

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Arquitetura de Software Distribuído

Anderson Felipe de Paiva

SISTEMA DE CONTROLE AMBIENTAL

Belo Horizonte

2019

Anderson Felipe de Paiva

SISTEMA DE CONTROLE AMBIENTAL

Trabalho de Conclusão de Curso de Especialização
em Arquitetura de Software Distribuído como
requisito parcial à obtenção do título de especialista.

Orientador(a):

Belo Horizonte

2019

RESUMO

Este projeto aborda a arquitetura de solução para um sistema de gestão ambiental, com amplas integrações com sistemas externos como um sistema de normas ambientais e sistemas de consultorias e assessorias contratadas pelas mineradoras.

Com os recentes eventos que ocorreram em algumas regiões do Brasil, envolvendo principalmente problemas de segurança de barragens e com suas consequências catastróficas de rompimento, existe a devida preocupação da solução aumentar o controle dos processos ambientais e melhorar a comunicação com as comunidades no entorno das mineradoras. Para chegar a esse resultado, será abordado a solução arquitetônica de módulos de controle de processos minerários, monitoramento de barragens, segurança e comunicação, inteligência negocial entre outros.

Palavras-chave: arquitetura de software, mineradoras, meio-ambiente, segurança, barragens, tecnologia, informação.

SUMÁRIO

1. Objetivos do trabalho.....	5
2. Descrição geral da solução	6
2.1. Apresentação do problema.....	6
2.2. Descrição geral do software (Escopo)	6
3. Definição conceitual da solução	6
3.1. Requisitos Funcionais	6
Módulo Cadastro de Ativos.....	6
Módulo de Processos Minerários	7
Módulo de Monitoramento de Barragens	7
Módulo de Segurança e Comunicação	8
Módulo de Inteligência Negocial	8
3.2 Requisitos Não-Funcionais	8
3.3. Restrições Arquiteturais	10
3.4. Mecanismos Arquiteturais	10
4. Modelagem e projeto arquitetural.....	12
4.1. Casos de Uso.....	12
4.2. Modelo de componentes	14
4.3. Modelo de implantação	16
5. Prova de Conceito (POC) / protótipo arquitetural	17
5.1. Implementação e Implantação	17
5.2 Interfaces/ APIs.....	19
6. Avaliação da Arquitetura.....	23
6.1. Análise das abordagens arquiteturais	23
6.2. Cenários	27
6.3. Avaliação	29
6.4. Resultado.....	35
7. Conclusão.....	36
REFERÊNCIAS.....	37
APÊNDICES.....	38
CHECKLIST PARA VALIDAÇÃO DOS ITENS E ARTEFATOS DO TRABALHO ..	39

1. Objetivos do trabalho

Os objetivos do projeto é apresentar uma solução para controle de processos ambientais de empresas mineradoras, aumentando o controle e monitoria, para prover dados suficientes para resolver ou mitigar cenários de extremo risco para a sociedade e meio-ambiente.

Os objetivos específicos são:

- Implementar módulo web de cadastro de ativos, que contemplará fazer a gestão de ativos da atividade fim da empresa, além de inspeções, permitindo agendar e controlar as já realizadas e planejadas;
- Implementar módulo web de controle de processos minerários, onde será feito a gestão dos processos permitindo o controle da lista de atividades, paradas e problemas durante o turno de produção.
- Implementação de módulo de monitoramento de barragens, que consiste em um módulo web e mobile integrado com os sensores existentes nas barragens, além de permitir o monitoramento periodico por consultores contratados. Proverá integrações com os sistemas externos.
- Implementação do módulo de segurança e comunicação, que será responsável por disparar alertas quando identificado potenciais eventos catastróficos, está diretamente integrado com os dados do módulo de monitoramento de barragens.
- Implementação do módulo de inteligência negocial, que será responsável por gerar informação com base nos dados dos demais módulos, consequentemente encontrando padrões e anomalias para poder disparar alertas pelo módulo de segurança e comunicação.
- Implementação do portal de controle ambiental, que consiste em um portal que agregará todos os módulos em uma única interface, conforme os módulos implantados, visando entregar uma melhor experiência ao usuário.

2. Descrição geral da solução

2.1. Apresentação do problema

Recentes episódios de catástrofes ocasionadas por rompimentos de barragens, levam a reflexão de o quão evitável são essas tragédias, se seria possível evitar ou tomar alguma medida para que o efeito colateral desses eventos não fossem tão agressivos. Pois esses eventos acabam por gerar um débito socioambiental que terá impacto em muitas gerações, com rios extremamente poluídos por rejeitos de minérios, comunidades que perderam bens materiais tomadas pela lama, doenças devido a contaminação da água.

2.2. Descrição geral do software (Escopo)

A solução visa entregar uma melhor gestão de processos ambientais para mineradoras, buscando prover ferramentas para melhor gerir seus processos afim de entregar melhores mecanismos de monitoria, e poder mitigar os riscos de novos eventos catastróficos.

Para isso a solução proverá meios de controle de ativos das atividades de mineração, cronogramas de manutenções de equipamentos, integrações com sistemas satélites. Meios de monitoramento das barragens, com emissão de alertas, além de análise contínua utilizando algoritmos de aprendizado de máquina para gerar indicadores e informações dos processos envolvidos.

3. Definição conceitual da solução

3.1. Requisitos Funcionais

Módulo Cadastro de Ativos

- Cadastro de Equipamentos
 - O módulo deve permitir o cadastro de equipamentos e insumos utilizados nos processos de mineração, devendo indicar marca, categoria, data de fabricação e número de série.
 - O módulo deve permitir cadastrar categorias de equipamentos.
- Cronogramas de Manutenção
 - O módulo deve permitir cadastrar cronogramas de manutenção por equipamento e/ou por categoria de equipamento, indicando a categoria ou

equipamento específico, periodicidade da manutenção preventiva e itens a serem revisados.

- O módulo deve permitir indicar manutenções corretivas em determinado equipamento, indicando o equipamento, data da manutenção e atividades e itens trocados na manutenção.
- Integração com o Sistema de Aquisições
 - O módulo deve permitir a integração automática com o sistema de aquisições da corporação, gerando lançamentos de possíveis aquisições necessárias.

Módulo de Processos Minerários

- Fluxo de Trabalho
 - O módulo deve permitir o cadastro de fluxos de trabalho, contemplando listas de atividades diárias, setor e turno.
 - O módulo deve permitir cadastrar setores envolvidos nos processos de mineração.
 - O módulo deve permitir visualizar graficamente o fluxo das atividades cadastradas.
- Gestão de Ocorrências
 - O módulo deve permitir registrar as paradas que ocorreram durante o dia, sendo obrigatório indicar o motivo da parada e horário.

Módulo de Monitoramento de Barragens

- Integração com Sensores das Barragens
 - O módulo deve receber dados dos sensores das barragens.
 - O módulo deve permitir visualizar relatório dos indicadores recebidos dos sensores das barragens.
- Integração com o Módulo de Segurança e Comunicação
 - O módulo deve permitir parametrizar condições para gerar as mensagens de alerta para a equipe de engenheiros da corporação, para assim eles decidirem emissão de um alerta para interessados externos à corporação.
- Integração Sistemas Externos
 - O módulo deve possuir fácil integração com sistemas externos.

Módulo de Segurança e Comunicação

- Emissão de Alertas
 - O módulo deve permitir a emissão de alertas para diferentes perfis, como engenheiros e comunidade por exemplo.
 - O módulo deve permitir emitir alertas por SMS.
 - O módulo deve permitir emitir alertas manualmente, sendo obrigatório indicar um motivo.

Módulo de Inteligência Negocial

- Dashboards
 - O módulo deve permitir montar relatórios e dashboards.
- Aprendizado de máquina:
 - O módulo deve possuir a capacidade de aplicar aprendizado de máquina para gerar *insights* para outros componentes da arquitetura.

3.2 Requisitos Não-Funcionais

A seguir os requisitos não funcionais da solução:

- Acessabilidade – A solução deve ter responsabilidade para dispositivos móveis

Estímulo	Usuário visualizando relatórios de monitoramento de barragens.
Fonte do Estímulo	Usuário visualizando os relatórios de monitoramento de barragens por um dispositivo móvel.
Ambiente	Funcionamento, carga normal.
Artefato	Módulo de Monitoramento de Barragens.
Resposta	A camada de apresentação se adaptou as resoluções do dispositivo, permitindo o usuário interagir com as informações de forma fluída.
Medida de Resposta	Redimensionamento e reorganizações dos

	componentes conforme a resolução do dispositivo.
--	--

- Resiliência – A solução deve conseguir se recuperar de falhas transientes nas integrações entre os sistemas

Estímulo	Sensor enviando dados de monitoramento para o Módulo de Monitoramento de Barragens.
Fonte do Estímulo	Sensor de barragem envia dados de leitura.
Ambiente	Módulo Segurança e Comunicação indisponível.
Artefato	Módulo de Monitoramento de Barragens e Segurança e Comunicação.
Resposta	O módulo de monitoramento de barragens deverá publicar a mensagem no message broker com sucesso.
Medida de Resposta	O alerta deve ser recebido assim que o microserviço Segurança e Comunicação estiver normalizado.

