**PONTÍFICIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**

**Pós-graduação Latu Sensu em Arquitetura de Software Distribuído**

**Luiz Fernando Dias Santos**

**SISTEMA DE GESTÃO E CONTROLE DE INFORMAÇÕES AMBIENTAIS**

Brasília

2020

**Luiz Fernando Dias Santos**

**SISTEMA DE GESTÃO E CONTROLE DE INFORMAÇÕES AMBIENTAIS**

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

Orientador: Prof. Dr. Pedro A. Oliveira

Brasília

2020

*Ao meu querido pai José Dias, por ter me ensinado o valor do trabalho.*

*A minha família, pelo carinho*

**RESUMO**

Devido aos impactos no meio ambiente decorrente da atividade industrial e empresarial, bem como a adequação a legislação brasileira, torna-se fundamental que a empresa tenha controle das informações do processo de licenciamento ambiental.

Este projeto aborda a criação de sistema que pode ser acessado de qualquer dispositivo ou computador com segurança, a empresa poderá lançar dados de licenciamento ambiental do empreendimento, as multas aplicadas, gerenciamento da obtenção das licenças ambientais do empreendimento, de modo a auxiliar a empresa na construção da política ambiental, a ter uma atuação ecologicamente correta e aprimorar sua gestão ambiental, com simplicidade e usabilidade.

O projeto arquitetural aborda uma solução eficiente e simples, para atender as necessidades da empresa com escalabilidade e praticidade, com possibilidade de integração com outras aplicações de forma segura e com *frameworks* modernos.

**Palavras-chave:** Arquitetura de Software, projeto de software, software de gestão ambiental.

**SUMÁRIO**

[**SUMÁRIO** 5](#_Toc41771283)

[1.Objetivos do trabalho 6](#_Toc41771284)

[**2. Descrição Geral da Solução** 7](#_Toc41771285)

[2.1 Apresentações do Problema 7](#_Toc41771286)

[2.2 Definições gerais do *Software* (Escopo) 8](#_Toc41771287)

[**3. Definição da Solução** 8](#_Toc41771288)

[3.1 Requisitos Funcionais 8](#_Toc41771289)

[3.2 Requisitos Não-Funcionais 10](#_Toc41771290)

[3.3 Restrições Arquiteturais 13](#_Toc41771291)

[3.4 Mecanismos Arquiteturais 13](#_Toc41771292)

[4.Modelagem e Projeto Arquitetural 14](#_Toc41771293)

[4.1 Diagrama de Processo 14](#_Toc41771294)

[4.2 Descrição resumida dos casos de Uso 15](#_Toc41771295)

[4.3 Modelo de Componentes 20](#_Toc41771296)

[4.4 Modelo de Implantação 21](#_Toc41771297)

[5 Prova de Conceito (POC)/Projeto Arquitetural 22](#_Toc41771298)

[5.1 Implantação e Implementação 22](#_Toc41771299)

[5.2 Interfaces/ APIs 24](#_Toc41771300)

[6. Avaliação da Arquitetura 25](#_Toc41771301)

[6.1 Análises das abordagens arquiteturais 25](#_Toc41771302)

[6.2 Cenários 26](#_Toc41771303)

[6.3 Avaliação 26](#_Toc41771304)

[7. Conclusão 33](#_Toc41771305)

[REFERÊNCIAS 34](#_Toc41771306)

[APÊNDICES 35](#_Toc41771307)

CHECKLIST PARA VALIDAÇÃO DOS ITENS E ARTEFATOS DO TRABALHO......................37

**1. Objetivos do trabalho**

Este trabalho tem como objetivo apresentar uma proposta de arquitetura para o desenvolvimento de um sistema de gestão ambiental, com o intuito de fornecer uma solução de controle de informações das diversas etapas do processo de aquisição da licença de funcionamento do empreendimento, com boa manutenibilidade, portabilidade e usabilidade.

Os objetivos específicos são:

1. Criação de um módulo de autenticação para o controle de acesso dos colaboradores, o módulo deverá permitir diferentes níveis de acesso bem como permitir que a senha seja recuperada através de e-mail. A aplicação deverá fornecer uma solução de autenticação moderna e segura.
2. Criar um módulo de gestão das licenças ambientais, garantindo o atendimento da legislação e dos processos de licenciamento ambiental do empreendimento, conforme o fluxo pertinente ao processo, deverá também observar o cadastro das condicionantes bem como gerenciar o fluxo das licenças.
3. Desenvolver o sistema de tal forma que a camada de persistência funcione bem com diversos bancos de dados, sejam eles proprietários ou não, de forma que se há a necessidade de se adequar a uma solução de banco de dados existente no cliente, que seja feito com o mínimo de impacto em termos de desenvolvimento para sua adaptação – em outras palavras, deverá utilizar um *framework* objeto-relacional.

4. O sistema deverá permitir o cadastro de colaboradores bem como a definição do nível de acesso, propiciando diferentes visões de negócio de acordo com o cargo do usuário no empreendimento. Deverá observar também que o profissional deve ser legalmente habilitado, conforme estabelecido em resolução, logo deverá abranger também a sua habilitação ou especialidade, deverá também fornecer ao usuário uma solução que permita um acompanhamento adequado e eficiente dos prazos do licenciamento.

5. O sistema deverá ser portável – em outras palavras – deverá utilizar tecnologia que facilite o acesso através de diferentes dispositivos e plataformas, deverá ser acessível de um *smartphone*, de um *tablet* ou de um *notebook* por exemplo, através de um framework moderno e responsivo, ou seja, o *layout* da aplicação deverá se adaptar de modo a permitir uma melhor usabilidade do sistema.

6. Criar um módulo para a gestão dos documentos, estudos e projetos necessários à aquisição de licença ambiental, conforme estabelecido pelos órgãos ambientais competentes.

7. Deverá incluir um módulo de cadastros básicos, isso vai tornar o sistema mais dinâmico e permitir que o usuário final possa customizar os valores das tabelas de domínio do sistema, por exemplo, a inclusão de um novo tipo de condicionante ser facilmente ajustada pelo administrador do sistema.

# **2. Descrição Geral da Solução**

2.1 Apresentações do Problema

Para que o empreendimento esteja de acordo com a legislação vigente e de modo a conciliar o desenvolvimento econômico com a conservação do meio ambiente, é necessário que o empreendedor buscar o licenciamento junto aos órgãos competentes, logo, é fundamental que os responsáveis pelo empreendimento estejam guarnecidos de informação do correto estágio de licenciamento ambiental, e de como podem utilizar essa informação para verificar a sua adequação e aderência aos processos em relação ao licenciamento ambiental estabelecidos em lei.

Além do que foi exposto acima, para que a empresa obtenha sua adesividade as normas do mercado, torna-se necessário o desenvolvimento de um sistema que possibilite a empresa ter uma maior aderência aos padrões do mercado e das normas, em especial, a ISO 14001.

A ISO 14001 é uma norma que estabelece critérios e requisitos de um sistema de gestão ambiental e que permite que as empresas e organizações criem uma estrutura de proteção do meio ambiente bem como mitigação dos possíveis impactos ambientais.

A implementação de um sistema de gestão ambiental permite que a imagem da empresa seja melhor para a sociedade bem como ganho de credibilidade perante os investidores, o empreendimento também irá consumir menos insumos e por consequência irá economizar.

2.2 Definições gerais do *Software* (Escopo)

O Sistema SIGA (Sistema Integrado de Gestão Ambiental) foi criado com o propósito de auxiliar o processo de aquisição da licença ambiental para o empreendimento. O objetivo é fornecer uma solução simples e focada nas necessidades do usuário, de fácil manutenção e flexível, de modo a minimizar os custos de sua manutenção.

Deverá ser acessível por diferentes aparelhos, deverá ser seguro, uma vez que possuí informações importantes do empreendimento e possibilitar o acompanhamento de todo o processo. Deverá ser simples e que traga agilidade aos processos da organização e também deverá ser de fácil utilização.

# **3. Definição da Solução**

3.1 Requisitos Funcionais

Módulo de autenticação

O sistema deverá fornecer autenticação e autorização para a aplicação, deverá ser seguro de forma a proteger as informações contra ataques e outras formas de intrusão. A tecnologia escolhida foi o *Spring Security*, que está contido no ecossistema do *framework*. O *Spring Security* também permite integração com diversos padrões de autenticação, no SIGA iremos utilizar o *Oauth* 2*.*0 como padrão.

O modulo de autenticação deverá fornecer um modo que o usuário recupere sua senha através de e-mail bem como permitir que um usuário específico seja desativado. Através do usuário informado, o sistema vai definir o perfil de acesso. Abaixo são descritos os perfis:

**Administrador:** O administrador do sistema deverá ter acesso ao cadastro de usuários e de documentos, estudos ou projetos. Terá acesso a todas as etapas do processo de licenciamento. O usuário com o perfil de administrador poderá acompanhar todas as etapas do processo de licenciamento. O administrador do sistema terá acesso ao módulo de cadastros básicos.

**Gestão:** O usuário do sistema com o perfil de gestor terá acesso a relatórios gerenciais, bem como visualizar detalhes dos documentos a respeito do processo. Entende-se como gestor o responsável pelo empreendimento.

**Operacional:** Vai permitir o cadastro das informações das licenças ambientais. Entende-se como operacional o profissional habilitado. Essas informações podem ser inseridas nas mais diversas etapas do licenciamento.

Módulo de licenciamento ambiental.

De acordo com a Resolução nº 237, de 19 de dezembro de 1997, definimos licenciamento ambiental como “procedimento administrativo pelo qual o órgão ambiental competente licencia a localização, instalação, ampliação e a operaçãode empreendimentos e atividades utilizadoras de recursos ambientais,consideradas efetiva ou potencialmente poluidoras ou daquelas que, sob qualquer forma, possam causar degradação ambiental, considerando as disposições legais e regulamentares e as normas técnicas aplicáveis ao caso”.

