




Ministério da Justiça

Projeto: ORCRIM

Nota Técnica

MJ	ORCRIM – Nota Técnica	
-----------	------------------------------	--

Revisão	Descrição	Autor	Data
1.0	Construção do documento	Israel Branco	26/02/2020

1 Sumário

2	Introdução.....	4
3	Apresentação do cenário atual.....	5
3.1	Front-End.....	6
3.2	Orcrim-Integration.....	7
3.3	Orcrim-Util.....	7
3.4	Orcrim-Entity.....	7
3.4	Orcrim-Presentation.....	7
3.2	Tecnologias utilizadas.....	8
3.3	Modelagem de dados.....	9
4	Análise técnica.....	12
4.1	SonarQube.....	12
4.2	OWASP Dependency Check.....	14
4.3	NPM Audit.....	15
4.4	Análise sobre os resultados.....	16
4.4.1	Manutenibilidade de código.....	16
4.4.2	Confiabilidade.....	17
4.4.3	Performance e estabilidade.....	17
4.4.3	Escalabilidade.....	17
4.4.3	UX – User experience.....	18
5	Recomendações.....	20
6	Conclusão.....	22

2 Introdução

Este documento visa reportar o resultado da análise efetuada na aplicação ORCRIM. Para este estudo foram desconsiderados todo o contexto comercial ao qual a ferramenta está inserida, também foram desconsideradas o ambiente ao qual a ferramenta está 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.



3 Apresentação do cenário atual

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

O ORCRIM 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.

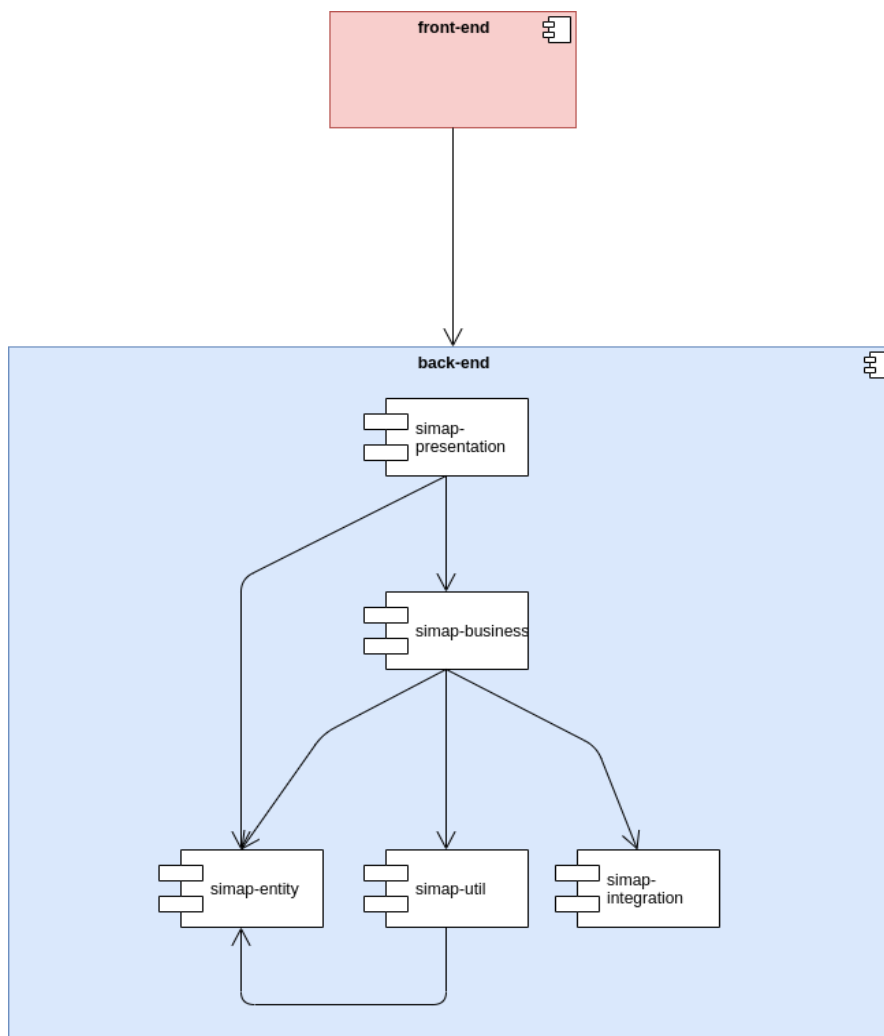


Figura 1: Diagrama de componentes

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.

