Departamento de Polícia Rodoviária Federal

Projeto: DAT - Declaração de Acidente de Trânsito

Absorção de Sistema

Revisão	Descrição	Autor	Data
1.0	Construção do documento	Israel Branco	24/03/2020

1 Sumário

2 Considerações iniciais

O presente documento tem por objetivo a verificação do processo de absorção de tecnologia do projeto DAT. Este processo consiste em analisar as necessidades para preparação de ambiente local de desenvolvimento, impedimentos tecnológicos para continuidade da solução, análise estática de código e análise de vulnerabilidade de dependências.

E para atender ao objetivo expresso acima, compreende-se que o termo transferência de tecnologia é definido como um processo entre duas entidades sociais, em que o conhecimento tecnológico é adquirido, desenvolvido, utilizado e melhorado por meio da transferência de um ou mais componentes de tecnologia. Existe ainda a necessidade de inovar e evoluir com autonomia em busca de novas funcionalidades, de forma continuada, preservando os interesses originais encontrados no desenvolvimento do projeto.

2.1 Motivação

O sistema DAT tem por objetivo realizar o registro de acidentes em rodovias federais ao qual não houveram vítimas (nem mesmo lesões leves), derramamento ou vazamento de produto perigoso, danos aos bens públicos e por fim não pode ter ocorrido incêndio.

O sistema possui dois módulos de acesso sendo eles, acesso externo de uso público em geral ao qual o objetivo é o registro de ocorrência de acidente e o segundo de acesso interno à DPRF para validações dos registros.

3 Apresentação do cenário atual

Os módulos de acesso interno e externo denominados aqui respectivamente como DAT-WEB-VALIDACAO E DAT-WEB-DECLARANTE foram construídos para trabalhar em ambiente web com tecnologia Java. A arquitetura de ambos módulos é bem similar e ambas são dotadas como monólitos (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).

O diagrama a seguir representa a composição dos módulos da solução em alto nível.

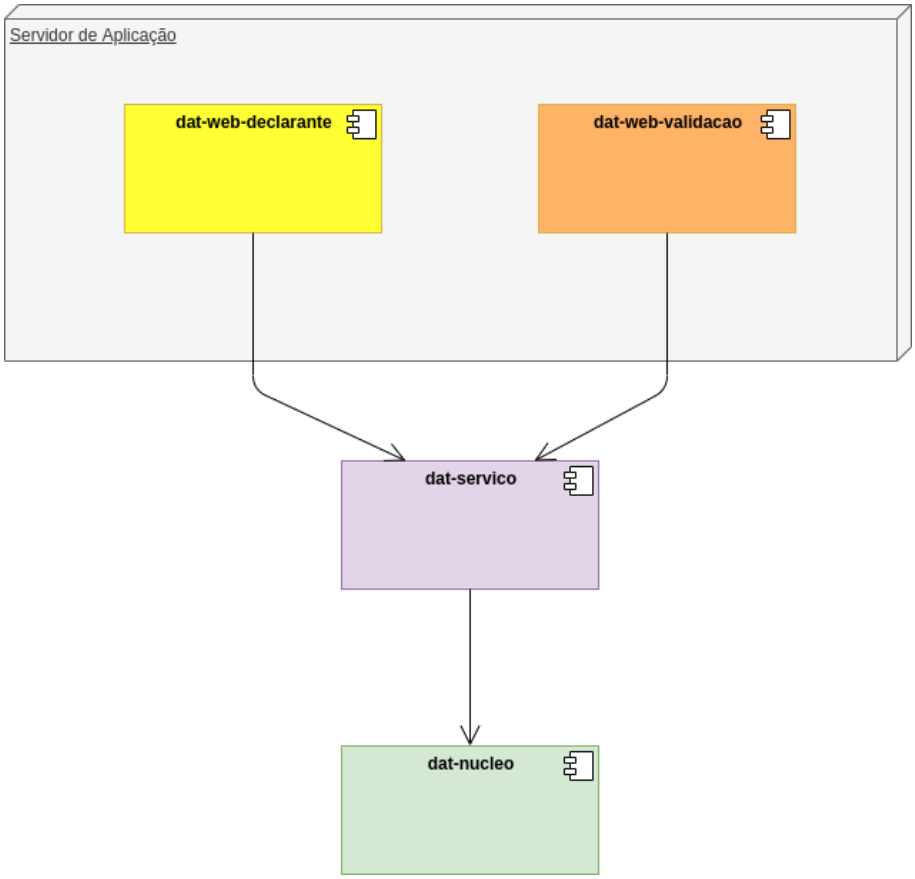

Figura 1: Diagrama de componentes

- **dat-nucleo**: Este componente possui o mapeamento ORM de classes Java com as tabelas de banco de dados. A estrutura deste projeto está feito de forma organizada e com boa separação de pacotes.
- **dat-servico**: Componente responsável por tratar regras negociais, possui em sua composição classes de serviço, camada de acesso a dados, classes utilitárias em geral e classes de validações.
- **dat-web-declarante**: Aplicação web externa utilizada para acesso publico em geral para efetuar as declarações de acidentes.
- **dat-web-validacao**: Aplicação web interna utilizada com acesso restrito para validações das declarações de acidentes.

