Departamento de Polícia Rodoviária Federal

Projeto: Sistema de Acompanhamento de Emissão de CIF

SPIF

Nota Técnica

Revisão	Descrição	Autor	Data
1.0	Construção do documento	Israel Branco	02/06/2021

1 Sumário

2 Considerações iniciais

Este documento visa reportar o resultado da análise efetuada no **Sistema de Acompanhamento de Emissão de CIF** denominada neste documento simplesmente como **SPIF**. 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, modelo relacional de banco de dados e concepção arquitetural.

Para a realização desta análise, gerou-se a tag *ctis-nota-tecnica-20210531* disponível no repositório GIT https://git.prf/inovacao/spif/-/tree/ctis-nota-tecnica-20110531 com referência branch master na data de 31/05/2021.

3 Apresentação do cenário atual

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

O sistema SPIF foi construído para funcionar em ambiente WEB, utiliza tecnologia Java e está estruturada arquiteturalmente como uma aplicação monolítica (entende-se por este termo quando o sistema é composto por camadas de interface com usuário, camada de aplicação de regras negociais e camada de acesso a dados combinadas em uma única aplicação), utiliza o banco de dados *PostgreSQL.*

Sua arquitetura é composta apenas por um componente com segregação em pacotes que contem: controladores, persistencia, mapeamento ORM, DTO, camada de serviço, camada de acesso a dados (DAO), validadores customizados, exceções, conversores customizados, exceções e classes utilitárias. Os componentes visuais (folhas de estilo, páginas web, java script) estão segregados em diretórios específicos segundo a padronização da tecnologia utilizada para a camada de apresentação.

O diagrama a seguir representa o modelo de componentes ao qual a aplicação está construída.

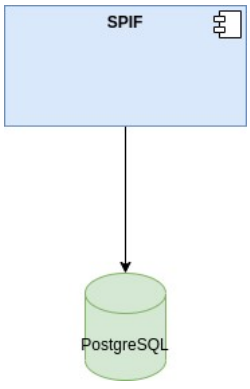

Figura 1: Componentes

A aplicação utiliza o modelo MVC para a segregação de responsabilidades em camadas onde as requisições http são oriundas das páginas JSF. O diagrama a seguir representa os fluxos encontrados durante o processo de análise da aplicação, o diagrama representa o fluxo principal encontrado no processamento de tarefas agendadas.

