



Ministério da Justiça

**Projeto:** FORPDI

# Nota Técnica

Rev isão	Descrição	Autor	Data
1.0	Construção do documento	Israel Branco	26/03/ 2020

# 1 Sumário

2 Introdução.....	4
3 Apresentação do cenário atual.....	5
3.1 Front-End.....	6
3.2 Back-end.....	7
3.2 Tecnologias utilizadas.....	8
3.3 Modelagem de dados.....	10
4 Análise técnica.....	11
4.1 SonarQube.....	11
4.2 OWASP Dependency Check.....	13
4.3 NPM Audit.....	14
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 FORPDI. 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.

FORPDI é uma plataforma de código aberto disponibilizada sob a licença Apache 2.0, sua plataforma está distribuída e mantida no repositório GitHub (<https://github.com/forpdi/plataforma-for>). Esta plataforma possui vasta documentação disponibilizada, dentre elas capacitação online por vídeo aulas, diagramas de banco de dados, dicionário de dados, manuais de usuário e documentações técnicas do sistema. Vale observar que na data da escrita desta nota técnica todos os links para disponibilização da documentação estão fora do ar, com a exceção dos vídeos de capacitação.

Dentro do Ministério da Justiça, esta plataforma está disponível no repositório <https://gitlab.mj.gov.br/cgsis/plataforma-forpdi> e sua estrutura está fidedigna a versão disponibilizada no GitHub.

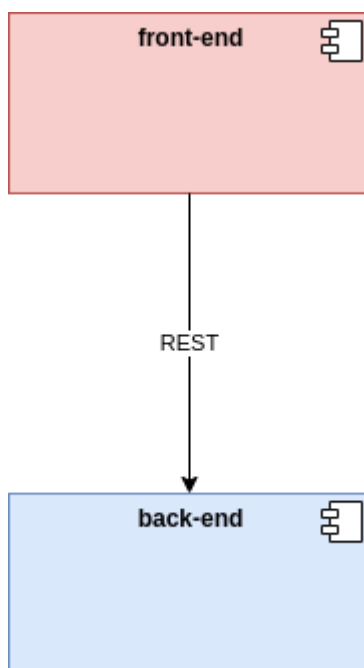
### 3 Apresentação do cenário atual

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

FORPDI 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 React.

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

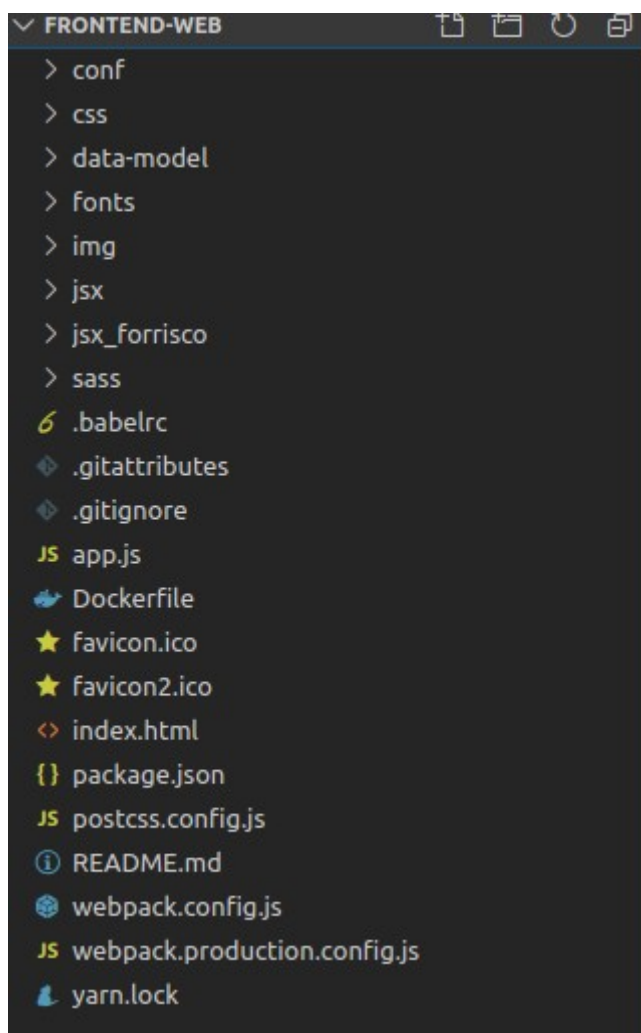


*Figura 1: Diagrama de componentes*

### 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.

O projeto front-end carrega consigo o arquivo de imagem Dockerfile que contempla todas as configurações para sua execução em ambiente de container.