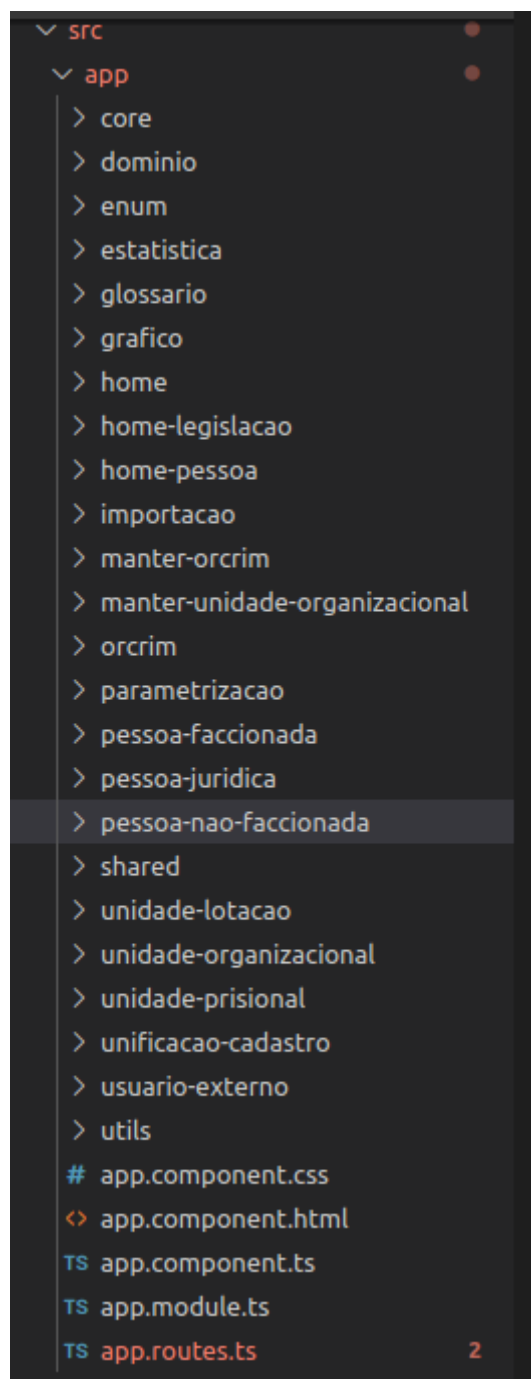


Figura 1: Organização do projeto front-end

Vale ressaltar que há uma grande gama de funcionalidades no sistema e quando há um crescimento não orgânico a organização do projeto tende a ser prejudicada, não sendo este o caso do ORCRIM. É perceptível a coesão do componentes e sua segregação.

3.2 Orcrim-Integration

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

3.3 Orcrim-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 Orcrim-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 Orcrim-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 designe 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	2.4.1	Biblioteca para serializar/deserializar quando se trabalha com Pojos/Json	
Junit	4.11	Utilizado para a criação de testes automatizados.	
Jboss Arquillian	1.1.5	Framework para realização testes de integração.	
Apache POI	4.0.1	Componente utilizado para manipular documentos baseado em formato MS Office.	
Keycloak	3.1.0	Framework para gerenciamento de identidades e acesso.	
Oracle		Banco de dados relacional	

3.3 Modelagem de dados

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

OBS:as tabelas deste schema estão sendo relacionadas no schema Orcrim.