- Segurança – A solução deve apresentar altos padrões de segurança

Estímulo	Acessar uma página privada sem autenticação.
Fonte do Estímulo	Usuário administrador
Ambiente	Em funcionamento e com carga normal.
Artefato	Todos os módulos.
Resposta	Usuário deve ser redirecionado para a página de autenticação.
Medida de Resposta	O sistema não deve permitir o acesso a páginas privadas.

Estímulo	Enviar uma requisição HTTP sem autenticação.
Fonte do Estímulo	Aplicação Externa
Ambiente	Em funcionamento e com carga normal.
Artefato	Todos os módulos.
Resposta	Deve ser respondido com HTTP 401.
Medida de Resposta	A solução não deve permitir acesso a recursos que exijam autenticação.

- Escalabilidade – A solução deve ser possível escalar horizontalmente

Estímulo	Alta carga de envio de SMS.
Fonte do Estímulo	Indicador crítico de barragens.
Ambiente	Alta volumetria de mensagens.
Artefato	Segurança e comunicação.
Resposta	A solução deve notificar todos os contatos.
Medida de Resposta	Enviar vinte e cinco mil mensagens em menos de 10 minutos.

3.3. Restrições Arquiteturais

- O sistema deve abrir de forma responsiva em dispositivos móveis.
- O sistema deve ser modular para facilitar implantação e manutenção.

3.4. Mecanismos Arquiteturais

Mecanismo de Análise	Mecanismo de Design	Mecanismo de Implementação
Persistência	Banco de dados orientado a documentos.	MongoDB
Versionamento	Versionamento do código da aplicação	Git
Barramento de Mensagens	Integrações entre os	RabbitMQ

	componentes através de mensagens	
Front-End	Interface de comunicação com o usuário do sistema	VueJS e Vuetify
Integração Contínua	Solução para automação de compilação e publicação dos artefatos nos servidores	Azure DevOps
Serviço de Descoberta	Registro de Serviços	Eureka
Log	Solução para gestão de logs	Slf4j para gerar o log em arquivo texto, LogStash para tratamento dos logs, Elasticsearch para armazenamento e Kibana para visualização em interfaces gráficas.
Servidor de Aplicação .Net	Servidor de aplicação para os componentes .Net	Kestrel
Servidor de Aplicação Java	Servidor de aplicação para os componentes Java	Jetty
Integração entre os módulos	Tipo de conteúdo padrão para integração entre os módulos	JSON
Autenticação e Autorização	Protocolo de autenticação e autorização da solução	OpenID
Configuração	Solução para gestão dos arquivos de configuração dos módulos	Spring Boot ConfigServer
Injeção de Dependências	Solução para gestão de ciclo de vida de objetos e injeção de dependências.	Spring
Aplicação Gateway	Solução para implementar aplicação Gateway para centralização de requisições	Ocelot
Servidor de Identidade	Framework para implementação de autenticação e autorização	IdentityServer 4

Documentação de API	Solução para documentação das APIs do projeto	Swagger
Inteligência Negocial	Solução para todo o processo de Inteligência Negocial	Pentaho
Persistências para os relatórios	Solução para persistir os dados consolidados para a montagem de relatórios	PostgreSQL
Persistência Big Data	Solução de persistência para o processo de Big Data	Hbase
Componente Gráfico para fluxos	Solução para visualização de fluxos de trabalho	Vue Flowy

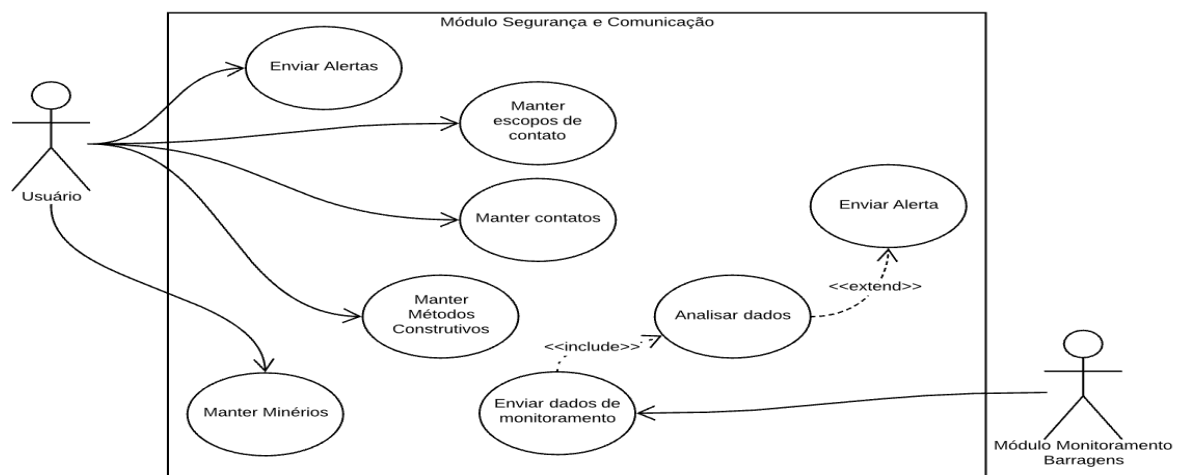
4. Modelagem e projeto arquitetural

Nesta seção são apresentados os diagramas que permitem entender a arquitetura da aplicação, detalhando-a suficientemente para viabilizar sua implementação.

4.1. Casos de Uso

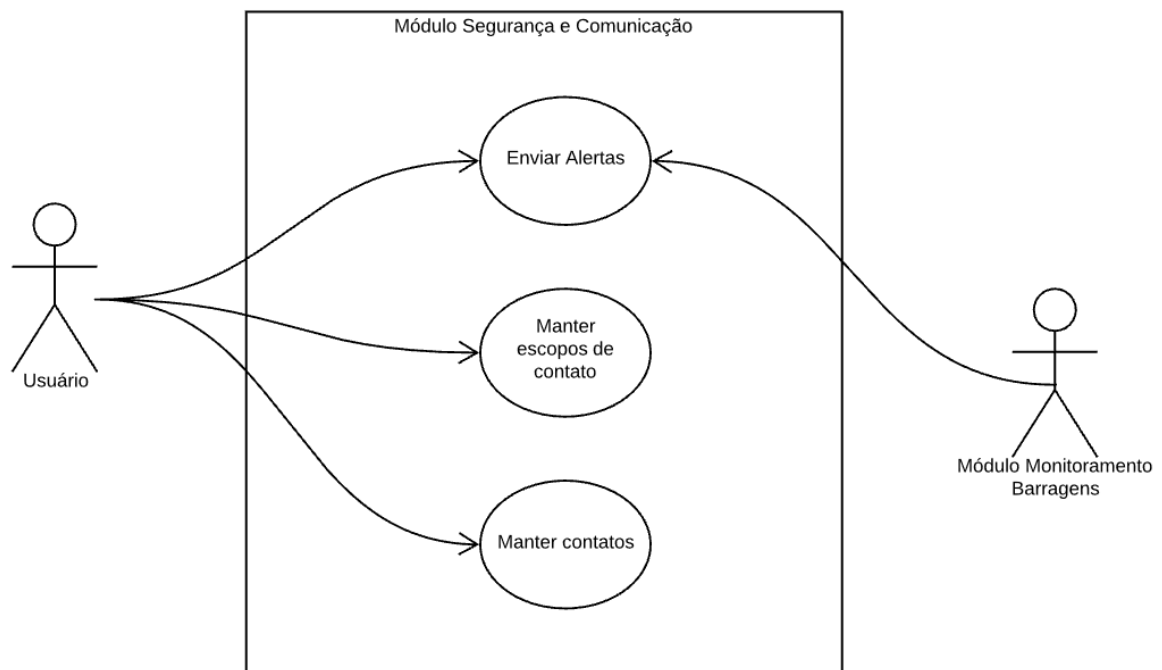
Abaixo estão listados os casos de uso chaves para essa prova de conceito. Foram segmentados por módulos para melhor entendimento.

Módulo Monitoramento de Barragens



Este caso de uso ilustra as funcionalidades envolvidas no módulo de Monitoramento de Barragens que foram necessários para a elaboração dessa prova de conceito. Consiste em funcionalidades de manter parâmetros de alerta, que será utilizados pelo caso de uso analisar dados, manter barragens, manter métodos construtivos e manter minérios. O ator sensor de barragem envia dados de monitoramento para o módulo de Monitoramento de Barragens, que irá analisá-los conforme os parâmetros de alerta, e se atender algum desses parâmetros irá enviar o alerta, que tecnicamente consiste em uma integração com o módulo Segurança e Comunicação.

