Departamento de Polícia Rodoviária Federal

Projeto: Alerta Brasil - Central

Absorção de Sistema

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

1 Sumário

2 Considerações iniciais

O presente documento tem por objetivo a verificação do processo de absorção de tecnologia do projeto Alerta Brasil Central. 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 ecossistema Alerta Brasil tem por objetivo a análise de dados dos veículos por intermédio da interceptação de imagens de veículos que trafegam nas rodovias federais. Esta análise consiste em fornecer informações estratégicas que auxiliam os policiais rodoviários federais na tomada de decisão quanto ao critério de abordagem dos veículos.

Chamado aqui de ecossistema, o Alerta Brasil possui diversos módulos em sua composição contudo o modulo em questão denominado aqui como Alerta Brasil Central referenciado daqui por diante como AB Central, alvo desta análise, tem por objetivo a

manutenção de informações base para o funcionamento deste ecossistema e concentra as informações gerenciais de todo o fluxo de processamento utilizado neste ecossistema.

O AB Central possui diversas funcionalidades em sua composição que com o passar do tempo foram sendo descontinuadas e/ou substituídas por sistemas especialistas. No momento da escrita deste documento não foi possível levantar e categorizar as funcionalidades existentes neste módulo, contudo, trata-se de uma ferramenta de missão crítica utilizada pela área de inteligência do Departamento Rodoviário Federal e demais inspetores lotados nos diversos postos rodoviários espalhados pelo Brasil.

3 Apresentação do cenário atual

O modulo AB Central foi construído para trabalhar em ambiente web com tecnologia Java, sua arquitetura está dotada como um monólito (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).

Há uma dificuldade de mensurar todas as dependências necessárias para a utilização deste módulo em ambiente produtivo, a documentação de arquitetura existente data de 07/2017 em sua última atualização e muitos dos recursos listados encontram-se obsoletos.

O diagrama a seguir representa a composição do ecossistema Alerta Brasil em alto nível, mesclando o cenário apresentado durante reunião de apresentação da solução (em azul) e o cenário atual visto em análise de código fonte do sistema AB Central (em vermelho). No momento da escrita deste documento, a proposição destacada em vermelho encontra-se obsoleta conforme repasse de informações da equipe de desenvolvimento do projeto (área de segurança da DPRF).