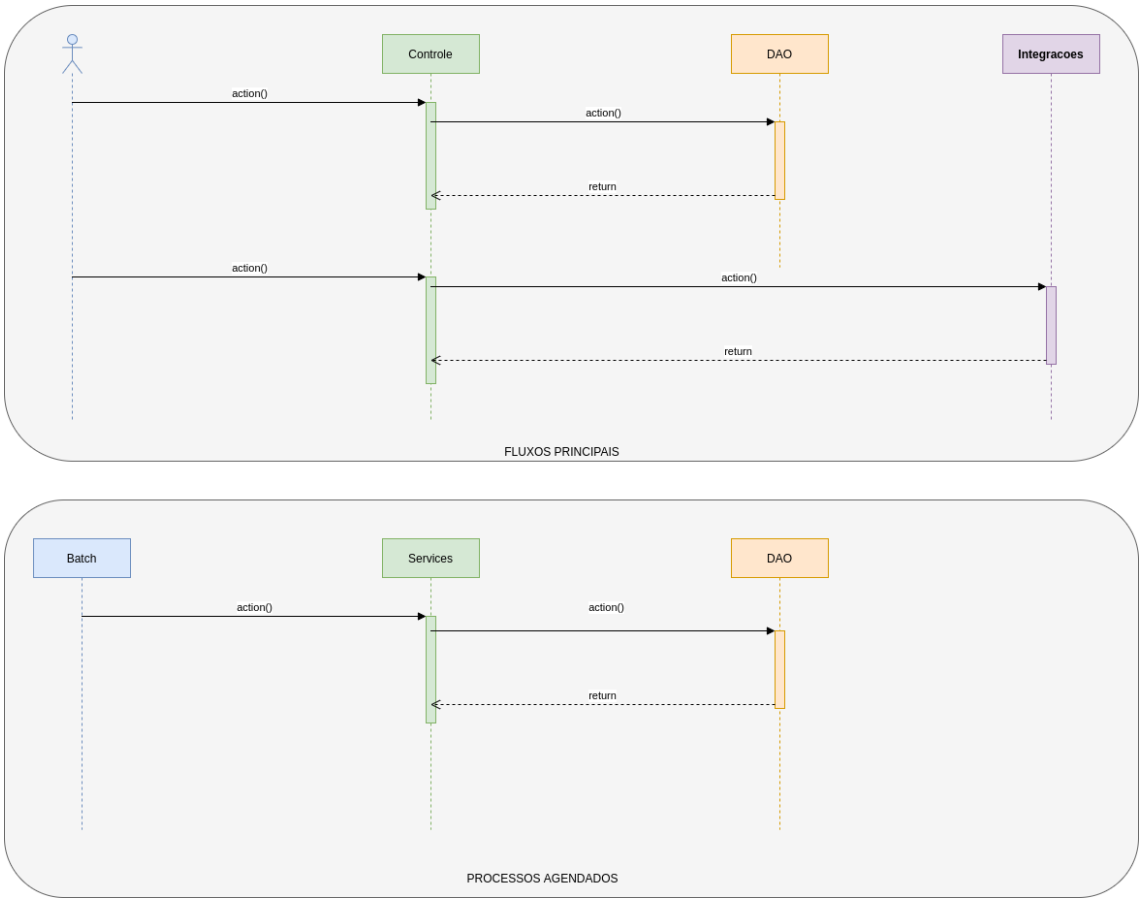

Figura 2: Diagrama de sequências

A solução utiliza o schema dbidentfuncioal que contendo um total de 11 tabelas.