*Figura 2: Organização do projeto front-end*

Vale ressaltar que há estrutura adotada neste projeto suporta o acréscimo de novos módulos de forma orgânica e organizada, é perceptível a coesão do componentes e sua segregação.

### 3.2 Back-end

Este componente trabalha com o microuniverso de cada funcionalidade do sistema. Os pacotes são criados por funcionalidades e em cada pacote está contemplado classes de transição de valores (DTO), classes que efetuam mapeamento ORM (VO), classes que contemplam a exposição da api Rest (Controller) e as classes de persistência. A segregação de responsabilidades aplicada neste projeto é coesa e de fácil entendimento.



*Figura 3: 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.

Os motivadores das escolhas tecnológicas deste projeto podem ser encontradas na seção 4.3.3 do documento FORPDI\_documento dosistema.pdf

([https://gitlab.mj.gov.br/cgsis/plataforma-forpdi/blob/2.0.9/frontend-web/data-model/FORPDI\\_documentacao\\_do\\_sistema.pdf](https://gitlab.mj.gov.br/cgsis/plataforma-forpdi/blob/2.0.9/frontend-web/data-model/FORPDI_documentacao_do_sistema.pdf)).

Nome	Versão	Utilização	Observação
Java	1.8	Linguagem de programação	
VRaptor	4	Framework MVC	
CDI	1.2	Container de IOC	
Hibernate		Framework ORM	
Maven	3.x	Gerenciador de dependências e build	
iText	5.5.10	Biblioteca para criar e manipular documentos com extensão PDF.	
Bootstrap	3.4.1	Biblioteca para desenvolvimento de aplicações WEB	
Quartz		Biblioteca para realização de tarefas agendadas	
SASS	7.1	Extensão do CSS	
ReacrJS	15	Framework JavaScript para criação de itnerfaces gráficas	
MDI	1.5.54	Ícones para utilização no	