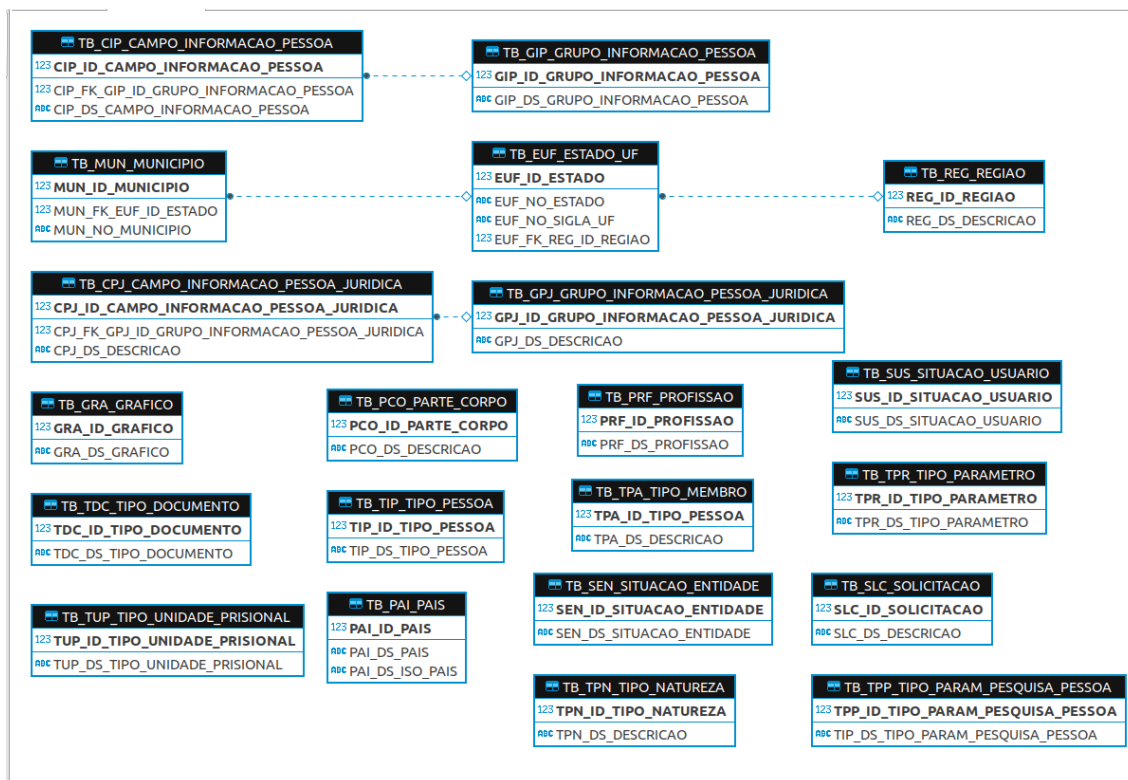


Figura 3: Modelo relacional Schema Apoio

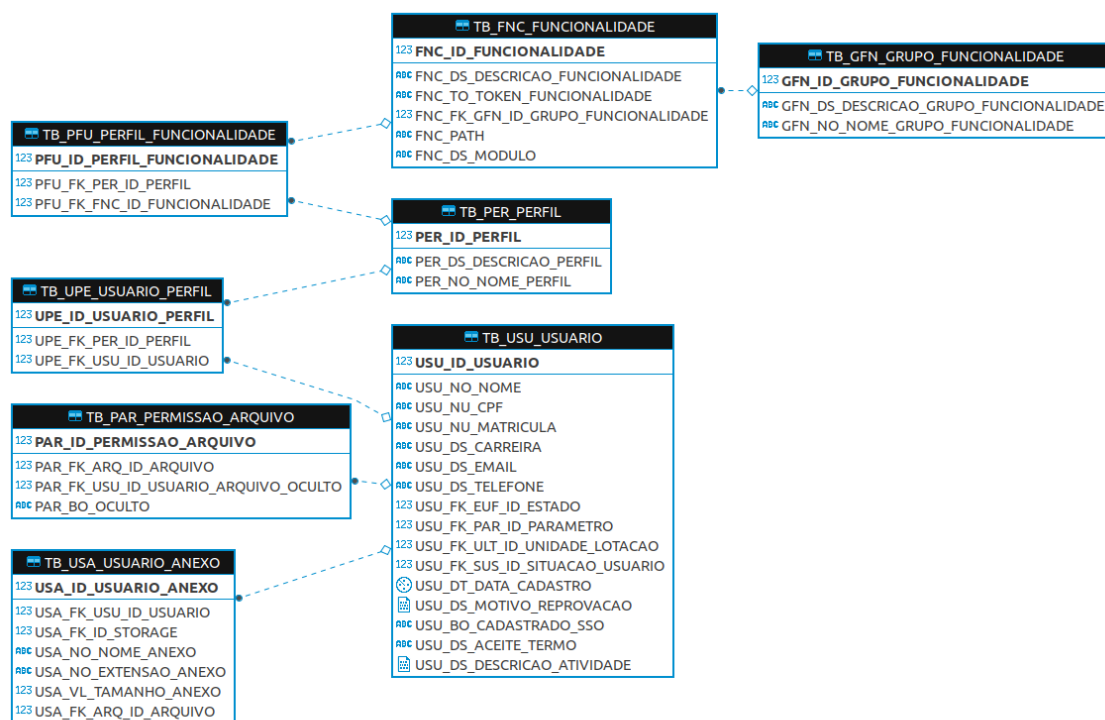
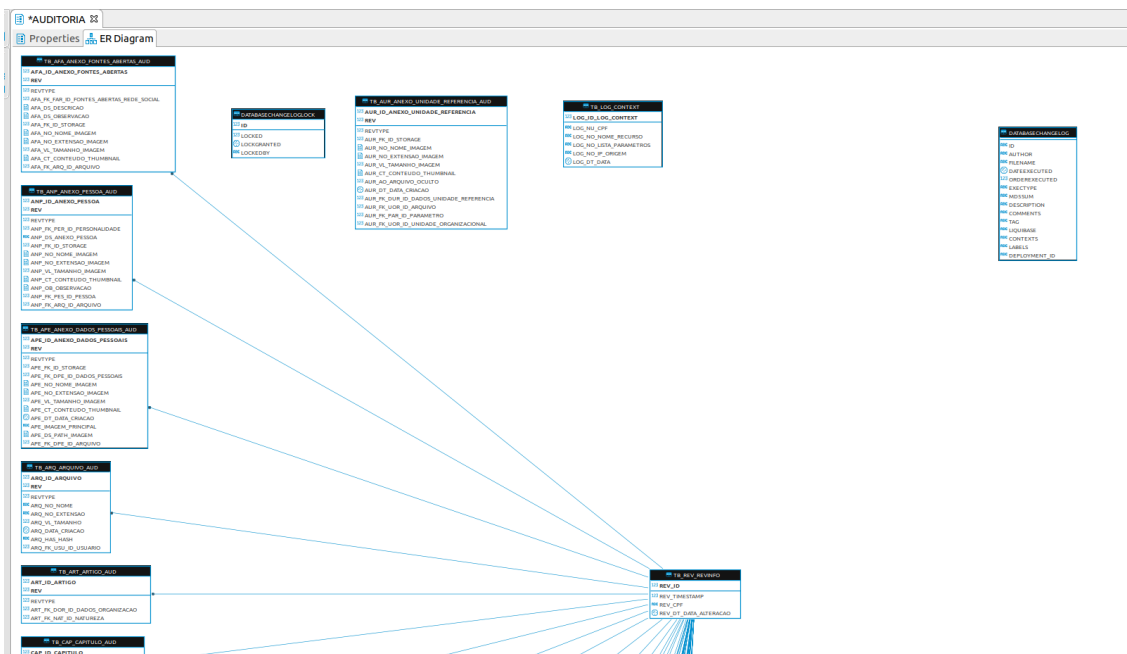