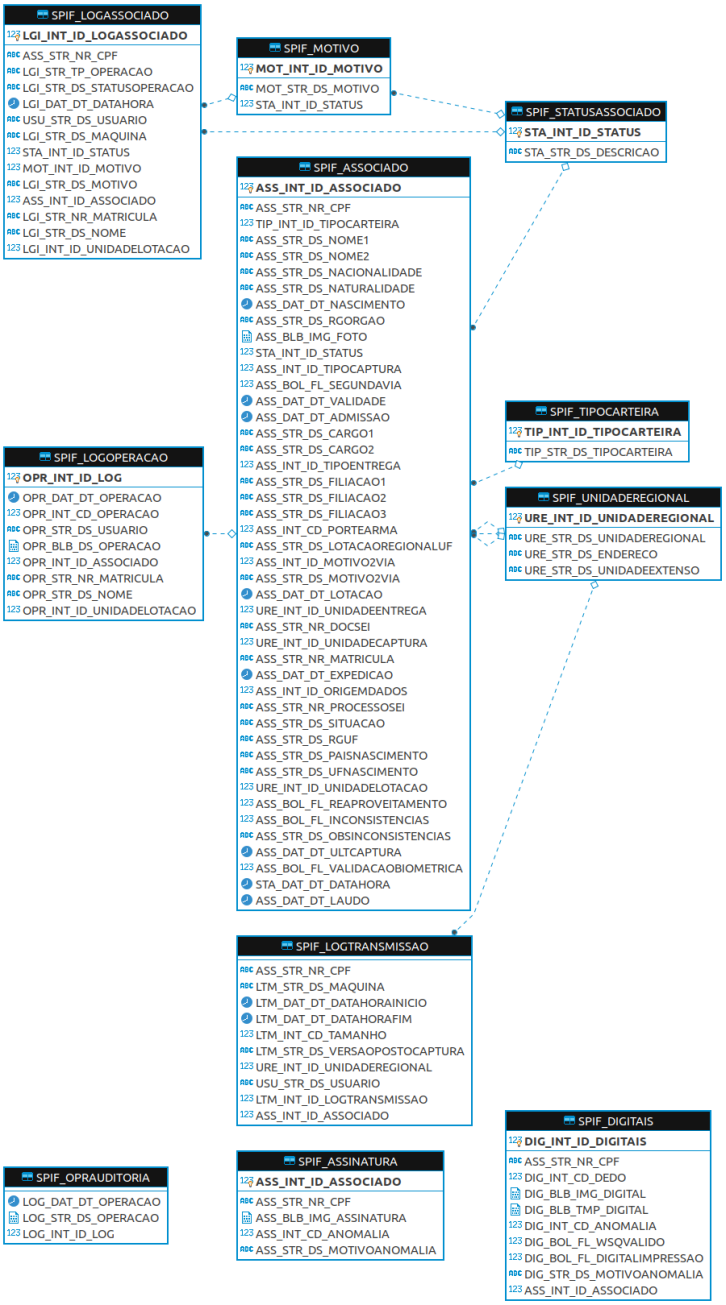

Figura 3: Modelo entidade relacional

3.1 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.	
Hibernate	5.x	Framework ORM.	
Primefaces	5.1	Extensão de componentes JSF.	
CDI	1.2		
Wildfly	10.x	Servidor de aplicação JEE.	Utiliza container CDI e Servlet.
PostgreSQL		Banco de dados relacional.	
Apache POI	3.10	Biblioteca para leitura e gravação de arquivos MS Office e PDF.	

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 da DPRF, contudo foram utilizadas as regras de qualidade padrões de análise da ferramenta.

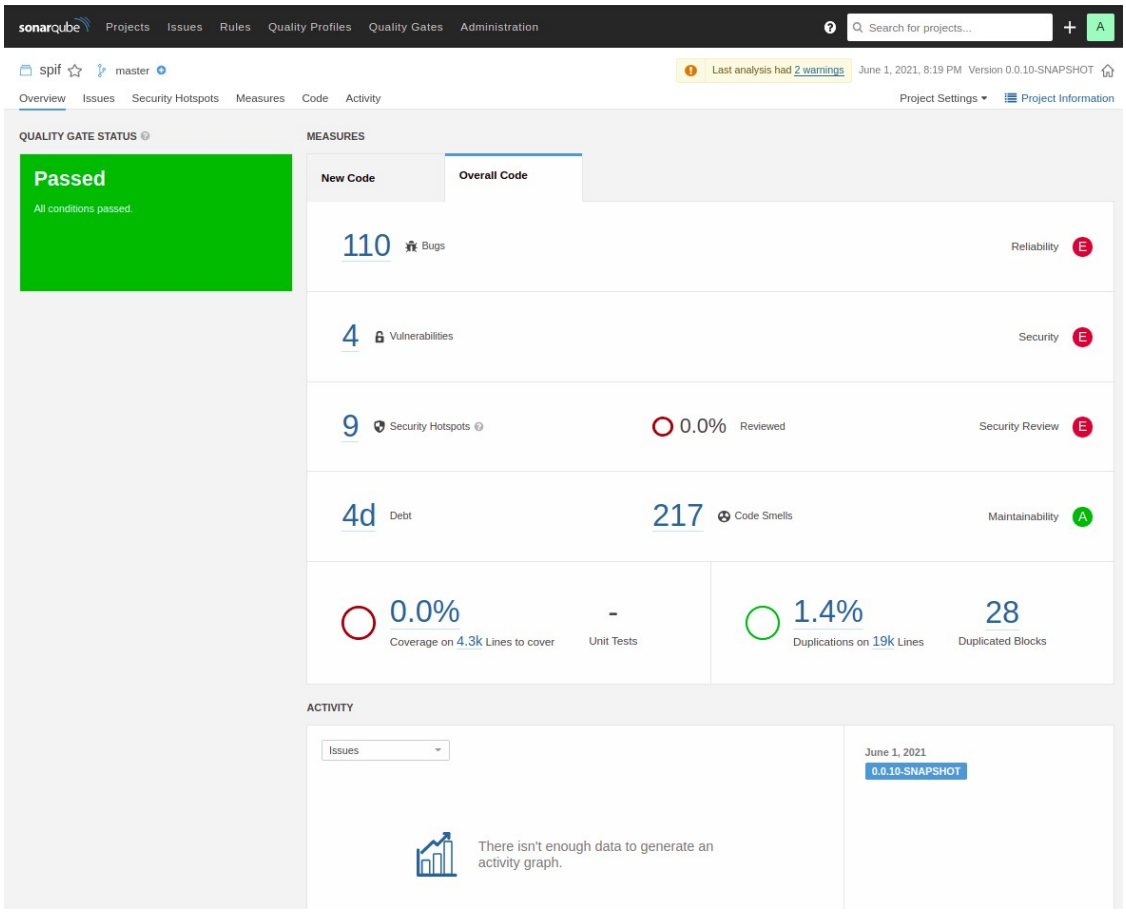

Figura 4: Análise estática de código

- 110 bugs;
- 4 vulnerabilidades de código;
- 9 violações de segurança;
- 217 violações de código ruim (complexidade cognitiva , complexidade ciclomática e débito técnico);
- 1,6% de duplicidade de código