Módulo Segurança e Comunicação

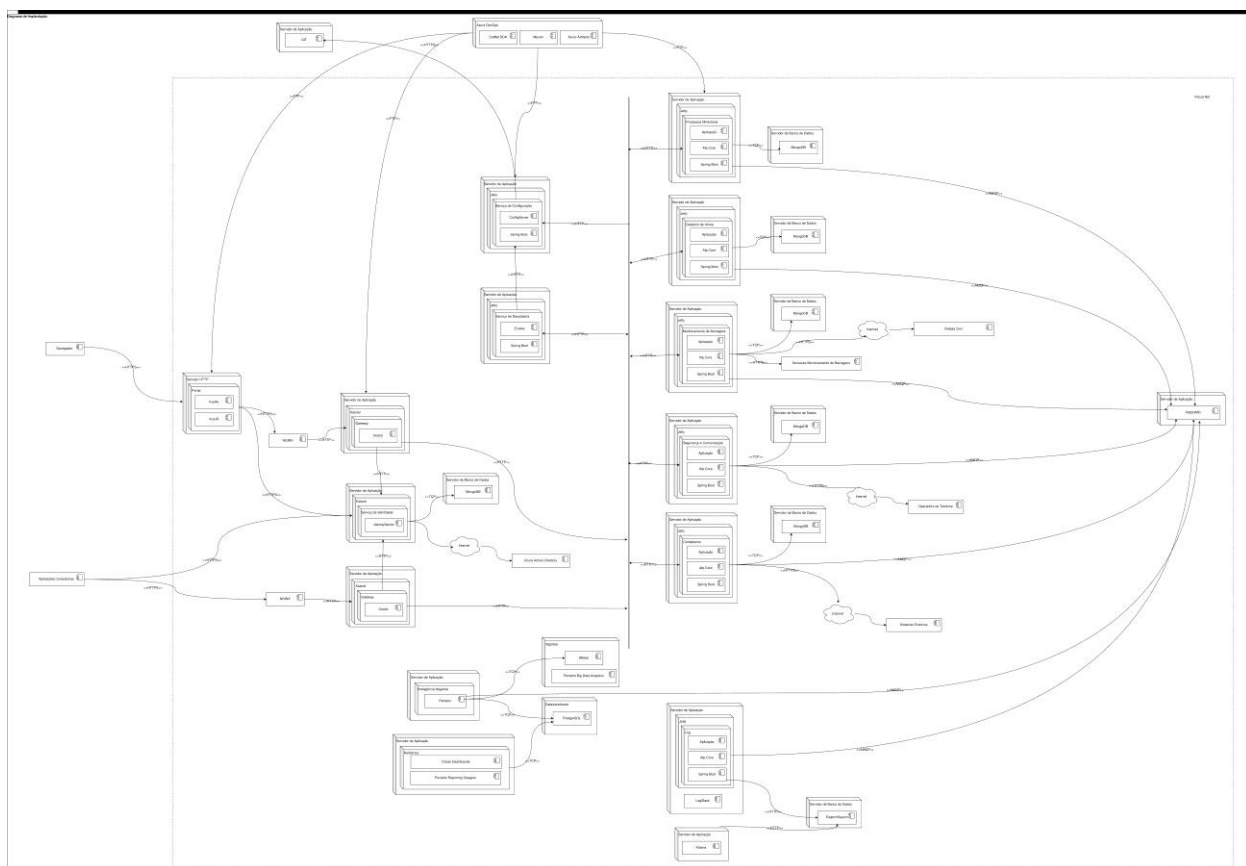


Este caso de uso ilustra as funcionalidades envolvidas no módulo de Segurança e Comunicação que foram necessários para a elaboração dessa prova de conceito. Consiste em funcionalidades de manter escopos de contato, manter contatos e enviar alertas. O caso de uso enviar alertas também é invocado pelo módulo de Monitoramento de Barragens.

Microserviço SCA	Representa todos os microserviços que implementam as regras do Sistema de Controle Ambiental. Está implementado em Java 8, sobre o framework Spring Boot. Todos os serviços consomem uma biblioteca kernel que é o Afp.Core, essa biblioteca possui implementações abstratas que são reaproveitadas em todos os serviços, como classes de repositório genéricas, configuração do Swagger para documentação de APIs, registro no Serviço de Descoberta Eureka, handler para exceções não tratadas entre outras funcionalidades transversais.
Afp.Log	Microserviço de log transversal, consome uma fila do RabbitMQ que receberá as mensagens publicadas pelo Afp.Core, os gravará em um arquivo com extensão log, que será tratado e persistido pelo LogStash no Elasticsearch, para posteriormente ser possível a elaboração de relatórios utilizando o Kibana.
Pentaho	Solução de código aberto para inteligência negocial, será responsável pelo processo de extração, transformação e carga dos dados, utilizando a ferramenta Data Integration. A extração dos dados podem ser consumidos das filas do RabbitMQ, ou diretamente das bases de dados dos microserviços SCA. Para os dados estruturados e a elaboração de Dashboards (Ctools) será persistido na base PostgreSQL, e no Hbase para dados não estruturados que será consumido pelo Pentaho Big Data Analytics.
RabbitMQ	Message Broker que exerce o papel de barramento de eventos. Os microserviços SCA publicam eventos que são consumidos pelos outros componentes, buscando sempre o menor acoplamento possível entre os microserviços.
Afp.DiscoveryServer	Serviço de descoberta implementado utilizando o Spring Boot e Eureka. O serviço é responsável por prover o endereço de todos os microserviços da arquitetura SCA, assim quando necessário realizar uma requisição, o microserviço necessita

	conhecer apenas o Serviço de Descoberta.
Afp.ConfigServer	Serviço de configuração implementado utilizando Spring Boot ConfigServer. O serviço é responsável pela centralização de arquivos de configuração, os consumindo de um repositório Git.
Azure Active Directory	Utilizado como provedor de autenticação, pode ser exportado todas as contas do Active Directory da empresa para o Azure e reaproveitar as contas para o login na aplicação.

4.3. Modelo de implantação



Visualizar em alta resolução: <https://sendeyo.com/up/d/21ea55aa6b>

Modalidade de execução em nuvem, abaixo a descrição dos artefatos.

Componente	Descrição
Servidores de Aplicação dos microserviços SCA	Máquinas virtuais Windows que hospedarão as aplicações em Java e .Net Core.
Servidor de Banco de Dados	Representa a máquina virtual de hospedagem do MongoDB, PostgreSQL ou ElasticSearch.
Virtual Network	Representa uma rede que irá envolver os componentes, não permitindo acesso externo diretamente a eles, somente as aplicações Gateways, que farão o redirecionamento para as APIs.
NGINX	Balanceador de Carga
Azure DevOps	Responsável pelo processo de geração dos artefatos compilados e publicação dos mesmos, solução em nuvem.
Servidor de Aplicação das Soluções de Inteligência Negocial.	Essas soluções serão executadas em máquinas virtuais na nuvem.
Servidor de Aplicação RabbitMQ	Máquina virtual Linux com o RabbitMQ instalado.

O quadro acima descreve os componentes e ambiente de hardware utilizado para a execução da prova de conceito. Em um ambiente de produção o ideal seria alterar as aplicações que compõem a arquitetura para contêineres Docker e delegar a gestão de escalabilidade para uma solução como o Kubernetes, assim sendo possível automatizar de forma mais facilitada a operação de escalabilidade sob demanda das aplicações.

5. Prova de Conceito (POC) / protótipo arquitetural

5.1. Implementação e Implantação

A prova de conceito desse projeto visa atender as necessidades críticas dos módulos de Monitoramento de Barragens, Segurança e Comunicação, Portal e Serviço de Identidade. A seguir será demonstrado evidências dos testes sobre o protótipo, visando atender os requisitos não funcionais de segurança, acessabilidade, escalabilidade e resiliência.

- As tecnologias utilizadas na sua implementação estão descritas no item de 3.4 Mecanismos Arquiteturais.
- Para os cenários que há interação com interface gráfica, podem ser vistas nas evidências.

Seguem os requisitos não funcionais que pretende-se validar:

Acessabilidade

Requisito não funcional escolhido devido a preocupação em garantir que o sistema seja responsivo a dispositivos móveis.

Critérios de aceite:

- As telas e componentes devem apresentar facilidade de interação em dispositivos móveis.
- O sistema deve manter-se com os mesmos padrões de cores e objetos.

Segurança

Requisito não funcional escolhido devido a preocupação manter os dados seguros e evitar vulnerabilidades de segurança no projeto.

Critérios de aceite:

- Usuário não autenticado não pode ter acesso ao Portal SCA, deve ser redirecionado para a tela de login do provedor de autenticação.
- Cliente aplicativo, como aplicações externas integrando com componentes de nossa arquitetura, não podem conseguir fazer requisições sem um token válido.

Resiliência

Requisito não funcional escolhido devido a preocupação em manter um baixo acoplamento entre os componentes arquiteturais, e que a indisponibilidade de um deles não impacte nos

demais componentes além de, assim que normalizado o funcionamento, as integrações que ocorreram no momento da indisponibilidade não sejam perdidas.