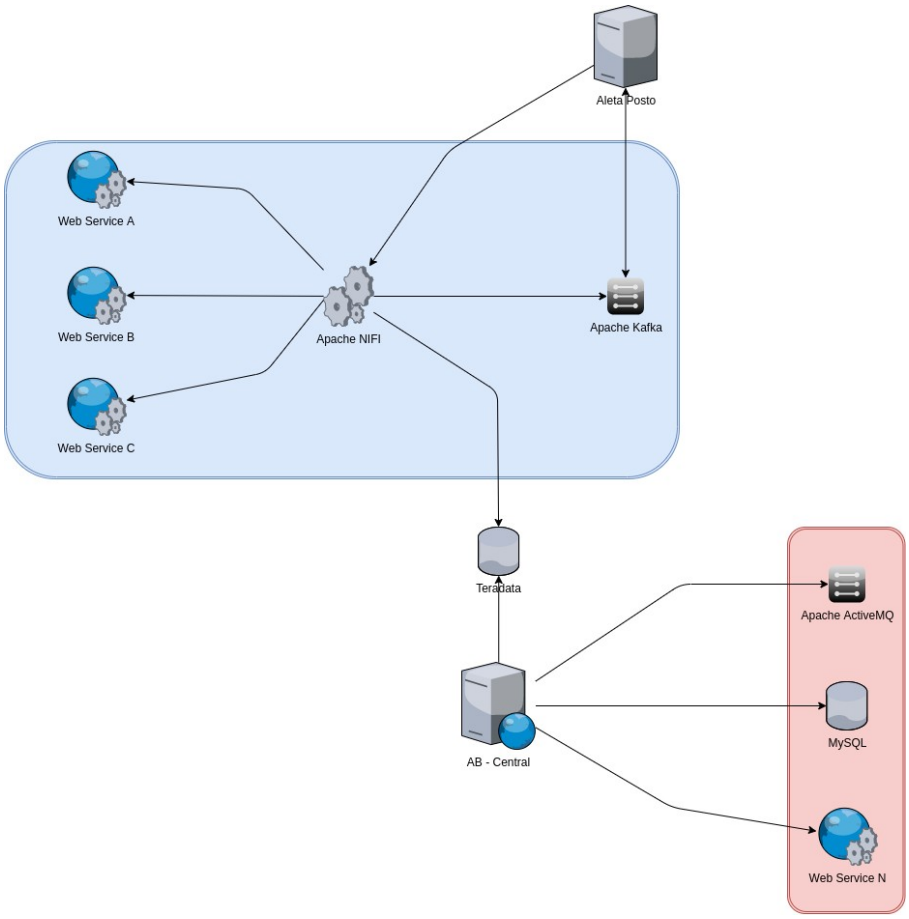

Figura 1: AB - Visão geral

Os diagramas encontrado na documentação de referência da arquitetura não representam o estado atual da solução, contudo trazem consigo informações de grande valia para a análise das tecnologias utilizadas na construção deste projeto.

3.1 Documentação existente

Os documentos entregues no momento desta análise encontram-se em sua maioria desatualizados com a proposição atual da solução, a baixo.

Código fonte: Disponibilidade completa no repositório SVN https://www170.prf.gov.br/svn/repositorios/alertabrasil/sistema_monitoramento/trunk/03-Implementacao/C%c3%b3digo%20Fonte/;

Documentação de requisitos: Não é possível determinar se os documentos disponibilizados neste repositório contemplam a versão atual da solução https://www170.prf.gov.br/svn/repositorios/alertabrasil/sistema_monitoramento/trunk/02-Requisitos/;

Documentação de implantação: Não é possível determinar se os documentos disponibilizados neste repositório contemplam a versão atual da solução https://www170.prf.gov.br/svn/repositorios/alertabrasil/sistema_monitoramento/trunk/04-Implantacao/;

Documentação para criação de ambiente: Documento reflete a preparação para o desenvolvimento local https://www170.prf.gov.br/svn/repositorios/alertabrasil/sistema_monitoramento/trunk/08-Ambiente/;

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 tecnologia, não sendo possível assim estimar os recursos de hardware necessários para a operação da solução.

3.2 Principais bibliotecas e frameworks

- Bibliotecas para conversão para formato de arquivo PDF e relatórios

  - iText 5.3.0
  - Jettison 1.3.4
  - jTidy r938
  - Flying-saucer
  - JasperReports 4.1.2
- Biblioteca para utilizar filas assíncronas de mensageria
  - ActiveMQ 5.9.0
- Bibliotecas para tratativa da camada de persistência
  - Hibernate 4.2.6 (core, ehcache, entity-manager)
  - Hibernate JPA 1.0.1
  - Hibernate search 4.3.0
  - Hibernate validator 4.3.1
- Spring Framework 3.2.4
  - Expression
  - JDBC
  - Aspects
  - AOP
  - Web
  - Web MVC
  - Secutity
- Framework para camada de visão (front-end)
  - JSF-API 2.1.24
  - Primefaces 4.0
  - Omnifaces 1.6
- Bibliotecas e frameworks para WebSocket w WebServices
  - Atmosphere 2.1.4
  - Java-WebSocket 1.3.0
  - Axis 1.4
  - Json 20080701
- Biblioteca para agendamento
  - Quartz 2.1.5

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 (trunk branch):