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 na construção deste projeto, a seguir temos as principais informações extraídas desta análise, a relação completa desta análise está disponível no Anexo I deste documento.

Dependency	Package	Highest Severity	CVE Count
httpclient-4.5.3.jar	pkg:maven/org.apache.httpcomponents/httpclient@4.5.3	MEDIUM	1
jasperreports-6.3.1.jar	pkg:maven/net.sf.jasperreports/jasperreports@6.3.1	HIGH	5
itext-2.1.7.js5.jar	pkg:maven/com.lowagie/itext@2.1.7.js5	HIGH	1
bcmail-jdk14-138.jar	pkg:maven/bouncycastle/bcmail-jdk14@138	MEDIUM	1
bcprov-jdk14-138.jar	pkg:maven/bouncycastle/bcprov-jdk14@138	HIGH	18
spring-aop-3.0.6.RELEASE.jar	pkg:maven/org.springframework/spring-aop@3.0.6.RELEASE	CRITICAL	13
spring-core-3.0.6.RELEASE.jar	pkg:maven/org.springframework/spring-core@3.0.6.RELEASE	CRITICAL	14
poi-3.10-FINAL.jar	pkg:maven/org.apache.poi/poi@3.10-FINAL	HIGH	7
commons-beanutils-1.8.3.jar	pkg:maven/commons-beanutils/commons-beanutils@1.8.3	HIGH	2
primefaces-5.3.jar	pkg:maven/org.primefaces/primefaces@5.3	CRITICAL	3
hibernate-validator-5.1.3.Final.jar	pkg:maven/org.hibernate/hibernate-validator@5.1.3.Final	MEDIUM	1
cdi-api-1.2.jar	pkg:maven/javax.enterprise/cdi-api@1.2	MEDIUM	1
javax.faces-api-2.2.jar	pkg:maven/javax.faces/javax.faces-api@2.2	MEDIUM	1
shiro-core-1.2.3.jar	pkg:maven/org.apache.shiro/shiro-core@1.2.3	CRITICAL	7
postgresql-9.3-1100-jdbc41.jar	pkg:maven/org.postgresql/postgresql@9.3-1100-jdbc41	CRITICAL	11
jackson-databind-2.1.4.jar	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.1.4	CRITICAL	18
jai-imageio-core-1.3.1.jar	pkg:maven/com.github.jai-imageio/jai-imageio-core@1.3.1	HIGH	1
bootstrap.min.js	pkg:javascript/bootstrap@4.0.0-alpha	MEDIUM	4
jquery.js	pkg:javascript/jquery@1.9.1	MEDIUM	4
primefaces-5.3.jar: jquery.js	pkg:javascript/jquery@1.11.3	MEDIUM	4

4.3 OWASP ZAP

Ferramenta funciona como scanner de segurança, utilizada para realização de testes de vulnerabilidade de aplicações WEB. Atualmente trata-se de um dos projetos mais ativos na comunidade de software livre.