O diagrama de sequência a seguir demonstra a comunicação básica entre as camadas que compõe as aplicações internas e externas.

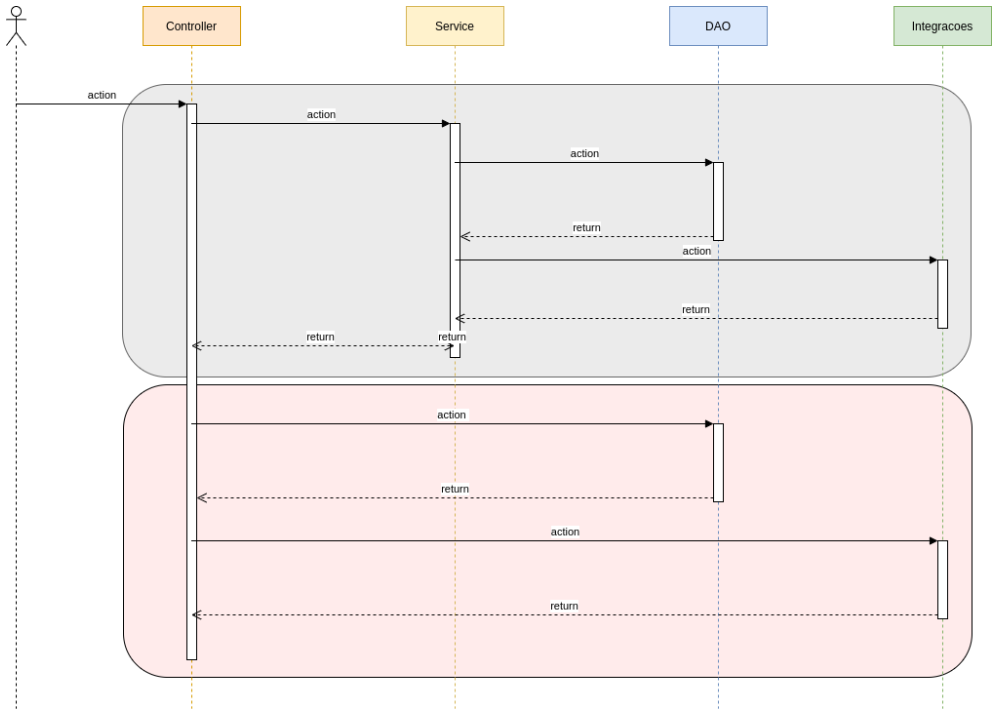

Figura 2: Diagrama de sequências

Destacado com o fundo azul temos a sequência básica da transmissão de dados entre as camadas, essa sequência é mantida por quase toda aplicação contudo, é possível enxergar em ocorrências menores comportamentos destacados com o fundo vermelho. A classe ConsultasMB.java com a designação de um Managed Bean da especificação JSF possui o comportamento destacado em vermelho, essa sequência tira o padrão de comunicação MVC do restante da aplicação.