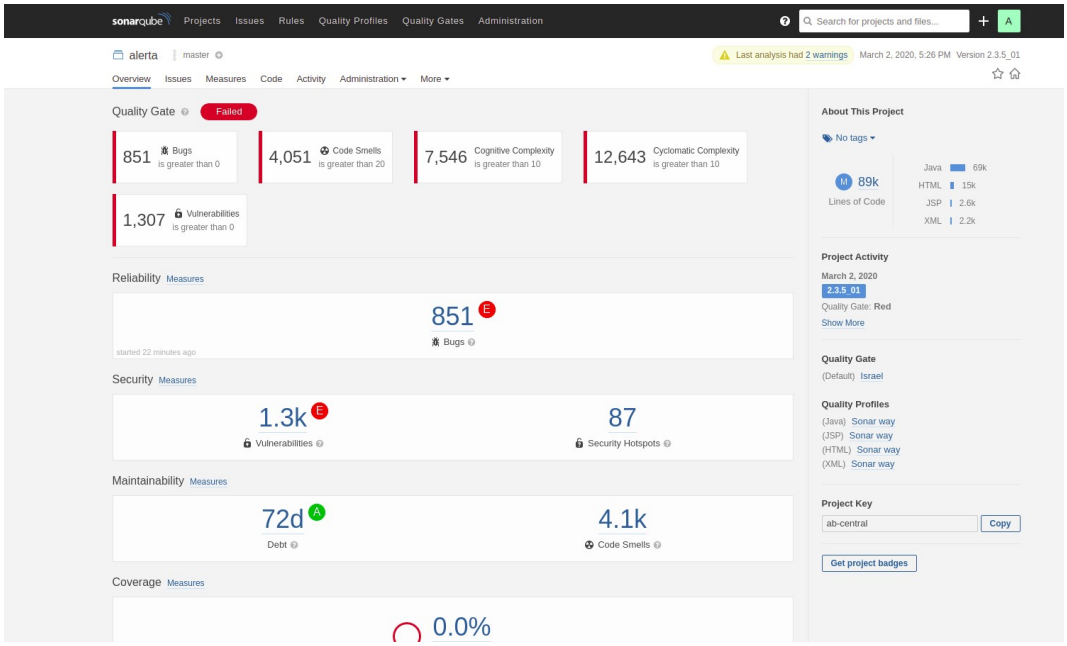

Figura 2: Análise estática de código

Nesta análise obtivemos os seguintes resultados:

- 851 bugs;
- 4.051 violações de más práticas;
- 7.546 violações de complexidade cognitiva (dificuldade de entendimento de código);
- 12.643 violações de complexidade ciclomática (complexidade de código);
- 1.307 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
itextpdf-5.3.0.jar	HIGH	1	High	25
activemq-all-5.9.0.jar	CRITICAL	18	Highest	27
dom4j-1.6.1.jar	HIGH	1	Highest	25
mysql-connector-java-5.1.25.jar	HIGH	4	Highest	36
solr-core-3.6.2.jar	HIGH	12		28
solr-solrj-3.6.2.jar	HIGH	10		27
tika-core-1.4.jar	CRITICAL	8	Highest	30
pdfbox-1.8.1.jar	HIGH	3	Highest	25
jempbox-1.8.1.jar	HIGH	3	Highest	27
bcprov-jdk15-1.45.jar	0.0	15	Highest	27
poi-3.9.jar	HIGH	7	Highest	29
xmpcore-5.1.2.jar	HIGH	1		33
xercesImpl-2.8.1.jar	0.0	2	Low	67
jackson-mapper-asl-1.9.12.jar	CRITICAL	11	High	30
hibernate-validator-4.3.1.Final.jar	MEDIUM	1	Highest	33
jsoup-1.7.2.jar	MEDIUM	1	Highest	27
logback-core-1.0.13.jar	CRITICAL	1	Highest	32
guava-11.0.jar	MEDIUM	1	Highest	22
commons-fileupload-1.3.jar	CRITICAL	4	Highest	36
commons-beanutils-1.8.3.jar	HIGH	2	Highest	35
commons-collections-3.2.1.jar	CRITICAL	3	Highest	35
spring-core-3.2.4.RELEASE.jar	CRITICAL	13	Highest	26
spring-tx-3.2.4.RELEASE.jar	CRITICAL	7	Highest	26
spring-aspects-3.2.4.RELEASE.jar	CRITICAL	13	Highest	12
spring-web-3.2.4.RELEASE.jar	CRITICAL	10	Highest	26
spring-webmvc-3.2.4.RELEASE.jar	CRITICAL	9	Highest	28
spring-security-core-3.1.4.RELEASE.jar	CRITICAL	4	Highest	32
spring-security-web-3.1.4.RELEASE.jar	HIGH	1	Highest	31
spring-security-config-3.1.4.RELEASE.jar	HIGH	1	Highest	31
jsf-impl-2.1.24.jar	MEDIUM	1	Highest	45
primefaces-4.0.jar	CRITICAL	3	Highest	20
jstl-1.2.jar	HIGH	1		26
axis-1.4.jar	HIGH	4	Highest	15
javamelody-core-1.47.0.jar	0.0	6	Highest	15
jasperreports-4.1.2.jar	HIGH	4	High	29
bcprov-jdk14-138.jar	HIGH	13	Highest	23
castor-1.2.jar	MEDIUM	1	Highest	19
quartz-2.1.5.jar	CRITICAL	1		22
c3p0-0.9.1.1.jar	HIGH	1	Highest	26
javamelody-core-1.47.0.jar: prototype.js	HIGH	1		3
primefaces-4.0.jar: jquery.js	MEDIUM	2		3
activemq-all-5.9.0.jar (shaded: org.apache.activemq:activemq-camel:5.9.0)	CRITICAL	18	Highest	11
activemq-all-5.9.0.jar (shaded: org.jasypt:jasypt:1.9.1)	HIGH	1	Highest	13
activemq-all-5.9.0.jar (shaded: org.apache.activemq:activemq-amqp:5.9.0)	CRITICAL	18	Highest	11
activemq-all-5.9.0.jar (shaded: log4j:log4j:1.2.17)	CRITICAL	1	Highest	13