Figura 4: Schema Seq

OBS: O schema auditoria representa a auditoria das tabelas do schema orcrim, sua característica de esta pautada segundo a utilização do framework Hibernate-Envers.





OBS: o schema Orcrim possui 80 tabelas com um alto nível de relacionamento entre elas. A tabela tb_arq_arquivo deste schema é a única que não apresenta relacionamento, contudo esta possui uma relação conceitual com a tabela tb_usa_usuario_anexo do schema seq.



Figura 5: Schema orcrim

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:

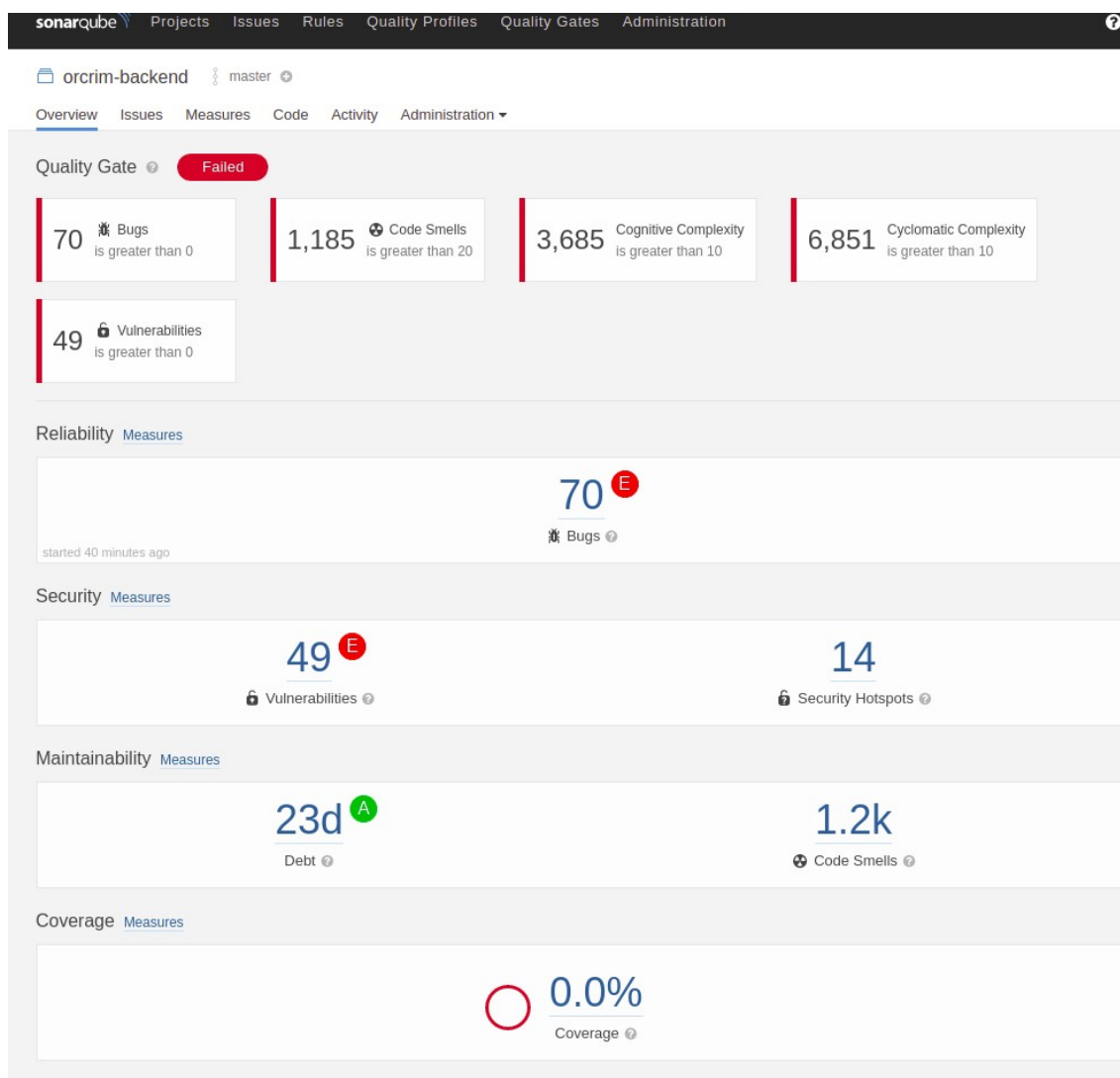


Figura 6: Análise estática de código

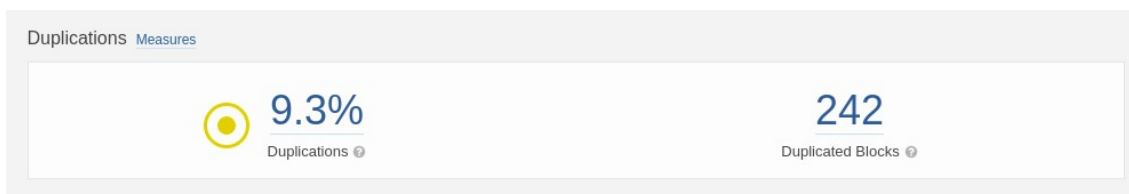


Figura 7: Análise estática de código

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

- 70 bugs;
- 1185 violações de más práticas;
- 3685 violações de complexidade cognitiva (dificuldade de entendimento de código);
- 6851 violações de complexidade ciclomática (complexidade de código);
- 49 vulnerabilidades;
- 9.3% 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.

```

Israel@fswLenovo: ~/Documents/projetos/mj/orcrlm/orcrlm-backend/codigos_fonle
File Edit View Search Terminal Tabs Help

Results :
Tests in error:
  ParametroRestTest.br.gov.mj.test.business.ParametroRestTest » Runtime Could no...
  PaisRestTest.br.gov.mj.test.business.PaisRestTest » Runtime Could not invoke d...
  TituloRestTest.br.gov.mj.test.business.TituloRestTest » Runtime Could not invo...
  MunicipioRestTest.br.gov.mj.test.business.MunicipioRestTest » Runtime Could no...
  TipoParametroRestTest.br.gov.mj.test.business.TipoParametroRestTest » Runtime ...
  UfRestTest.br.gov.mj.test.business.UfRestTest » Runtime Could not invoke depla...
  TipoUnidadePrisionalRestTest.br.gov.mj.test.business.TipoUnidadePrisionalRestTest » Runtime