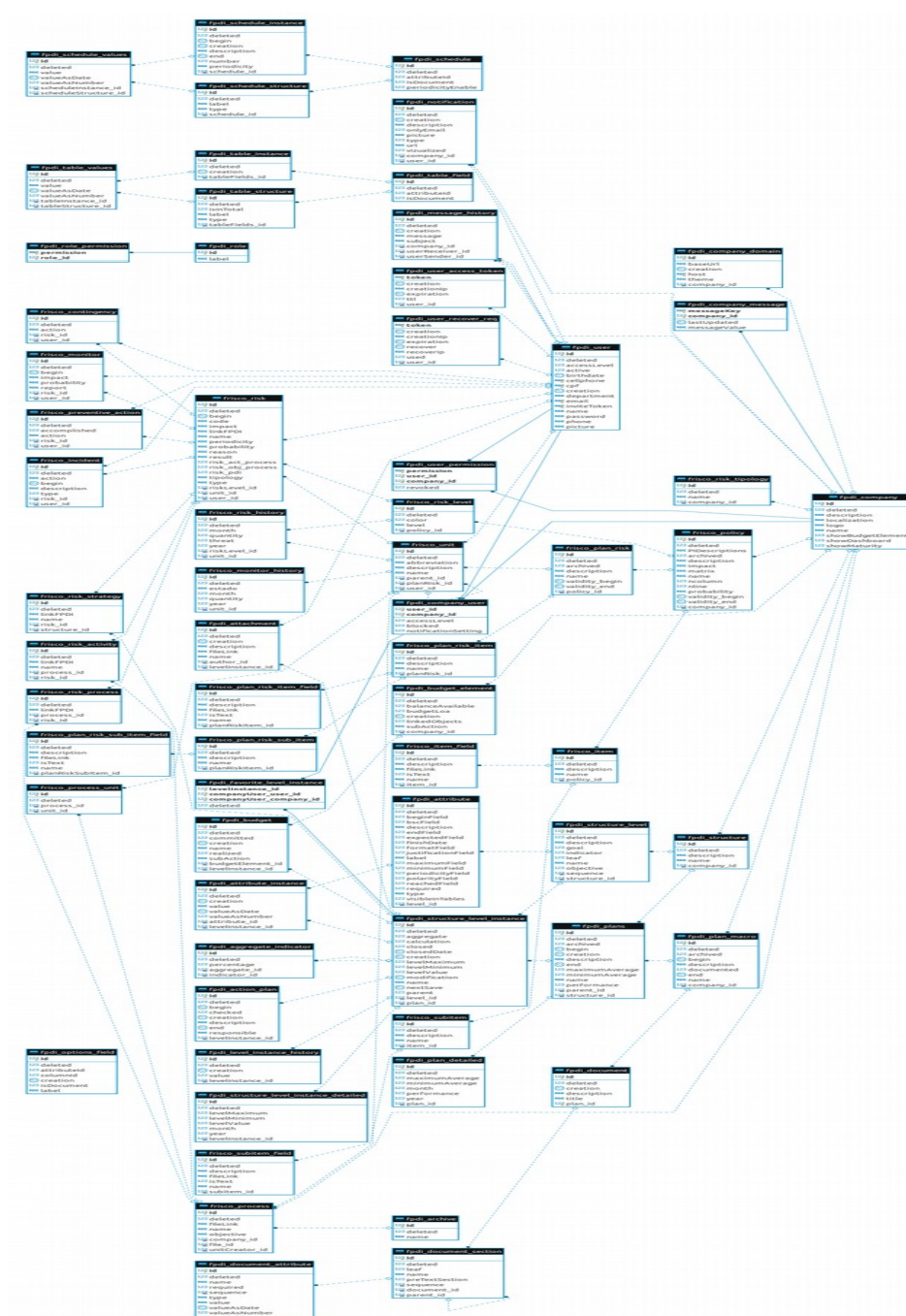


<b>MJ</b>	<b>FORPDI - Nota Técnica</b>	
-----------	------------------------------	--

		design da aplicação	
MySQL	5.7	Servidor de banco de dados	
Wildfly	9.0.2	Servidor de aplicação	

### 3.3 Modelagem de dados

A estrutura de banco de dados utiliza um único schema que agrega estruturas relacionadas ao FORPDI e FORRISCO.



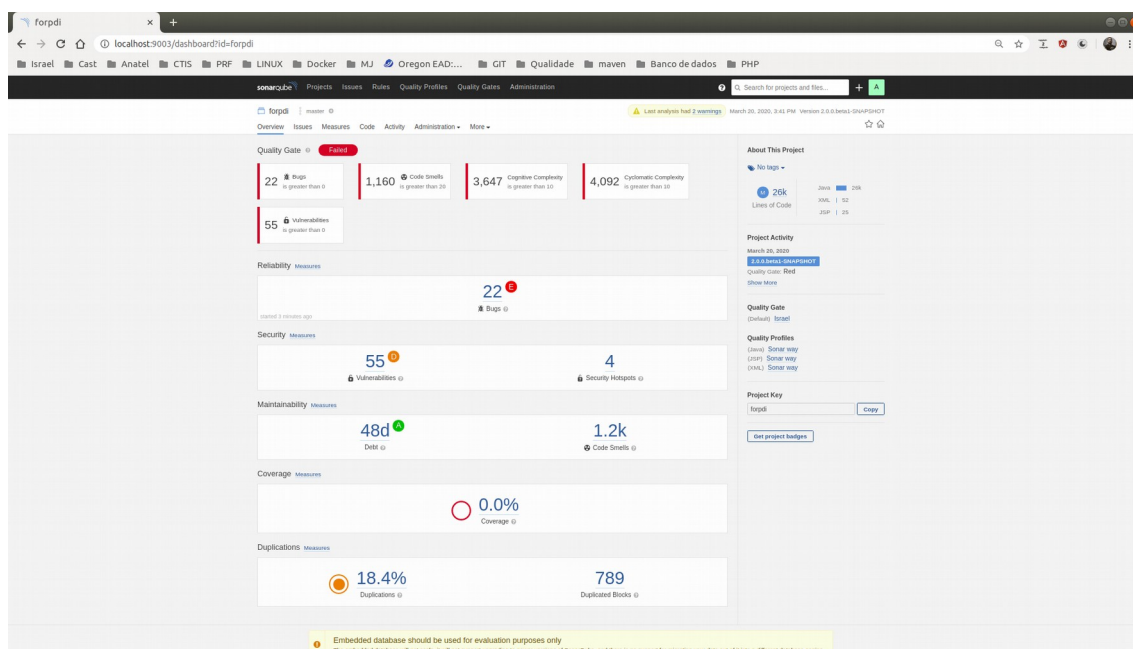
*Figura 4: MER - FORPDI e FORRISCO*

## 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 5: Análise estática de código*

<b>MJ</b>	<b>FORPDI - Nota Técnica</b>	
-----------	------------------------------	--