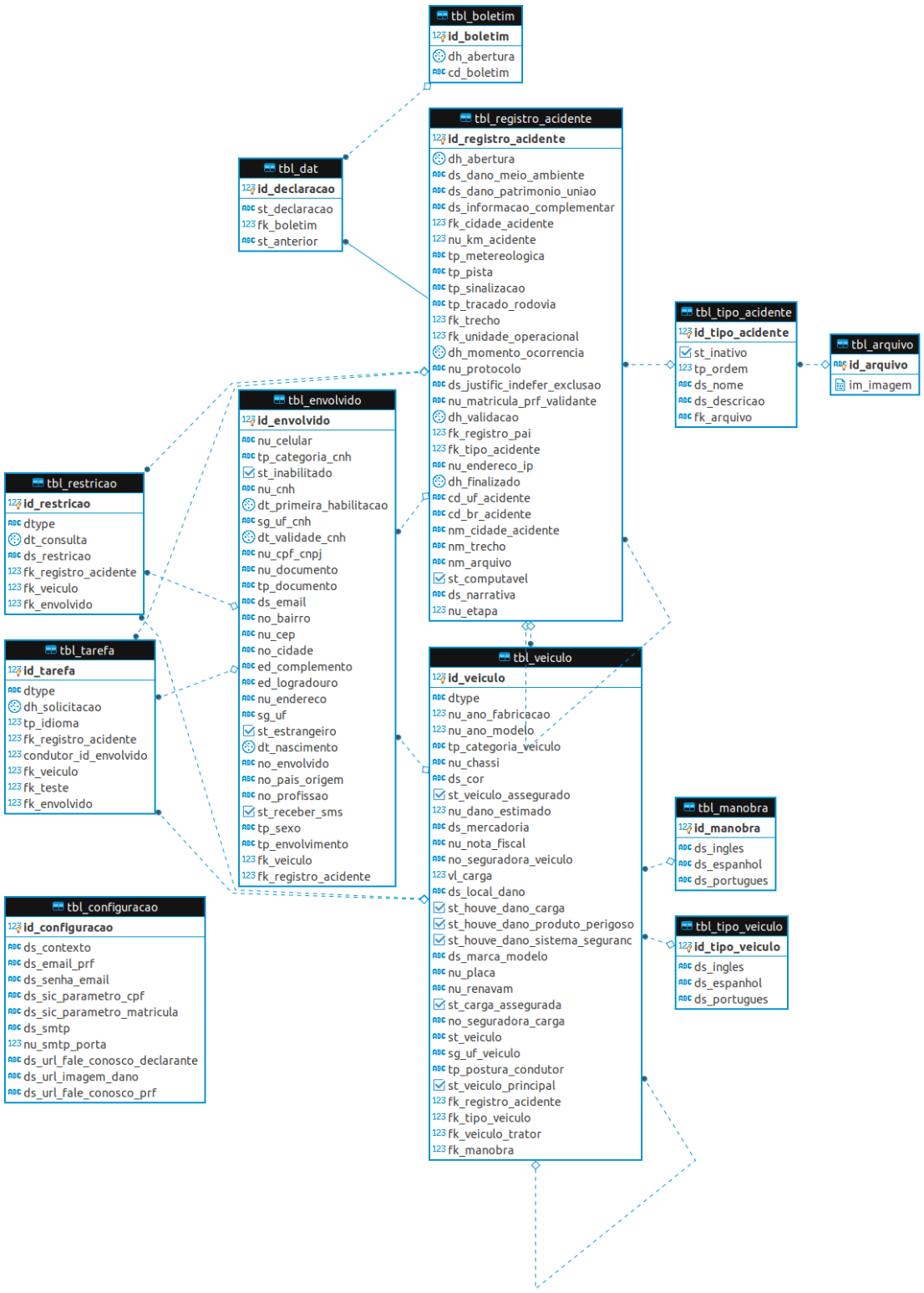

Figura 3: MER - Banco dbedat Schema edat

3.1 Documentação existente

Código fonte: Disponibilidade completa no repositório GIT https://git.prf/acidente/dat;

Documentação de requisitos: Não é disponível.

Documentação de implantação: Não é disponível;

Documentação para criação de ambiente: Não é disponível;

3.2 Implantação

Por se tratar de um sistema oriundo a equipe de inteligência do Departamento de Polícia Rodoviária Federal, todo ambiente de desenvolvimento e produção da solução esta segregado das demais implantações mantidas pelo departamento de operações, não sendo possível assim estimar os recursos de hardware necessários para a operação da solução.

Atualmente a implantação dos dois módulos WEB estão sendo feitas no servidor de aplicação Wildfly versão 10.1.x. Durante a escrita deste documento percebeu-se a ausência da dependência DRPFWSCliente versão 1.6.15 e de suas dependências no repositório de componente Nexus (http://nexus1.datacenter1:8081/nexus/content/repositories/release), para a compilação do projeto se fez necessário atualizar a versão desta dependência para a versão 1.7.2, sendo necessário a mesma ação para a dependência ClienteMotoConsultas que saiu da versão 2.1.4 para a versão 2.1.9.

3.3 Build, configuração e deploy

Para o processo de build se faz necessário a utilização do gerenciador apache Maven 3.3.x, JDK 1.8, Wildfly 10.1.x, driver JDBC PostgreSQL 9.4.x e modulo Hibernate Spatial 5.0.7.

Após a execução do processo de build (mvn install), será necessário configurar os módulos adicionais no servidor de aplicação.

3.3.1 Driver JDBC

Deposite o driver jdbc no diretório %WILDFLY_HOME%/modules/system/layers/base/org/postgresql/main. Crie neste diretório o arquivo module.xml com o seguinte conteúdo:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
      <resources>
         <resource-root path="postgresql-9.4.1211.jar"/>
      </resources>
      <dependencies>
          <module name="javax.api"/>
         <module name="javax.transaction.api"/>
      </dependencies>
   </module>
```

Adicione no arquivo standalone.xml entre as chaves <datasources></datasources> o conteúdo (substitua as partes ${env.xxx} com o conteúdo apropriado:

```
<datasource jndi-name="java:jboss/datasources/edatDS" pool-name="edatDS">
                <connection-url>jdbc:postgresql://${env.host}:5432/${env.db}</connection-url>
                <driver>postgresql</driver>
                <pool>
                   <min-pool-size>1</min-pool-size>
                   <max-pool-size>20</max-pool-size>
                </pool>
                <security>
                   <user-name>${env.user}</user-name>
```

```
            <password>${env.passwd}</password>
        </security>
        <timeout>
            <idle-timeout-minutes>3</idle-timeout-minutes>
        </timeout>
    </datasource>
```

Adicione no arquivo standalone.xml entre as chaves <drivers></drivers> o conteúdo:

```
<driver name="postgresql" module="org.postgresql">
        <driver-class>org.postgresql.Driver</driver-class>
</driver>
```

3.3.2 Módulo Hibernate Spatial

Faça o download via maven do seguintes arquivos:

```
mvn dependency:copy -Dartifact=org.hibernate:hibernate-spatial:5.0.7.Final:jar -DoutputDirectory=.
mvn dependency:copy -Dartifact=org.geolatte:geolatte-geom:1.0.1:jar -DoutputDirectory=.
mvn dependency:copy -Dartifact=com.vividsolutions:jts:1.13:jar -DoutputDirectory=.
```

Copie os arquivos para o diretório %WILDFLY_HOME%/modules/system/layers/base/org/hibernate/main e adicione nas tags <resources></resources> no arquivo module.xml o seguinte conteúdo:

