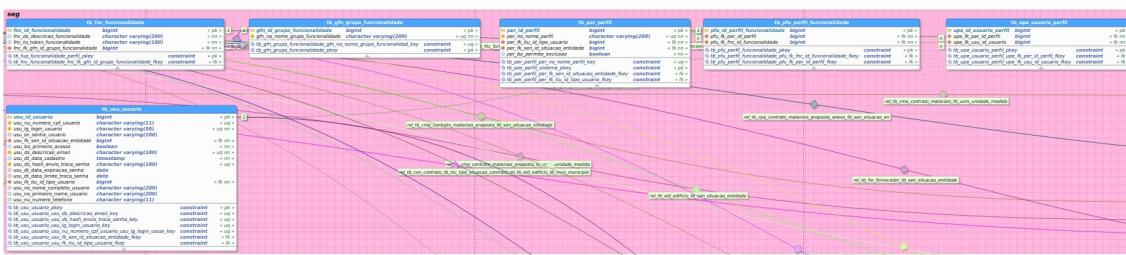


Figura 4: schema seq

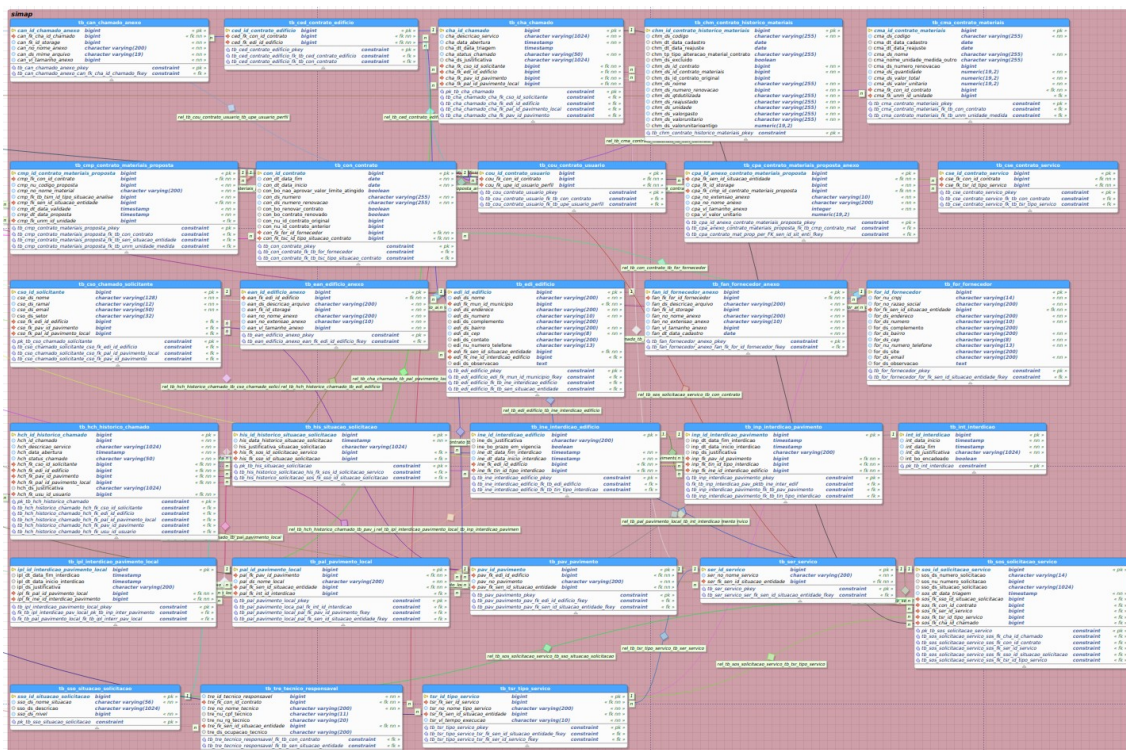


Figura 5: schema simap

4 Análise técnica

Este tópico descreve a ferramenta do ponto de vista técnico, tanto nos aspectos de codificação, análise estática de código, análise de vulnerabilidade de dependências e particularidades de implementação.