Deverá armazenar as licenças ambientais bem como armazenar as informações nas diferentes etapas do licenciamento. As licenças são descritas abaixo:

Licença Prévia (LP) – Concedida na etapa de planejamento, atesta a viabilidade de concepção e determina as condicionantes que deverão ser atendidas para a próxima fase.

Licença de Instalação (LI) – Fornecida com o intuito de autorizar o responsável pelo empreendimento a construção ou instalação da obra de acordo com as especificações constantes nos planos, projetos e especificações aprovados, principalmente as medidas de mitigação e controle ambiental para a próxima fase.

Licença de Operação (LO) – Licença de funcionamento. Após a constatação de que todas as exigências e condicionantes foram cumpridas. O empreendedor fica responsável por implementar todas as medidas de mitigação e controle ambiental, sob pena de ter a LO suspensa ou caçada pelo órgão responsável. O sistema deverá entender que uma licença transita entre os três tipos informados acima, deverá, portanto, compreender o fluxo e guardar o histórico do licenciamento. O módulo de licenciamento ambiental deverá abordar também o cadastro de condicionantes.

Módulo Administrativo

No módulo administrativo serão cadastrados os usuários do sistema, bem como atribuição das suas habilitações e controle, conforme requerido por legislação. O módulo administrativo deve incluir também os cadastros básicos da aplicação.

3.2 Requisitos Não-Funcionais

*Framework* Objeto-Relacional

A aplicação deverá funcionar de acordo com a solução de banco de dados encontrada pelo cliente, em outras palavras, o sistema deverá ser independente de banco de dados de forma que o responsável pelo empreendimento possa optar por uma solução paga de banco de dados já adquirida pelos responsáveis pelo empreendimento, bem como utilizar uma opção gratuita que lhe aprouver sem prejuízo de ter que refazer boa parte do sistema, ou ter que se adotar um padrão de *SQL* e se desprezar as nuances de cada banco de dados, como era comum nos sistemas antigos antes da popularização dos *frameworks objeto-relacionais*.

|  |  |
| --- | --- |
| **Estímulo** | Mudança de Banco de dados. |
| **Fonte de estímulo** | A empresa optou por uma solução de banco de dados com um melhor custo-benefício, ou decisão do gestor. |
| **Ambiente** | Funcionamento, carga normal |
| **Artefato** | Camada de persistência da aplicação |
| **Resposta** | Não houve impactos, porém é necessário um estudo prévio e migração dados da solução antiga para a nova |
| **Medida da resposta** | Funcionamento normal da aplicação. |

* Manutenibilidade – O sistema deverá ter manutenção facilitada

O sistema deverá construído observando as boas práticas de desenvolvimento, com a finalidade de proporcionar a facilidade de manutenção evolutiva e correção de *bugs*, um sistema com uma manutenção mais fácil também é mais flexível e com um custo menor para futuras evoluções. Também deve comtemplar um mecanismo de *log* e de tratamento de erro que os torne facilmente identificável, assim, quando ocorrer um erro no sistema, esses logs deverão estar devidamente separados por dia, e ser de fácil leitura e interpretação.

|  |  |
| --- | --- |
| **Estímulo** | Ajuste no cadastro de condicionantes para incluir o tipo de condicionante |
| **Fonte de estímulo** | Correção de bug |
| **Ambiente** | Carga normal, em funcionamento. |
| **Artefato** | Módulo de gestão das licenças. |
| **Resposta** | Manutenção facilitada, sobretudo ao uso de JPA (*Java Persitence API)* na arquitetura. |
| **Medida da Resposta** | O sistema deverá utilizar um *framework* objeto-relacional moderno e eficiente. |

* Segurança – O sistema deverá implementar a segurança através do padrão Oauth2

O Oauth 2.0 é definido como um protocolo que permite que os usuários de uma aplicação ter acesso limitado a seus recursos sem necessariamente expor suas credenciais (REACHER e SANSO, 2017). E utilizado por um número enorme de aplicações em todo o mundo.

|  |  |
| --- | --- |
| **Estímulo** | Tentar realizar o acesso ao sistema sem estar autenticado. |
| **Fonte de estímulo** | Tentativa de acesso não identificado. |
| **Ambiente** | Em funcionamento, com carga normal. |
| **Resposta** | O sistema redireciona o usuário a tela de login. |
| **Medida da resposta** | O sistema não deverá permitir o acesso sem que o usuário esteja devidamente cadastrado. |

* Desempenho – O sistema deverá executar sem lentidão ou travamento

|  |  |
| --- | --- |
| **Estimulo** | Navegação nas telas do sistema. |
| **Fonte de estímulo** | Usuário autenticado. |
| **Ambiente** | Em funcionamento, com carga normal. |
| **Artefato** | Módulos do sistema. |
| **Resposta** | Normal. |
| **Medida da resposta** | O interagiu entre as telas do sistema de forma normal, sem travamentos ou lentidão. |

* Acessibilidade - O sistema deve suportar o acesso através de diferentes mecanismos e plataformas

|  |  |
| --- | --- |
| **Estímulo** | Usuário com o perfil operacional logado no sistema. |
| **Fonte do estímulo** | Usuário com o perfil operacional acessando o sistema a partir de um *tablet*. |
| **Ambiente** | Em funcionamento, com carga normal. |
| **Artefato** | Módulo de autenticação. |
| **Resposta** | A tela do módulo de autenticação funcionou perfeitamente, devido a tecnologia *frontend* utilizada. |
| **Medida da Resposta** | Identidade visual semelhante em diversas resoluções, teste positivo. |

* Interoperabilidade – O sistema deverá ter um mecanismo de comunicação com terceiros

|  |  |
| --- | --- |
| **Estímulo** | Teste de conexão |
| **Fonte do estímulo** | Script de teste externo simulando conexão com o sistema. |
| **Ambiente** | Funcionamento normal, com carga normal. |
| **Artefato** | Módulo de comunicação. |
| **Resposta** | Foi enviada uma mensagem ao sistema com sucesso. |
| **Medida da resposta** | A comunicação ocorreu com sucesso. |

3.3 Restrições Arquiteturais

* O sistema deverá utilizar um *framework* objeto-relacional, de modo a não sofrer impactos na hipótese da alteração do banco de dados;
* Deverá ser desenvolvido utilizando a abordagem de serviços;
* Deverá utilizar o protocolo *Oauth 2.0* para autenticação;
* Deverá ser portável de modo a funcionar nas mais diferentes plataformas, sem impactar na usabilidade do produto.
* Deverá possuir mecanismo de logs, para uma boa rastreabilidade.

3.4 Mecanismos Arquiteturais

|  |  |  |
| --- | --- | --- |
| **Mecanismo de Análise** | **Mecanismo de *Design*** | **Mecanismo de implementação** |
| Comunicação entre processos | Servidor de aplicação e servidor HTTP | *Wildfly 18.01* |
| Versionamento | Versionamento do código fonte da aplicação | *Github* |
| Autenticação e Autorização | Verificação das credenciais através de *token* | *Spring Security, Spring Boot e Oauth 2.0*. |
| *Frontend* | Interface de comunicação com os usuários do sistema | Angular, Bootstrap |
| *Build* | Geração de artefato. | *Gradle* |
| *Log* | Mecanismo de log da aplicação | Log4J |
| *ESB* | Integração de mensagens da aplicação com outros sistemas/barramento | RabbitMQ |

4. Modelagem e Projeto Arquitetural

Através desta seção, vamos compreender alguns diagramas que permitem um melhor entendimento da estrutura fundamental da aplicação.

4.1 Diagrama de Processo

O diagrama de processo retrata as principais etapas do fluxo de licenciamento ambiental, conforme é descrito na resolução CONAMA nº 237, ou seja, através do diagrama, pode-se visualizar o fluxo básico que o sistema deverá atender.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Fig. 1 – Diagrama de processo criado a partir da resolução CONAMA nº 237**

4.2 Descrição resumida dos casos de Uso

Caso de uso 1: Autenticação na aplicação

Descrição resumida

Este caso de uso descreve como o software SIGA deve permitir que a autorização do usuário seja feita na aplicação, a figura abaixo ilustra como deverá funcionar a configuração do Oauth 2.0 em termos de arquitetura. Observamos que o sistema tem embutido um servidor de autorização e autenticação, que é fornecido pelo *framework* *Spring*.

Existem outras formas de autenticação e autorização com o *Spring Security*, porém foi escolhido o *Oauth2* em detrimento de JWT, pois apesar de similares, o JWT não é um método de autenticação e sim um mensageiro, apenas transporta a informação, enquanto que o Oauth2 é amplamente utilizado em diversas aplicações ao redor do mundo. Existe a possibilidade de também trabalhar com a *Kerberos*, porém, acredito que o Oauth2 supra todas as necessidades de autenticação da aplicação.

