LIGA DE ENSINO

CENTRO UNIVERSITÁRIO DO RIO GRANDE DO NORTE

ESPECIALIZAÇÃO EM DESENVOLVIMENTO DE SISTEMAS CORPORATIVOS

RAFAEL HOLANDA DE LIMA

MONITORAMENTO DO BARRAMENTO DE SERVIÇOS SOB A PLATAFORMA MULE ESB

NATAL/RN

2015

RAFAEL HOLANDA DE LIMA

MONITORAMENTO DO BARRAMENTO DE SERVIÇOS SOB A PLATAFORMA MULE ESB

Trabalho de Conclusão de Curso apresentado ao Centro Universitário do RN como parte dos requisitos para a obtenção do título de Especialista em Desenvolvimento de Sistemas Corporativos.

Orientador: Prof. Dr. Gibeon Aquino

NATAL/RN

2015

Resumo

Este trabalho apresenta uma proposta de solução para o monitoramento de um barramento de serviços corporativo (ESB). A solução apresentada permite um monitoramento assistido, a partir de uma aplicação WEB, e um monitoramento remoto, a partir de um aplicativo Android. A aplicação WEB acompanha métricas do ESB com uma determinada frequência e as armazena em um banco dados, o qual é utilizado para a construção de gráficos a serem exibidos para o usuário. O aplicativo Android, por sua vez, permite que o usuário seja notificado quando os limites, definidos no próprio aplicativo, são excedidos pelo ESB.

Lista de Figuras

Sumário

1. Introdução

Este capítulo apresenta a motivação para a criação de uma aplicação responsável por monitorar um barramento de serviços (Seção 1.1), em que também serão apresentados os objetivos (Seção 1.2) e o escopo deste documento (Seção 1.3).

1.1 Motivação

Diante da evolução no desenvolvimento de Software, novos modelos de arquitetura foram sendo criados, entre eles foi a arquitetura orientada a serviços (SOA), um conceito de arquitetura corporativo que promove a integração entre o negócio e a TI por meio de conjuntos de interfaces de serviços acoplados. Um dos componentes mais importantes em SOA é o ESB (Enterprise Service Bus - Barramento de Serviços Corporativos), o qual não implementa a arquitetura, mas oferece as funcionalidades para implementá-la. O barramento provê uma camada de abstração que permite a integração entre diversas aplicações.

Com a necessidade da utilização de um barramento de serviços em um de seus produtos, a empresa Softplan Planejamento de Sistemas desenvolveu um barramento sob a plataforma Mule que será capaz de se conectar a WebServices instalados em cada estado brasileiro. Sobre essa empresa, considera-se uma das maiores do Brasil no desenvolvimento de softwares de gestão. Atualmente, suas soluções estão presentes em todos os estados brasileiros. Desde 1990, a companhia atua de modo a tornar gestão pública e privada no Brasil mais transparente, eficiente e ágil como o uso de tecnologias modernas e inovadoras.

A partir disso, como o barramento desenvolvido deverá acoplar diversos serviços em diferentes regiões ou infraestruturas, notou-se a necessidade de ser implementada uma aplicação de monitoramento, a qual será relatada nesse trabalho, em que permitam serem acompanhados os parâmetros de um ESB, que possam indicar uma falha de conexão, um elevado número de requisições recebidas, um elevado processamento, erros de execução e outros fatores que possam prejudicar o desempenho ou o funcionamento do barramento, acarretando assim em um resultado não esperado pelo produto.

1.2 Objetivo

O principal objetivo deste trabalho é possibilitar o monitoramento de qualquer barramento de serviços (ESB) sob a plataforma Mule, seja de maneira assistida ou remota.

A solução é composta de um barramento já desenvolvido, que deverá ser acrescido de um ajuste para que possa aceitar a comunicação com a aplicação web implementada neste trabalho, a qual deverá consultar o barramento com determinada frequência, armazenando as informações obtidas e exibindo-as de maneira compreensível para o usuário do sistema.

O monitoramento poderá ser realizado de forma remota através de notificações recebidas no dispositivo móvel do usuário, que deverá definir quando deseja que seja informada a situação em se encontra o ESB.

1.3 Escopo

Este documento possui o seguinte escopo: o Capítulo 2 apresenta toda a fundamentação teórica para elaboração desta solução, listando as principais tecnologias e uma breve explicação das suas características. O Capítulo 3 explana o funcionamento da solução em que exibe as interfaces que possibilitam a interação com o usuário final. O Capítulo 4 detalha a estrutura implementada das camadas dos projetos envolvidos a partir de diagramas. O Capítulo 5 por fim, expõe as considerações finais.

2. Fundamentação Teórica

Nesta seção serão abordadas as tecnologias e conceitos, os quais possibilitaram a elaboração deste trabalho.

2.1. Spring Boot

O Spring Boot é um projeto do Spring criado para facilitar o desenvolvimento de aplicações Java para web, utiliza a convenção sobre configuração, o qual exige menos arquivos de configuração (devido às convenções) sem perder a liberdade (ainda é possível configurar) visando simplicidade e agilidade principalmente no momento da criação da aplicação.

O mesmo também possibilita ao desenvolvedor a utilização de diversos requisitos não funcionais já pré-configurados como, por exemplo, acesso a base de dados, servidor de aplicação embarcado, segurança, web services, scheduler e outros.

A partir do Spring Boot torna-se fácil criar aplicações “stand-alone”, em que é permitido “apenas executar”, ou seja, ao gerar um pacote jar da aplicação, pode-se subir a mesma apenas com o comando “java -jar” e assim o servidor de aplicação embarcado inicializa ficando disponível na porta configurada, por padrão a 8080.

2.2. JavaServerFaces e PrimeFaces

Uma tecnologia que nos permite criar aplicações Java para Web, o Java Server Faces (JSF), utiliza componentes visuais pré-prontos, permitindo que o desenvolvedor deixe de se preocupar com Javascript e HTML. Pode-se encontrar diversos componentes, tais como calendários, tabelas ou formulários, os quais serão renderizados e exibidos em formato html.

Para que possa ser permitido a utilização de componentes mais sofisticados, existem várias extensões do JSF que seguem o mesmo ciclo e modelo da especificação. Exemplos dessas bibliotecas são PrimeFaces, RichFaces e IceFaces, todas definem componentes JSF que vão muito além da especificação.

O PrimeFaces foi uma das extensões escolhidas para o desenvolvimento deste trabalho por ser uma das mais utilizadas e por possuir diversos componentes para gráficos, em que é possível utilizá-lo para desenhar gráficos de linha, de barra, de pizza, de área, gráficos para atualização dinâmica. Também é possível exportar e animar gráficos.

2.3. Scheduler e CronExpression

O agendamento de tarefas, scheduler, é útil quando se precisa que determinada função da aplicação seja evocada em um momento específico do dia, como por exemplo no monitoramento de alguma aplicação com determinada frequência. A repetição de tarefas em um intervalo de tempo pré-definido é útil para checagem de parâmetros e

armazenagem dos mesmos em um banco de dados para possibilitar a criação de um histórico.

No Spring, o scheduler, ou “agendador”, é configurado através de cronExpression com a sintaxe do cron, em que podem ser definidos o segundo, o minuto, a hora, o dia do mês, o mês e o dia da semana. Conforme exemplo, “0 43 9 ? 8 FRI”, ou seja, a tarefa será executada aos “0 segundo, 43 minutos, 9 horas, qualquer dia do mês (?) de agosto (8), às sextas-feiras (FRI)”.

2.4. MongoDB

O MongoDB é um banco de dados não relacional orientado a documentos, dessa, forma ao invés de armazenar dados relacionados em uma área de armazenamento separado, os bancos de dados de documentos integram esses dados ao próprio documento.

Possui como características ser de código-fonte aberto, possuir alta perfomance, não possuir esquemas, ser escrito em C++, multiplataforma e ser formado por um conjunto de aplicativos JSON. É um banco de dados schemaless, ou seja, podem ser adicionados novos campos quando for necessário, o que aumenta a flexibilidade da solução.

O MongoDB é rápido tanto para inclusão quanto para consultas, suporta milhares de inserts por segundo, o que o torna bastante útil quando deseja-se realizar um monitoramento, em que se é necessário a inserção dos dados a cada verificação em um curto espaço de tempo.

2.5. Web services RESTful

Representational State Transfer (Transferência de estado representacional) ou somente REST é baseado no design do protocolo HTTP, utiliza os métodos POST, GET, PUT e DELETE para se comunicar, porém não existe um padrão obrigatório, pode ser implementado somente o que é necessário em seu contexto. Esses métodos podem ser perfeitamente mapeados respectivamente com as funções create, read, update e delete (CRUD).