4.1 SonarQube

Ferramenta utilizada para verificação de estática de código. Para esta análise não foram utilizadas as métricas de qualidade implantadas no SonarQube do Ministério da Justiça, contudo foram utilizadas as regras padrões de análise da ferramenta. Os resultados foram os seguintes para a aplicação back-end:

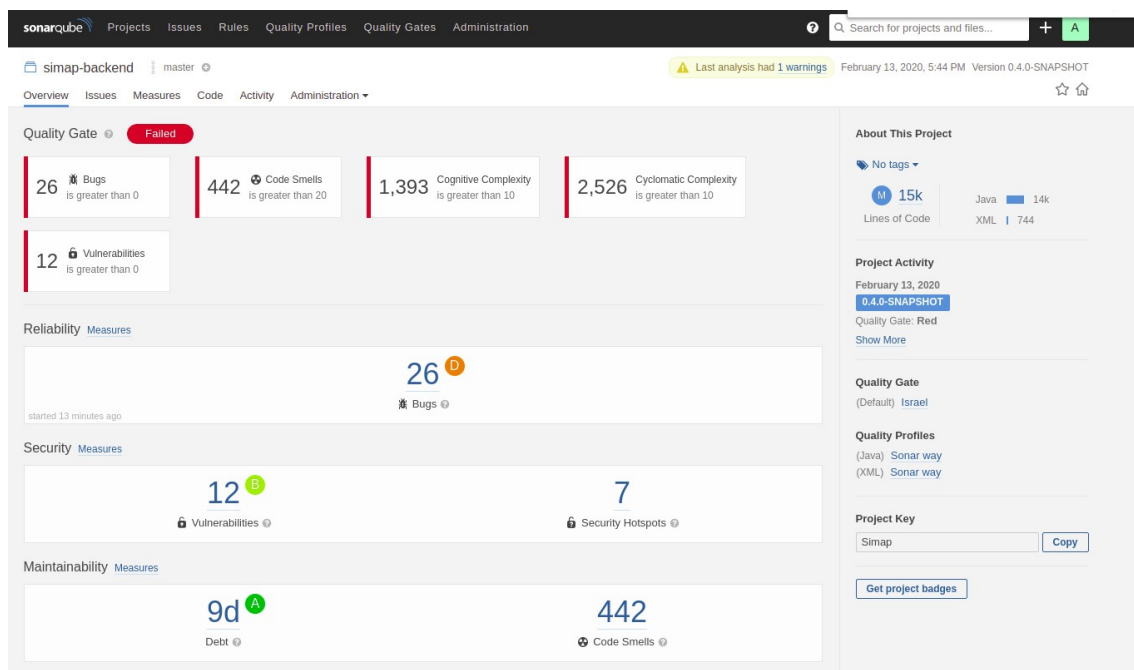


Figura 6: Análise estática de código

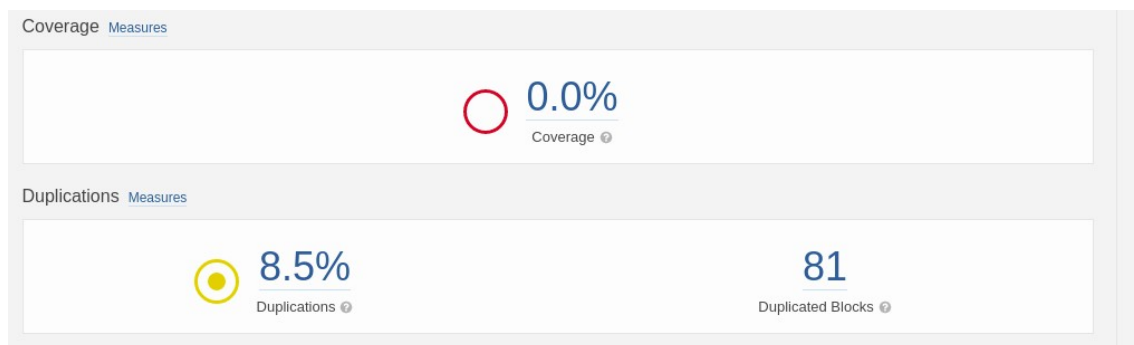


Figura 7: Cobertura de testes e duplicidade de código



Para a obtenção dos resultados, fora utilizado o código fonte com último histórico de commit (correção_issues_capgemini):