```
<resource-root path="hibernate-spatial-5.0.7.Final.jar"/>
<resource-root path="jts-1.13.jar"/>
<resource-root path="geolatte-geom-1.0.1.jar"/>
```

Adicione também o conteúdo abaixo entre as tag <dependencies></dependencies>:

```
<module name="org.slf4j"/>
```

Após estes passos, deposite os arquivos com extensão .war no diretório %WILDFLY_HOME%/standalone/deployments.

3.4 Principais bibliotecas e frameworks

- Bibliotecas para tratativa da camada de persistência
  - Hibernate 4.3.11 (core, entity-manager)
  - Hibernate Spatial 4.3.0
  - Hibernate validator 5.2.2
- JavaEE Web API 7
- JVM 1.8
- Wildfly 10.1.x
- Framework MVC, relatórios e WEB em geral
  - JSF-API 2.2.13
  - Primefaces 6.1
  - Omnifaces 2.6.6
  - Bootstrap 1.0.10
  - Freemarker 2.3.8
  - Jasperreports 6.3.0

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 padrões de análise da ferramenta. Os resultados foram os seguintes para as aplicações (master branch):

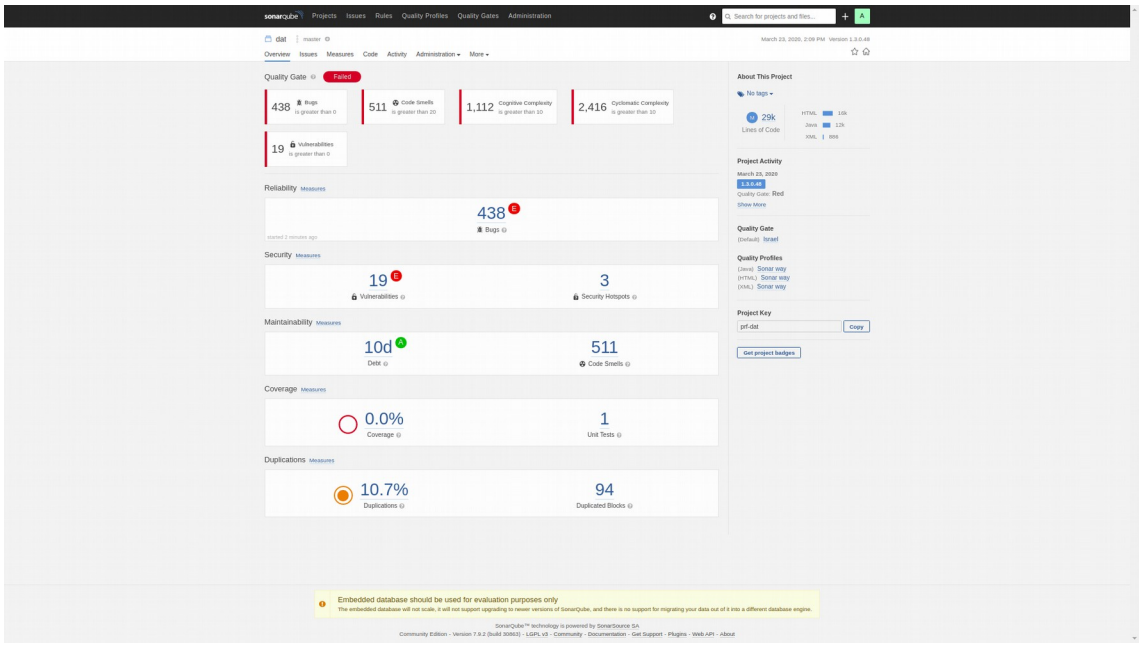

Figura 4: Análise estática de código

Nesta análise obtivemos os seguintes resultados:

- 438 bugs;

- 511 violações de más práticas;
- 1112 violações de complexidade cognitiva (dificuldade de entendimento de código);
- 2416 violações de complexidade ciclomática (complexidade de código);
- 19 vulnerabilidades;