4.3 Estrutura do projeto

A nomenclatura utilizada na organização dos pacotes está projetada e disposta de forma intuitiva e coesa. A quantidade de pacotes e a falta da componentização deste monólito dificulta o entendimento do mesmo, trazendo consigo aumento na curva de aprendizado para novos ingressos a equipe de desenvolvimento do projeto.

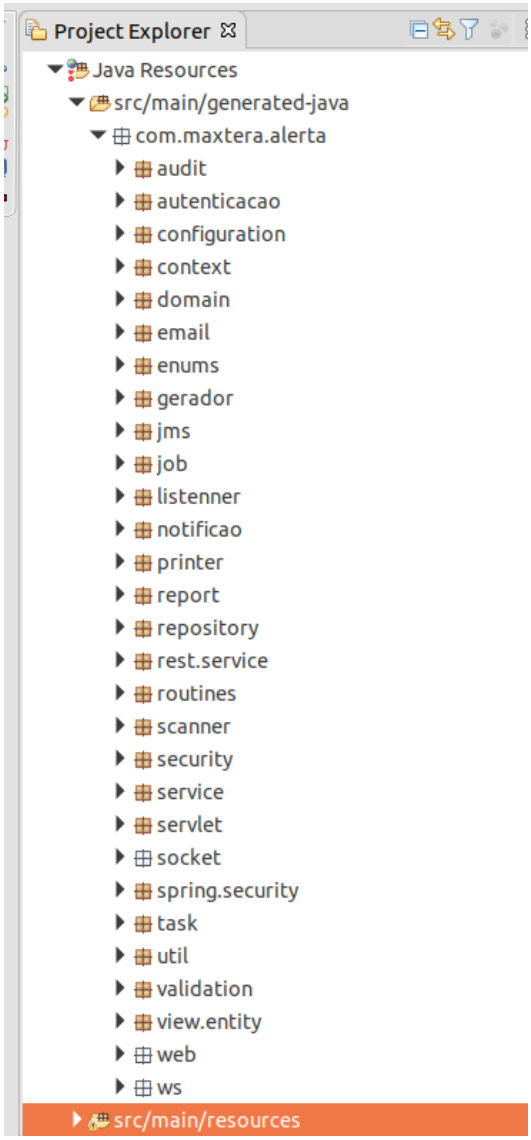

Figura 3: Estrutura do projeto

4.4 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).

```
public static JSONArray convertVeiculoRestricaoList(com.maxtera.alerta.util.RetornoNifi.Veiculo.Restri
        JSONArray jsonArray = new JSONArray();
        JSONObject json;
        Posto posto = CacheHolder.postoGet(postoCod);
        boolean tipoRestricaoValida;
        for (com.maxtera.alerta.util.RetornoNifi.Veiculo.Restricao vr : vra) {
                TipoRestricao tipoRestricao = CacheHolder.tpRestWSGet(vr.tipoDeRestricao);
                if (tipoRestricao.getAtivo()) {
                        tipoRestricaoValida = true;
                        if (tipoRestricao.getListaPostosBloqueados() != null && !tipoRestricao.getLista
                                tipoRestricaoValida = !tipoRestricao.containsPosto(posto);
                                if (!tipoRestricaoValida) {
                                        tipoRestricaoValida = verificaDesbloqueio(posto, tipoRestricao
                                }
                        }
                        if (tipoRestricaoValida) {
                                json = convertVeiculoRestricao(vr);
                                jsonArray.put(json);
                        }
                }
        }
        return jsonArray;
```