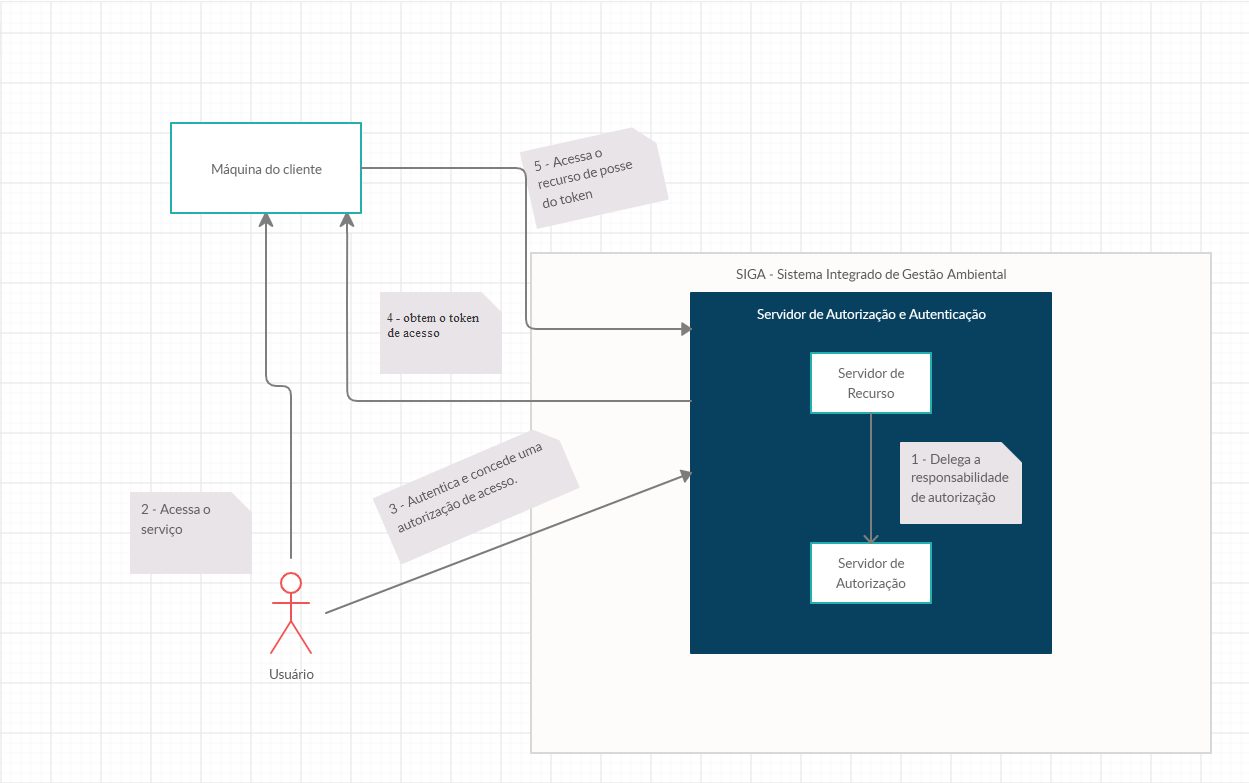
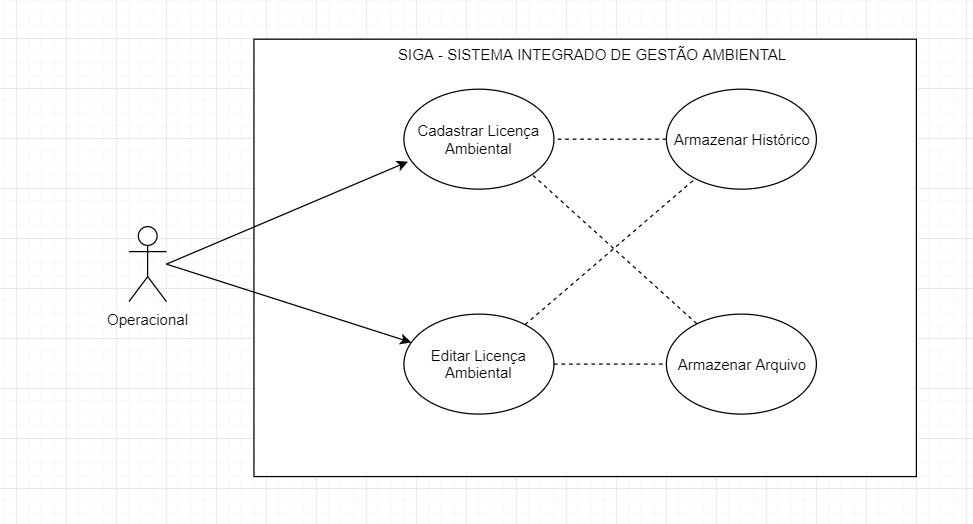


Fig. 2 – Autenticação e Autorização no SIGA

|  |  |
| --- | --- |
| **ID** | 1 |
| **Estória do usuário** | Realizar o *login* no sistema |
| **Criador** | Usuário |
| **Descrição** | Necessidade de realizar a autenticação, devido a segurança que deverá ser fornecida aos dados |
| **Valor do negócio** | Módulo administrativo |
| **Prioridade** | Alta |

Caso de uso 2: Cadastro de licença ambiental



**Fig. 3 – Cadastro de Licença Ambiental**

**Descrição resumida**

Este caso de uso descreve como deve permitir ao usuário com o perfil operacional cadastrar os dados de uma licença ambiental, ele deve fornecer as informações e poderá anexar arquivos, o sistema deverá guardar o usuário que está entrando com as informações. O sistema também deverá atentar aos prazos e ao correto fluxo do licenciamento ambiental, conforme legislação específica.

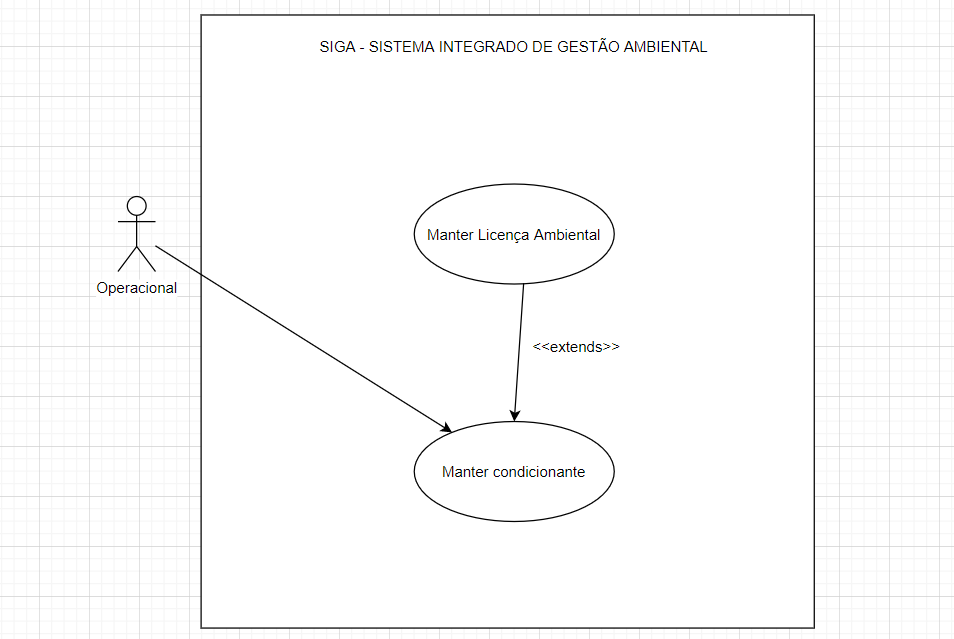
Abaixo, temos uma relação dos campos que a aplicação deverá contemplar, com as respectivas máscaras.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome | Tipo | Tamanho | Máscara | Descrição/Observação |
| Data de validade | Data | N/A | \*\*/\*\*/\*\*\*\* | Campo obrigatório |
| Nome do Empreendimento | Texto | 200 Caracteres | N/A | Campo obrigatório |
| Horário de Funcionamento | Tempo | N/A | \*\*:\*\* à \*\*:\*\* | -- |
| Número de funcionários | Inteiro | N/A | N/A | -- |
| CNPJ | Alfanumérico | N/A | \*\*.\*\*\*.\*\*\*/\*\*\*\*-\*\* | Campo obrigatório |
| Número | Alfanumérico | 10 caracteres | N/A | Campo obrigatório |
| Complemento | Texto | 200 caracteres | N/A | Campo Obrigatório |
| Bairro | Texto | 200 caracteres | N/A | Campo obrigatório |
| Cidade | Texto | 250 caracteres | N/A | Campo obrigatório |
| Estado | Texto | 250 caracteres | N/A | N/A Campo Obrigatório |
| UF | Texto | 2 caracteres | N/A | Campo Obrigatório |
| Conteúdo | Texto | N/A | N/A | O campo deverá ser inlimitado |
| Emitente | Texto | 200 caracteres | N/A | -- |
| Atividade Principal | Texto | 500 caracteres | N/A | -- |
| Descrição da Atividade principal | Texto | N/A | N/A | Campo obrigatório |
| Arquivos anexos | Arquivo | Deve suportar *n* arquivos | N/A | Campo obrigatório |
| Área | *Float* | N/A | N/A | -- |
| Área construída | *Float* | N/A | N/A | -- |

Devemos observar também que o sistema deverá armazenar o histórico das mudanças, uma vez que o licenciamento ambiental é um processo.

|  |  |
| --- | --- |
| **ID** | 2 |
| **Estória do usuário** | Cadastrar licença ambiental |
| **Criador** | Com usuário com o perfil operacional |
| **Descrição** | Necessidade de cadastrar as informações das licenças ambientais, de modo que o sistema guarde o correto fluxo do processo, assim como anexar arquivo PDF e JPG referentes ao licenciamento |
| **Valor do negócio** | Módulo principal do sistema |
| **Prioridade** | Alta |