- 26 bugs;
- 442 violações de más práticas;
- 1393 violações de complexidade cognitiva (dificuldade de entendimento de código);
- 2526 violações de complexidade ciclomática (complexidade de código);
- 12 vulnerabilidades;
- 8.5% de duplicação de código;

A ferramenta apresenta 0% de cobertura de testes tendo em vista que no momento desta análise não foram reportados os resultados os mesmos. Embora tenhamos a presença de testes unitários no código , o mesmo é feito de forma inexpressiva para a métrica de cobertura de testes e sua execução apresentou erros nos testes de integração.

```

File Edit View Search Terminal Tabs Help
Israel@fwlnewov: ~/Documents/projetos/mj/simap/Build/codigos_fonte/backend/simap

Tests run: 1, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.304 sec <<< FAILURE! - in br.gov.mj.test.business.UFRestTest
br.gov.mj.test.business.UFRestTest Time elapsed: 0.304 sec <<< ERROR!
java.lang.RuntimeException: Could not invoke deployment method: public static org.jboss.shrinkwrap.api.Archive br.gov.mj.test.business.UFRestTest.createDeployment()
    at org.jboss.shrinkwrap.resolver.impl.maven.MavenWorkingSessionImpl.loadPomFromFile(MavenWorkingSessionImpl.java:191)
    at org.jboss.shrinkwrap.resolver.impl.maven.task.LoadPomTask.execute(LoadPomTask.java:84)
    at org.jboss.shrinkwrap.resolver.impl.maven.PonlessResolveStageBaseImpl.loadPomFromFile(PonlessResolveStageBaseImpl.java:93)
    at org.jboss.shrinkwrap.resolver.impl.maven.MavenResolverSystemBaseImpl.loadPomFromFile(MavenResolverSystemBaseImpl.java:178)
    at br.gov.mj.test.base.TestBase.criaMavenPadrao(TestBase.java:19)
    at br.gov.mj.test.business.UFRestTest.createDeployment(UFRestTest.java:43)

Running br.gov.mj.test.business.SituacaoEntidadeRestTest
Feb 14, 2020 4:00:52 PM org.jboss.shrinkwrap.resolver.impl.maven.logging.LogTransferListener transferFailed
WARNING: Failed downloading org.jboss.arquillian:arquillian-bom/1.1.5.Final/arquillian-bom-1.1.5.Final.pom from http://repo1.maven.org/maven2/. Reason:
org.eclipse.aether.transfer.ArtifactTransferException: Could not transfer artifact org.jboss.arquillian:arquillian-bom:pom:1.1.5.Final from/to central (http://repo1.maven.org/maven2): Failed to look for f
ile: http://repo1.maven.org/maven2/org.jboss.arquillian:arquillian-bom/1.1.5.Final/arquillian-bom-1.1.5.Final.pom. Return code is: 501
Tests run: 1, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.338 sec <<< FAILURE! - in br.gov.mj.test.business.SituacaoEntidadeRestTest
br.gov.mj.test.business.SituacaoEntidadeRestTest Time elapsed: 0.337 sec <<< ERROR!
java.lang.RuntimeException: Could not invoke deployment method: public static org.jboss.shrinkwrap.api.Archive br.gov.mj.test.business.SituacaoEntidadeRestTest.createDeployment()
    at org.jboss.shrinkwrap.resolver.impl.maven.MavenWorkingSessionImpl.loadPomFromFile(MavenWorkingSessionImpl.java:191)
    at org.jboss.shrinkwrap.resolver.impl.maven.task.LoadPomTask.execute(LoadPomTask.java:84)
    at org.jboss.shrinkwrap.resolver.impl.maven.PonlessResolveStageBaseImpl.loadPomFromFile(PonlessResolveStageBaseImpl.java:93)
    at org.jboss.shrinkwrap.resolver.impl.maven.MavenResolverSystemBaseImpl.loadPomFromFile(MavenResolverSystemBaseImpl.java:178)
    at br.gov.mj.test.base.TestBase.criaMavenPadrao(TestBase.java:19)
    at br.gov.mj.test.business.SituacaoEntidadeRestTest.createDeployment(SituacaoEntidadeRestTest.java:41)

Results :
Tests in error:
  MunicipioRestTest.br.gov.mj.test.business.MunicipioRestTest > Runtime Could no...
  UFRestTest.br.gov.mj.test.business.UFRestTest > Runtime Could not invoke depla...
  SituacaoEntidadeRestTest.br.gov.mj.test.business.SituacaoEntidadeRestTest > Runtime

Tests run: 3, Failures: 0, Errors: 3, Skipped: 0

[INFO] -----
[INFO] Reactor Summary for simap-backend 0.4.0-SNAPSHOT:
[INFO]
[INFO] simap-backend ..... SUCCESS [ 0.368 s]
[INFO] simap-entity ..... SUCCESS [ 10.159 s]
[INFO] simap-util ..... SUCCESS [ 0.140 s]
[INFO] simap-integration ..... SUCCESS [ 0.262 s]
[INFO] simap-business ..... SUCCESS [ 11.285 s]
[INFO] simap-presentation ..... FAILURE [ 3.893 s]
[INFO] simap-application ..... SKIPPED
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 26.445 s
[INFO] Finished at: 2020-02-14T16:00:52-03:00
[INFO] -----
[INFO] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.17:test (default-test) on project simap-presentation: There are test failures.
[ERROR] Please refer to /home/Israel/Documents/projetos/mj/simap/Build/codigos_fonte/backend/simap/presentation/target/surefire-reports for the individual test results.
[ERROR]