Na arquitetura REST, dados e funcionalidades são considerados recursos e são acessados através de Uniform Resource Identifiers (URIs), tipicamente links na Web. Essa identificação proporciona um espaço de endereçamento global de recursos e de descoberta de serviço.

Os serviços desenvolvidos em RESTful são altamente escaláveis, enquanto são fracamente acoplados aos dados subjacentes e possibilitam o armazenamento em cache, resultando em um melhor desempenho, escalabilidade e manutenibilidade, o que permite um melhor funcionamento na Web e em dispositivos móveis que costumam possuir um baixo nível de processamento.

Os recursos são separados de sua representação para que o seu conteúdo possa ser acessado em uma variedade de formatos, como HTML, XML, texto simples, PDF, JPEG, JSON e outros. Os metadados sobre o recurso está disponível e pode ser utilizado, por exemplo, para controlar o cache, detectar erros de transmissão, negociar formato de representação adequada, e realizar a autenticação ou controle de acesso.

2.6. MuleESB

O MuleESB é uma plataforma de integração da MuleSoft com uma leve linguagem de programação em Java, a qual permite integrar ou se comunicar com múltiplas aplicações através de uma abordagem orientada a serviços, possibilitando uma fácil integração de sistemas existentes, independentemente das diferentes tecnologias que os aplicativos usam, incluindo JMS, serviços web, JDBC e HTTP.

Um barramento de serviços (ESB – Enterprise Service Bus), possui como principal tarefa, tratar as mensagens entre os diversos aplicativos, sendo a espinha dorsal da integração de uma empresa. Suas principais funcionalidades são validar a validação de esquema, enriquecer, tranformar, rotear e operar (executar operações) - VETRO.

Segue na Figura 1 abaixo um diagrama que exemplifica a utilização de um ESB, onde só precisa conectar cada aplicativo ao ESB para que cada um possa se comunicar uns com os outros.

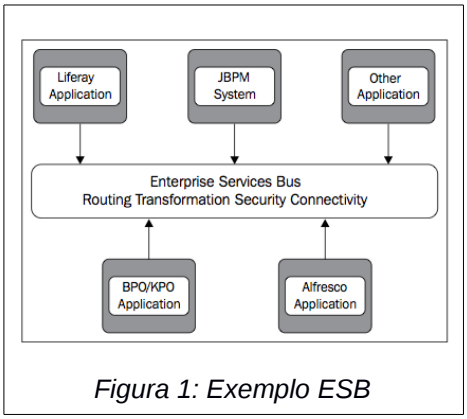

Figura 1: Exemplo ESB

2.7. Java Management Extensions (JMX)

A JMX é uma tecnologia permite o monitoramento da máquina virtual Java (JVM), a qual fornece ferramentas para o gerenciamento de recursos como aplicações, dispositivos, serviços e outros.

A arquitetura JMX é dividida em três níveis, instrumentação, agente e gerente. Essa divisão traz flexibilidade, permitindo que subconjuntos da especificação sejam utilizados individualmente por diferentes comunidades de desenvolvedores que utilizam tecnologia Java.

Um recurso gerenciável é um recurso que foi instrumentado conforme o nível de especificação de instrumentação, o qual pode ser uma aplicação de negócio, um dispositivo, ou uma implementação de “software” de um serviço ou política. Um objeto

Mbean é um objeto Java que representa um recurso JMX gerenciável que segue o modelo JavaBeans e permitem o mapeamento entre os componentes e o seu gerenciamento. Os objetos Mbeans podem ser adaptados a qualquer agente JMX, pois permitem instrumentação de forma padrão dos recursos gerenciados.

Um agente JMX é uma entidade de gerência implementada conforme a especificação de agente JMX, composto de um servidor de Mbean que fornece os serviços que permitem a manipulação de objetos Mbeans, um kit de Mbeans que representa os recursos gerenciados e por, pelo menos, um adaptador de protocolo, o qual possibilita sobre uma variedade de protocolos (HTTP, HTTPS, IIOP) que as aplicações de gerência tenham acesso a um agente JMX. Pode-se conter serviços de gerência, que são representados como Mbeans.

Implementado conforme a especificação de gerência JMX, um gerente JMX é uma entidade que fornece uma interface para que as aplicações de gerência possam interagir com o agente, distribuir ou consolidar informações de gerência, e prover segurança. Os gerentes JMX podem controlar qualquer número de agentes, simplificando a estrutura de gerência de sistemas complexos e distribuídos.

A especificação JMX define a interface para alguns serviços básicos geralmente utilizados, tais como, um registro de Mbeans, consultas desses registros, operações sobre os recursos e transmissão de eventos de volta aos gerentes, carga dinâmica de um novo Mbean, criação de dependências entre Mbeans, funções de tempo e monitoração de atributos.

2.8. Jolokia

Jolokia é uma abordagem baseada em agentes JMX com suporte para várias plataformas, considerada uma ponte JMX-HTTP. Além das operações básicas JMX ele melhora a comunicação remota JMX com características únicas, como as requisições em massa e as políticas de segurança. O seu funcionamento é bem simples, o qual se conecta a um servidor Mbean e expões os dados via REST.

O MuleESB fornece uma API de gerenciamento que um agente Jolokia dedicado conecta-se muito bem. Este agente inclui um servidor Jetty embutido para fornecer acesso HTTP-JMX.

2.9. AndroidAnnotations

AndroidAnnotations é um framework de código-fonte aberto para acelerar o desenvolvimento para Android, tornando o código mais robusto e assim facilitando a sua manutenção.

A partir de anotações Java, os desenvolvedores podem mostrar apenas quais suas intenções e deixar que o framework gere em tempo de compilação o código correspondente ao desenvolvimento nativo para o Android.

Além de sua manutenibilidade e robustez, o AndroidAnnotations possui outras características como a injeção de dependências, que permite serem injetados views, extras, system services, resources dentre outras dependências que nos deparamos ao desenvolver um aplicativo Android; um modelo de threads para execução em background simplificado, anotando apenas o método que deseja ser executado por uma thread em background; uma melhor implementação de eventos vinculados a views, sem a necessidade de classes listeners, conforme é definido na forma padrão de desenvolvimento Android; criação de clientes REST apenas com a definição de sua interface com seus métodos anotados, que o framework gera a implementação necessária para o seu funcionamento. Como o código é gerado em tempo de compilação, o mesmo pode ser acessado facilmente para que possa ser compreendido como é feita a implementação pelo framework.

2.10. Google Cloud Messaging

O Google Cloud Messaging (GCM) é um serviço para Android que permite que sejam enviados dados de um servidor para os usuários de dispositivos Android, e também que receba mensagens a partir da mesma conexão. O serviço GCM lida com todos os aspectos de enfileiramento de mensagens e entrega para um aplicativo Android em execução no dispositivo de destino, e é totalmente gratuito.

Conforme ilustrado na Figura 2, o funcionamento do GCM ocorre a partir das seguintes etapas:

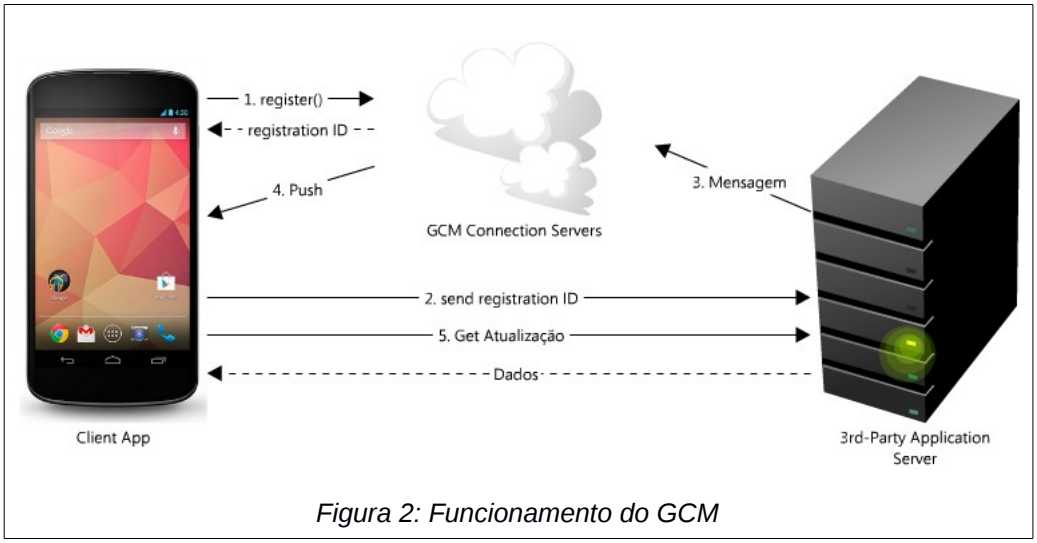

Figura 2: Funcionamento do GCM

1. Registro do aplicativo no GCM que retorna um identificador que será necessário para que o serviço identifique qual dispositivo o servidor deseja enviar a mensagem.