Caso de uso 3 – Cadastro de Condicionante



**Fig. 4 – Manter condicionante**

**Descrição Resumida**

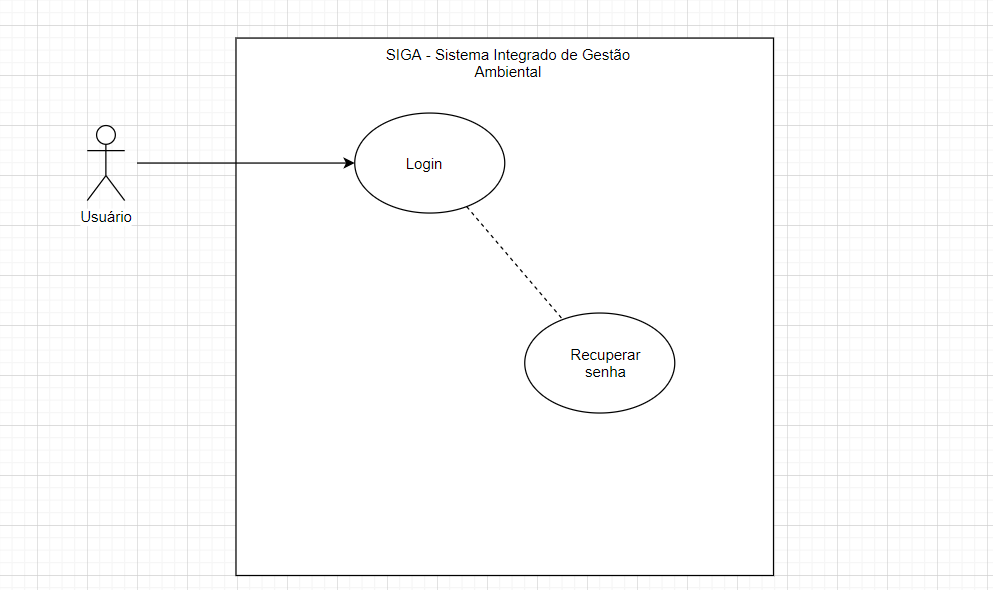
De acordo com Coli e Dias (2017, pag. 22) temos que:

As condicionantes representam obrigações que são impostas pelo órgão ambiental, com base nos estudos e monitoramentos desenvolvidos ao longo do licenciamento e na análise dos projetos apresentados, que determinam a forma em que foi autorizada a concepção implantação ou operação da atividade.

O usuário com o perfil operacional vai poder cadastrar essas condicionantes, vinculando a um processo de licenciamento determinado, em qualquer fase da aquisição.

|  |  |
| --- | --- |
| **ID** | 3 |
| **Estória do usuário** | Cadastrar condicionante |
| **Criador** | Com usuário com o perfil operacional |
| **Descrição** | Utilidade de cadastrar as obrigações que o responsável ou responsáveis pelo empreendimento com os prazos, também é preciso selecionar um processo de licenciamento e atrelar essa condicionante cadastrada. |
| **Valor do negócio** | Módulo principal do sistema |
| **Prioridade** | Alta |

Caso de uso 4 – Recuperação de senha para autenticação no sistema



**Fig 5. Recuperação de senha**

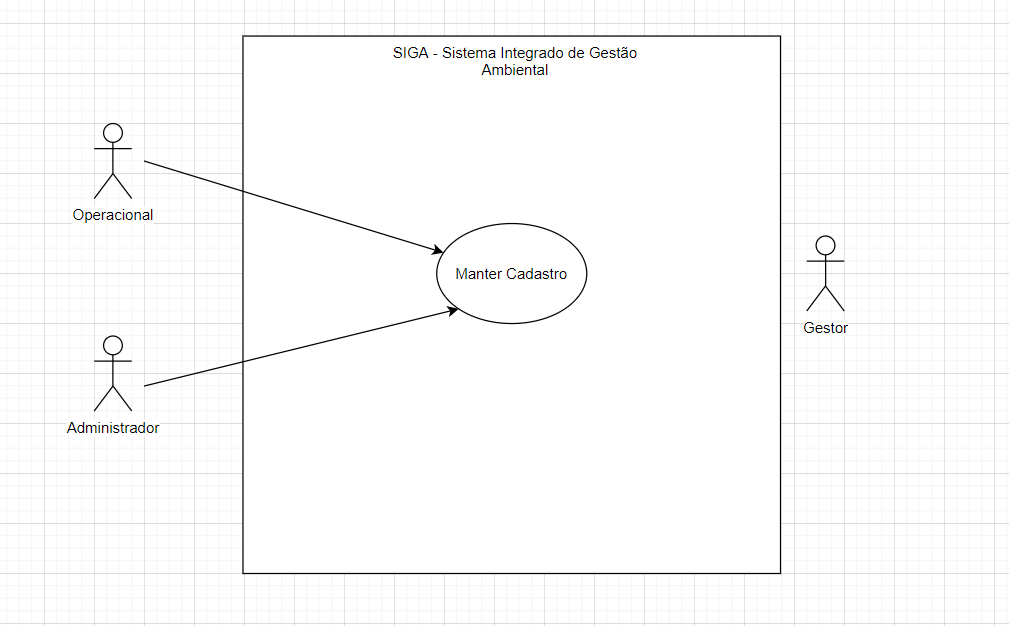
**Descrição Resumida**

Este caso de uso detalha a funcionalidade de recuperar a senha através de uma opção na tela de login, através desta, será encaminhado um e-mail para o usuário cadastrado para redefinição de senha.

|  |  |
| --- | --- |
| **ID** | 4 |
| **Estória do usuário** | Recuperar a senha para login no sistema |
| **Criador** | Com usuário com o perfil operacional |
| **Descrição** | É preciso que o sistema forneça na tela de login uma funcionalidade de envio de senha para o e-mail do usuário cadastrado |
| **Valor do negócio** | Módulo de autenticação do sistema, fundamental para realizar o *login.* |
| **Prioridade** | Médio |

Caso de uso 5 – Cadastro de Usuário

**Descrição Resumida**



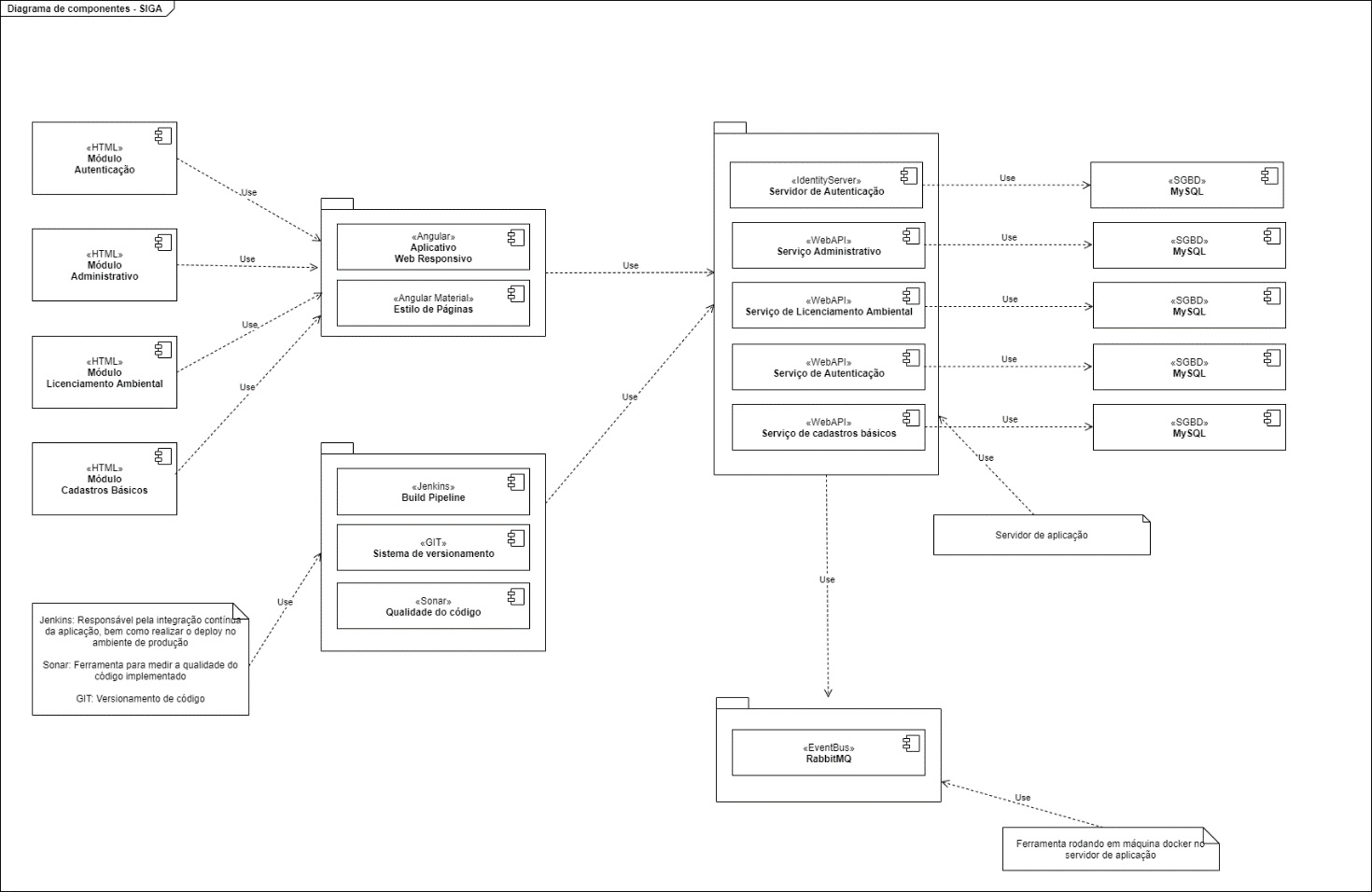
**Fig. 6 – Manter cadastro**

Caso de uso que especifica o cadastro do usuário no módulo administrativo, importante ressaltar que o sistema deverá ser seguro e não permitir o cadastro de qualquer colaborador, já o usuário com o perfil de administrador poderá inclusive criar um pré-cadastro

|  |  |
| --- | --- |
| **ID** | 5 |
| **Estória do usuário** | Realizar o cadastro de usuário |
| **Criador** | Com usuário com o perfil administrador cadastra um usuário |
| **Descrição** | Necessidade de cadastrar um usuário ao sistema, ele pode utilizar os dados das redes sociais para realizar uma espécie de pré-cadastro. |
| **Valor do negócio** | Módulo administrativo |
| **Prioridade** | Alta |

4.3 Modelo de Componentes

Como cita Melo (2010, pág. 214), o diagrama de componentes é uma especificação de um conjunto de construções que especifica um sistema de software, abaixo segue diagrama:



**Fig. 7 – Diagrama de Componentes**

4.4 Modelo de Implantação

Abaixo está o modelo de implantação do sistema, De acordo com MARTINS (2007, pág. 444), temos que:

O diagrama de implantação mostra como várias partes físicas do sistema serão distribuídas no ambiente computacional e se comunicarão entre si, como, por exemplo, máquinas clientes de usuário, servidores de banco de dados, servidores de comunicação e outros.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

# **Fig. 8 – Diagrama de implantação do sistema**

Ainda, de acordo com Guedes (2018, p. 363) destaca-se que:

O diagrama de implantação é o diagrama com a visão mais física da UML. Enfoca a questão da organização da arquitetura física sobre o qual o software será implantado e executado, em termos de hardware, ou seja, as máquinas (computadores pessoais, servidores etc.) que suportarão o sistema, além de definir como estas estarão conectadas e por meio de quais protocolos se comunicarão e transmitirão informações.