```

8: Erros na execução de testes unitários

4.2 OWASP Dependency Check

A utilização de bibliotecas de terceiros aumenta substancialmente a produtividade na construção de um software, contudo estas podem trazer consigo vulnerabilidades que afetam diretamente a segurança da aplicação. A ferramenta Dependency Check tem como propósito efetuar análise de vulnerabilidade de dependências utilizadas no projeto back-end, a seguir temos as principais informações extraídas desta análise.

SIMAP-APPLICATION					
DEPENDENCY	HIGHEST SEVERITY	CVE COUNT	CONFIDENCE	EVIDENCE COUNT	
poi-3.10-FINAL.jar	HIGH	7	Highest	28	
poi-ooxml-3.10.1.jar	HIGH	5	Highest	27	
dom4j-1.6.1.jar	HIGH	1	Highest	25	
log4j-1.2.16.jar	CRITICAL	1	Highest	29	
postgresql-9.0-801.jdbc4.jar	CRITICAL	3	Highest	18	
commons-collections4-4.0.jar	HIGH	1	Highest	37	
keycloak-core-3.4.0.Final.jar	CRITICAL	12	Highest	36	
bcprov-jdk15on-1.56.jar	CRITICAL	3	Low	47	
jackson-databind-2.5.4.jar	CRITICAL	22	Highest	40	
simap-presentation-0.4.0-SNAPSHOT.war: jackson-databind-2.4.1.jar	CRITICAL	22	Highest	37	
SIMAP-BUSINESS					
poi-3.10-FINAL.jar	HIGH	7	Highest	28	
poi-ooxml-3.10.1.jar	HIGH	5	Highest	27	
dom4j-1.6.1.jar	HIGH	1	Highest	25	
keycloak-core-3.4.0.Final.jar	CRITICAL	12	Highest	36	
bcprov-jdk15on-1.56.jar	CRITICAL	3	Low	47	
jackson-databind-2.5.4.jar	CRITICAL	22	Highest	40	
log4j-1.2.16.jar	CRITICAL	1	Highest	29	
postgresql-9.0-801.jdbc4.jar	CRITICAL	3	Highest	18	
commons-collections4-4.0.jar	HIGH	1	Highest	37	
SIMAP-ENTITY					
keycloak-core-3.4.0.Final.jar	CRITICAL	12	Highest	36	
bcprov-jdk15on-1.56.jar	CRITICAL	3	Low	47	
jackson-databind-2.5.4.jar	CRITICAL	22	Highest	40	
dom4j-1.6.1.jar	HIGH	1	Highest	25	
SIMAP-INTEGRATION					
keycloak-core-3.4.0.Final.jar	CRITICAL	12	Highest	36	
bcprov-jdk15on-1.56.jar	CRITICAL	3	Low	47	
jackson-databind-2.5.4.jar	CRITICAL	22	Highest	40	
resteasy-jaxrs-3.0.9.Final.jar	HIGH	3		21	
SIMAP-PRESENTATION					
log4j-1.2.16.jar	CRITICAL	1	Highest	29	
postgresql-9.0-801.jdbc4.jar	CRITICAL	3	Highest	18	
commons-collections4-4.0.jar	HIGH	1	Highest	37	
jackson-databind-2.4.1.jar	CRITICAL	22	Highest	38	
keycloak-core-3.4.0.Final.jar	CRITICAL	12	Highest	36	
bcprov-jdk15on-1.56.jar	CRITICAL	3	Low	47	
poi-3.10-FINAL.jar	HIGH	7	Highest	28	
poi-ooxml-3.10.1.jar	HIGH	5	Highest	27	
dom4j-1.6.1.jar	HIGH	1	Highest	25	
SIMAP-UTIL					
dom4j-1.6.1.jar	HIGH	1	Highest	25	
keycloak-core-3.4.0.Final.jar	CRITICAL	12	Highest	36	
bcprov-jdk15on-1.56.jar	CRITICAL	3	Low	47	
jackson-databind-2.5.4.jar	CRITICAL	22	Highest	40	