  SituacaoEntidadeRestTest.br.gov.mj.test.business.SituacaoEntidadeRestTest » Runtime
  GrupoRestTest.br.gov.mj.test.business.GrupoRestTest » Runtime Could not invoke...
  CapituloRestTest.br.gov.mj.test.business.CapituloRestTest » Runtime Could not ...
  NaturezaRestTest.br.gov.mj.test.business.NaturezaRestTest » Runtime Could not ...
  OrganizacaoCriminosaRestTest.br.gov.mj.test.business.OrganizacaoCriminosaRestTest » Runtime
  LegislacaoRestTest.br.gov.mj.test.business.LegislacaoRestTest » Runtime Could ...
  UnidadeLotacaoRestTest.br.gov.mj.test.business.UnidadeLotacaoRestTest » Runtime
  UnidadePrisionalRestTest.br.gov.mj.test.business.UnidadePrisionalRestTest » Runtime

Tests run: 15, Failures: 0, Errors: 15, Skipped: 0

[INFO] -----
[INFO] Reactor Summary for orcrlm-backend 1.1.7:
[INFO]
[INFO] orcrlm-backend ..... SUCCESS [ 0.413 s]
[INFO] orcrlm-entlty ..... SUCCESS [ 3.949 s]
[INFO] orcrlm-util ..... SUCCESS [ 0.114 s]
[INFO] orcrlm-integration ..... SUCCESS [ 0.395 s]
[INFO] orcrlm-business ..... SUCCESS [ 3.783 s]
[INFO] orcrlm-presentation ..... FAILURE [04:50 min]
[INFO] orcrlm-application ..... SKIPPED
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 04:59 min
[INFO] Finished at: 2020-02-20T17:21:49-03:00
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.17:test (default-test) on
project orcrlm-presentation: There are test failures.
[ERROR]
[ERROR] Please refer to /home/Israel/Documents/projetos/mj/orcrlm/orcrlm-backend/codigos_fonle/presentation
on/target/surefire-reports for the individual test results.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
[ERROR]
[ERROR] After correcting the problems, you can resume the build with the command
[ERROR] mvn <args> -rf :orcrlm-presentation
Israel@fswLenovo: ~/Documents/projetos/mj/orcrlm/orcrlm-backend/codigos_fonle