Acima é exposto de como deverá ficar distribuído o sistema na implantação. Para o *deploy* da aplicação será utilizada o Jenkins, e na máquina teremos o *Sonar* para o controle de qualidade de código e o GIT para o versionamento de código. Será utilizada também para a mensageria no sentido de fornecer integração com outras aplicações um *container* Docker com a aplicação RabbitMQ.

A aplicação utiliza Oauth2, haverá a possibilidade de utilizar o Google, *Facebook* e *LinkedIn* para autenticação bem como recuperar a senha através de e-mail conforme diagrama acima, como será utilizado serviço na web, não há a necessidade de se ter uma máquina de servidor de e-mail. Abaixo segue a descrição dos componentes acima:

|  |  |
| --- | --- |
| **Componente** | **Descrição** |
| Navegador | *Software* utilizado para acessar o aplicativo *web* |
| Servidor de Integração contínua | Servidor onde deverá ficar o *software* de automação, o sistema de versionamento GIT e também a ferramenta Sonar. |
| Servidor de aplicação | Máquina onde ficará distribuída a aplicação *backend* |
| Servidor *mySQL* | Máquina que vai hospedar a base de dados |
| *Docker* | O *Docker* é uma plataforma de código aberto para criação de *containers* |

5 Prova de Conceito (POC)/Projeto Arquitetural

5.1 Implantação e Implementação

Como é citado por Pressman (2016), entendemos como prova de conceito é um teste que demostra a viabilidade de uma tecnologia, além disso, visa também verificar o atendimento dos requisitos funcionais e não funcionais. Abaixo as validações dos requisitos não funcionais:

**O sistema deverá utilizar um framework objeto-relacional**

Este Requisito não-funcional foi definido devido a necessidade de se construir uma aplicação moderna e independente de banco de dados.

Os critérios para validação são descritos a seguir:

* A aplicação não deverá sofrer retrabalho se o cliente adotar uma solução de banco de dados divergente do MySQL em termos de código. A preocupação deverá ser somente migração de dados.
* O sistema não deverá ser impactado, em outras palavras, o sistema deverá utilizar bem a solução de persistência utilizada, de modo que não impacte na performance.

**O sistema deverá ser responsivo, pois será acessado de diferentes dispositivos**

Devido a evolução dos dispositivos portáteis, torna-se fundamental que o sistema utilize um framework de *frontend* moderno e responsivo, de modo que a página não sofra distorções ou que a experiência de acessar o sistema através de um celular ou um smartphone não prejudique a usabilidade.

Os critérios de validação são descritos a seguir:

* O sistema deverá se manter acessível de modo a não apresentar distorções ou que a usabilidade seja prejudicada, uma vez que o sistema é acessado através de um tablet ou smartphone.
* O sistema deverá ser compatível com os navegadores mais utilizados pelo mercado.
* O sistema também não deverá sofrer prejuízo em relação a seu esquema de cores.

**O sistema deverá ser seguro**

Como o sistema contém informações importantes sobre o processo de licenciamento ambiental do empreendimento, foi utilizado o protocolo Oauth2 utilizando a implementação fornecida pelo framework *Spring Security*.

Os critérios de validação são informados abaixo:

* O sistema não deverá permitir que o usuário não autenticado acesse páginas com informação sensível do sistema sem informar as suas credenciais.
* O sistema deve possuir o total controle das credenciais, de modo que o usuário de determinado perfil não acesse funcionalidades de outro, ou seja, o sistema tem que gerenciar de modo apropriado as permissões.
* Se a sessão do usuário estiver expirada, então o sistema deverá redirecionar o usuário para a tela de login.
* Se o usuário tentar acessar uma URL diferente do login sem estar autenticado, então ele vai ser redirecionado para a tela de login.
* O sistema deve permitir que o usuário recupere a senha através de e-mail.

**O sistema deverá possuir um bom desempenho**

É necessário prover ao usuário um sistema que possua um bom desempenho de forma que a navegação seja fluída e diminuir ao máximo a possibilidade de travamentos ou lentidão.

Para validação, é necessário:

* Que o sistema possua uma boa navegabilidade, sem travamentos ou lentidão.
* Que o sistema seja programado de forma que utilize o framework objeto-relacional de modo correto, para que não sobrecarregue o banco de dados com consultas desnecessárias bem como que otimize o processamento.
* Que o software seja programado de acordo com as boas práticas de desenvolvimento, evitando o retrabalho e que o recurso computacional não seja desperdiçado com código que não otimize o hardware disponível, como código que carregue uma mesma lista em memórias várias vezes, por exemplo.

**Testabilidade – O sistema deverá ter facilidade de teste, ou utilizar alguma ferramenta que facilidade sua manutenção.**

A qualidade é o requisito mais importante do software a ser desenvolvido, uma vez que se o software possuir boa qualidade vai sofrer menos manutenção, em consequência disso, vamos economizar em termos de mão de obra. Se o software tem boa qualidade, então a manutenção vai ser facilitada. Abaixo são enumerados os elementos para validação:

* O sistema deverá permitir o teste unitário através de alguma ferramenta ou framework.
* O sistema deve ser desenvolvido dentro de um fluxo de trabalho onde as funcionalidades sejam testadas utilizando um *bug tracker* como o Mantis ou ferramenta similar.

**O sistema deverá ter boa usabilidade**

O sistema deverá ser fácil de utilizar, facilitando o treinamento com o software desenvolvido, bem como o seu acesso através de dispositivo portátil. Deve ser intuitivo, prático e simples.

Para a validação, temos que:

* O sistema não deve ter um número grande de inputs de formulário por tela, de modo a tornar o cadastramento da licença um processo demorado e cansativo.
* As telas do sistema devem ser focadas e objetivas. Deve haver praticidade**.**

**Interoperabilidade – O sistema deve possuir uma interface de comunicação com outros sistemas**

O sistema deverá abrir um método de comunicação com outros sistemas ou serviços eficiente, Isso significa que não deverá simplesmente disponibilizar um serviço, pois quando há queda do sistema, pode haver perca de informação. Os pontos de validação, são, portanto:

* O sistema deverá abrir meio de comunicação eficiente, de modo que não ocorra perca de informação quando um sistema enviar uma solicitação de um serviço, como ocorreria normalmente com um sistema que apenas disponibiliza um SOAP.
* Deve utilizar uma ferramenta de barramento.

5.2 Interfaces/ APIs

Sessão 1: *Interface* do SIGA

O sistema expõe uma interface para autenticação para acesso ao *Oauth* do *Spring Security*, o método utilizado do protocolo HTTP é o *POST.* A senhas são criptografas utilizando o algoritmo *bcryt.* Caso o usuário forneça corretamente a senha e usuário corretos, então é retornado pelo *Spring Security* um *token* que o *frontend* armazena e utiliza para obter autorização da aplicação para acesso aos serviços.

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo** | **Descrição** | **Obrigatório** |
| Usuário | *String(50)* | Nome de usuário | Sim |
| Senha | *String(50)* | Senha | Sim |

Exemplo de retorno (json):

{

"access\_token":"2e6499fe-ba37-49bb-9a1b-deeb18b18d2d",

"token\_type":"bearer",

"refresh\_token":"3495fe8f-f3dc-4ef6-ae5a-81ec8ef93140",

"expires\_in":48037,

"scope":"all"

}

Acima, o json que é retornado pela aplicação em caso de sucesso, com o *token* de acesso, o tipo de *token* que é fornecido pelo *Oauth* e o tempo que o token expira ( sessão ). O *refresh* *token* é uma espécie de *token* que é utilizada pela aplicação para requerer outro *access token*.

Em caso de falha na autenticação, como se, por exemplo, forem fornecidas credenciais inválidas, o json retornado pela API é diferente:

{

    "error": "invalid\_grant",

    "error\_description": "Bad credentials"

}

6. Avaliação da Arquitetura

6.1 Análises das abordagens arquiteturais

Para a criação do SIGA – Sistema Integrado de Gestão Ambiental foi utilizado amplamente o framework Spring devido a gama de recursos oferecidos, para autenticação e autorização, como o Spring Security quanto para persistência, como o Spring Data. O *Spring* *WebFlow* também fornece um framework conciso e maduro quando se trata de solução MVC utilizando a linguagem Java.

É comum encontrar em projetos legados uma quantidade grande de *metadados* em XML, no projeto, foi utilizada *annotations*, por questões de leitura. O *Spring Boot* também facilita bastante a configuração do projeto.

Foi proposta uma arquitetura simples, porém eficiente e com grande escalabilidade, com o propósito principal de atender tanto o responsável pelo empreendimento com um hardware simples, bem como um empreendedor que tenha diversos escritórios e um hardware mais robusto.