Critérios de aceite:

- O envio de uma mensagem do módulo Monitoramento e Barragens para o disparo de alertas por parte do módulo Segurança e Comunicação não deve ser perdido enquanto o módulo de Segurança e Comunicação estiver indisponível, e nem ocasionar qualquer comportamento inesperado no módulo Monitoramento de Barragens.
- Assim que o módulo Segurança e Comunicação estiver com seu funcionamento normalizado, deve processar as mensagens de alerta.

Escalabilidade

Requisito não funcional escolhido devido a preocupação em que a solução possua escalabilidade horizontal facilitada, sem que gere conflitos entre as diferentes instâncias, e nem comportamentos inesperados.

Critérios de aceite:

- Em alta volumetria de mensagens de alerta, o sistema deve suportar processar as mensagens com múltiplas instâncias do módulo Segurança e Comunicação, sem exceções e conflitos entre uma instância e outra.

5.2 Interfaces/ APIs

Afp.Core

Biblioteca base utilizada em todos os módulos em Java do projeto, possui classes base com código genérico para ser reaproveitado. Para utilizar basta adicionar no arquivo pom.xml a referência da biblioteca:

```

<dependency>
  <groupId>br.com.andersondepaiva</groupId>
  <artifactId>core</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</dependency>

```

Possuí classes e implementações genéricas para reaproveitar como:

```


v core [pucminas-tcc master]
  v src/main/java
    > br.com.andersondepaiva.core
    v br.com.andersondepaiva.core.business
      > Business.java
      > IBusiness.java
    > br.com.andersondepaiva.core.client
    v br.com.andersondepaiva.core.dto
      > BaseDto.java
      > Comparasion.java
      > ParamDto.java
      > PessoaDto.java
      > UserChangeDto.java
      > UsuarioDto.java
    > br.com.andersondepaiva.core.infra.controller
    > br.com.andersondepaiva.core.infra.exception
    v br.com.andersondepaiva.core.infra.exception.model
      > BaseException.java
      > BusinessException.java
      > ErrorDto.java
      > ErrorResponse.java
      > ExceptionDto.java
    v br.com.andersondepaiva.core.infra.exception.producer
      > ExceptionProducer.java
    > br.com.andersondepaiva.core.infra.exception.producer.interfaces
    > br.com.andersondepaiva.core.infra.persistence
    > br.com.andersondepaiva.core.infra.security
    v br.com.andersondepaiva.core.infra.utils
      > JsonService.java
    v br.com.andersondepaiva.core.model
      > BaseModel.java
      > UserChange.java
    v br.com.andersondepaiva.core.mq
      > Consumer.java
      > CustomMessagePostProcessor.java
      > IConsumer.java
      > IProducer.java
      > Producer.java
    v br.com.andersondepaiva.core.repository
      > IRepositoryMongoDb.java

```

- Classes base para o modelo, com propriedades de InseridoEm, AlteradoEm e outras abstrações.
- Código interceptador de exceções na aplicação, para gerar mensagens no RabbitMQ que serão consumidas pelo serviço de log transversal.
- Classes base para mensageria.
- Classes base para os objetos de transferência de dados.
- Classes de repositórios genéricas para o MongoDB
- Operações de escrita, leitura, exclusão e atualização genéricas, necessitando apenas implementar a classe modelo e de transferência de dados, e consumir nos controladores

Segurança e Comunicação

Abaixo o swagger utilizado para documentar a API, podendo ser reutilizado para qualquer integração externa:


swagger
default (/v2/api-docs) ▾
Explore

Api Documentation

Api Documentation

[Apache 2.0](#)

contato-controller : Contato Controller
Show/Hide
List Operations
Expand Operations

GET	/contatos	GetAll
POST	/contatos	Post
GET	/contatos/{id}	GetById

escopo-controller : Escopo Controller
Show/Hide
List Operations
Expand Operations


GET	/escopos	GetAll
POST	/escopos	Post
GET	/escopos/{id}	GetById

solicitacao-mensagem-controller : Solicitacao Mensagem Controller
Show/Hide
List Operations
Expand Operations

GET	/solicitacoes-mensagem	GetAll
POST	/solicitacoes-mensagem	Post
GET	/solicitacoes-mensagem/{id}	GetById

Monitoramento de Barragens

Abaixo o swagger utilizado para documentar a API, podendo ser reutilizado para qualquer integração externa:


swagger
default (/v2/api-docs) ▾
Explore

Api Documentation

Api Documentation

[Apache 2.0](#)

barragem-controller : Barragem Controller

Show/Hide | List Operations | Expand Operations

GET	/barragens	GetAll
POST	/barragens	Post
GET	/barragens/{id}	GetById

metodo-contrutivo-controller : Metodo Construtivo Controller

Show/Hide | List Operations | Expand Operations

GET	/metodos-contrutivos	GetAll
POST	/metodos-contrutivos	Post
GET	/metodos-contrutivos/{id}	GetById

minerio-controller : Minerio Controller

Show/Hide | List Operations | Expand Operations

GET	/minerios	GetAll
POST	/minerios	Post
GET	/minerios/{id}	GetById

monitoramento-barragem-controller : Monitoramento Barragem Controller

Show/Hide | List Operations | Expand Operations

GET	/monitoramento-barragens	GetAll
POST	/monitoramento-barragens	Post
GET	/monitoramento-barragens/barragem/{id}	GetAllByBarragem
GET	/monitoramento-barragens/{id}	GetById

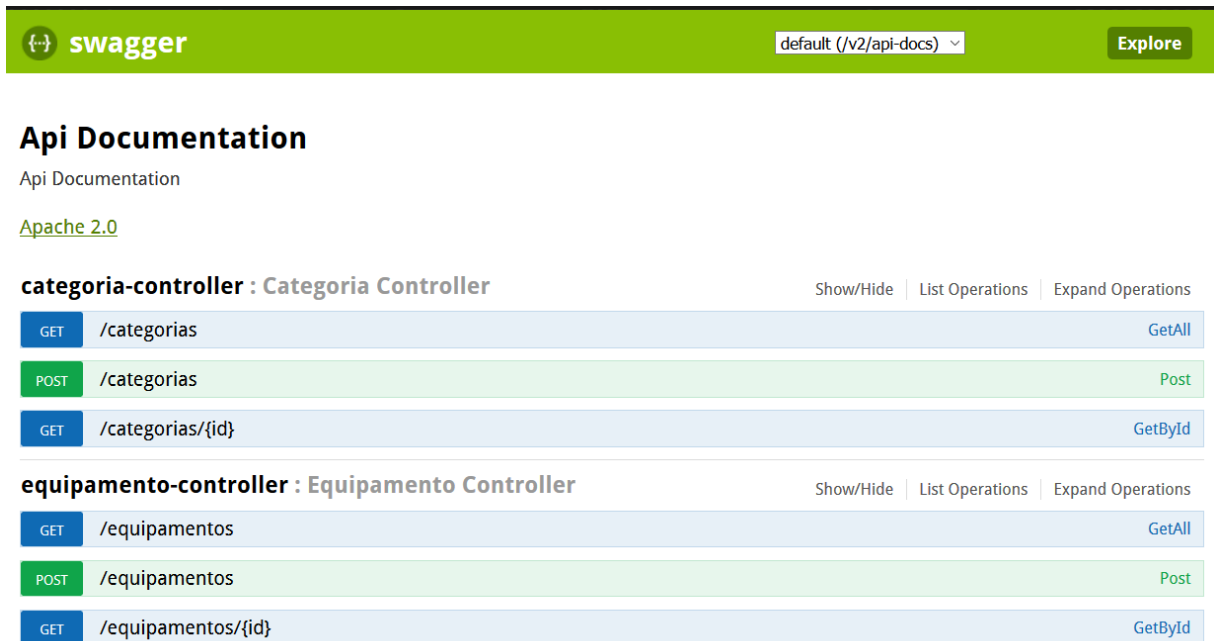
parametro-alerta-controller : Parametro Alerta Controller

Show/Hide | List Operations | Expand Operations

GET	/parametros-alerta	GetAll
POST	/parametros-alerta	Post
GET	/parametros-alerta/{id}	GetById

Cadastro de Ativos

Abaixo o swagger utilizado para documentar a API, podendo ser reutilizado para qualquer integração externa:



The image shows a Swagger UI interface for an API. At the top, there's a green header with the Swagger logo, a dropdown menu set to 'default (/v2/api-docs)', and an 'Explore' button. Below the header, the title 'Api Documentation' is displayed, followed by 'Api Documentation' and a link to 'Apache 2.0'. The main content area lists two controllers: 'categoria-controller : Categoria Controller' and 'equipamento-controller : Equipamento Controller'. Each controller has a table of operations. For 'categoria-controller', the operations are: GET /categorias (GetAll), POST /categorias (Post), and GET /categorias/{id} (GetById). For 'equipamento-controller', the operations are: GET /equipamentos (GetAll), POST /equipamentos (Post), and GET /equipamentos/{id} (GetById). Each operation row has a colored button (blue for GET, green for POST) and a link to the operation details.