2. Envio do identificador registrado no GCM para o servidor responsável pela origem das mensagens.

3. Envio da mensagem do servidor para o GCM com o identificador do dispositivo de destino desejado para receber a mensagem.

4. Envio da mensagem pelo GCM para o dispositivo quando o mesmo se encontra conectado a uma conexão de Internet, caso contrário, o serviço ficará aguardando até que o destino esteja conectado.

Dois tipos de mensagens são permitidos pelo serviço, o *Send to Sync Messages*, são as notificações para que a aplicação saiba que o servidor possui novos dados a serem sincronizados; e o *Send Data Messages*, são mensagem de dados que possuem até 4KB de informações incluídas no corpo da mensagem.

3. Proposta de Solução

O projeto desenvolvido é composto por quatro aplicações (Figura 3), uma aplicação WEB, um aplicativo em Android, o serviço GCM da Google e por final um ESB sob a plataforma Mule.

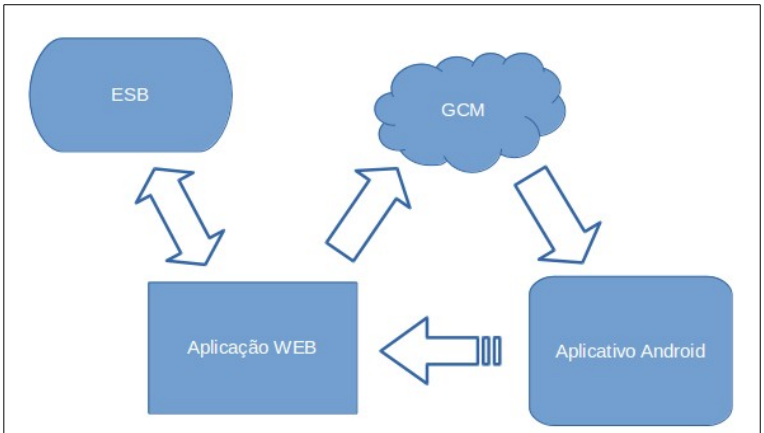

Figura 3: Proposta de Solução

3.1. Aplicação WEB

Para ser realizado o monitoramento foi desenvolvida uma aplicação central, a qual é responsável por fazer a verificação de tempos em tempos de todos os parâmetros do ESB e assim armazenando os seus valores atuais em uma base de dados gerando um histórico, o qual é exibido em gráficos para os usuários que acessam a aplicação. A mesma também é responsável por notificar os usuários cadastrados quando um determinado parâmetro extrapola os limites definidos, enviando uma requisição POST para o serviço GCM da Google.

A aplicação WEB para o seu devido funcionamento deve-se iniciar com a configuração do ESB a ser monitorado, a qual deverá ser feita através de um arquivo chamado application.properties (Figura 4) encontrado dentro do projeto (esbmonitor/src/main/resources/). No arquivo será informado a URL onde está disponível o projeto MuleESB (mule.url) e o nome do projeto (mule.projeto), como também a chave fornecida ao cadastrar o projeto no serviço GCM da Google.

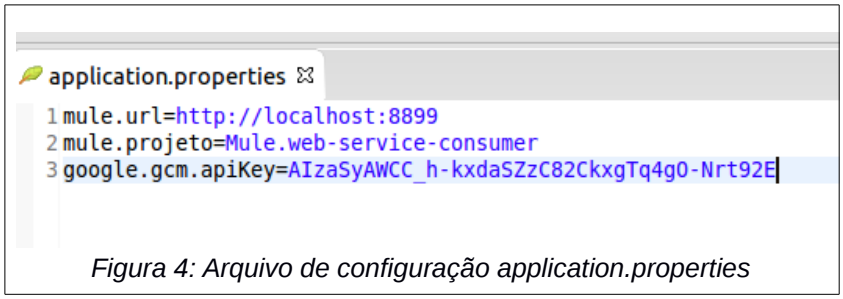

Figura 4: Arquivo de configuração application.properties

Após configuração devidamente realizada, o próximo passo para dar inicio ao monitoramento do ESB, será o cadastro dos fluxos escolhidos para o seu acompanhamento no uso da aplicação, utilizando uma tela de cadastro conforme observado na Figura 5.

Figura 5: Tela de Cadastro de Fluxo

O botão com um icone de “+” que redireciona para a tela de cadastro de fluxo poderá ser encontrada na tela inicial do sistema, onde serão exibidos os gráficos de monitoramento baseados nos dados armazenados referentes a cada fluxo cadastrado anteriormente conforme Figura 6.

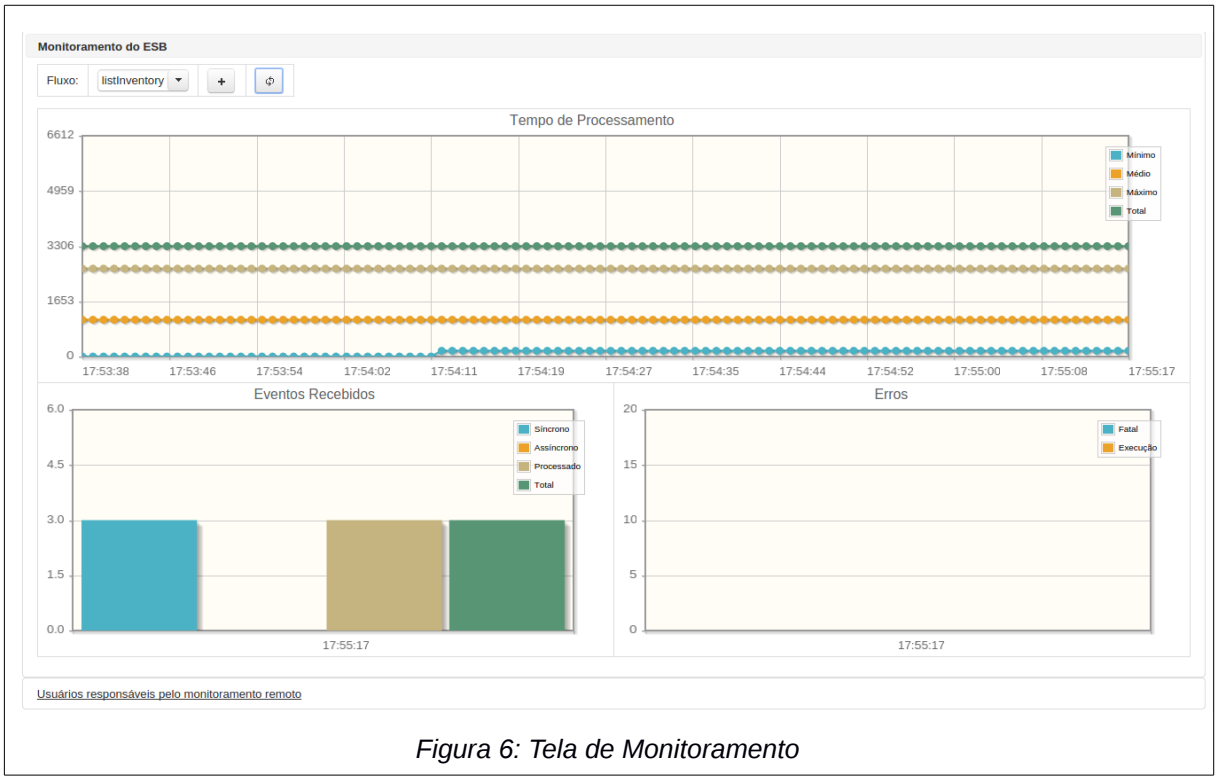

Figura 6: Tela de Monitoramento

Na tela de monitoramento poderá ser escolhido cada fluxo que deseja-se verificar três gráficos, o primeiro deverá monitorar o tempo de processamento obtido através de 4 parâmetros (mínimo, médio, máximo e o total), o segundo, localizado abaixo do gráfico de tempo de processamento, deverá exibir os eventos recebidos verificados a partir de 4 parâmetros (síncrono, assíncrono, processado e o total) e por último, ao lado do gráfico de eventos recebidos, serão exibidos os erros ocorridos identificados diante de 2 parâmetros (fatal e de execução).

Ao final da tela de monitoramento estará disposto um link para o cadastro de usuários responsáveis pelo monitoramento remoto, o qual deverá redirecionar para uma tela de cadastro de usuário (Figura 7), onde serão informados aqueles que possuíram acesso ao aplicativo móvel onde receberá notificações a partir desta aplicação.