A planilha acima apresenta as vulnerabilidades encontradas nas dependências de cada módulo, o detalhamento encontra-se no Anexo I deste documento.

4.3 NPM Audit

Após realizar a instalação dos módulos de dependências gerenciados pelo NodeJS (npm install), foram reportados existências de 208 vulnerabilidades na aplicação/dependências e foram classificadas como: 22 são de baixo risco, 38 de riscos moderados, 144 são alto risco e 4 são críticos.

```
added 1183 packages from 1221 contributors and audited 6338 packages in 122.728s
found 208 vulnerabilities (22 low, 38 moderate, 144 high, 4 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details
israel@fswlenovo:~/Documents/projetos/mj/simap/Build/codigos_fonte/frontend/simap$ npm audit fix
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
```

Figura 9: Execução npm install

A partir da versão 6 do gerenciador de pacotes NPM, fora disponibilizado a ferramenta “Audit” que além de auxiliar na descoberta dos itens vulneráveis auxilia através do comando “fix” a atualização/resolução de parte dos problemas. Após a execução do mesmo restaram 23 vulnerabilidades que requerem intervenção manual.

```
* grunt@1.0.4
added 9 packages from 22 contributors, removed 16 packages and updated 21 packages in 208.014s
fixed 142 of 208 vulnerabilities in 6338 scanned packages
  23 vulnerabilities required manual review and could not be updated
  3 package updates for 43 vulns involved breaking changes
(use 'npm audit fix --force' to install breaking changes; or refer to 'npm audit' for steps to fix these manually)
```

Figura 10: Execução npm audit fix

```
added 1 package from 1 contributor, updated 2 packages and audited 6300 packages in 17.446s
found 59 vulnerabilities (15 low, 13 moderate, 21 high, 1 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details
```

Figura 11: Resultado após execução

O relatório completo pode ser acessado através da execução do comando “rpm audit”.