Quando necessário integrar com qualquer aplicação externa, deverá ser exposto pelo gateway, gerando o devido `client_id` e `client_secret` para a aplicação do parceiro e configurando somente as rotas necessárias para a referida integração.

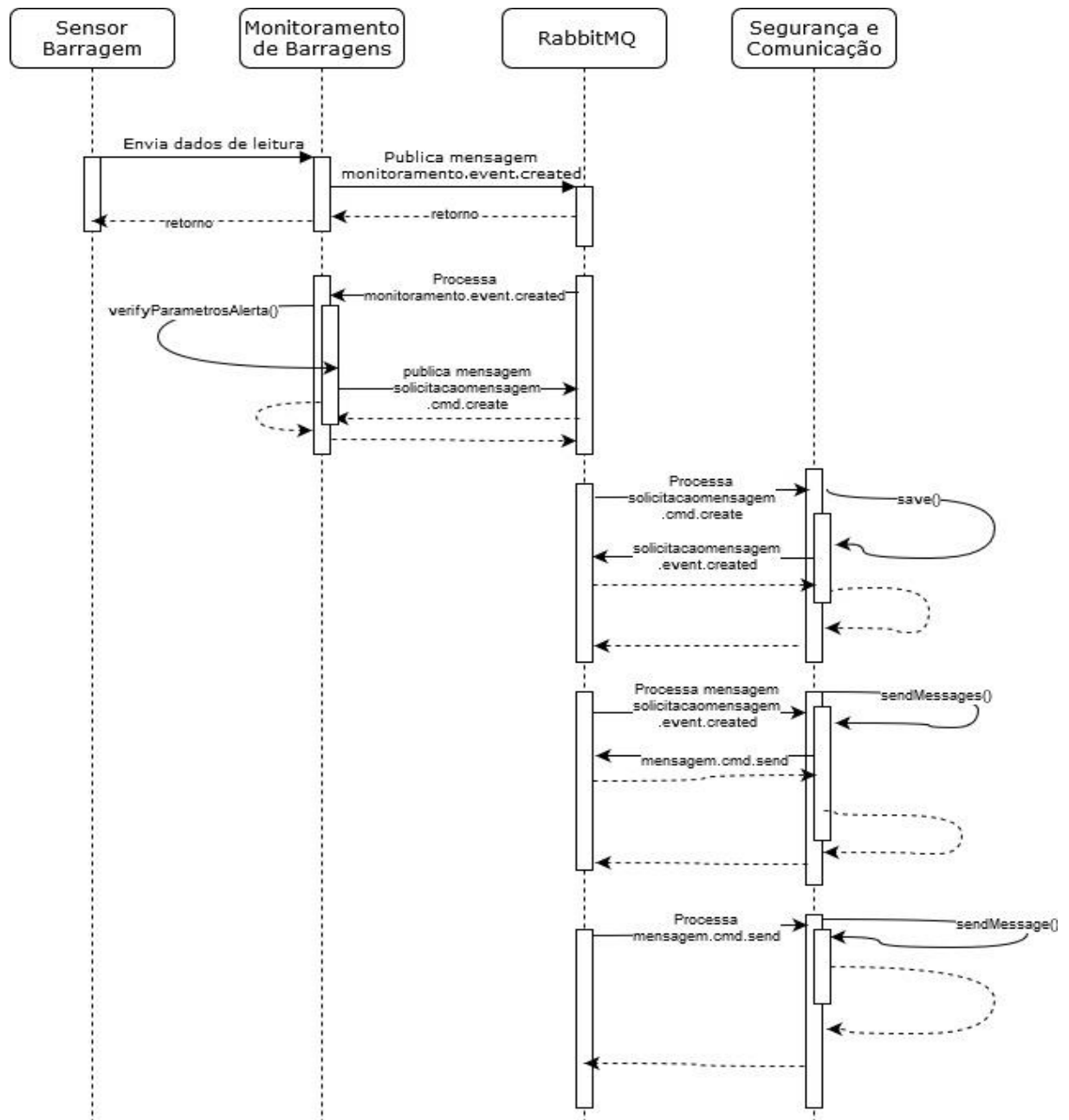
6. Avaliação da Arquitetura

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção, visando avaliar se atende ao que foi proposto.

6.1. Análise das abordagens arquiteturais

As principais características dessa abordagem arquitetural é entregar uma solução com baixo acoplamento entre os componentes, resiliência nas operações, manutenabilidade e segurança. Uma das principais abordagens que permite essa entrega é a utilização de mensageria utilizando o RabbitMQ, que nos de uma forma fácil políticas de retentativas, mensagens persistentes e desempenho. Para a utilização buscamos seguir uma abordagem de arquitetura orientada a eventos, onde cada operação gera um evento de referente ao context

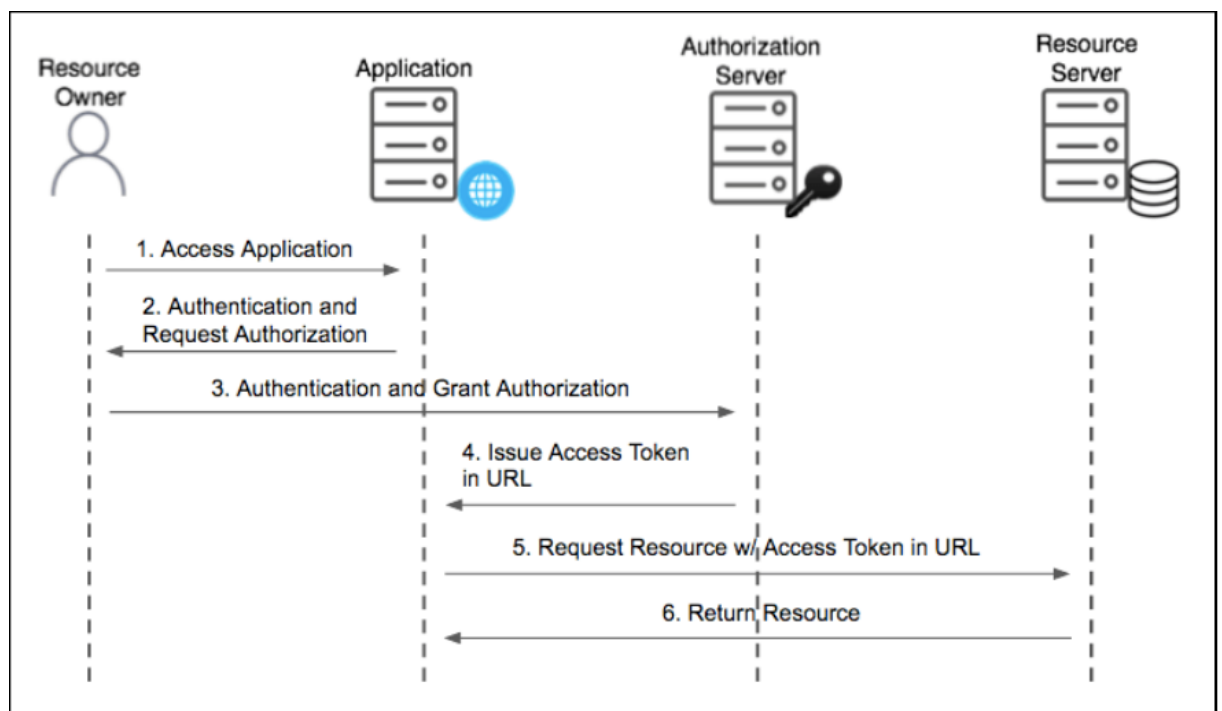
daquele domínio, abaixo um diagrama de sequência explicando essa abordagem sobre a funcionalidade de envio de alertas:



No protótipo o consumidor de mensagens está na mesma aplicação das Web APIs, mas em produção para facilitar a escalabilidade e performance, deve-se separar o consumidor da aplicação que responde as requisições, assim conseguimos entregar desempenho e escalabilidade em processamentos em segundo plano, sem impactar na responsividade das requisições HTTP.