Figura 7: Tela de Cadastro de Usuário

3.2. Aplicativo Android

Como meio de possibilitar um monitoramento remoto, em caso que é necessário um acompanhamento de determinados parâmetros sem que o usuário fique verificando constantemente os gráficos exibidos na aplicação WEB, foi desenvolvido um aplicativo sob a plataforma Android em que é permitido ser informado limites mínimos e máximos dos parâmetros de tempo de processamento, eventos recebidos e erros durante o monitoramento. O aplicativo deve receber requisições através do serviço GCM e assim tratá-las para que sejam enviadas ao usuário do dispositivo como uma notificação do aplicativo.

Antes de ser disponibilizado o aplicativo, deverá ser alterada a classe MainActivity.java, informando na constante PROJECT_NUMBER, o número do projeto encontrado no cadastro realizado no serviço do GoogleCloudMessage.

O aplicativo ao ser iniciado deverá exibir uma tela de login (Figura 8) onde serão informados os dados do usuário responsável pelo monitoramento remoto, os quais serão validados com a aplicação WEB e em caso de sucesso, será solicitado ao serviço GCM da Google um registro de identificação daquele aplicativo instalado no dispositivo do usuário que o está utilizando e assim esse registro deverá ser enviado para a aplicação e associado ao usuário.

Figura 8: Tela de Login

Após ser realizada autenticação do usuário, deverá ser exibida uma tela de formulário (Figura 9), que deverá possuir a opção de seleção do fluxo que deseja ser notificado e os valores dos seus parâmetros (total médio de processamento, total de eventos recebidos, total de erros de execução e total de erros fatais) a serem informados para os limites em que o mesmo deverá possuir, assim ao ser clicado no botão salvar, os dados serão enviados para aplicação WEB que passará a acompanhar esses limites em seu monitoramento.

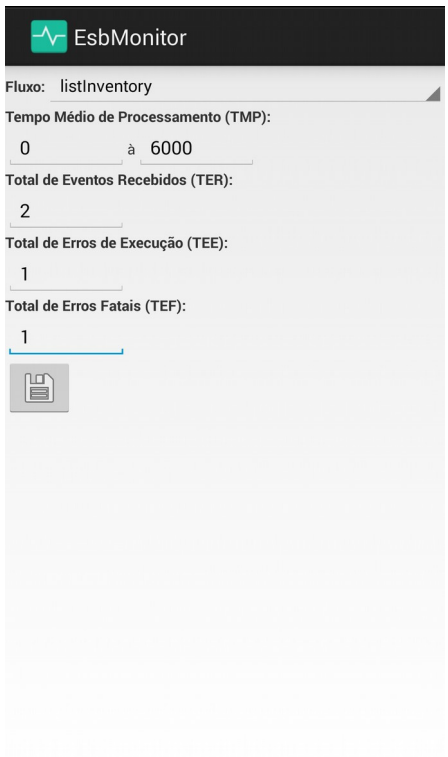

Figura 9: Tela de Limite de Parâmetro do ESB

Ao ser extrapolado algum dos limites definidos, o sistema WEB enviará para o GCM, o registro de identificação referente ao usuário que definiu aquele limite em conjunto com a informação de qual fluxo quebrou o limite e qual parâmetro foi excedido, e assim deverá ser exibida uma notificação no dispositivo do usuário responsável pelo monitoramento remoto conforme exemplo ilustrado na Figura 10.

Figura 10: Exemplo de Notificação

4. Estrutura de Implementação da Proposta

Neste capitulo será explicada a estrutura utilizada na implementação dos projetos desenvolvidos para um melhor funcionamento da proposta de solução e uma melhor manutenibilidade.

4.1. Aplicação WEB

Os seguintes diagramas de classes demonstram como foi desenvolvida a estrutura da aplicação WEB, composta pelas camadas de configuração, domínio, DAO, visão, conexão, integração, monitoramento e negócio.

4.1.1. Camada de Configuração

A camada de configuração trata-se das classes necessárias para que o SpringBoot funcione corretamente (Figura 11), atendendo todos os requisitos não-funcionais da aplicação. Camada composta pela classe principal (Application) de execução a qual deve conter o método main de execução Java, pela classe de configuração do jsf (JsfConfig) que deverá criar os beans necessários para que o Spring possa reconhecer o framework corretamente em conjunto com as classes ServletInitializer e a classe ServletListenerRegistrationBean. Todas essas classes possuem anotações especificas que o framework de injeção de dependências exige.

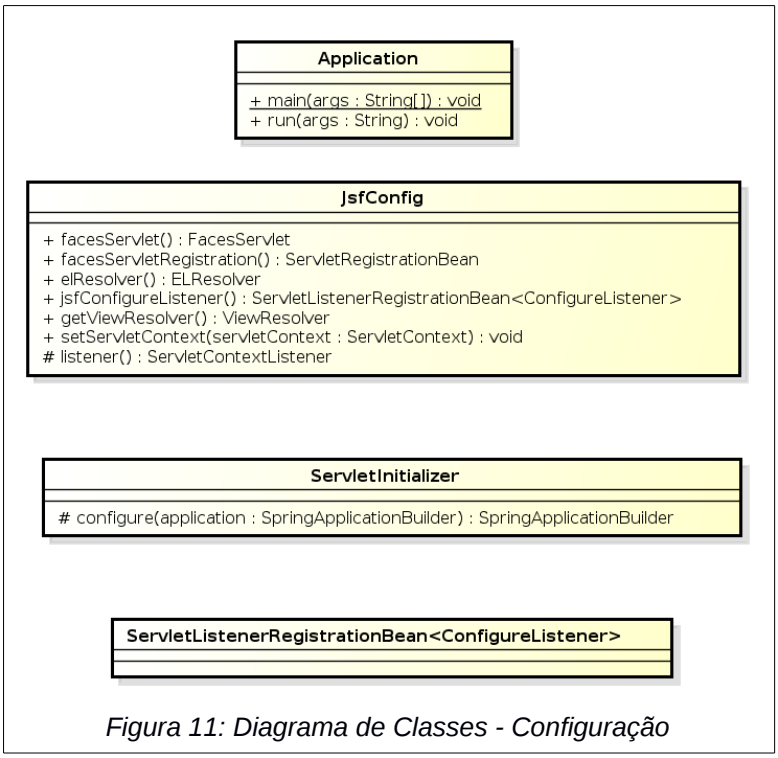

Figura 11: Diagrama de Classes - Configuração

4.1.2. Camada de Domínio

Na Figura 12 pode-se observar as classes que representam a camada de domínio da aplicação, as quais também formam a modelagem do banco de dados MongoDB:

- A classe RegistroFluxo representa todos os valores recebidos do ESB referente a determinado fluxo e a determinado parâmetro.
- A classe Fluxo identifica cada fluxo que será monitorado a partir do ESB configurado.
- A classe ParametroFluxo trata-se dos possíveis parâmetros de monitoramento de cada fluxo do ESB.
- A classe LimiteParametroFluxo se refere aos limites definidos pelos usuário, os quais são utilizados para notificações e o funcionamento do monitoramento remoto.
- A classe NotificacaoUsuario representa a notificação enviada para o usuário de acordo com o limite definido pelo mesmo que excedeu seus valores, criado para impedir que a mesma notificação seja enviada mais de uma vez enquanto o ESB mantiver naquela determinada situação.
- A classe Usuario possui as informações necessárias para que sejam identificados aqueles usuários que serão responsáveis pelo monitoramento remoto e deverão ser notificados.