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	Highest Severity	CVE Count	Confidence	Evidence Count
Dat Nucleo				
commons-beanutils-1.7.0.jar	HIGH	2	Highest	22
dom4j-1.6.1.jar	HIGH	1	Highest	25
hibernate-validator-5.2.2.Final.jar	HIGH	1	Highest	34
httpclient-4.2.6.jar	MEDIUM	2	Highest	34
postgresql-8.4-701.jdbc4.jar	HIGH	10	Highest	18
Dat Service				
commons-beanutils-1.8.3.jar	HIGH	2	Highest	35
commons-email-1.3.3.jar	HIGH	2	Highest	38
jasperreports-6.3.0.jar	HIGH	4	Highest	29
bcprov-jdk14-138.jar	HIGH	13	Highest	23
spring-aop-3.0.6.RELEASE.jar	CRITICAL	11	Highest	28
spring-core-3.0.6.RELEASE.jar	CRITICAL	13	Highest	28
jackson-databind-2.1.4.jar	CRITICAL	28	Highest	38
httpclient-4.2.6.jar	MEDIUM	2	Highest	34
Dat Web Declarante				
bootstrap-4.0.0-beta.2.jar: bootstrap.bund	MEDIUM	4		3
bootstrap-4.0.0-beta.2.jar: bootstrap.bund	MEDIUM	4		3
bootstrap-4.0.0-beta.2.jar: bootstrap.js	MEDIUM	4		3
bootstrap-4.0.0-beta.2.jar: bootstrap.min.j	MEDIUM	4		3
commons-beanutils-1.7.0.jar	HIGH	2	Highest	22
httpclient-4.2.6.jar	MEDIUM	1	Highest	34
jquery-1.10.2.min.js	MEDIUM	2		3
jquery-3.0.0.jar: jquery.js	MEDIUM	1		3
jquery-3.0.0.jar: jquery.min.js	MEDIUM	1		3
jquery-migrate-1.2.1.min.js	medium	1		3
jquery-ui.min.js	MEDIUM	1		3
popper.js-1.12.3.jar: jquery.min.js	MEDIUM	2		3