Para a obtenção dos resultados, fora utilizado o código fonte da tag 2.0.9 sendo esta a última tag gerada até a escrita desta nota técnica:

- 22 bugs;
- 1160 violações de más práticas;
- 3647 violações de complexidade cognitiva (dificuldade de entendimento de código);
- 4092 violações de complexidade ciclomática (complexidade de código);
- 55 vulnerabilidades;
- 18.4% 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.

## 4.2 OWASP Dependency Check

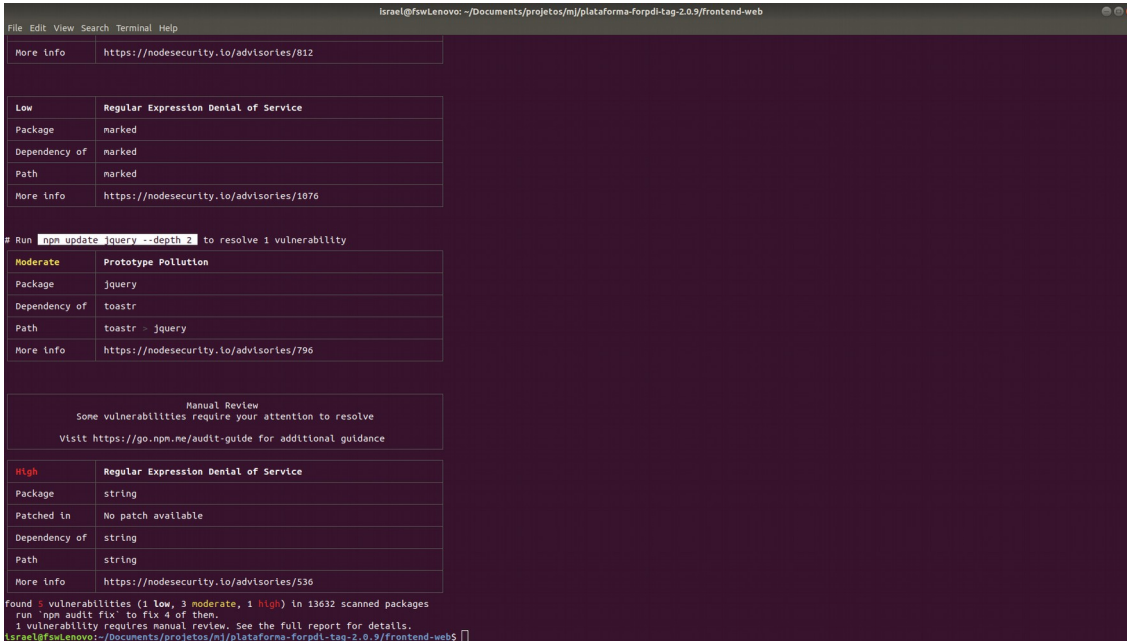
A utilização de bibliotecas de terceiros aumenta substancialmente a produtividade na construção de um software, contudo estas podem trazer consigo vulnerabilidades que afetam diretamente a segurança da aplicação. A ferramenta Dependency Check tem como propósito efetuar análise de vulnerabilidade de dependências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

Dependency	Highest Severity	CVE Count	Confidence	Evidence Count
<a href="#">sms-gateway-1.0.jar</a>	LOW	1	High	30
<a href="#">itextpdf-5.5.10.jar</a>	HIGH	1	High	31
<a href="#">quartz-2.2.2.jar</a>	CRITICAL	1		44
<a href="#">mysql-connector-java-5.1.35.jar</a>	HIGH	4	Highest	38
<a href="#">cdi-api-1.2.jar</a>	MEDIUM	1	Low	36
<a href="#">jstl-1.2.jar</a>	HIGH	1		26
<a href="#">guava-15.0.jar</a>	MEDIUM	1	Highest	20
<a href="#">xstream-1.4.7.jar</a>	HIGH	2	Highest	44
<a href="#">dom4j-1.6.1.jar</a>	HIGH	1	Highest	25
<a href="#">c3p0-0.9.2.1.jar</a>	HIGH	1	Highest	25
<a href="#">poi-3.12.jar</a>	HIGH	4	Highest	29
<a href="#">commons-email-1.3.3.jar</a>	HIGH	2	Highest	38
<a href="#">commons-fileupload-1.3.1.jar</a>	CRITICAL	2	Highest	37