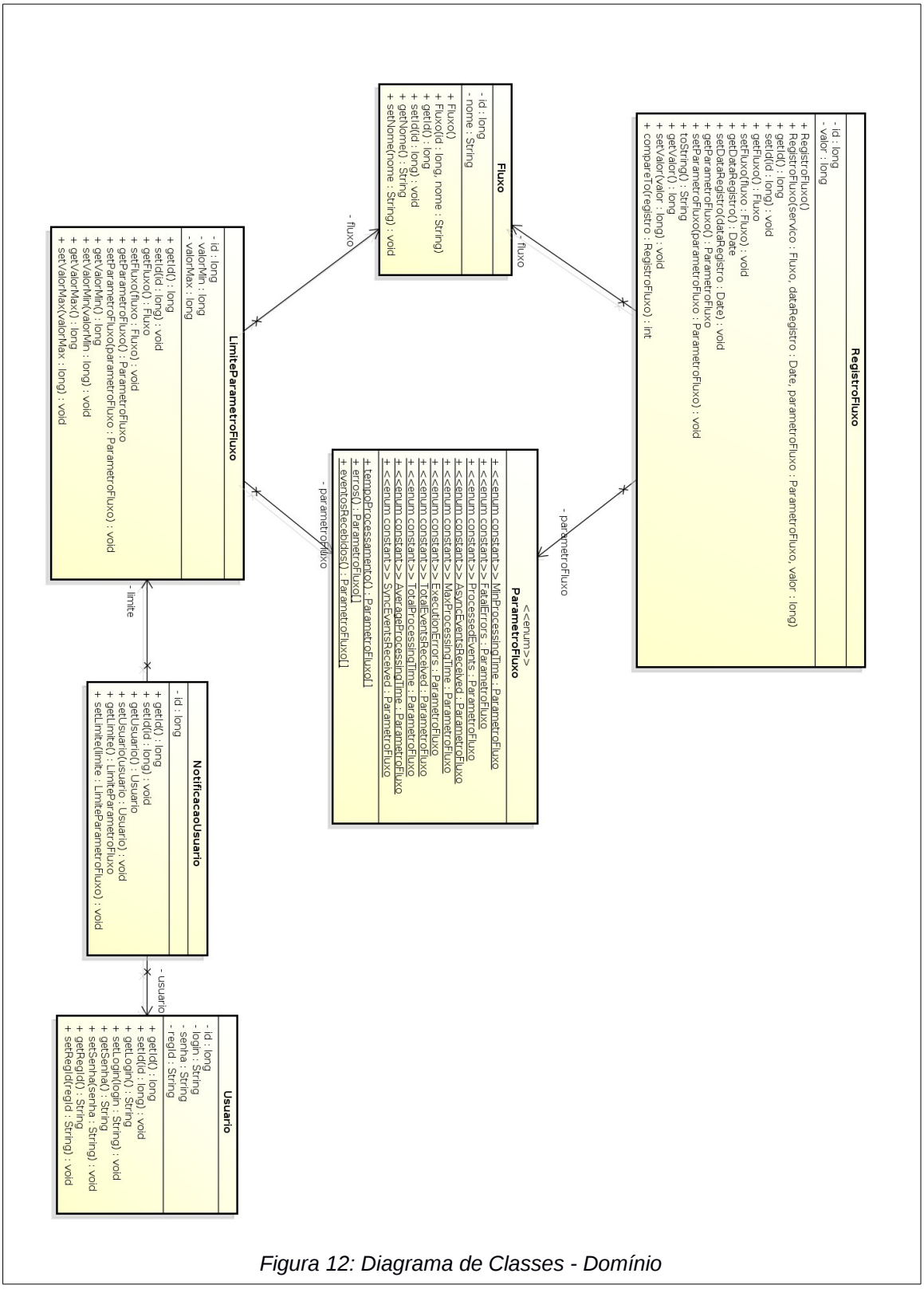

Figura 12: Diagrama de Classes - Domínio

4.1.3. Camada DAO

A partir das dependências do framework utilizado, para criação da camada de acesso ao banco de dados MongoDB, apenas foi necessário a criação de interfaces (Figura 13) que herdam de uma interface “MongoRepository<Entidade, Long>”, onde a “Entidade” deve ser a classe de dominio a qual o DAO pretende realizar as operações na base, e a deve ser informado o tipo da chave primária (“Long”).

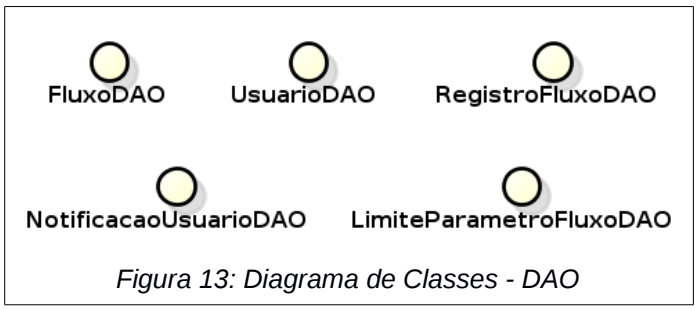

Figura 13: Diagrama de Classes - DAO

4.1.4. Camada de Visão

A camada de visão é composta pelas classes de implementação do Beans do JSF (Figura 14), classes que tratam as requisições vindas das telas que interagem com o usuário e enviam para a camada de negócio.

As classes FluxoBean e UsuarioBean são responsáveis pela criação e remoção dos dados das entidades Fluxo e Usuario respectivamente, assim, por possuírem funcionalidades parecidas, criou-se uma classe abstrata AbstractCrudBean que deve centralizar essas funcionalidades em comum, diferenciando apenas a classe de negócio e dominio.

A classe MonitorBean é responsável por gerar os gráficos na tela de monitoramento realizando a consulta na base de dados.

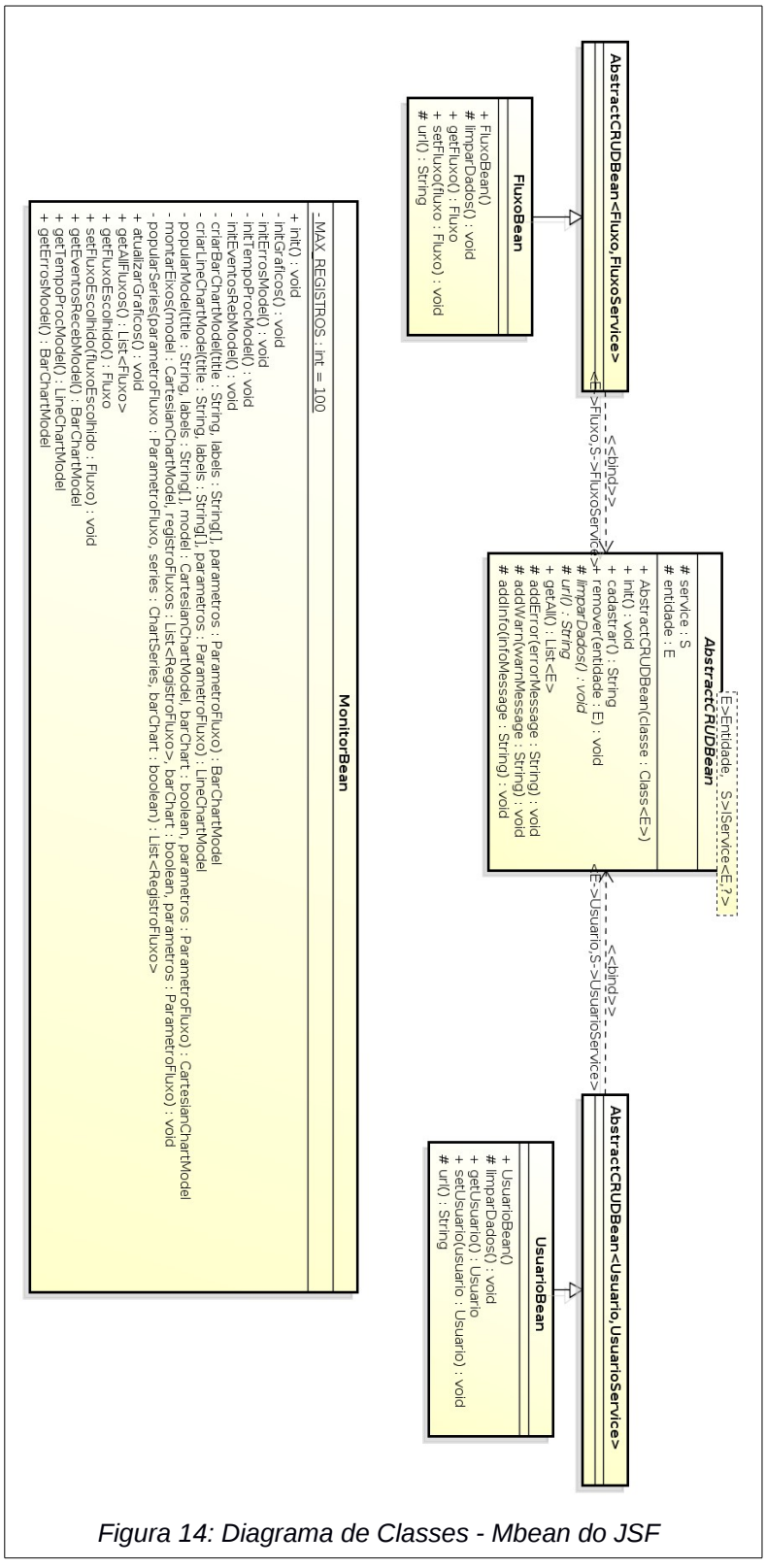

Figura 14: Diagrama de Classes - Mbean do JSF

4.1.5. Camada de Conexão

Esta camada é composta apenas por uma classe (Figura 15), a qual é responsável por implementar o cliente de conexão com o Jolokia instalado no ESB a ser monitorado.