primefaces-6.1.jar: jquery.js	MEDIUM	2		3
shiro-core-1.3.2.jar	MEDIUM	1	Highest	32
spring-core-3.0.6.RELEASE.jar	CRITICAL	11	Highest	28
xstream-1.4.7.jar	HIGH	2	Highest	44
Dat Web Validacao				
bootstrap-4.0.0-beta.2.jar: bootstrap.bund	MEDIUM	4		3
bootstrap-4.0.0-beta.2.jar: bootstrap.bund	MEDIUM	4		3
bootstrap-4.0.0-beta.2.jar: bootstrap.js	MEDIUM	4		3
bootstrap-4.0.0-beta.2.jar: bootstrap.min.j	MEDIUM	4		3
commons-beanutils-1.7.0.jar	HIGH	2	Highest	22
httpclient-4.2.6.jar	MEDIUM	1	Highest	34
jquery-3.0.0.jar: jquery.js	MEDIUM	1		3
jquery-3.0.0.jar: jquery.min.js	MEDIUM	1		3
jquery-3.2.1.jar: jquery.js	MEDIUM	1		3
jquery-3.2.1.jar: jquery.min.js	MEDIUM	1		3
jquery-3.2.1.jar: jquery.slim.js	MEDIUM	1		3
jquery-3.2.1.jar: jquery.slim.min.js	MEDIUM	1		3
jquery.js	MEDIUM	2		3
primefaces-6.1.jar: jquery.js	MEDIUM	2		3
shiro-core-1.2.3.jar	HIGH	2	Highest	32
spring-core-3.0.6.RELEASE.jar	CRITICAL	11	Highest	28

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.

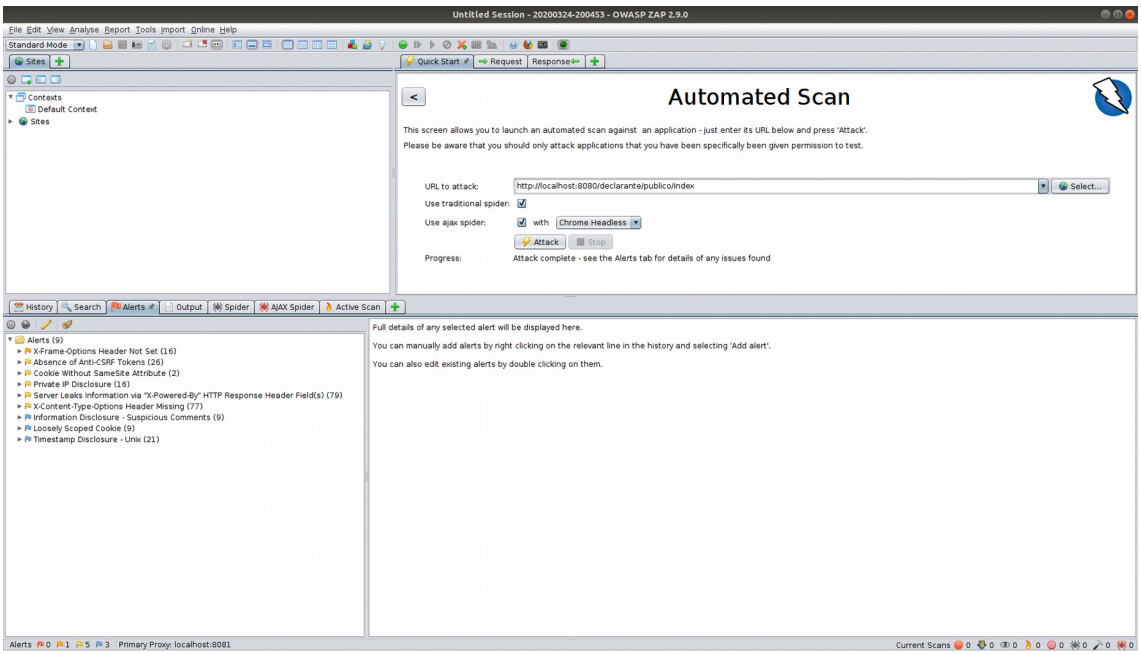

Figura 5: OWASP ZAP - Teste de intrusão - DAT Web Declarante

- 0 vulnerabilidade de severidade alta;
- 1 vulnerabilidade de severidade média;
- 5 vulnerabilidades de baixa média;
- 3 vulnerabilidades a nível informativo;

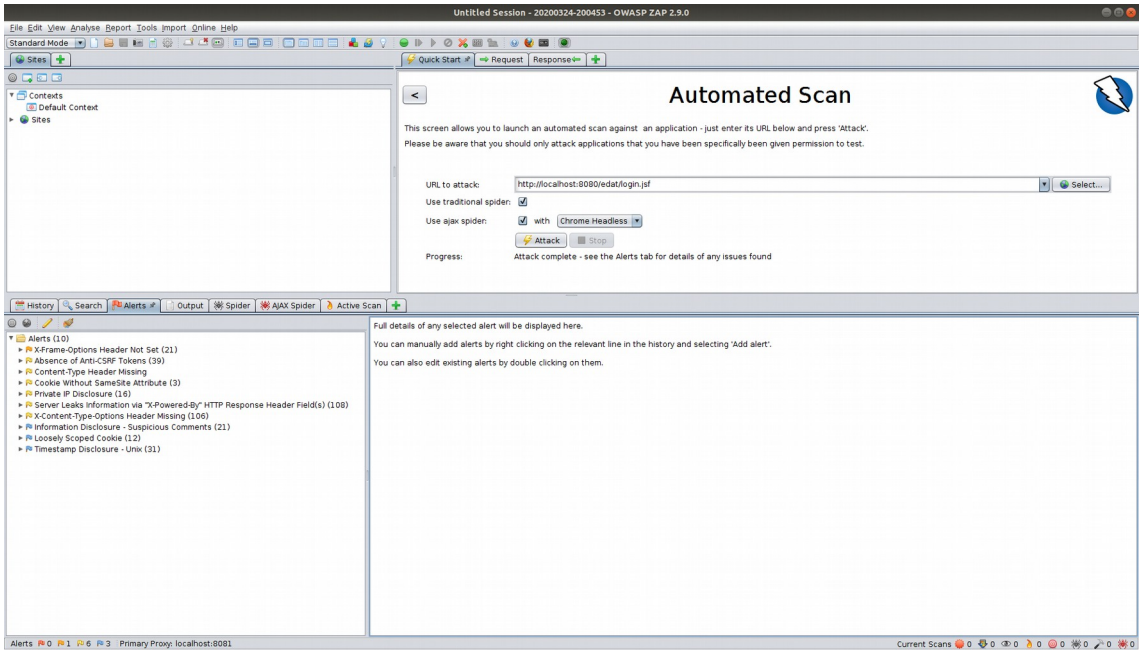