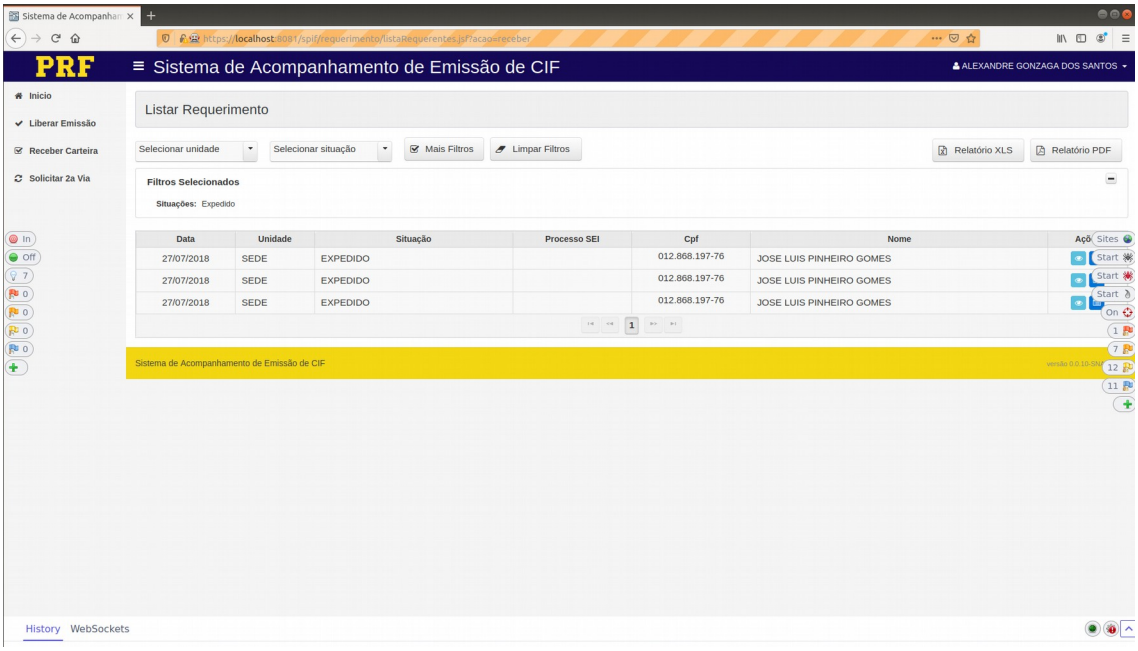

Figura 5: OWASP ZAP - Execução de Pentest

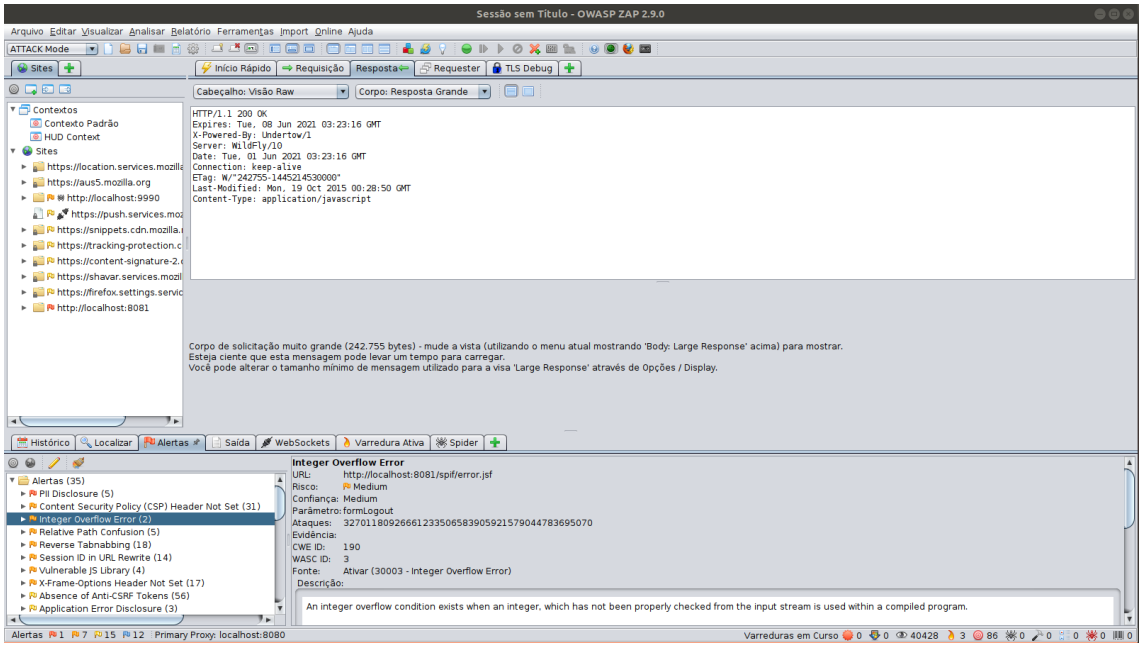

Figura 6: OWASP ZAP - Execução de Pentest

- 1 vulnerabilidade de severidade alta;
- 8 vulnerabilidade de severidade média;
- 35 vulnerabilidades de baixa média;
- 31 vulnerabilidades a nível informativo;

O relatório completo dos testes aplicados estão disponíveis no anexo I deste documento.