=== npm audit security report ===	
# Run <code>npm install --save-dev @angular/cli@9.0.2</code> to resolve 17 vulnerabilities	
SEMVER WARNING: Recommended action is a potentially breaking change	
Low	Incorrect Handling of Non-Boolean Comparisons During Minification
Package	uglify-js
Dependency of	@angular/cli [dev]
Path	@angular/cli > postcss-url > directory-encoder > handlebars > uglify-js
More info	https://nodesecurity.io/advisories/39
Low	Regular Expression Denial of Service
Package	uglify-js
Dependency of	@angular/cli [dev]
Path	@angular/cli > postcss-url > directory-encoder > handlebars > uglify-js
More info	https://nodesecurity.io/advisories/48
High	Cross-Site Scripting
Package	handlebars
Dependency of	@angular/cli [dev]
Path	@angular/cli > postcss-url > directory-encoder > handlebars
More info	https://nodesecurity.io/advisories/61
Moderate	Regular Expression Denial of Service
Package	mime
Dependency of	@angular/cli [dev]

Figura 12: Execução npm audit



4.4 Análise sobre os resultados

Este tópico tratará tecnicamente a análise baseado nos resultados obtidos pelas ferramentas citadas juntamente com a análise amostral do código fonte.

4.4.1 Manutenibilidade de código

Os relatórios apresentados pela ferramenta SonarQube demonstram uma série de vícios adotados durante o processo de construção do software e alinhado a estes vícios.

A inexistência de cobertura de testes de unidade que trazem a dificuldade no processo de refactoring da aplicação, uma vez que não há condições de mensurar impactos durante o processo de manutenção corretiva/adaptativa. Alinhado a esta questão, a alta complexidade ciclomática e a falta de coesão dificultam este processo de refactoring, a ilustração abaixo demonstra ambos cenários apresentados em um único método (OBS: a característica apresentada é utilizada de forma recorrente em diversos momentos do código).

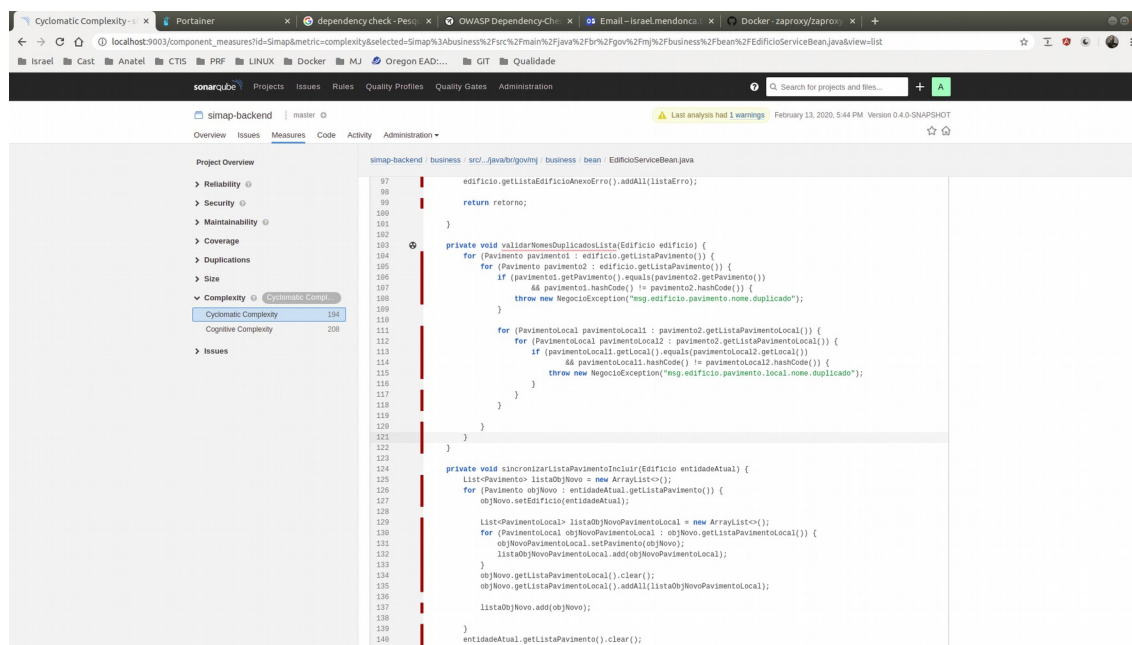



Figura 13: Demonstração de alta complexidade ciclomática - Camada de serviço

MJ	SIMAP – Nota Técnica	
-----------	-----------------------------	--

4.4.2 Confiabilidade

Existe o tratamento de controle transacional na camada de serviço da aplicação, este é o tratamento segue boas práticas de desenvolvimento de aplicações sendo esta a camada responsável por orquestrar as execuções em banco de dados. Este controle transacional garante as propriedades ACID do SGBD.