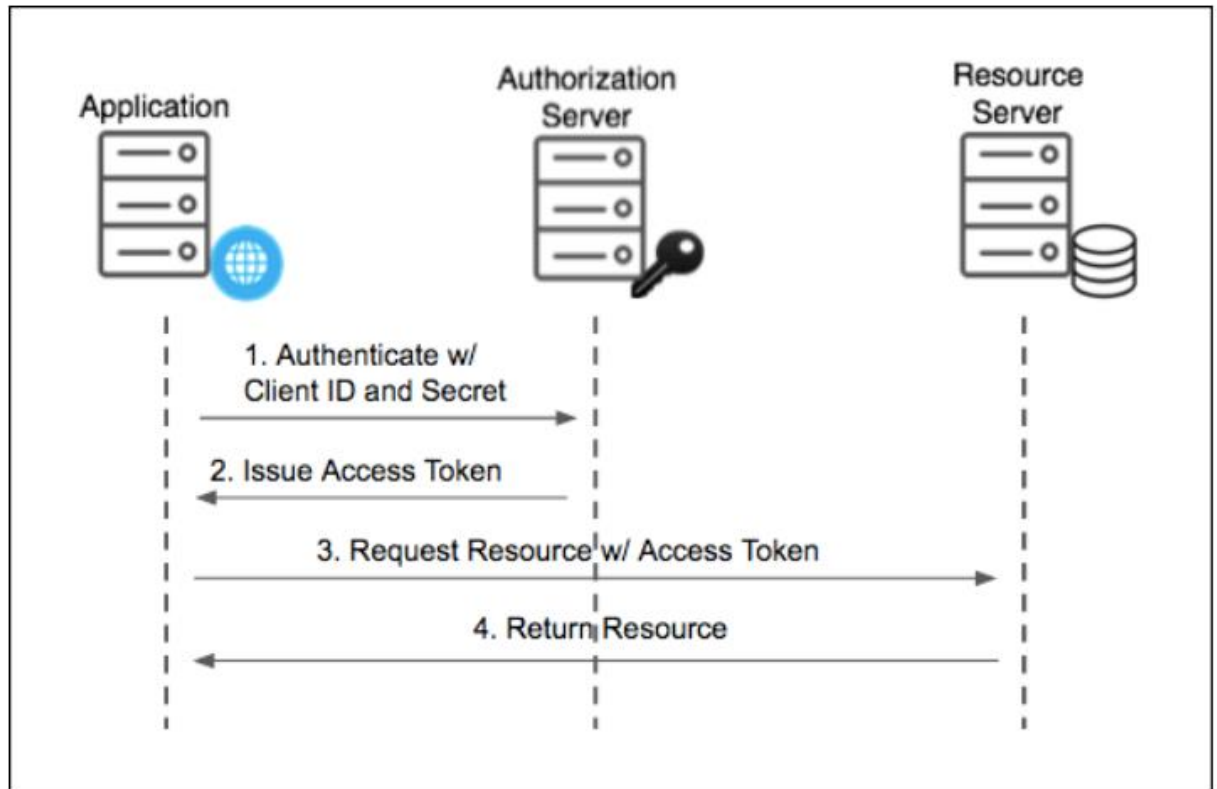
No quesito segurança foi escolhido implementar um serviço de identidade para abstrair do provedor de autenticação, assim podendo integrar com os provedores mais utilizados no mercado, como Google Cloud, Azure e Amazon, sem impactar nos demais componentes da arquitetura. Com a implementação do protocolo OpenId, foram escolhidos dois fluxos de autenticação, um para as aplicação SPA e outra para clientes aplicativos externos.

Para aplicações SPA foi escolhido o fluxo implícito que é explicado no diagrama abaixo montado pela empresa Pivotal:



Fonte: <https://docs.pivotal.io/p-identity/1-9/configure-apps/single-page-js-app.html>

Para aplicações externas foi escolhido o fluxo de credenciais de cliente que é explicado no diagrama abaixo montado pela empresa Pivotal:



Fonte: <https://docs.pivotal.io/p-identity/1-9/configure-apps/service-to-service-app.html>

A aplicação gateway que é responsável por validar o token da requisição, assim como os escopos do token conforme a rota consumida. Para esse quesito foi implementado o padrão Backends for Frontends, que busca ter uma aplicação gateway para cada tipo de cliente aplicativo, assim podemos definir escopos específicos no JWT para cada tipo de cliente.

Para elaboração e visualização de relatórios, foi escolhido a suíte Pentaho, que consiste em uma solução de código aberto que possui todos os componentes necessários para elaboração da extração, transformação e carga dos dados, e elaboração dos relatórios sobre essas visões, assim como a abordagem de Big Data que é possível através do módulo Pentaho Big Data Analytics.

Para o processo de DevOps foi utilizado o Azure DevOps, devido ser uma solução em nuvem e muito versátil. Durante a prova de conceito foram criados automações para a plataforma Heroku:

The image shows two screenshots from the Azure DevOps web interface. The top screenshot displays the 'All builds' history for a project, listing various builds with their commit IDs, build numbers, pipeline names, branches, and durations. The bottom screenshot shows the configuration of a specific pipeline named 'Afp.MonitoramentoBarragens-CI', including its tasks (Get sources, Agent job 1, Maven, Copy Files, Publish Artifact) and the configuration for the agent pool (Azure Pipelines, ubuntu-16.04).

Commit	Build #	Pipeline	Branch	Queued ↓	Duration
heroku Manual build for Anderson De Paiva	20190902.1	Afp.SegurancaComuni...	master	2019-09-02 19:33	5:00.113
Added Heroku Manual build for Anderson De Paiva	20190902.1	Afp.CadastroAtivos-CI	master	2019-09-02 19:22	5:01.655
configserver Manual build for Anderson De Paiva	20190825.7	Afp.DiscoveryServer-CI	master	2019-08-25 22:50	2:05.909
https://sca-discoveryserver.herokuapp.com/eureka Manual build for Anderson De Paiva	20190825.6	Afp.DiscoveryServer-CI	master	2019-08-25 22:39	2:00.762
https://sca-discoveryserver.herokuapp.com/eureka Manual build for Anderson De Paiva	20190825.5	Afp.ConfigServer-CI	master	2019-08-25 22:39	3:12.913
port9012 Manual build for Anderson De Paiva	20190825.22	Afp.MonitoramentoBa...	master	2019-08-25 22:25	4:27.477
\$(PORT) Manual build for Anderson De Paiva	20190825.21	Afp.MonitoramentoBa...	master	2019-08-25 22:17	4:22.846
\$(PORT) Manual build for Anderson De Paiva	20190825.4	Afp.ConfigServer-CI	master	2019-08-25 22:16	2:58.230
\$(PORT) Manual build for Anderson De Paiva	20190825.5	Afp.DiscoveryServer-CI	master	2019-08-25 22:14	2:01.744
port Manual build for Anderson De Paiva	20190825.3	Afp.ConfigServer-CI	master	2019-08-25 22:04	3:01.507
port Manual build for Anderson De Paiva	20190825.4	Afp.DiscoveryServer-CI	master	2019-08-25 22:04	2:10.565

Pipeline Configuration: Afp.MonitoramentoBarragens-CI

- Name:** Afp.MonitoramentoBarragens-CI
- Agent pool:** Azure Pipelines
- Agent Specification:** ubuntu-16.04
- Parameters:**
 - Maven POM file: Afp.MonitoramentoBarragens/pom.xml
- Tasks:**
 - Get sources (Repository: andersondepaiva/pucminas-etc, Branch: master)
 - Agent job 1 (Run on agent)
 - Maven (Maven)
 - Copy Files to: \$(build.artifactstagingdirectory)
 - Publish Artifact: drop (Publish build artifacts)

6.2. Cenários

Cenário 1 – Acessabilidade

Ao acessar o Portal SCA por dispositivos com diferentes resoluções o sistema deve entregar uma facilidade de interação com os componentes da tela e os mesmos padrões de objetos e cores, visando entregar usabilidade ao usuário.

Cenário 2 – Segurança

Ao tentar acessar o Portal SCA sem estar autenticado, o usuário deve ser redirecionado para a página de login do provedor de autenticação (Azure AD), e após a autenticação deve ser novamente redirecionado para a página principal do Portal SCA. Assim garantindo acesso ao Portal e suas informações somente quem possui a devida permissão.

Cenário 3 – Segurança

O cliente aplicativo, sendo uma aplicação externa que integra com a nossa solução, ao tentar efetuar uma requisição HTTP sem um token válido, deve receber como resposta o código HTTP 401. E após devida autenticação com seu `client_id`, `client_secret` e escopos de acesso, deve receber os dados. Assim também garantindo segurança nas integrações com aplicações externas.

Cenário 4 – Resiliência

Simular o envio de dados de leitura por parte de um sensor de barragem para o módulo Monitoramento de Barragens, enquanto o módulo de Segurança e Comunicação encontra-se fora de operação, deve processar os dados normalmente e ao identificar uma anomalia deve propagar uma mensagem no RabbitMQ para o envio de alertas, que será processado pelo módulo de Segurança e Comunicação assim que estiver com seu funcionamento normalizado. Esse cenário visa testar que a indisponibilidade em um componente não impacta em outro, assim como as integrações não se percam, e serão processadas assim que o componente responsável normalize sua operação.

Cenário 5 – Escalabilidade

Simular o envio de dados de leitura por parte de um sensor de barragem para o módulo Monitoramento de Barragens, que deve processar os dados e ao identificar uma anomalia deve propagar uma mensagem no RabbitMQ para o envio de vinte e cinco mil alertas, que será processado pelo módulo de Segurança e Comunicação. O módulo de Segurança e Comunicação estará operando com cinco instâncias simultâneas, e o resultado esperado é que se processe todas essas mensagens em menos de dez minutos, sem ter impacto ou conflito entre uma instância e outra. Visando garantir que da forma que está planejada a arquitetura possa ser facilmente escalado por método automatizados, como Kubernetes.