Figura 6: Figura 5: OWASP ZAP - Teste de intrusão - DAT Web Validação

- 0 vulnerabilidade de severidade alta;
- 1 vulnerabilidade de severidade média;
- 6 vulnerabilidades de baixa média;
- 3 vulnerabilidades a nível informativo;

O relatório completo dos testes aplicados estão disponíveisno anexo I deste documento.

4.4 Estrutura do projeto

A nomenclatura utilizada na organização nos pacotes está projetada e disposta de forma intuitiva e coesa. A componentização dos monólitos facilita o entendimento, trazendo consigo diminuição de tempo na curva de aprendizado para novos ingressos a equipe de desenvolvimento do projeto.

Figura 7: DAT Núcleo - Estrutura do projeto

Figura 8: DAT Serviço - Estrutura do projeto

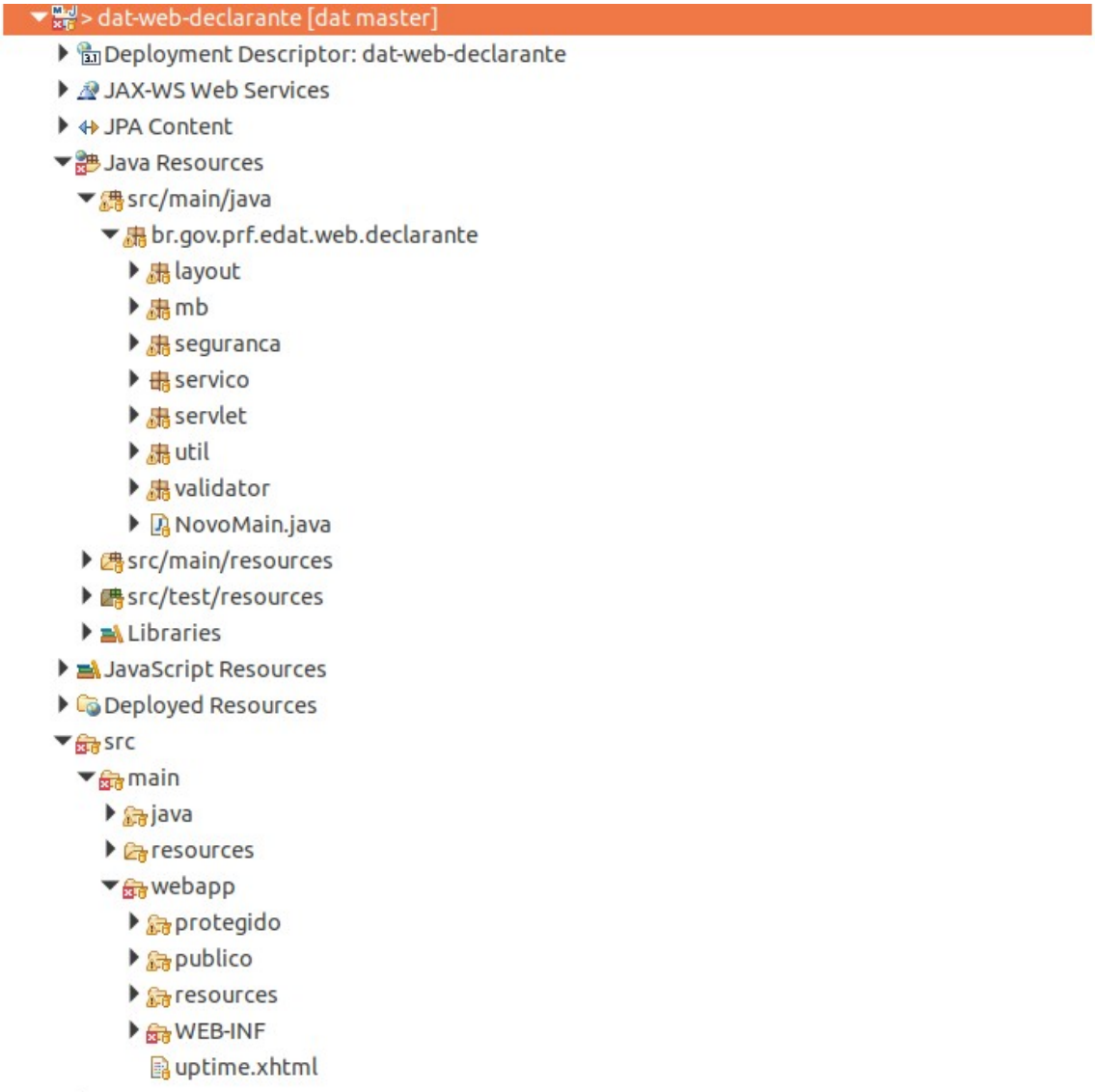

Figura 9: DAT Web Declarante - Estrutura do projeto

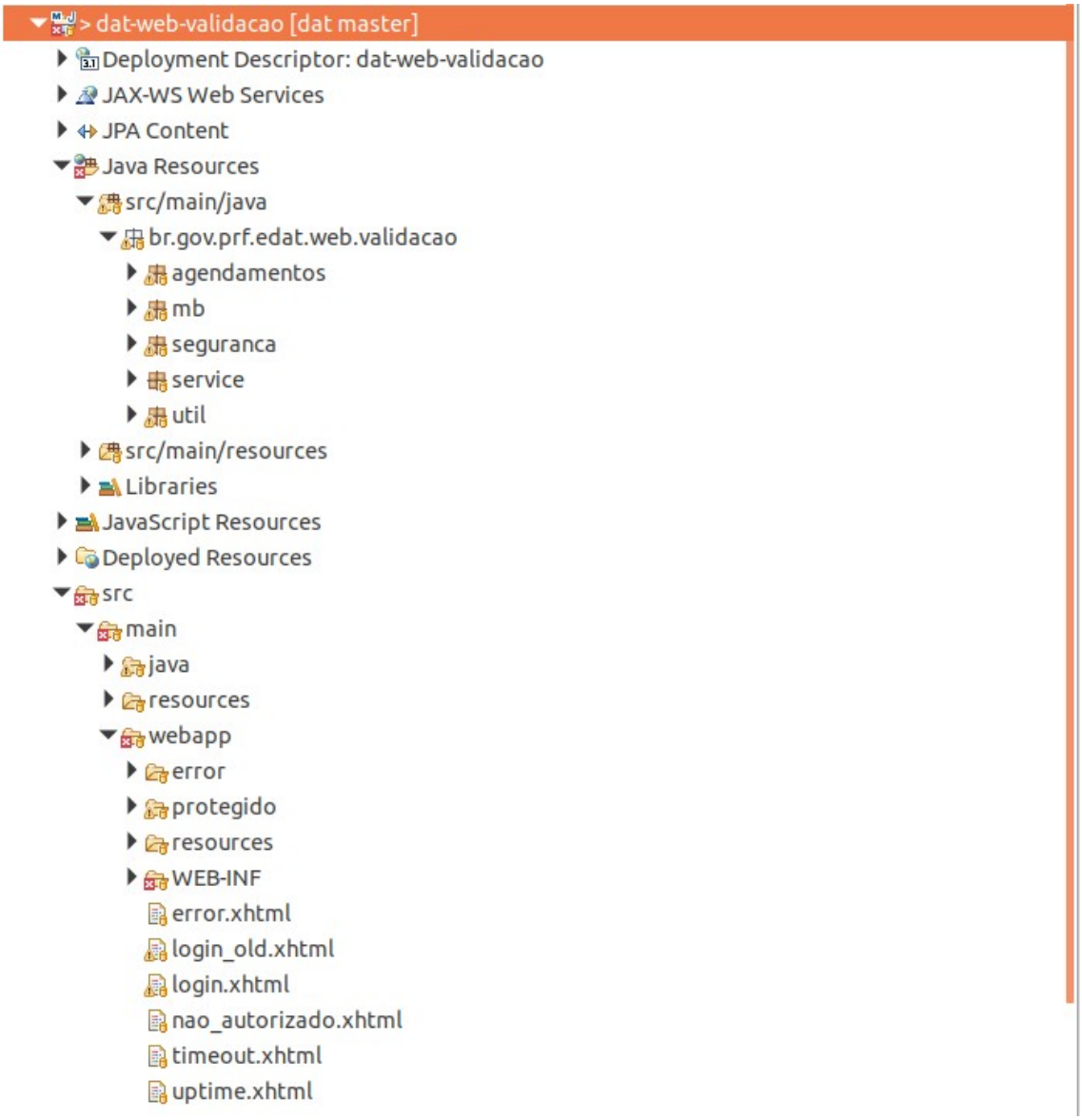

Figura 10: DAT Web Validação - Estrutura do projeto

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 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.

A alta complexidade ciclomática e a falta de artefatos de testes de unidade dificultam o processo de refactoring, a ilustração que seguem demonstram o cenário apontado (OBS: a característica apresentada é utilizada de forma recorrente em diversos momentos do código).