```

Figura 8: Execução de processo de build com execução de testes de unidade

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
Orcrim-application				
commons-beanutils-1.8.3.jar	HIGH	2	Highest	35
commons-compress-1.18.jar	HIGH	1	Highest	45
dom4j-1.6.1.jar	HIGH	1	Highest	25
jackson-databind-2.9.5.jar	CRITICAL	24	Highest	41
keycloak-core-5.0.0.jar	CRITICAL	8	Highest	39
log4j-1.2.16.jar	CRITICAL	1	Highest	29
nextcloud-api-11.0.2.jar	MEDIUM	3	Highest	24
orcrim-presentation-1.1.7.war: jackson-databind-2.4.1	CRITICAL	23	Highest	37
poi-4.0.1.jar	MEDIUM	1	Highest	29
postgresql-9.0-801.jdbc4.jar	CRITICAL	3	Highest	18
Orcrim-business				
keycloak-core-3.4.0.Final.jar	CRITICAL	12	Highest	36
bcprov-jdk15on-1.56.jar	CRITICAL	3	Low	47
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
resteasy-jaxrs-3.0.10.Final.jar	HIGH	2		21
poi-4.0.1.jar	MEDIUM	1	Highest	29
commons-compress-1.18.jar	HIGH	1	Highest	45
commons-beanutils-1.8.3.jar	HIGH	2	Highest	35
jackson-databind-2.9.5.jar	CRITICAL	24	Highest	41
nextcloud-api-11.0.2.jar	MEDIUM	3	Highest	24
Orcrim-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	23	Highest	40
dom4j-1.6.1.jar	HIGH	1	Highest	25
Orcrim-integration				
keycloak-core-5.0.0.jar	CRITICAL	8	Highest	39
jackson-databind-2.9.5.jar	CRITICAL	24	Highest	41
resteasy-client-3.1.0.Final.jar	HIGH	1	Highest	36
resteasy-jaxrs-3.1.0.Final.jar	HIGH	2	Highest	32
Orcrim-application				
log4j-1.2.16.jar	CRITICAL	1	Highest	29
postgresql-9.0-801.jdbc4.jar	CRITICAL	3	Highest	18
poi-4.0.1.jar	MEDIUM	1	Highest	29
commons-compress-1.18.jar	HIGH	1	Highest	45
commons-beanutils-1.8.3.jar	HIGH	2	Highest	35
nextcloud-api-11.0.2.jar	MEDIUM	3	Highest	24
dom4j-1.6.1.jar	HIGH	1	Highest	25
jackson-databind-2.4.1.jar	CRITICAL	23	Highest	38
keycloak-admin-client-5.0.0.jar	CRITICAL	8	Highest	35
keycloak-core-3.4.0.Final.jar	CRITICAL	12	Highest	36
bcprov-jdk15on-1.56.jar	CRITICAL	3	Low	47
resteasy-jaxrs-3.0.10.Final.jar	HIGH	2		21
Orcrim-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	23	Highest	40

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

Há certa recorrência no relatório de vulnerabilidade de dependências entre os componentes do sistema, estas estão correlacionadas a arquitetura de referencia e necessitam ser revistas.

4.3 NPM Audit

Após realizar a instalação dos módulos de dependências gerenciados pelo NodeJS (npm install), foram reportados existências de 52 vulnerabilidades na aplicação/dependências e foram classificadas como: 17 são de baixo risco, 11 de riscos moderados, 23 são alto risco e 1 são críticos.

```
added 1368 packages from 1253 contributors and audited 9820 packages in 22.095s
found 52 vulnerabilities (17 low, 11 moderate, 23 high, 1 critical)
```

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 6 vulnerabilidades que requerem intervenção manual.

```
added 4 packages from 6 contributors, removed 2 packages and updated 13 packages in 9.787s
fixed 10 of 52 vulnerabilities in 9820 scanned packages
6 vulnerabilities required manual review and could not be updated
3 package updates for 36 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