4.4 Estrutura do projeto

Os componentes que envolvem o escopo da ferramenta possui boa organização por contexto de responsabilidade.

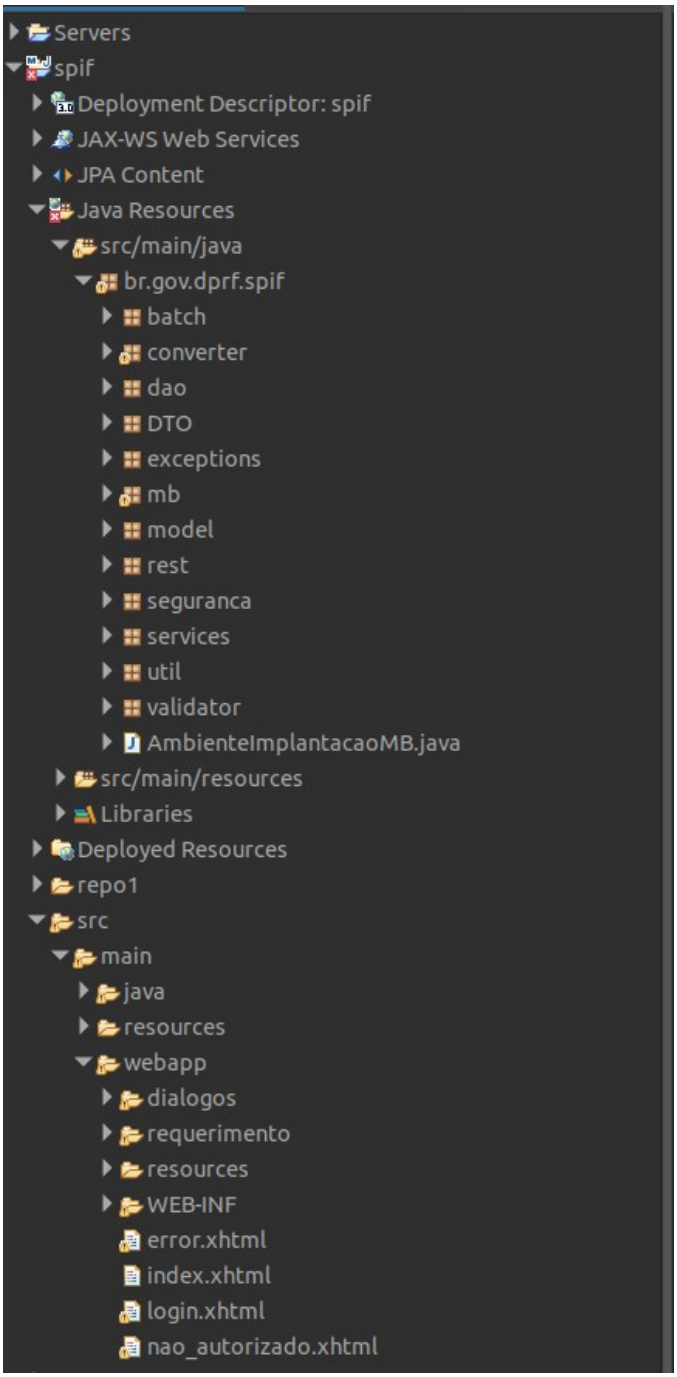

Figura 6: Organização do projeto

4.5 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 testes de unidade 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.

Embora tenhamos apontamentos de bugs e code smells altos relativo ao tamanho da aplicação, a existência de baixa complexidade ciclomática favorece o entendimento negocial da aplicação.

Fatores dificultadores para a manutenibilidade do código estão voltadas para o gerenciamento de dependências da aplicação. Duas versões de bibliotecas apontadas (em comentário de código) não estão presentes no repositório maven atual conforme a figura 7. Para o favorecimento desta nota técnica, as versões superiores mais próximas foram utilizadas.