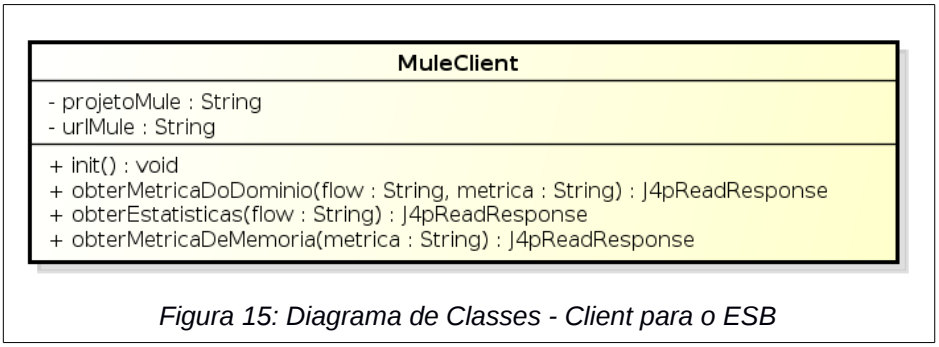

Figura 15: Diagrama de Classes - Client para o ESB

4.1.6. Camada de Integração

Para que seja permitida a integração entre o Aplicativo Android e a Aplicação WEB, foram criadas duas classes (Figura 16) para implementação da comunicação via REST.

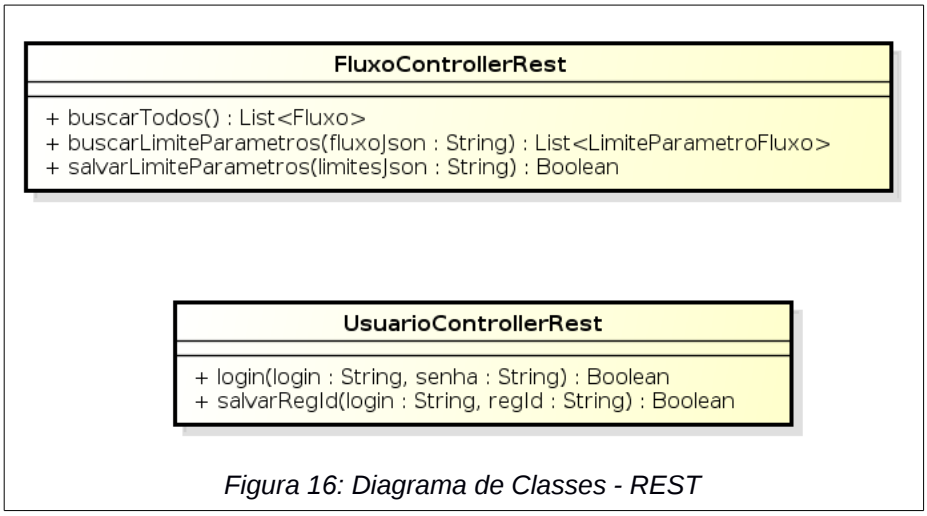

Figura 16: Diagrama de Classes - REST

4.1.7. Camada de Monitoramento

O monitoramento deve ser realizado com um determinada frequência, sendo assim, foi criada a classe MonitorTask (Figura 17) que trata-se de uma atividade programada que será executada conforme agendamento configurado a partir de anotações nessa classe em seus métodos principais.

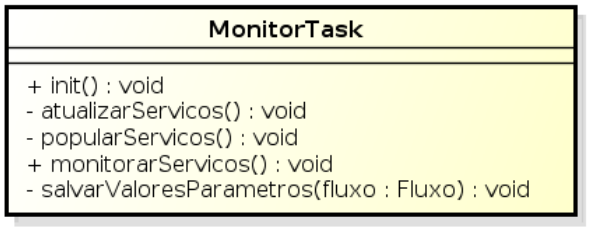

Figura 17: Diagrama de Classes - Task de Monitoramento

4.1.8. Camada de Negócio

Responsável por gerenciar as regras de negócio envolvidas, a camada de negócio é composta por classes associadas a cada classe de domínio (Figura 18), e a cada interface DAO, pois os dados que saem da camada de visão, da camada de conexão, da camada de integração e da camada de monitoramento, chegam a camada de negócio, a qual deverá invocar cada DAO responsável pelo armazenamento ou consulta dessas informações. E também onde será responsável por realizar as notificações, enviadas ao GCM (“NotificacaoUsuarioService”).

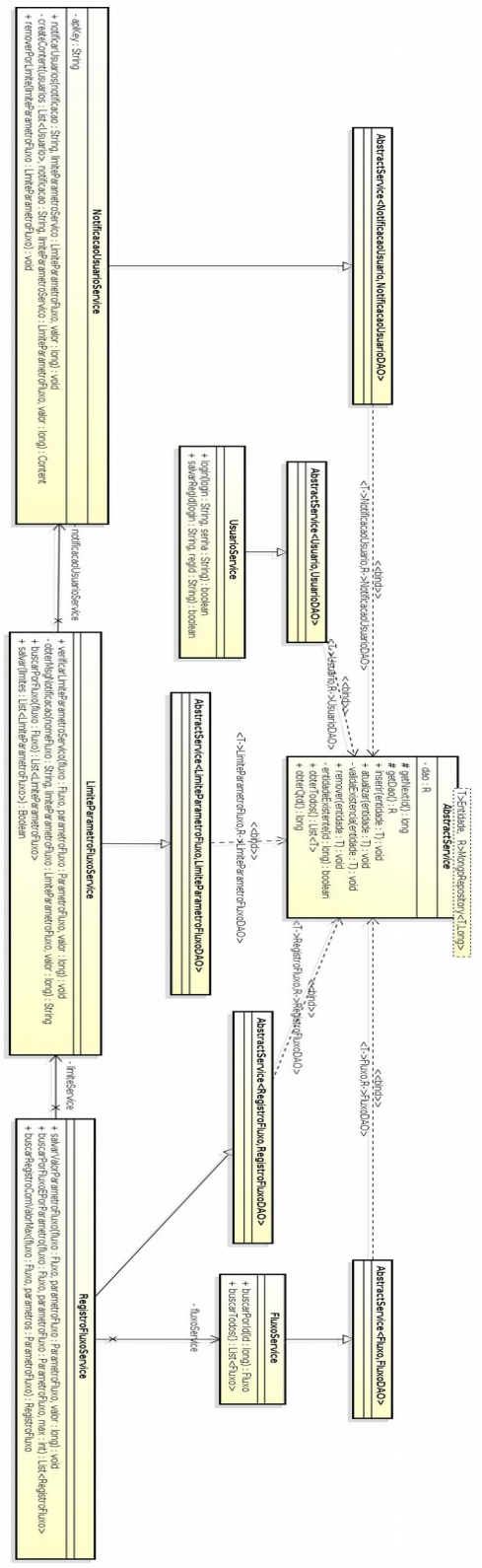

Figura 18: Diagrama de Classes - Camada de Negócio

4.1.9. Classes Utilitárias

No decorrer da implementação do projeto foram necessárias a criação de classes (Figura 19) que permitam auxiliar na exibição de mensagens ao usuário do JSF

(MensagemJsfUtil), no envio de requisição POST para o servidor da Google responsável pelo GCM (POST2GCM) e criação do conteúdo desta requisição (Content).

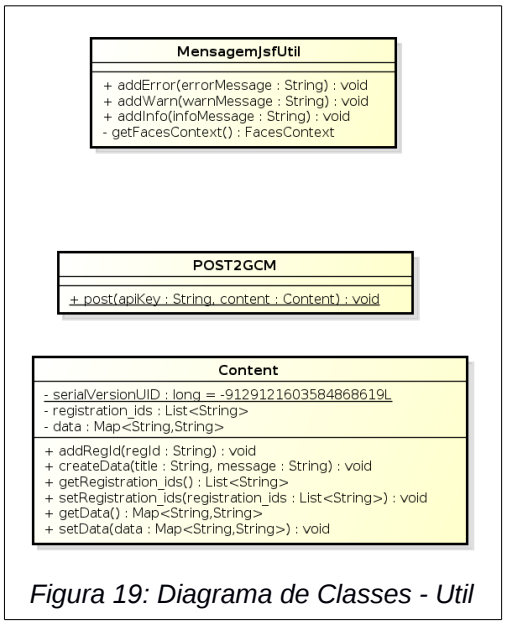

Figura 19: Diagrama de Classes - Util

4.1.10. Tratamento de Exceção

Para o tratamento de exceções, foi criada a classe DAOException (Figura 20), que deve ser instanciada e lançada quando ocorre algum erro no acesso ou operação na base de dados.