Na priorização foi utilizado o método de Árvore de utilidade reduzida e com prioridades. Foi categorizado de acordo os atributos de qualidade a que estão relacionados e então classificados em função de sua importância e complexidade, considerando a percepção de negócio e arquitetura. As duas variáveis de priorização "Importância" e "Complexidade", apresentadas nas colunas IMP. e COM. respectivamente foram classificadas em alta (A), média (M) e baixa (B) de acordo com as características do requisito.

Atributos de Qualidade		Cenário	Importância	Complexidade
F u n c i o n a l i d a d e	Acessabilidade	Cenário 1: Responsividade em dispositivos móveis	M	B
	Segurança	Cenário 2: Deve apresentar altos padrões de segurança.	A	M
		Cenário 3: Deve apresentar altos padrões de segurança.		
E f i c ê n c i a	Resiliência	Cenário 4: Solução conseguir se recuperar de instabilidades	A	A
	Escalabilidade	Cenário 5: Solução ser escalável e comportar processamento com múltiplas instâncias	A	A

6.3. Avaliação

Apresente as evidências dos testes de avaliação. Apresente as medidas registradas na coleta de dados. Para aquilo que não for possível quantificar apresente uma justificativa baseada em evidências qualitativas que suportem o atendimento aos requisitos não-funcionais. As evidências das avaliações neste item são fundamentais.

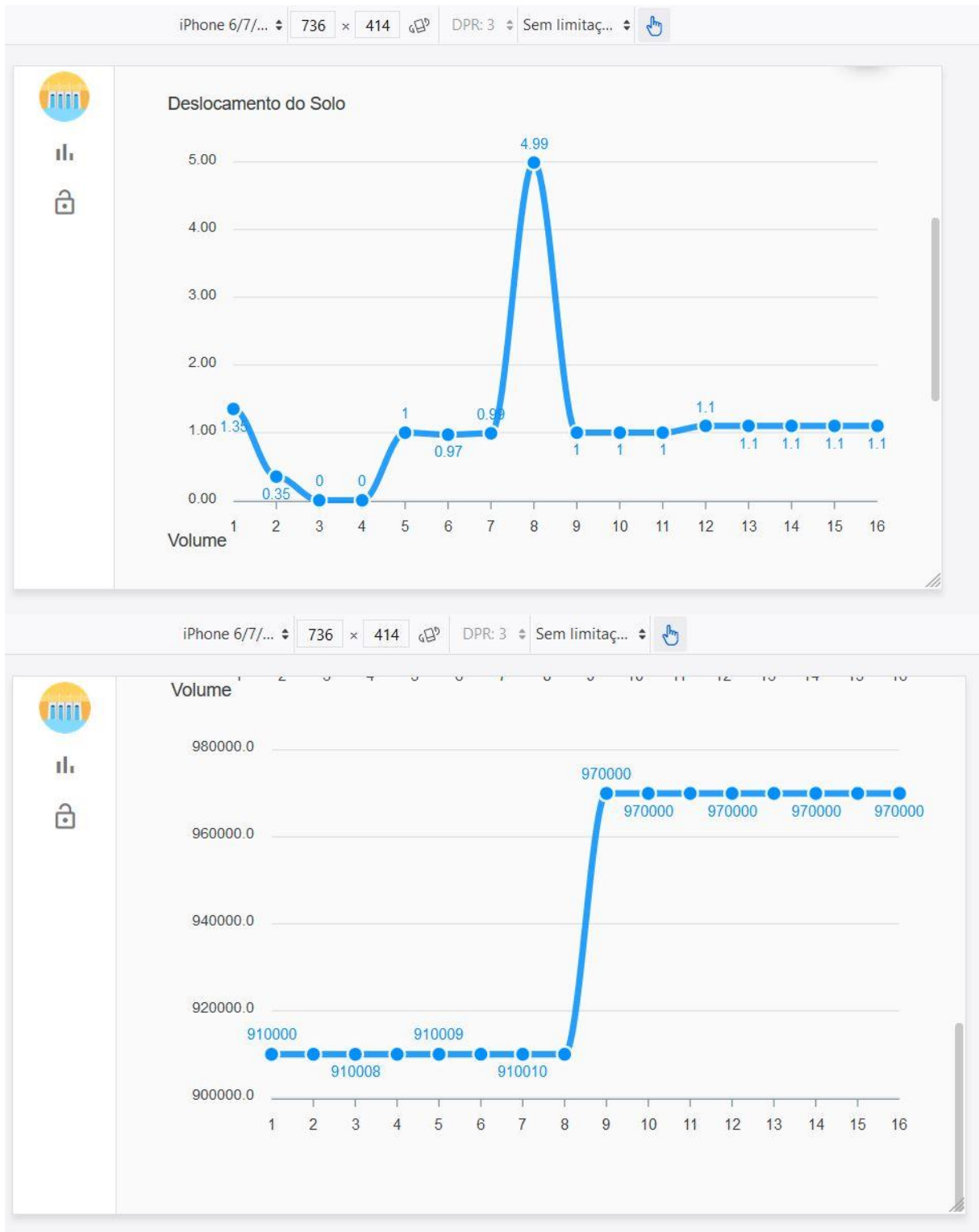
Cenário 1

Atributo de Qualidade:	Acessabilidade
Requisito de Qualidade:	A solução deve ser responsiva conforme o dispositivo
Preocupação:	

As telas e componentes devem apresentar facilidade de interação em dispositivos móveis, e deve manter-se com os mesmos padrões de cores e objetos	
Cenário(s):	
Cenário 1	
Ambiente:	
Operação normal	
Estímulo:	
Usuário acessar o Portal via dispositivo móvel, visualizar relatório de monitoramento e enviar uma mensagem de alerta manualmente.	
Mecanismo:	
Utilização framework Vuetify de componentes responsivos	
Medida de Resposta:	
As telas e componentes devem apresentar facilidade de interação em dispositivos móveis e manter-se com os mesmos padrões de cores e objetos.	
Considerações sobre Arquitetura:	
Riscos:	Possível lentidão em redes móveis.
Pontos de Sensibilidade:	Não existe
Tradeoff:	Curva de aprendizado da equipe de desenvolvimento na utilização do framework de componentes.

Evidências:





iPhone 6/7/... 736 x 414 DPR: 3 Sem limitaç...

Enviar Aviso

Mensagem
Alerta de perigo

Motivo
Indicadores Criticos

Escopos
Comunidade Engenheiros

ENVIAR

Cenário 2 e 3

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	A solução deve possuir altos padrões de segurança
Preocupação:	
Manter os dados seguros e evitar vulnerabilidades de segurança no projeto.	
Cenário(s):	
Cenário 2 e 3	
Ambiente:	
Operação normal	
Estímulo:	
Cenário 2: Usuário não autenticado tentar acessar o Portal SCA.	
Cliente aplicacional tentar enviar requisição não autenticada	
Mecanismo:	
Cenário 2: Implementar no Portal SCA o fluxo implícito do protocolo OpenId.	
Cenário 2 e 3: Implementar o Serviço de Identidade para gerir os tokens de acesso e a	

integração com o provedor de autenticação, e filtro na aplicação gateway para validação do token nas requisições.	
Medida de Resposta:	
Cenário 2: O usuário deve ser redirecionado para a página de login.	
Cenário 3: Resposta HTTP 401	
Considerações sobre Arquitetura:	
Riscos:	Riscos na área de segurança sempre são críticos, qualquer vulnerabilidade pode comprometer a solução.
Pontos de Sensibilidade:	Requisições externas devem ser todas sobre o protocolo HTTPS.
Tradeoff:	Complexidade de implementação do Serviço de Identidade e integração com o provedor de autenticação, e gestão dos filtros na aplicação gateway.

Link da evidência na seção de apêndices.

Cenário 4

Atributo de Qualidade:	Resiliência
Requisito de Qualidade:	A solução deve conseguir se recuperar de momentos de instabilidade e possuir baixo acoplamento entre os componentes arquiteturais.
Preocupação:	
Possuir baixo acoplamento entre os componentes arquiteturais, e que a indisponibilidade de um deles não impacte nos demais componentes além de, assim que normalizado o funcionamento as integrações, que ocorreram no momento da indisponibilidade não sejam perdidas.	
Cenário(s):	
Cenário 4	

Ambiente:	
Módulo de Segurança e Comunicação indisponível temporariamente	
Estímulo:	
Módulo de Monitoramento de Barragens recebe dados de um sensor de barragem.	
Mecanismo:	
Implementação de mensageria utilizando RabbitMQ, utilizando exchanges do tipo tópico e chaves de roteamento para as filas.	
Medida de Resposta:	
O módulo de Monitoramento de Barragens deverá publicar a mensagem no message broker com sucesso, e o alerta SMS deverá ser recebido após a inicialização do módulo de Segurança e Comunicação.	
Considerações sobre Arquitetura:	
Riscos:	Existe um ponto único de falha nessa abordagem que é o RabbitMQ, qualquer lentidão ou indisponibilidade impacta os demais componentes.
Pontos de Sensibilidade:	Possuir redundância no RabbitMQ.
Tradeoff:	Complexidade de desenvolvimento.