Figura 4: Alta complexidade ciclomática

4.5 Confiabilidade

As evidências demonstram a existência de tratativas de controle transacional na camada de serviço da aplicação e também há controle transacional na camada de acesso a dados, este é tratamento segue as boas práticas de desenvolvimento de aplicações sendo estas as camadas responsáveis 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.

4.6 Performance e estabilidade

Não foi analisado o funcionamento da aplicação para avaliar demais requisitos não funcionais e por inferência à apresentação realizada para demonstrar suas funcionalidades, a aplicação apresenta baixo rendimento performático em sua utilização. Não é possível estimar (neste momento) se a baixa performance apresentada no sistema está relacionado a segmentação de rede (podendo haver alta latência), quantidade simultânea de acessos (tendo em vista que o ambiente de produção fora utilizado para apresentação) ou mesmo relacionada a performance do banco de dados Teradata (sendo esta última a hipótese mais provável do baixo desempenho da aplicação).

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 honerosos.

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.

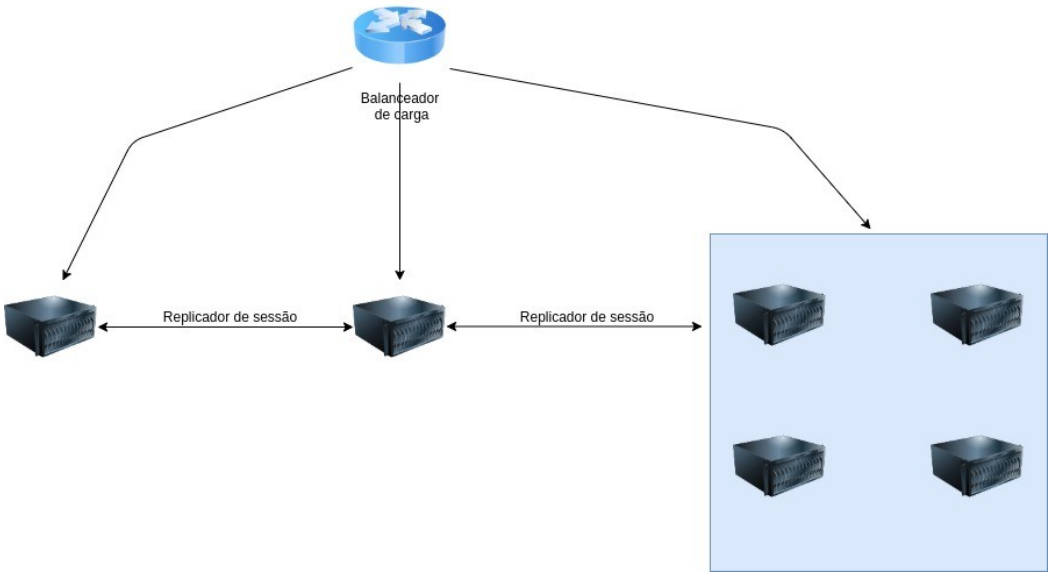

Figura 5: Escalabilidade do monólito

6 Recomendações

Recomenda-se que seja efetuado o levantamento das funcionalidades presentes no sistema atual, elencar destas todas as funcionalidades que estão atualmente em uso e a partir dai, iniciar processo de fork do projeto, objetivando eliminar funcionalidades obsoletas dependências desnecessárias.

Recomenda-se também que seja utilizado repositório central de artefatos (ex: Nexus ou Artifactory) para armazenamento e versionamento de componentes. Atualmente existem dependências que estão somente no repositório do desenvolvedor sem qualquer forma de gestão.

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 intrusão) 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 MySQL.

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.

Uma vez que o SGBD Teradata tem como maior função trabalhar em dados analíticos e estruturas OLAP de DW, recomenda-se a substituição de toda a parte transacional utilizada na aplicação para SBGB relacional (tal como PostgreSQL, SQL Server, Oracle ou outros).