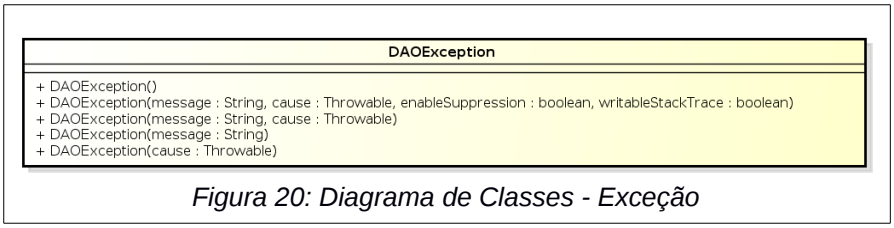

Figura 20: Diagrama de Classes - Exceção

4.2 Aplicativo Android

O aplicativo Android foi desenvolvido com o auxilio do framework AndroidAnnotations o que possibilitou um código mais enxuto e com poucas linhas de código. Pode-se observar a sua estrutura nos seguintes diagramas de classes.

4.2.1 Classes Activity

No desenvolvimento de aplicativos Android, para cada tela é necessária a implementação de uma classe Activity, assim, para a tela inicial de login foi implementada

a classe MainActivity, e para a tela de formulário dos limites de parâmetros a classe EsbMonitorActivity, conforme podemos observar na Figura 21.

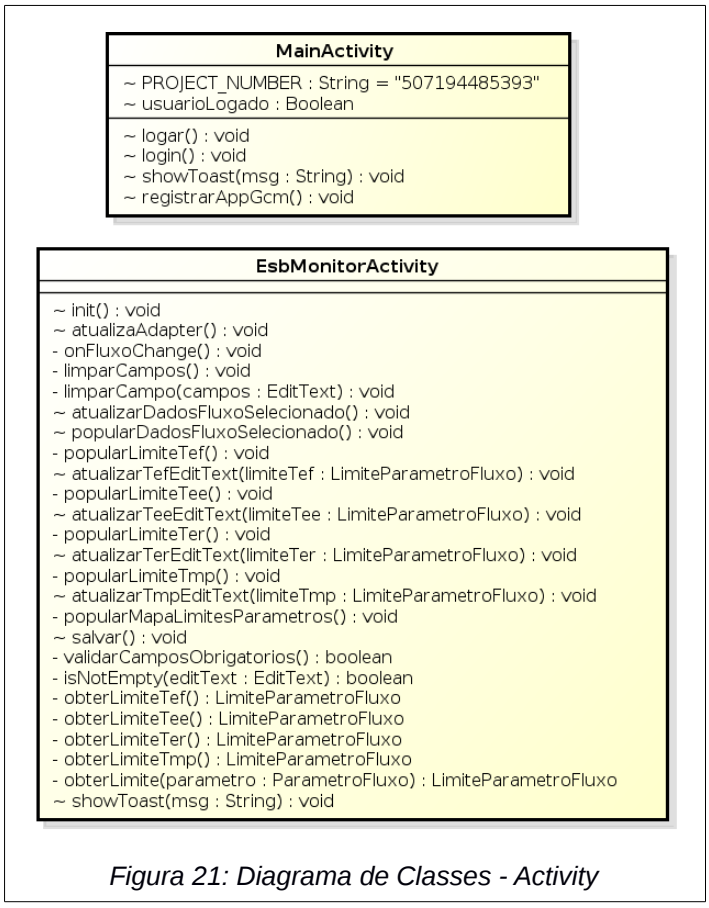

Figura 21: Diagrama de Classes - Activity

4.2.2. Classes DTO

Na integração entre o aplicativo Android e a Aplicação WEB foram utilizadas classes de transferências de dados, as quais possuem um representação com classes de domínio no sistema WEB, assim foram implementadas 3 classes (Figura 22), Fluxo, LimiteParametroFluxo e ParametroFluxo.

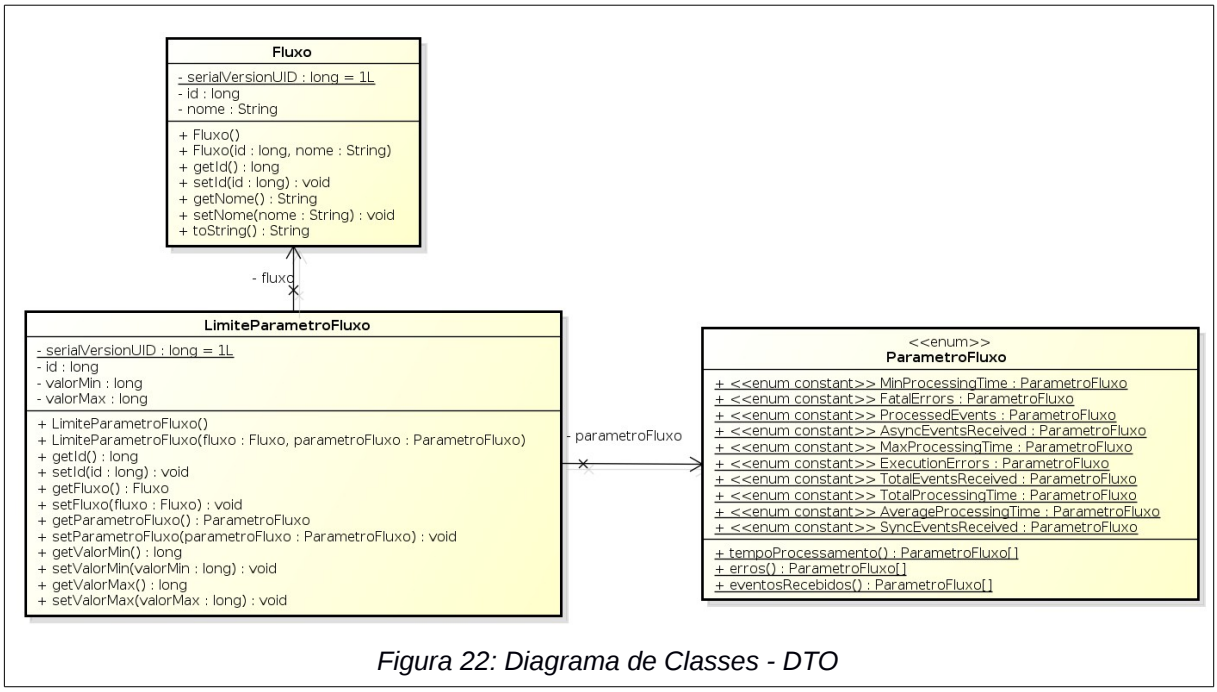

Figura 22: Diagrama de Classes - DTO

4.2.3. Classes de Implementação de Notificações via GCM

Para receber as mensagens vindas via GCM, criou-se as classes GcmBrodcastReceiver e GcmMessageHandler (Figura 23) que são responsáveis por interpretar a mensagem e criar a notificação a ser exibida para o usuário.

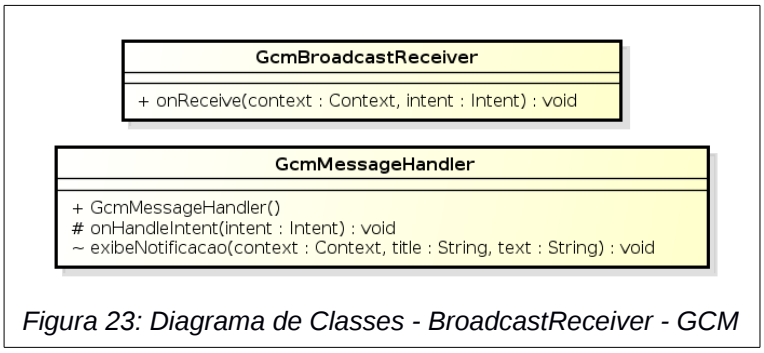

Figura 23: Diagrama de Classes - BroadcastReceiver - GCM

4.2.4. Classes e Interface para integração via REST

Como o projeto WEB possui sua camada de integração com implementação via REST e como foi utilizado o framework AndroidAnnotations, apenas foi necessário a criação da interface EsbMonitorClient que possui os métodos anotados para realizar a integração, sendo necessário também a criação de duas classes para configurações customizadas que permitem um tempo de timeout maior e especificam o tipo do conteúdo da mensagem enviada via JSON, ver Figura 24.