```
updated 1 package and audited 9820 packages in 8.04s
found 43 vulnerabilities (16 low, 7 moderate, 19 high, 1 critical)
```

Figura 11: Resultado após execução

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

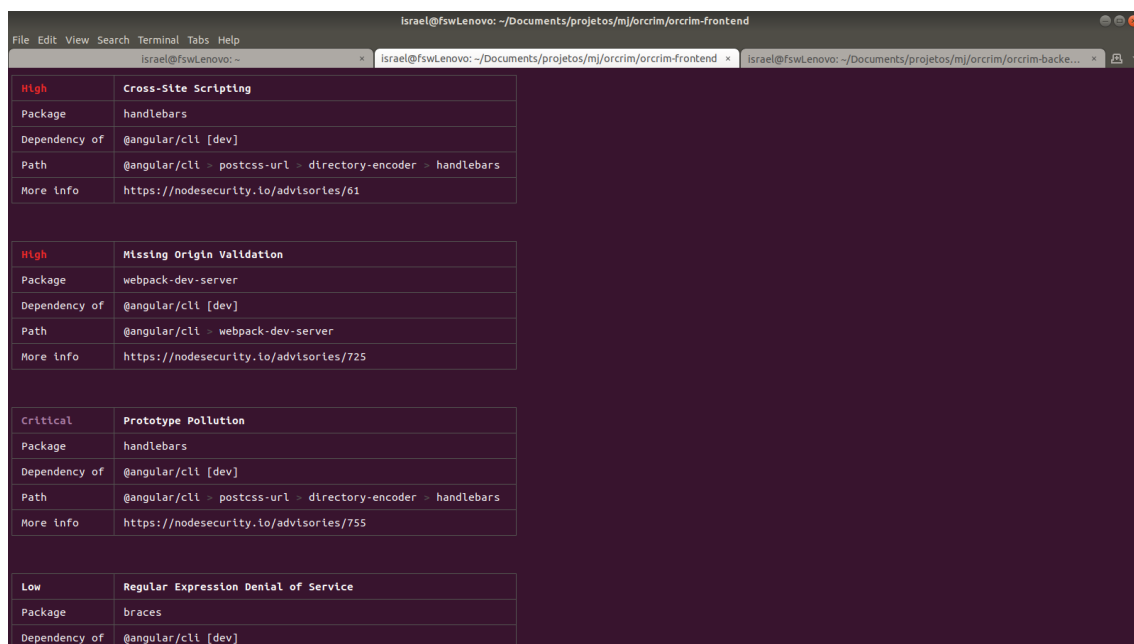


Figura 12: Execução do comando npm audit

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).

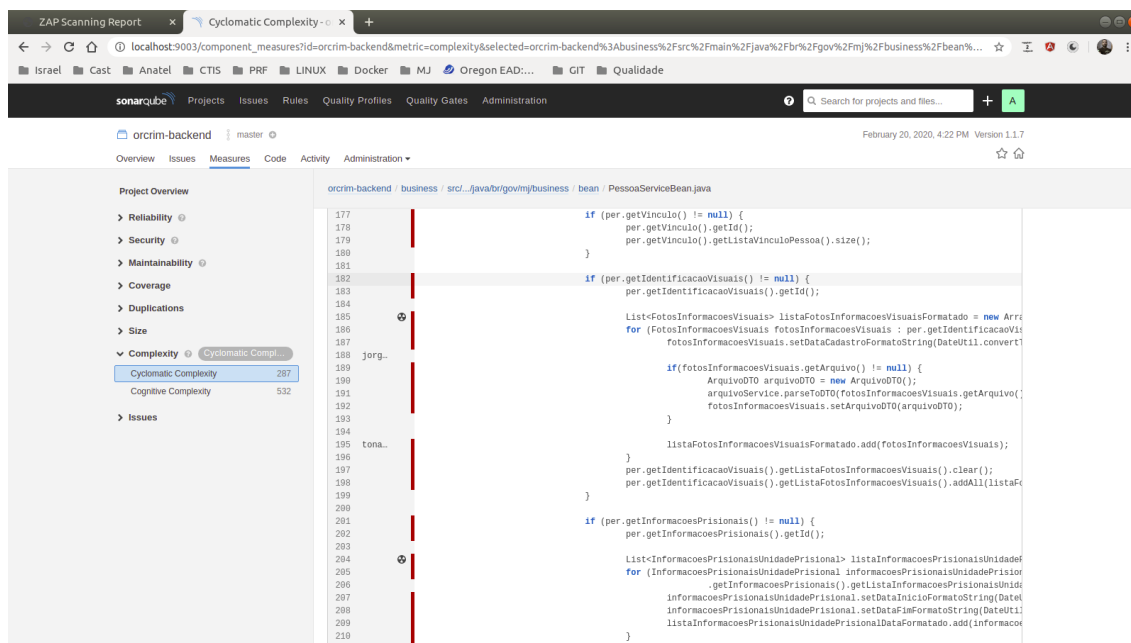


Figura 13: Demonstração de alta complexidade ciclomática - Classe PessoaServiceBean

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

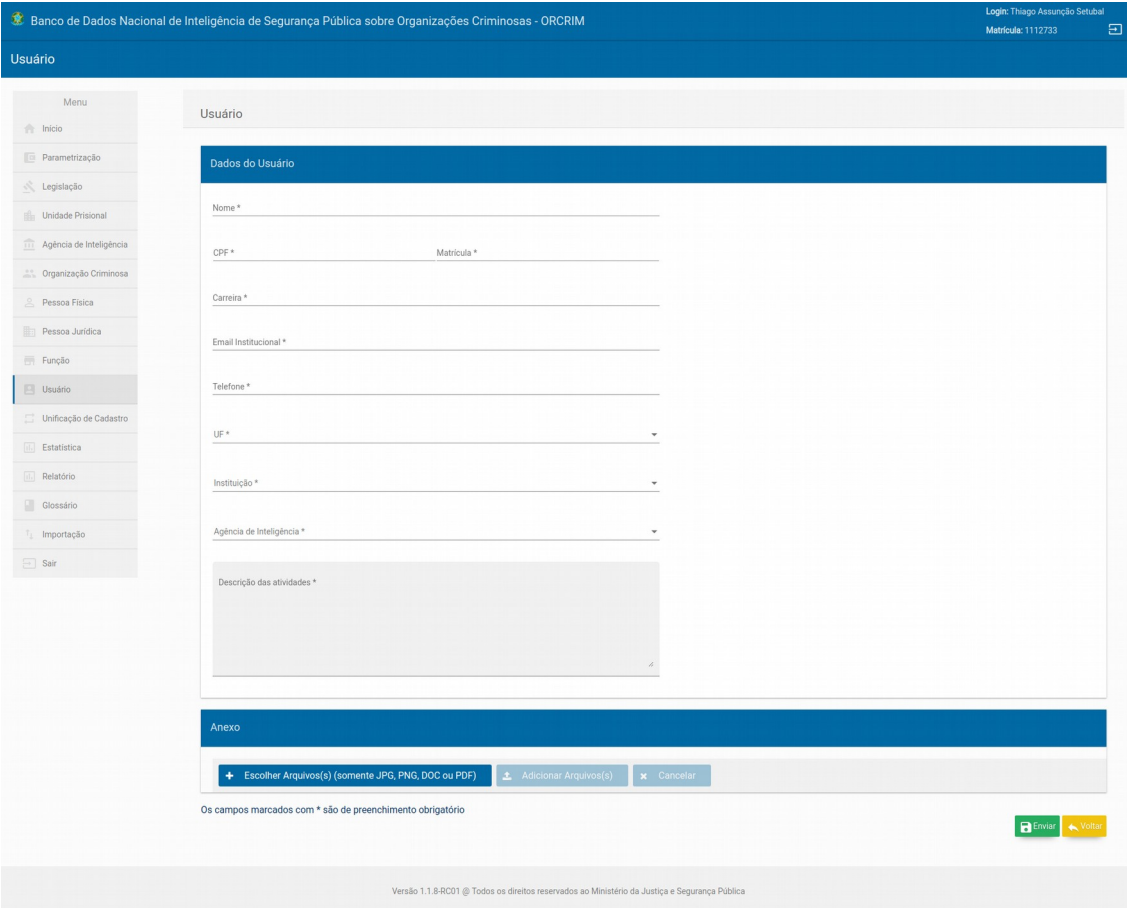
A arquitetura baseada em micro serviços e o comportamento sem estado (stateless) da aplicação back-end promove a esta uma boa capacidade de escalonamento na horizontal com a utilização de cluster e balanceadores de carga.

Esta arquitetura além de promover ambiente escalonável, favorece a utilização de ambientes redundantes com maior

probabilidade de tolerância a falhas.

4.4.3 UX - User experience

Durante a análise de dependência do projeto front-end foi notado a presença da dependência @angular/material versão 2 beta e da dependência PrimeNG versão 4.3. Ambas dependências objetivam trazer ao projeto a utilização de componentes visuais reutilizáveis, contudo a utilização de ambas em um mesmo projeto pode trazer (quando não devidamente tratado) dualidade nos padrões de componentes de interface, trazendo ao usuário uma experiência negativa dado a falta de padronização de telas e de seus componentes.



Banco de Dados Nacional de Inteligência de Segurança Pública sobre Organizações Criminosas - ORCRIM

Usuário: Thiago Assunção Setubal
Matrícula: 1112733

Usuário

Dados do Usuário

Nome *

CPF * Matrícula *

Carreira *

Email Institucional *

Telefone *

UF *

Instituição *

Agência de Inteligência *

Descrição das atividades *

Anexo

+ Escolher Arquivos(s) (somente JPG, PNG, DOC ou PDF) | Adicionar Arquivos(s) | Cancelar

Os campos marcados com * são de preenchimento obrigatório

Entrar Voltar

Versão 1.1.8-RC01 © Todos os direitos reservados ao Ministério da Justiça e Segurança Pública

Figura 14: Formulário com componentes Angular Material