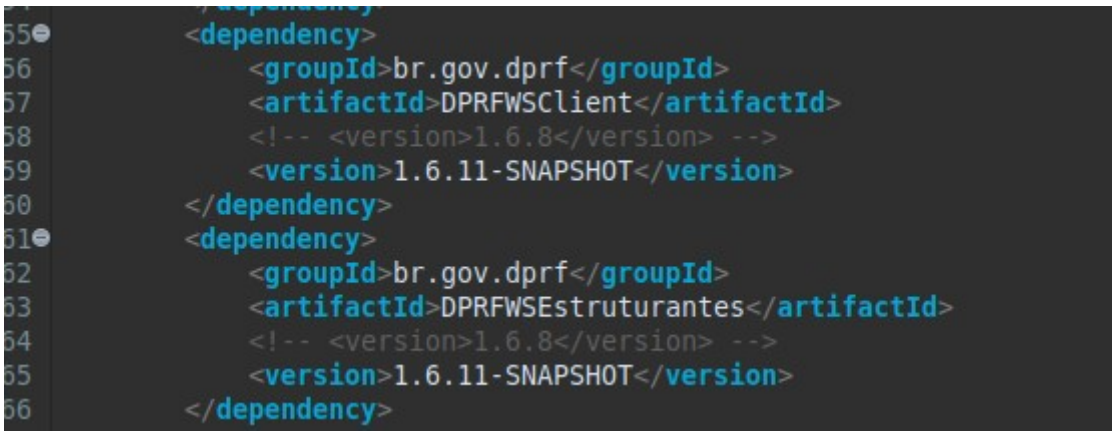

Figura 7: Versões de componentes DPRF

Ainda se tratando da execução do projeto, houve a necessidade de substituições e correções dos repositórios maven reportados no arquivo pom.xml. As figura 8 e 9 trazem o comparativo entre as configurações existentes (a esquerda) e as configurações necessárias

para a execução do projeto (a direita).

Figura 8: Configuração atual

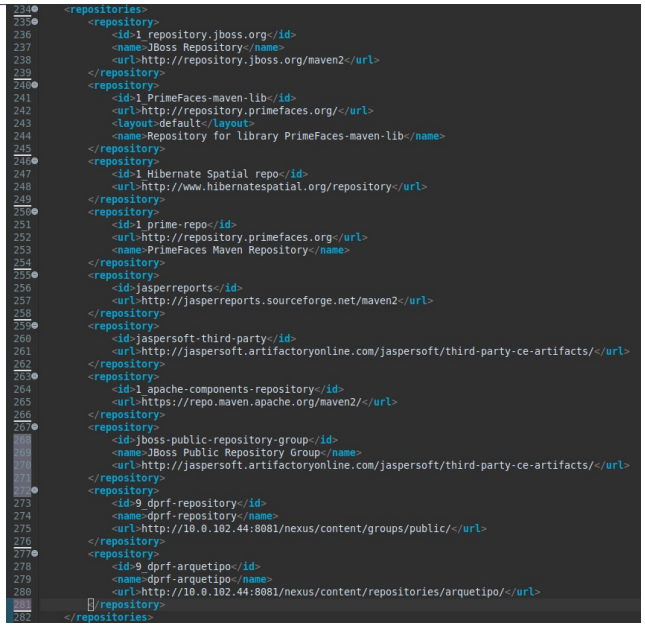

Figura 9: Configuração necessária

4.6 Confiabilidade

O controle de transação efetuado na aplicação está sendo feito em operações controladas a nível de aplicação na camada de acesso a tados quando as solicitações são requisitadas da camada de controller e a nível de aplicação quando requisitadas através das rotinas agendadas, sendo esta uma prática recomendada para que haja garantia das propriedades ACID.

A manutenção da consistência de dados é algo fortemente desejado, contudo esta não garante toda a confiabilidade da solução. A quantidade elevada de bugs, vulnerabilidades no código e nas bibliotecas de terceiros encontradas nos relatórios apresentados trazem riscos a confiabilidade da ferramenta.