Link da evidência na seção de apêndices.

Cenário 5

Atributo de Qualidade:	Escalabilidade
Requisito de Qualidade:	A solução deve possuir escalabilidade horizontal facilitada.
Preocupação:	
Possuir escalabilidade horizontal sem que gere conflitos entre as diferentes instâncias, e nem comportamentos inesperados.	
Cenário(s):	
Cenário 5	
Ambiente:	
Alta carga de alertas para envio e cinco instâncias do módulo Segurança e Comunicação	
Estímulo:	

Emissão de alerta por indicador crítico	
Mecanismo:	
Implementação de mensageria utilizando RabbitMQ, utilizando exchanges do tipo tópico e chaves de roteamento para as filas.	
Medida de Resposta:	
A solução deve notificar todos os contatos em menos de dez minutos.	
Considerações sobre Arquitetura:	
Riscos:	RabbitMQ pode se tornar um gargalo na aplicação. Para mitigar esse risco, poderá possuir mais de uma instância do message broker.
Pontos de Sensibilidade:	Múltiplas instâncias do RabbitMQ
Tradeoff:	Complexidade de desenvolvimento e automação da escalabilidade utilizando Kubernetes por exemplo.

Link da evidência na seção de apêndices.

6.4. Resultado

Considerando os atributos propostos de qualidade e requisitos não funcionais, foi verificado após a prova de conceito que o objetivo foi atingido, conseguindo entregar uma solução com performance, escalabilidade, resiliência, segurança e acessabilidade.

Requisito Não Funcional	Testado	Homologado
Segurança	Sim	Sim
Acessabilidade	Sim	Sim
Resiliência	Sim	Sim
Escalabilidade	Sim	Sim

O principais pontos atingidos nessa abordagem arquitetural foi os pontos de baixo acoplamento entre os serviços e escalabilidade com um nível granular por aplicação. Pontos de melhoria que deverão ser implementados no projeto final é a automação da escalabilidade da aplicação utilizando uma solução como Kubernetes. Outro ponto são os objetos propagados no RabbitMQ, o ideal seria implementar uma categoria de classe apenas para lidar com os eventos, no protótipo está sendo propagado o mesmo DTO utilizado pelo controlador. Fazer uma separação da biblioteca Afp.Core por conceitos, criando várias bibliotecas que possam ser consumidas conforme a necessidade, como Afp.Core.Data para os dados, Afp.Core.Mq para as classes de mensageria, assim deixando a aplicação mais coesa e consumindo somente o que é necessário. Outra melhoria seria separação da aplicação que consome as mensagens do RabbitMQ da aplicação que responde as requisições HTTP também é de suma importância para manter a escalabilidade e performance. O tempo de desenvolvimento de uma nova funcionalidade deve ser mais demorado, devido ao grande número de componentes arquiteturais e integrações entre eles, assim aumentando o tempo de desenvolvimento e sua complexidade, e esse ponto acaba também sendo um tradeoff dessa abordagem arquitetural.

7. Conclusão

Todo o processo de definição arquitetural e implementação do protótipo foi de grande importância para o apredizado, foi possível testar hipóteses e tomar notas de pontos de melhoria. O objetivo final desse projeto foi atingido e os pontos de melhoria, enumerados no capítulo anterior, com o devido tempo são totalmente atingíveis para uma versão final e apta para ir para produção.

REFERÊNCIAS

KOCHEN, Roberto. **Monitoramento e Segurança de Barragens.** <https://www.institutodeengenharia.org.br/site/wp-content/uploads/2019/03/3-Monitoramento-e-Seguran%C3%A7a-de-Barragens-IE-2019.pdf>, 2019.

Ambiente Livre, **Orquestrando Hbase, Cassandra e MongoDB com o Pentaho Big Data Analytics.** <https://pt.slideshare.net/ambientelivre/orquestrando-hbase-cassandra-e-mongodb-com-o-pentaho-big-data-analytics-124589293>, 2019.

BAILEY, Derick. **RabbitMQ – Best Practices For Designing Exchanges, Queues and Bindings?** <https://derickbailey.com/2015/09/02/rabbitmq-best-practices-for-designing-exchanges-queues-and-bindings/>, 2019.

JOHANSSON, Lovisa. **The Optimal RabbitMQ Guide: From beginner to advanced.** Suécia: Vulkan Bokförlag, 2018.

WOOLF, Bobby. **Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.** EUA: Addison-Wesley Professional, 2003.

APÊNDICES

URL Repositório GitHub: <https://github.com/andersondepaiva/pucminas-tcc>

<https://github.com/andersondepaiva/pucminas-tcc-apiconfigurations>

URL Portal implantado no Heroku: <https://sca-portal.herokuapp.com/>

Usuário: scapucminas@andersondepaiva92gmail.onmicrosoft.com

Senha: PucMinas2019

Swagger Segurança e Comunicação: <https://sca-segurancacomunicacao.herokuapp.com/swagger-ui.html>

Swagger Monitoramento de Barragens: <https://sca-monitoramentobarragens.herokuapp.com/swagger-ui.html>

Swagger Cadastro de Ativos: <https://sca-monitoramentobarragens.herokuapp.com/swagger-ui.html>

Eureka: <https://sca-discoveryserver.herokuapp.com/>

Evidência Segurança: <https://vimeo.com/358520239>

Evidência Resiliência: <https://vimeo.com/358519712>

Evidência Escalabilidade: <https://vimeo.com/358520602> - Parte 1

<https://vimeo.com/358520610> - Parte 2

CHECKLIST PARA VALIDAÇÃO DOS ITENS E ARTEFATOS DO TRABALHO

Nº	Item a ser cumprido	Sim	Não	Não se aplica
Completeza do documento				
1	Todos os elementos iniciais do documento (capa, contracapa, resumo, sumário...) foram definidos?			
2	Os objetivos do trabalho (objetivos gerais e pelo menos três específicos) foram especificados?			
3	Os requisitos funcionais foram listados e priorizados?			
4	Os requisitos não funcionais foram listados e identificados usando o estilo estímulo-resposta?			
5	As restrições arquiteturais foram definidas?			
6	Os mecanismos arquiteturais foram identificados?			
7	Um diagrama de caso de uso foi apresentado junto com uma breve descrição de cada caso de uso?			
8	Um modelo de componentes e uma breve descrição de cada componente foi apresentada?			
9	Um modelo de implantação e uma breve descrição de cada elemento de hardware foi apresentada?			
10	Prova de conceito: uma descrição da implementação foi feita?			
11	Prova de conceito: as tecnologias usadas foram listadas?			
12	Prova de conceito: os casos de uso e os requisitos não funcionais usados para validar a arquitetura foram listados?			
13	Prova de conceito: os detalhes da implementação dos casos de uso (telas, características, etc) foram apresentadas?			
14	Prova de conceito: foi feita a implantação da aplicação e indicado como foi feita e onde está disponível?			
15	As interfaces e/ou APIs foram descritas de acordo com um modelo padrão?			
16	Avaliação da arquitetura: foi feita uma breve descrição das características das abordagens da proposta arquitetural?			
17	Avaliação da arquitetura: Os atributos de qualidade e os cenários onde eles seriam validados foram apresentados?			
18	Avaliação da arquitetura: uma avaliação com as evidências dos testes foi apresentada?			
19	Os resultados e a conclusão foram apresentados?			
20	As referências bibliográficas foram listadas?			
21	As URLs com os códigos e com o vídeo da apresentação da POC foram listadas?			

Nº	Item a ser cumprido	Sim	Não	Não se aplica
Consistência dos itens do documento				
1	Todos os requisitos funcionais foram mapeados para casos de uso?			
2	Todos os casos de uso estão contemplados na lista de requisitos funcionais?			
3	Os requisitos não funcionais, mecanismos arquiteturais e restrições c arquiteturais estão coerentes com os modelos de componentes e implantação?			
4	Os modelos de componentes e implantação estão coerentes com os requisitos não funcionais, mecanismos arquiteturais e restrições arquiteturais?			
5	As tecnologias listadas na implementação estão coerentes com os requisitos não funcionais, mecanismos arquiteturais e restrições arquiteturais?			
6	Os casos de uso e os requisitos não funcionais listados na implementação estão coerentes com o que foi listado nas seções anteriores?			
7	Os atributos de qualidade usados na avaliação estão coerentes com os requisitos não funcionais na sessão 3?			
8	Os cenários definidos estão no contexto dos casos de uso implementados?			
9	O apresentado no item resultado está coerente com o que foi mostrado no item avaliação?			