Além disso, o sistema utiliza para comunicação uma ferramenta robusta, preparada para que em caso de indisponibilidade, não se perca a mensagens, como ocorre em uma integração, aonde se disponibilize apenas um *web service*. Entende-se que em caso de integração da ferramenta SIGA com outro *software*, caso este *software* fique indisponível, ou a ferramenta SIGA fique indisponível, as mensagens iriam se perder, levando há um retrabalho, pois isso poderia acarretar que as duas bases fossem analisadas, levando a erros e custos adicionais de algo que pode ser contornado com mecanismos de mensageria modernos e eficientes.

O *Docker*, que é um sistema que vai permitir uma camada de abstração para virtualização de sistema operacional vai ser executado com a ferramenta de mensageria RabbitMQ.

6.2 Cenários

Cenário 1: O sistema deverá se conectar com a base de dados utilizando uma solução objeto-relacional, de modo a minimizar o retrabalho em se adaptar o sistema a trabalhar com um banco de dados específico, bem como fornecer uma forma de se trabalhar com um bom nível de abstração dos dados. O sistema deverá criar as tabelas, caso elas ainda não exista;

Cenário 2: O sistema deverá utilizar um framework de autenticação de autorização que atenda o responsável pelo empreendimento, de modo a permitir um nível de segurança apropriado e que mantenha as informações estratégicas do empreendimento seguras. Deverá ser uma solução amplamente utilizada no mercado e com vasta documentação.

Cenário 3: O sistema deverá ser portável, ou em outras palavras, o sistema deverá permitir o acesso sem distorções em sua interface, de qualquer dispositivo. Deve-se, portanto, utilizar uma solução de *frontend* que permita que o sistema tenha essa característica.

Cenário 4: O sistema deverá utilizar uma solução moderna de mensageria, ou em outras palavras, o sistema deverá adotar uma ferramenta de comunicação que seja eficiente, e que trate uma eventual indisponibilidade da aplicação, não somente fornecer um serviço.

6.3 Avaliação

Cenário 1

|  |  |  |
| --- | --- | --- |
| Atributo de qualidade | | Persistência |
| Requisito de qualidade | | O sistema deverá utilizar um framework objeto-relacional |
| Preocupação | | |
| Necessidade do sistema de ser independente de banco de dados | | |
| Cenários | | |
| Cenário 1 | | |
| Ambiente | | |
| Sistema em operação normal | | |
| Estímulo | | |
| Criar as entidades dinamicamente no banco de dados | | |
| Mecanismo | | |
| Spring data com JPA | | |
| Medida da resposta | | |
| Criação do banco de dados a partir das classes do sistema | | |
| Considerações da arquitetura | | |
| Riscos | Para que a aplicação tenha um bom desempenho, é fundamental que o framework esteja configurado corretamente, de modo a não sobrecarregar o banco de dados. | |
| Pontos de sensibilidade | Camada de persistência | |
| *Tradeoff* | Não existe | |

Para este cenário é necessário adotar algumas abordagens na programação das entidades, como trazer somente a informação que eu preciso do banco de dados, evitando sempre que possível o *lazy loading* dos atributos.

O *pool* de conexões foi configurado no arquivo *application.properties*. Configurar o pool de conexões, embora pareça ser simples, é algo que pode otimizar o desempenho da aplicação.

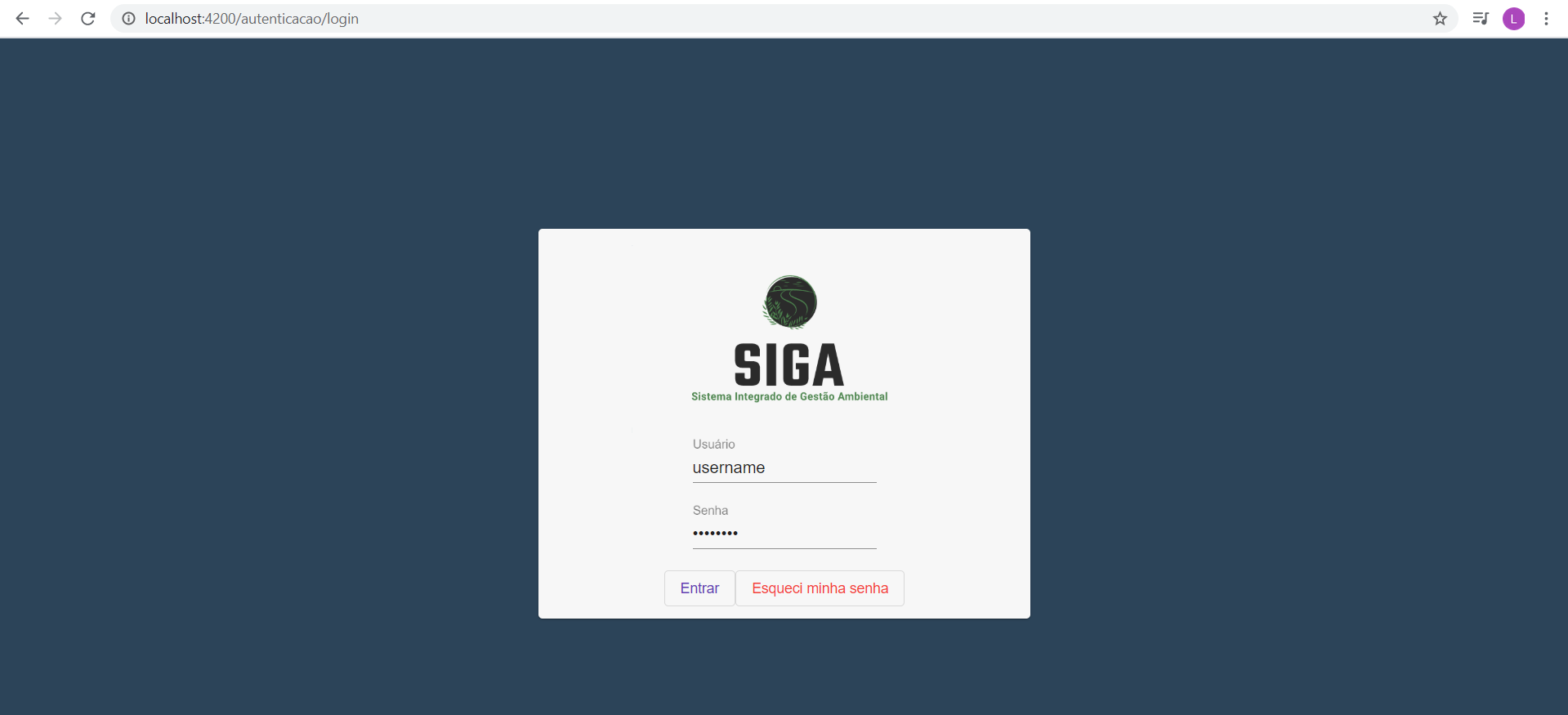
Outro ponto fundamental da implantação de um *framework* objeto-relacional é que se previne que a conexão seja utilizada de forma inadequada, programadores mais experientes que tiveram a oportunidade de trabalhar com sistemas legados sempre tiveram problemas em relação ao gerenciamento adequado da conexão, isso evita que determinado ponto da aplicação abra conexão sem fechar, levando a perda de performance da aplicação bem como queda do serviço do banco de dados.

Cenário 2

|  |  |  |
| --- | --- | --- |
| Atributo de qualidade | | O sistema deverá ser seguro |
| Requisito de qualidade | | O sistema deverá utilizar um framework seguro para autenticação e autorização do usuário |
| Preocupação | | |
| Necessidade de fornecer segurança ao responsável pelo licenciamento ambiental do empreendimento. | | |
| Cenários | | |
| Cenário 2 | | |
| Ambiente | | |
| Sistema em operação normal | | |
| Estímulo | | |
| Tentar realizar o login com credenciais inválidas e válidas para obtenção do *token* de acesso. Utilizar o token de acesso para checar se o *framework* está funcionando corretamente | | |
| Mecanismo | | |
| *Spring Security – Oauth 2.0* | | |
| Medida da resposta | | |
| Tentativa de *login* através do sistema | | |
| Considerações da arquitetura | | |
| Riscos | Acesso a recursos do sistema sem a autenticação poderia expor o negócio, prejudicando a estratégia da empresa, bem como invalidando todo o processo de licenciamento ambiental do empreendimento. É necessário também que a aplicação redirecione o usuário para a tela de login no *frontend.* | |
| Pontos de sensibilidade | Protocolo HTTPS | |
| *Tradeoff* | Não existe | |

Evidências do Cenário 2

Na figura abaixo, é mostrada a tela do sistema de login, caso o usuário tente acessar uma rota da aplicação e não estiver autenticado, o usuário é redirecionado ao login e mostrada uma mensagem que diz que é necessário fornecer as credenciais.



**Fig. 9 – Tela de login do sistema**

O usuário também é obrigado a informar o nome de usuário e a senha, caso o usuário tenha esquecido, ele pode optar pela opção esqueci minha senha, aonde um e-mail será pedido, e para este e-mail será encaminhado um link para redefinição de senha.