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 5 vulnerabilidades na aplicação/dependências e foram classificadas como: 1 de baixo risco, 3 de riscos moderados e 1 de alto risco.



```

File Edit View Search Terminal Help
Israel@fswLenovo: ~/Documents/projetos/mj/plataforma-forpdi-tag-2.0.9/frontend-web
More Info https://nodesecurity.io/advisories/812

Low Regular Expression Denial of Service
Package marked
Dependency of marked
Path marked
More Info https://nodesecurity.io/advisories/1876

# Run npm update jquery --depth 2 to resolve 1 vulnerability

Moderate Prototype Pollution
Package jquery
Dependency of toastr
Path toastr > jquery
More Info https://nodesecurity.io/advisories/796

Manual Review
Some vulnerabilities require your attention to resolve
Visit https://go.npm.me/audit-guide for additional guidance

High Regular Expression Denial of Service
Package string
Patched in No patch available
Dependency of string
Path string
More Info https://nodesecurity.io/advisories/536

Found 5 vulnerabilities (1 Low, 3 Moderate, 1 High) in 13832 scanned packages
run 'npm audit fix' to fix 4 of them.
1 vulnerability requires manual review. See the full report for details.
Israel@fswLenovo: ~/Documents/projetos/mj/plataforma-forpdi-tag-2.0.9/frontend-web$

```

*Figura 6: 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 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 traz 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 dificulta 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).

```
public List<StructureLevelInstance> listLevelInstancesByResponsible(List<User> users) {
    List<StructureLevelInstance> results = new ArrayList<StructureLevelInstance>();
    if (users.size() > 0) {
        Criteria criteria = this.dao.newCriteria(AttributeInstance.class);
        criteria.createAlias("attribute", "attribute", JoinType.INNER_JOIN);

        Disjunction or = Restrictions.disjunction();
        for (User user : users) {
            or.add(Restrictions.eq("value", String.valueOf(user.getId())));
        }
        criteria.add(or);

        criteria.add(Restrictions.eq("deleted", false));
        criteria.add(Restrictions.eq("attribute.deleted", false));
        criteria.add(Restrictions.eq("attribute.type", ResponsibleField.class.getCanonicalName()))
            .createAlias("levelInstance", "levelInstance", JoinType.INNER_JOIN)
            .add(Restrictions.eq("levelInstance.deleted", false));
        List<AttributeInstance> attrInsts = this.dao.findByCriteria(criteria, AttributeInstance.class);

        for (AttributeInstance attrInst : attrInsts) {
            boolean exists = false;
            for (StructureLevelInstance levelInst : results) {
                if (levelInst.getId() == attrInst.getLevelInstance().getId())
                    exists = true;
            }
            if (!exists)
                results.add(attrInst.getLevelInstance());
        }
    }
    return results;
}
```

Figura 7: Complexidade ciclomática - StructureBS.java

Vale ressaltar há inexistência de classes de serviços na estrutura do componente back-end, toda lógica está sendo realizada nos controladores e subsequentemente há uma submissão as classes de persistência denominadas pelo sufixo BS.

#### **4.4.2 Confiabilidade**

Não há evidências de controle transacional em nenhuma camada da aplicação, a falta deste tratamento fere boas práticas de desenvolvimento de aplicações e a falta do mesmo não 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. Durante o processo de análise de código fonte, não foram encontradas evidências que demonstrem impactos em performance da aplicação.

#### **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.

## 6 Conclusão

A aplicação front-end e back-end apresentam boa estruturação em sua construção o que facilita a sua manutenção corretiva/evolutiva.

Há boa documentação disponível sob todos os aspectos da aplicação, em destaque aos treinamentos de usuário que são contemplados inúmeras video aulas.

A utilização de containers Docker para a distribuição das aplicações front e back-end facilitam a implantação da ferramenta sendo que sua utilização esta ainda mais facilitada pelas instruções disponibilizadas no arquivo Readme.md.

A ausência de classes de serviço (service) aumentam a competência das classes Controladoras, consequentemente aumentam sua responsabilidade e complexidade ciclomática.

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