```
@Transactional
public String buscar() {
    this.declaracao = null;
    switch (tipoConsulta) {
        case DOCUMENTO:
            if (!StringUtil.isEmpty(documento)) {
                if (TipoDocumento.CPF.equals(this.tipoDocumento)) {
                    this.lista = dao.findDeclaracaoByCpf(documento);
                } else {
                    this.lista = dao.findDeclaracaoByTipoAndDocumento(tipoDocumento, documento);
                }
            } else {
                mensageiro.addMessage(messageUtil.getMensagem("label_erro", "label_erro_documentoNaoInformado"));
                return null;
            }
            break;
        case PROTOCOLO:
            if (!StringUtil.isEmpty(protocolo)) {
                this.lista = dao.findDeclaracaoByProtocolo(protocolo);
            } else {
                mensageiro.addMessage(messageUtil.getMensagem("label_erro", "label_erro_protocoloNaoInformado"));
                return null;
            }
            break;
        case PLACA:
            if (!StringUtil.isEmpty(placa)) {
                this.lista = dao.findDeclaracaoByPlaca(placa);
            } else {
                mensageiro.addMessage(messageUtil.getMensagem("label_erro", "label_erro_placaNaoInformada"));
                return null;
            }
    }
    if (this.lista.isEmpty()) {
        mensageiro.addMessage(messageUtil.getMensagem("label_erro", "label_erro_declaracaoNaoEncontradaRet"));
    }
    return null;
}
```

Figura 11: Alta complexidade ciclomática – AutenticarImprimirMB.java

Embora haja uma boa segregação de responsabilidade dos pacotes nos 4 projetos que compõe a solução, a nomenclatura das classes não leva nenhum identificador de sua camada/responsabilidade. Este fato dificulta a identificação de competência.

4.6 Confiabilidade

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

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.

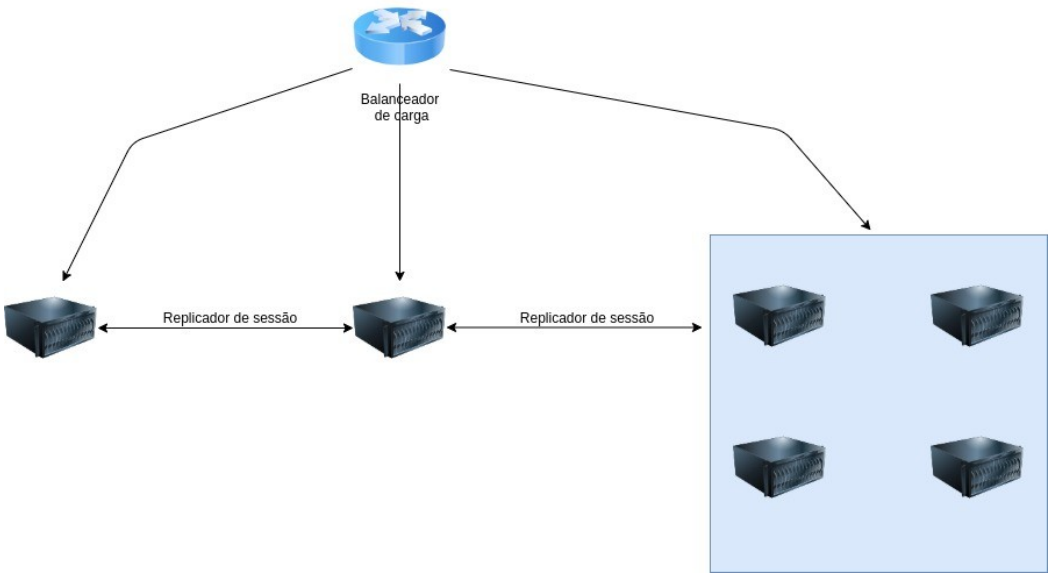

Figura 1: Escalabilidade do monólito

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 intrusãopara 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.