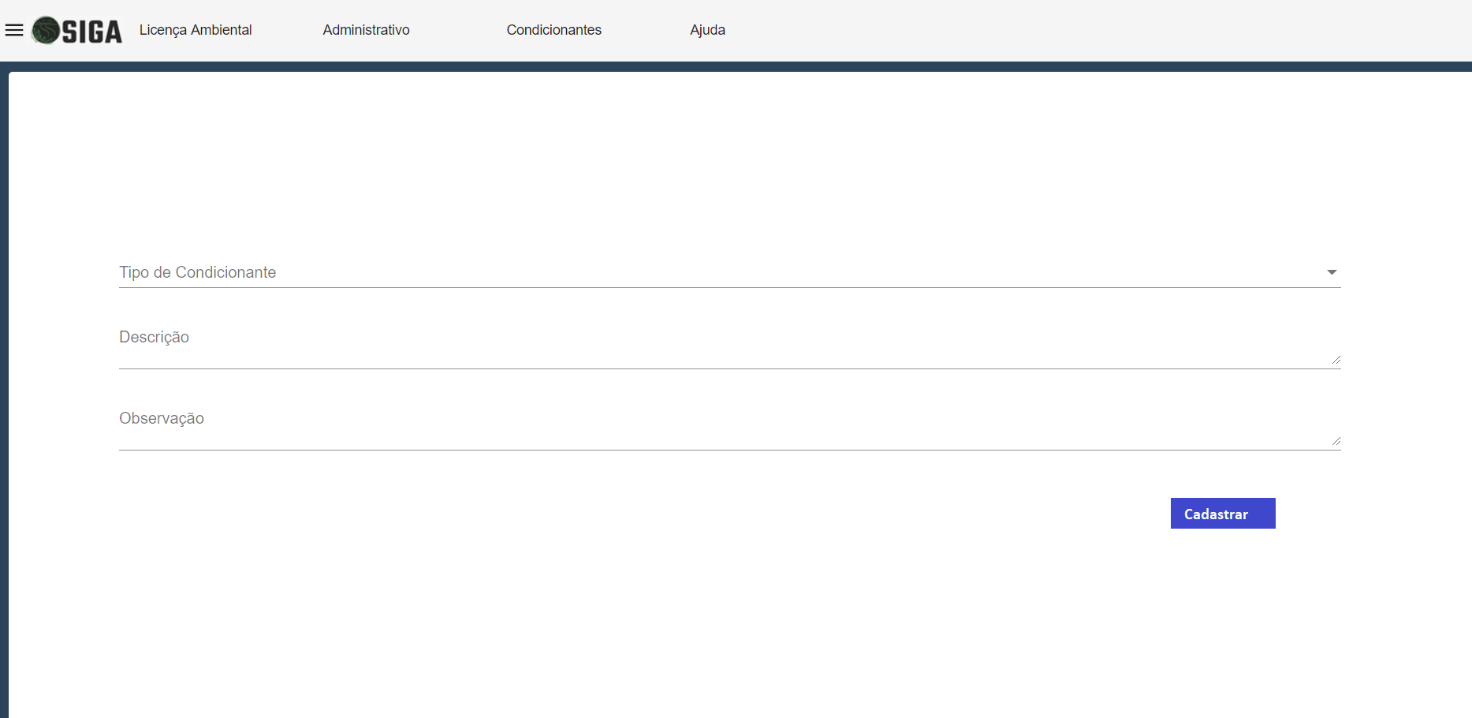
**Fig. 10 – Tela de recuperação de senha**

O usuário informa o e-mail informado no cadastro, e então é encaminhado para esse e-mail a senha e o usuário, caso exista na base de dados da aplicação, caso não exista, é exibida uma mensagem de erro e orientando a fazer o cadastro.

Cenário 3

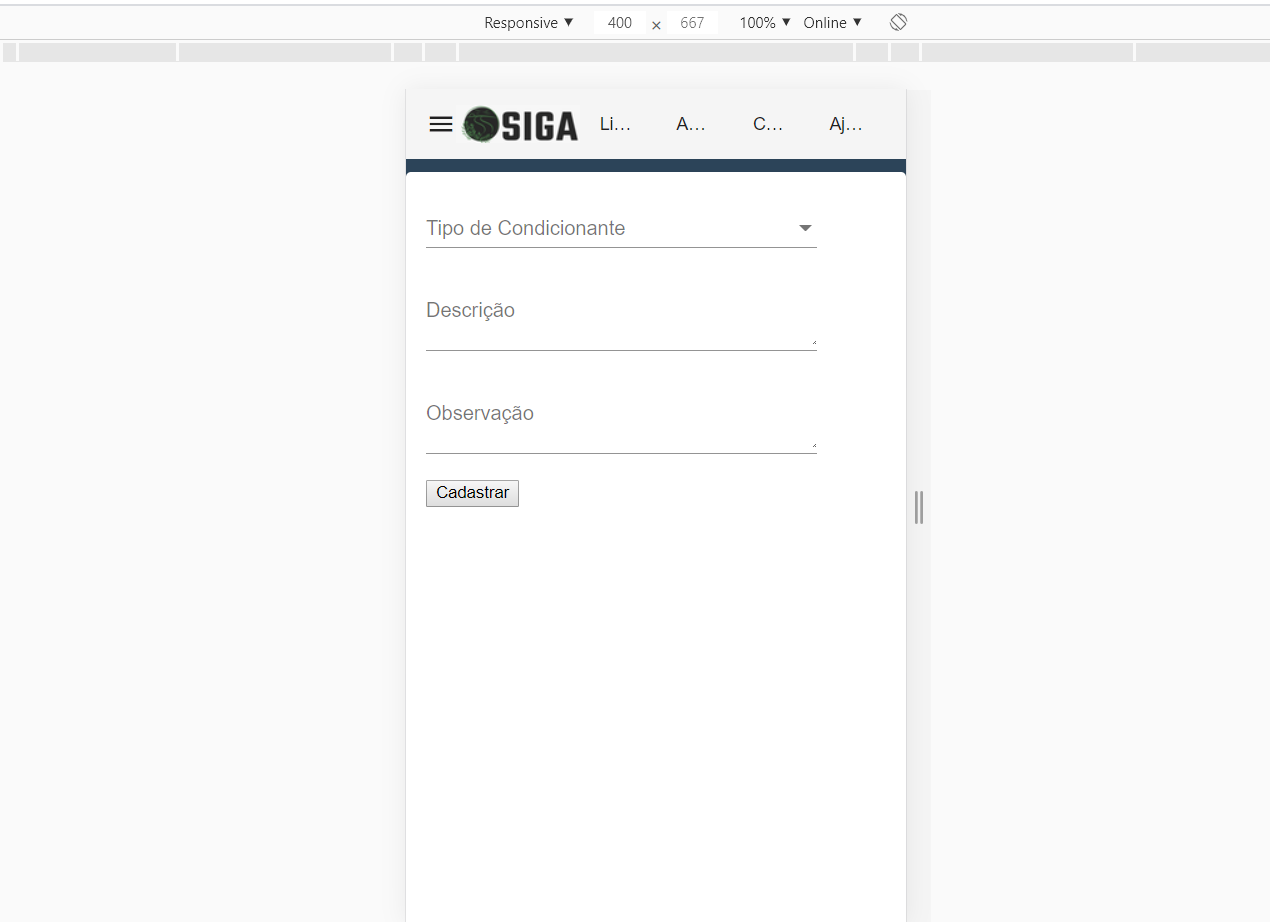
|  |  |  |
| --- | --- | --- |
| Atributo de qualidade | | O sistema deverá ser portável |
| Requisito de qualidade | | O sistema deverá utilizar um framework de *frontend* moderno que permita que a aplicação seja acessível de qualquer dispositivo. |
| Preocupação | | |
| É necessário fornecer uma camada de visão adequada que permita que a aplicação seja acessível a partir de diversos dispositivos.  É fundamental que o *frontend* do sistema suporte diferentes dispositivos. | | |
| Cenários | | |
| Cenário 3 | | |
| Ambiente | | |
| Sistema em operação normal | | |
| Estímulo | | |
| Acesso ao sistema | | |
| Mecanismo | | |
| *Angular, Angular Material* | | |
| Medida da resposta | | |
| Tela de cadastro de condicionante | | |
| Considerações da arquitetura | | |
| Riscos | Caso a aplicação não utilize um *framework* de *frontend* que permita o acesso de diferentes dispositivos, pode não fornecer uma boa usabilidade para o usuário, nem versatilidade.  Distorções provenientes de uma implementação restrita a determinado navegador, quando o usuário utilizar em outro. | |
| Pontos de sensibilidade | Protocolo HTTPS | |
| *Tradeoff* | Não | |

É fundamental que a aplicação tenha usabilidade, pois muito do trabalho do profissional com o perfil de operacional é realizado no empreendimento ou na obra, por exemplo, um profissional que visite um determinado terreno para realizar a medição pode tentar usar o *smartphone* ou um *tablet* para inserir informações. O *frontend* da aplicação deverá permitir que os dados cadastrados ou inseridos de forma fácil e prática.



**Fig. 11 - Tela de cadastro de condicionante**

Utilizando o *framework* Angular, podemos ver que a página se adapta ao acesso de um dispositivo de resolução menor como um *smartphone*, a tela do dispositivo é simulada pelo *Google Chrome.* Na aplicação utilizamos também o *Angular Material* que é uma suíte de componentes.