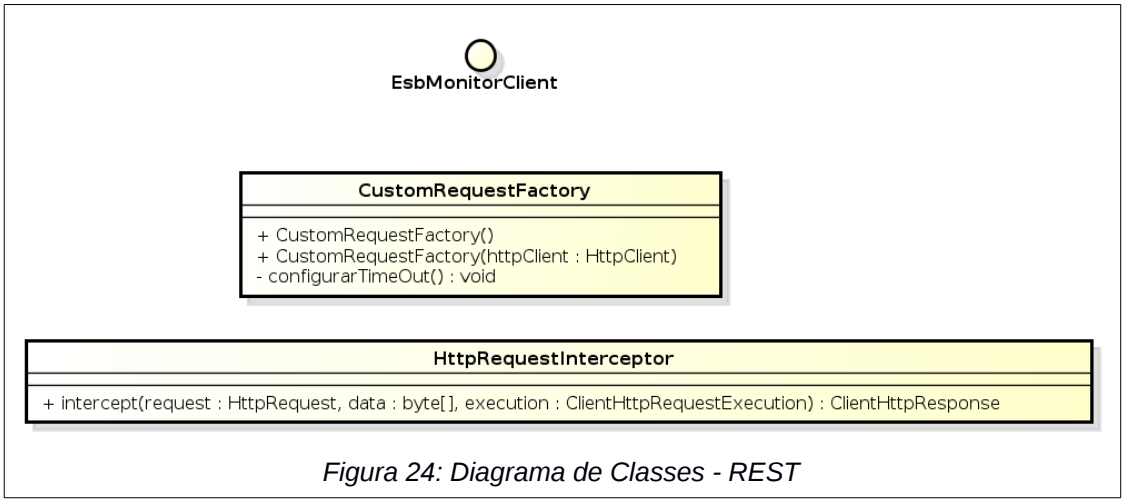

Figura 24: Diagrama de Classes - REST

5. Considerações Finais

A partir deste documento, foi possível observar a aplicação direta da teoria na prática, em que foi capaz de solucionar a necessidade de uma empresa utilizando conceitos de programação com banco de dados, desenvolvimento WEB, desenvolvimento para dispositivos móveis e interoperabilidade entre sistemas corporativos.

No decorrer do desenvolvimento foram utilizadas tecnologias que possibilitaram um desenvolvimento ágil, com a utilização do SpringBoot, o qual possui uma arquitetura com bibliotecas auxiliares, que conta com a injeção de dependências do Spring e um servidor de aplicação Tomcat embutido, e com a utilização do AndroidAnnotations, o qual tornou a codificação do aplicativo Android de forma mais robusta, utilizando anotações ao invés de sobrescrever métodos nativos. Como também, a utilização do framework JSF em conjunto com o PrimeFaces, que tornou a construção dos gráficos e telas da aplicação WEB de maneira mais direta e com uma melhor manutenibilidade.

Pode ser destacada também a utilização de novos conceitos, como o barramento de serviços corporativos (ESB), que implementa uma arquitetura SOA possibilitando a conexão entre vários serviços WEB, e o banco de dados não relacional MongoDB, o qual realiza operações de inserção e remoção mais eficientes quando comparados aos bancos de dados tradicionais.

A aplicação de monitoramento desenvolvida pode ser implantada em qualquer projeto MuleESB. Assim, essa aplicação poderá ser capaz de solucionar várias outras necessidades além da qual foi citada neste documento.

5.1. Trabalhos Futuros e Melhorias

Esta seção apresenta possíveis melhorias que podem ser necessárias na aplicação ou em trabalhos futuros que possam surgir a partir da proposta de solução apresentada neste documento.

- **Histórico de Monitoramento:** Na versão apresentada, os dados consultados do ESB são armazenados e exibidos apenas em uma faixa de tempo para o usuário. A partir dos dados armazenados, poderá ser criada uma funcionalidade que exiba o histórico do monitoramento de acordo com a necessidade do usuário.

- **Análise Inteligente dos Dados:** Os dados armazenados pela aplicação e a forma em que são exibidos podem significar vários estados em que o usuário consegue identificar do ESB, assim, pode ser criado a partir de padrões, uma forma de mostrar os dados armazenados com um significado mais inteligente e útil para o usuário.

- **Novas Notificações:** Além de poder ser notificado a partir de limites de parâmetros definidos por fluxo, novas notificações podem ser criadas a partir de determinada situação em que se encontra o ESB.

Referências bibliográficas

Pivotal Software, Spring. Disponível em: <http://spring.io/> Acesso em: 13/01/2015.

Paulo Cordeiro, Spring Boot. Disponível em: <http://www.paulocordeiro.com.br/?p=193> Acesso em: 13/01/2015.

Henrique Weissmann, Spring Boot: simplificando Spring. Disponível em: <http://www.devmedia.com.br/spring-boot-simplificando-spring-revista-java-magazine-135/31979> Acesso em: 13/01/2015.

Caelum, Introdução ao JSF e Primefaces. Disponível em: <http://www.caelum.com.br/apostila-java-testes-jsf-web-services-design-patterns/introducao-ao-jsf-e-primefaces/#7-2-caracteristicas-do-jsf> Acesso em: 14/01/2015.

Caelum, Gráficos com o Primefaces. Disponível em: <http://www.caelum.com.br/apostila-java-testes-jsf-web-services-design-patterns/graficos-interativos-com-primefaces/#9-2-graficos-com-o-primefaces> Acesso em: 14/01/2015.

Fernando Pucci, Agendando tarefas com Spring e Quartz. Disponível em: <http://www.devmedia.com.br/agendando-tarefas-com-spring-e-quartz-revista-easy-java-magazine-25/26774> Acesso em: 17/01/2015.

Higor Medeiros, Introdução ao MongoDB. Disponível em: <http://www.devmedia.com.br/introducao-ao-mongodb/30792> Acesso em: 08/02/2015.

Stefan Tikov, Uma rápida Introdução ao REST. Disponível em: <http://www.infoq.com/br/articles/rest-introduction> Acesso em: 09/02/2015.

Sérgio Júnior, Arquitetura REST com Java: JAX-RS. Disponível em: <http://blog.caelum.com.br/arquitetura-rest-com-java-jax-rs/> Acesso em: 10/02/2015.

Fernando Camargo, Construindo uma RESTful API. Disponível em: <http://www.devmedia.com.br/construindo-uma-restful-api-parte-1/29904> Acesso em: 10/02/2015.

João Longo, Boas práticas com web services RESTful. Disponível em: <http://www.devmedia.com.br/boas-praticas-com-web-services-restful-java-magazine-83/18020> Acesso em: 10/02/2015.

Oracle, What are RESTful Web Services?. Disponível em: <http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html> Acesso em: 10/02/2015.

MuleSoft Inc., What is Mule ESB?. Disponível em: <http://www.mulesoft.org/what-mule-esb> Acesso em: 03/03/2015.

Thiago Pagotto, Introdução ao Mule ESB. Disponível em: <http://www.thiagopagotto.com.br/introducao-ao-mule-esb/> Acesso em: 03/03/2015.

João Santana, Tutorial de Integração de aplicações utilizando Mule ESB. Disponível em: <http://www.das.ufsc.br/~rabelo/Ensino/DAS5316/MaterialDAS5316/ESB/TutorialESB%20(aulas-praticas).pdf> Acesso em: 03/03/2015.

José Carvilhe, Java Management Extension. Disponível em: <http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1757> Acesso em: 10/03/2015.

Thiago Vespa, Gerenciando o servidor com Server Mbean e JMX. Disponível em: <http://www.thiagovespa.com.br/blog/2012/12/17/gerenciando-o-servidor-com-server-mbean-e-jmx/> Acesso em: 10/03/2015.

Rolland Hub, Jolokia. Disponível em: <http://www.jolokia.org/> Acesso em: 21/03/2015.

Tomasz Nurkiewicz, Client-side server monitoring with Jolokia and JMX. Disponível em: <http://www.nurkiewicz.com/2012/02/client-side-server-monitoring-with.html> Acesso em: 21/03/2015.

Tomasz Nurkiewicz, Jolokia + Highcharts = JMX for human beings. Disponível em: <http://www.nurkiewicz.com/2011/03/jolokia-highcharts-jmx-for-human-beings.html> Acesso em: 21/03/2015.

Pierre-Yves Ricau, AndroidAnnotations. Disponível em: <http://androidannotations.org/> Acesso em: 27/03/2015.

Google Inc., Google Cloud Messaging for Android. Disponível em: <https://developer.android.com/google/gcm/index.html> Acesso em: 27/03/2015.

Heitor Carmo, Google Cloud Messaging: Introdução. Disponível em: <http://www.devmedia.com.br/google-cloud-messaging-introducao/29776> Acesso em: 27/03/2015.

Edson de Oliveira, Vantagens e Desvantagens de SOA. Disponível em: <http://www.devmedia.com.br/vantagens-e-desvantagens-de-soa/27437> Acesso em: 28/03/2015.