Banco de Dados Nacional de Inteligência de Segurança Pública sobre Organizações Criminosas - ORCRIM

 Login: Thiago Assunção Setubal
 Matrícula: 1112733

Cadastrar Pessoa Física

Dados Pessoais

Voltar

Dados Pessoais

☐ Personalidade Principal ☐ Óbito

Pai:

Tipo de Pessoa:

Mãe:

Nome:

Nacionalidade:

Nome Social (Se aplicável) - Conforme Decreto nº 8.727, de abril de 2016:

Naturalidade - UF:

Sexo:

Data de Nascimento:

Status:

Status:

Alcunha

Alcunha:

Data:

Fotos de Rosto

Documentos

Tipo de Documento:

Profissão

Profissão:

Referência - Dados Pessoais

Fonte:

Observação:

Versão 1.1.8-RC01 © Todos os direitos reservados ao Ministério da Justiça e Segurança Pública

Figura 15: Formulário com componentes PrimeNG



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. Ainda sobre as dependências, recomenda-se que seja removido as dependências que não estão sendo utilizadas no projeto a exemplo o driver jdbc do PostgreSQL, uma vez que neste projeto utiliza-se o SGBD Oracle.

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, differential loading module e novas features do Angular Material.

Independentemente da versão utilizada no framework Angular, é altamente recomendado a manutenção de somente uma extensão de componentes seja angular material (versão superior a 2 Beta) ou primeng.

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.

Dado complexidade relacional dos schemas utilizados nesta estrutura, é comum a perda de performance nas consultas dado a quantidade de joins que podem ser utilizadas, algumas estratégias podem ser utilizadas para acelerar as consultas, sendo elas:

- Oracle Database Flash Cache (utilizado quando a memória RAM do servidor for insuficiente para manutenção de Buffer Cache);
- Hibernate cache de 2 nível (necessário conhecer os principais processos da aplicação para sua devida utilização);
- Utilizar ambiente Oracle RAC para promover ambiente de alta disponibilidade e escalabilidade no SGBD;
- Quando couber a sua utilização para segregação de modelos transacionais e OLAP para a construção de relatórios e BI;

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.