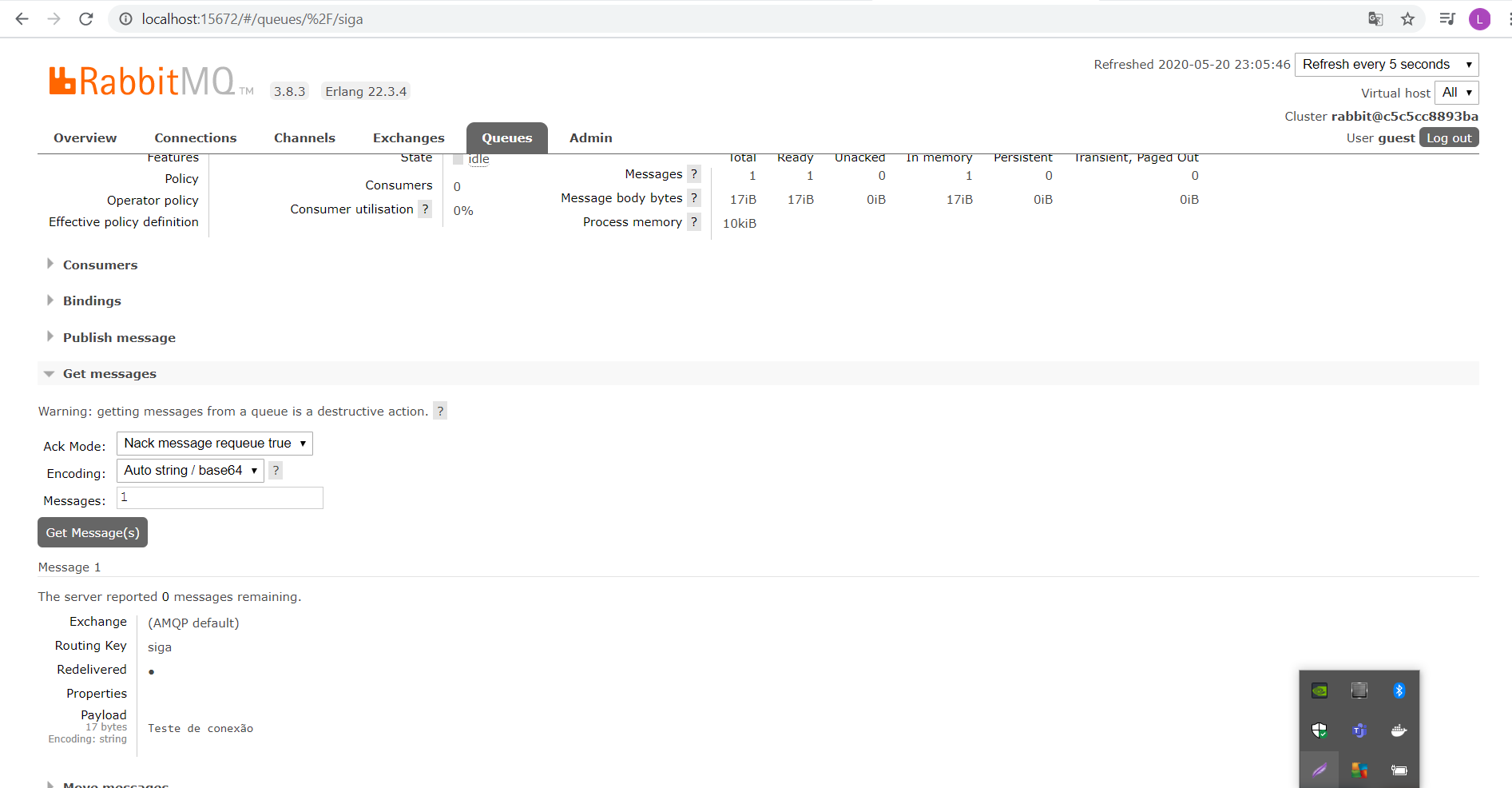
**Fig. 12 – Simulação de visualização da tela de cadastro de condicionante**

Cenário 4

|  |  |  |
| --- | --- | --- |
| Atributo de qualidade | | O sistema deverá utilizar uma ferramenta de mensageria |
| Requisito de qualidade | | O sistema deverá utilizar uma solução de mensageria. |
| Preocupação | | |
| É fundamental que a aplicação disponibilize uma solução de comunicação de tal forma que não se perca informação quando um sistema que vai receber essa informação fique indisponível, bem como que seja de fácil manutenção e implementação, com vasta biblioteca e comunidade.  Outro ponto determinante para a escolha do *RabbitMQ* é que ele tem suporte a diversos plug-ins, o que o torna uma opção mais interessante. | | |
| Cenários | | |
| Cenário 4 | | |
| Ambiente | | |
| Sistema em operação normal | | |
| Estímulo | | |
| Envio de mensagem através do sistema em produção normal | | |
| Mecanismo | | |
| *RabbitMQ* rodando em *Docker* | | |
| Medida da resposta | | |
| Classe de teste | | |
| Considerações da arquitetura | | |
| Riscos | Indisponibilidade da ferramenta | |
| Pontos de sensibilidade | Os diversos protocolos suportados pela ferramenta | |
| *Tradeoff* | Não | |

Evidências do cenário 4

Para realizar o teste, utilizamos o *Docker*, o download da imagem é feito através do comando *docker pull*, e com outro comando simples, é criado um container com a ferramenta *RabbitMQ* pronta para o uso, com uma interface simples e amigável. Outra vantagem enorme sobre o uso de SOAP é que é muito simples encaminhar mensagens para o *RabbitMQ* bem como mais seguro. Abaixo é mostrado a tela do *RabbitMQ* com a mensagem enviada pela classe de teste da aplicação.



**Fig. 12 – Tela do *RabbitMQ* com a mensagem**

**6.4 Resultado**

A avalição da arquitetura que proposta nos permitiu realizar uma checagem para ver se os requisitos funcionais e não funcionais foram atendidos bem como examinar pontos determinantes bem como possíveis limitações.

|  |  |  |
| --- | --- | --- |
| **Requisitos Não funcionais** | **Testado** | **Homologado** |
| RNF 1 – O sistema deverá utilizar um *framework* objeto-relacional. | SIM | SIM |
| RNF 2 – O sistema deverá ter boa Manutenibilidade | SIM | SIM |
| RNF 3 – O sistema deverá ser seguro e utilizar o protocolo de segurança *Oauth 2.0* implementado através do *Spring Security* | SIM | SIM |
| RNF 4 – O sistema deverá ter boa acessibilidade | SIM | SIM |
| RNF 5 – O sistema deverá ter boa comunicação | SIM | SIM |

O Angular proporciona uma camada de *frontend* que pode ser acessada de diferentes dispositivos sem distorções que ocorreriam se não fosse utilizado um framework responsivo, o que torna a arquitetura moderna e de layout interessante para o usuário final.

O *Spring Boot* facilita a criação de aplicações baseadas no *Spring Framework* para a camada de *backend*, o que permitiu que a configuração da aplicação utilizasse mais *annotations* e não XML, o que facilita o desenvolvimento, e o torna mais dinâmico e a utilização de um *framework* objeto-relacional faz com que a aplicação seja independente de uma solução de banco de dados específica.

A aplicação também utiliza um mecanismo moderno e eficiente de autenticação e autorização, o que faz com que os dados de negócio estratégicos sejam preservados.

A arquitetura proposta também utilizou uma forma de comunicação que tem por prioridade não ser somente um serviço disponibilizado de qualquer forma, no sentido de preservar as mensagens do sistema quando um sistema receptor não perca as informações por alguma eventual indisponibilidade.

Concluiu-se que arquitetura proposta se mostrou eficiente e de fácil manutenção, utiliza mecanismo de autorização eficiente e se mostrou satisfatória.

7. Conclusão

O trabalho apresenta a arquitetura de um sistema de gestão ambiental baseado em *Spring Boot, Spring Security* utilizando autenticação Oauth 2.0 e JPA. A arquitetura proposta é simples, porém eficiente para uma equipe pequena de desenvolvedores. O sentido para se manter uma arquitetura pequena e exuta é justamente atender o usuário final de forma satisfatória sem a necessidade de se criar um sistema complexo e difícil de utilizar, pouco flexível e que não atenda a demanda do cliente.

REFERÊNCIAS

BRASIL. Resolução nº 237 de 19 de dezembro de 1997. CONAMA

GUEDES, Gilleanes T.A . *UML – Uma Abordagem prática.*3ª ed. São Paulo, Editora Novatec, 2018.

COLI, Adriana; Dias, Pedro. *O Setor Elétrico e o meio ambiente*, 1º Ed. Rio de Janeiro, Editora Synergia, 2017

MELO, Ana Cristina. *Desenvolvendo Aplicações com UML 2.2.* 3º Ed. Rio de Janeiro, Editora Brasport, 2010

MARTINS, José Carlos Cordeiro. *Tecnicas Para Gerenciamento de Projeto de Software.* 1º Ed. Rio de Janeiro, Editora Brasport, 2007.

PRESSMAN, Roger S. ; Bruce R. Maxim. Engenharia de Software, Uma Abordagem Profissional, 8° Ed. Porto Alegre: AMGH, 2016.

RICHER, Justin; SANSO, Antônio. *Oauth 2 in Action*. 1º Ed. *Nova York:* Editora *Manning Plublications, 2017.*

APÊNDICES

*Usuário com perfil de administrador*

*Usuário:* admin

*Senha:* admin

*Usuário com o perfil de operacional*

*Usuário:* cliente

*Senha:* cliente

*CHECKLIST* PARA VALIDAÇÃO DOS ITENS E ARTEFATOS DO TRABALHO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Completeza do documento** | | | | |
| **Nº** | **Item a ser cumprido** | **Sim** | **Não** | **Não se aplica** |
| 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 do caso de uso? |  |  |  |
| 8 | Um modelo de componentes e uma breve descrição de cada componente foi apresentado? |  |  |  |
| 9 | Um modelo de implantação e uma breve descrição de cada modelo foi apresentada? |  |  |  |
| 10 | Um modelo de implantação e uma breve descrição de cada elemento do hardware foi apresentada? |  |  |  |
| 11 | Prova de conceito: As tecnologias foram listadas? |  |  |  |
| 12 | Prova de conceito: Os casos de uso e os requisitos não funcionais usados para validar a arquitetura foram apresentados? |  |  |  |
| 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: Os atributos de qualidade e os cenários onde eles seriam validados foram apresentados? |  |  |  |
| 19 | Avaliação da arquitetura: Os atributos de qualidade e os cenários onde eles seriam validados foram apresentados? |  |  |  |
| 20 | As referências bibliográficas foram listadas? |  |  |  |
| 21 | As URLs com os códigos e com o vídeo da apresentação da POC foram listadas? |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Consistência dos itens do documento** | | | | |
| **Nº** | **Item a ser cumprido** | **Sim** | **Não** | **Não se aplica** |
| 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 | Os modelos de componentes e implantação 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 dentro do contexto dos casos de uso implementados? |  |  |  |
| 9 | O apresentado no item resultado está coerente com o que foi mostrado no item avaliação? |  |  |  |