4.4.3 Performance e estabilidade

Não foi analisado a aplicação em funcionamento para avaliar demais requisitos não funcionais, contudo por análise de código fora identifica a presença de classes internas no arquivo LDAPFilter.java que efetuam a tratativa de array de bytes durante as requisições HTTP. Aparentemente estas classes não estão sendo utilizadas no contexto da aplicação, contudo requerem maior análise em nível de debug tendo em vista que sua utilização pode aumentar substancialmente o consumo de memória Heap.

4.4.3 Escalabilidade

A arquitetura baseada em micro serviços e o comportamento sem estado (stateless) da aplicação back-end promove a esta uma boa capacidade de escalonamento na horizontal com a utilização de cluster e balanceadores de carga.

Esta arquitetura além de promover ambiente escalonável, favorece a utilização de ambientes redundantes com maior probabilidade de tolerância a falhas.



5 Recomendações

É altamente recomendado que seja efetuado refactoring de código dos bugs e vulnerabilidades de código apontadas pelo SonarQube, estas atividades certamente trarão maior confiabilidade a ferramenta e estabilidade em seu uso. Para os demais itens apontados pela ferramenta SonarQube durante o processo de análise de código são altamente desejáveis, contudo este processo de ajuste de código é moroso e trás consigo risco em potencial e está diretamente aliado a falta de cobertura de testes de unidade.

Ajustar as dependências que trazem maior risco para a aplicação é altamente recomendável, contudo este trabalho deve ser feito de forma analítica e cautelosa afim de não prejudicar a estabilidade da ferramenta. Sugere-se a utilização de ferramentas de pentest (análise de vulnerabilidade) tais como OWASP ZAP (<https://owasp.org/www-project-zap/>) para que seja analisado as principais vulnerabilidades da aplicação e associá-las as dependências que oferecem tais riscos para os devidos ajustes. Esta recomendação esta embasada na interseção de resultados das ferramentas utilizadas e na otimização e na assertividade do trabalho de refactoring.

Recomenda-se também que seja instalado o agente da ferramenta de APM do Ministério da Justiça nos ambientes de homologação e produção, criar métricas e alarmes auxiliam na continuidade do serviço (monitoramento de processamento e memória por exemplo) tendo em vista que esta ferramenta fornece mecanismos para determinarmos o comportamento da solução (auxiliam no refactoring de código) também subsidia para o correto dimensionamento da infraestrutura.

Não há evidências que comprovem instabilidade na versão 4 do framework Angular, contudo caso exista o interesse em atualizá-lo recomenda-se a execução de estudo de viabilidade técnica e dimensionamento de esforço para a atualização da mesma para a versão mais estável mais atual (a versão 8 do framework é versão mais recente e estável no momento da escrita desta nota técnica). Dentre as vantagens da atualização da versão temos: suporte a versão 3.2 do TypeScript, otimização do processo de build, suporte a nova versão da biblioteca RxJS, lazy loading modules, diferencial loading module e novas features do Angular Material.

Recomenda-se também a análise de performance das operações executadas em banco de dados objetivando não somente a melhoria das mesmas, quanto subsídio para análise de viabilidade técnica que auxilie em sua mudança estrutural.

6 Conclusão

A aplicação apresenta uma boa estruturação em sua construção o que facilita a sua manutenção corretiva/evolutiva e aliado a necessidade de utilização dos recursos ofertados na versão 8 (ou superior) do framework Angular para a aplicação front-end, sugere-se o estudo de viabilidade técnica para o devido upgrade

Em relação aos ajustes de código da aplicação, sugestiona-se que os mesmos sejam efetuados com o auxílio/suporte de testes unitários tendo em vista que este caminho trará maior assertividade e menor risco para o não comprometimento da estabilidade da ferramenta.