4.7 Performance e estabilidade

Não foi analisado o funcionamento da aplicação para avaliar demais requisitos não funcionais, recomenda-se a utilização de ferramentas de APM para mensurar performance e recursos de máquina utilizados.

A arquitetura monolítica citada no tópico 3 deste documento prejudica a escalabilidade da ferramenta, os recursos empreendidos para a escalabilidade vertical (aumento de recursos de processamento, disco, memória e demais) são limitados e onerosos.

A escalabilidade vertical do monólito é possível levanto em consideração o aumento de nós no cluster, contudo esta escalabilidade é prejudicada tendo em vista que temos que escalar a aplicação como um todo, necessitando assim da mesma quantidade de recursos empreendidas nos demais nós existentes. Nesta arquitetura não há a possibilidade de escalar somente as funcionalidades/módulos que mais são demandados.

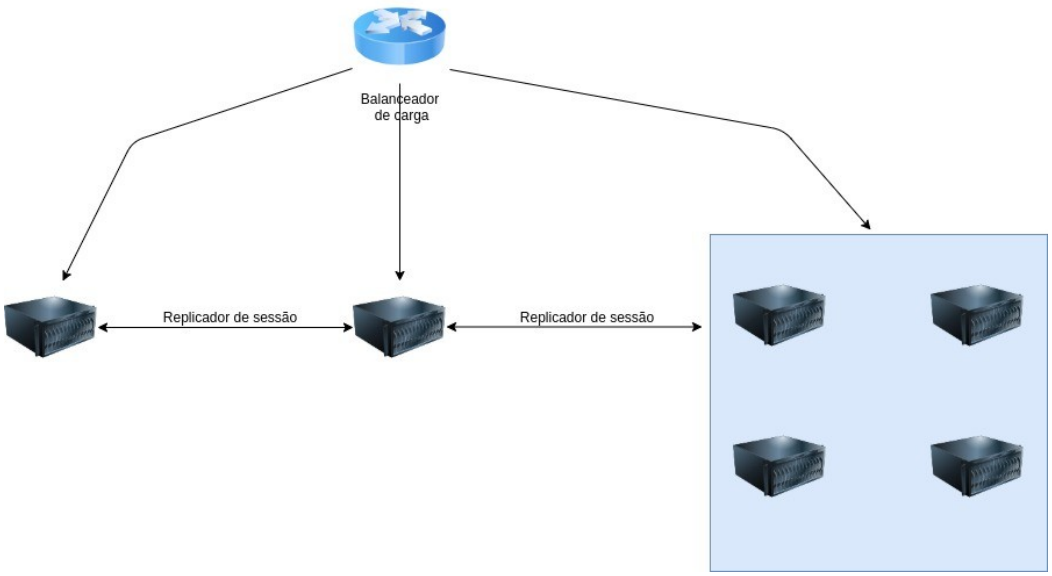

Figura 10: Escalabilidade do monólito

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 associação dos relatórios de análise de dependências com os relatórios de análise de intrusão para que sejam analisados 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 a implantação de ferramentas de APM para que sejam criadas métricas e alarmes que auxiliem na continuidade do serviço em ambiente produtivo(monitoramento de processamento e memória por exemplo), tendo em vista que este tipo de ferramenta fornece mecanismos para determinarmos o comportamento da solução (auxiliam no refactoring de código) e também subsidia para o correto dimensionamento da infraestrutura.

Recomenda-se o desacoplamento das rotinas agendadas da aplicação, uma vez que esta prática promove a concorrência de recursos da aplicação principal e dificulta a escalabilidade horizontal.

Atualmente os anexos submetidos à aplicação estão sendo

armazenados em banco de dados em estruturas de blobs, recomenda-se que os mesmos sejam utilizados em estruturas apropriadas para armazenamento de arquivos não estruturados, tal como Blob Storage.

A aplicação não possui integração com o sistema de RH da DPRF e a falta desta integração traz consigo a necessidade de manter dados dos servidores tais como nome, cargo, matricula, lotação e demais. A falta desta trás consigo a dificuldade da manutenibilidade do dado referente ao servidor, recomenda-se fortemente que estas informações sejam oriundas do sistema de RH atual da DPRF para a correta aplicabilidade da informação no contexto negocial.