**DOCUMENTAÇÃO DA API**

Sumário

[**1.** **Interdependências** 3](#_Toc24966138)

[**2.** **Solução** 3](#_Toc24966139)

[**2.1.** **Importador** 4](#_Toc24966140)

[**2.2.** **Diagrama Entidade Relacionamento - DER** 5](#_Toc24966141)

[**2.3.** **Portal WEB** 6](#_Toc24966142)

[**2.3.1.** **Login** 6](#_Toc24966143)

[**2.3.2.** **Usuários** 7](#_Toc24966144)

[**2.3.3.** **Token de Acesso** 8](#_Toc24966145)

[**2.3.4.** **Radares** 9](#_Toc24966146)

[**2.3.5.** **Trajetos** 9](#_Toc24966149)

[**2.3.6.** **Proposta de Análise Comparativa de Dados** 10](#_Toc24966152)

[**3.** **Requerimentos de Desempenho – Teste de Stress** 10](#_Toc24966155)

[**4.** **Requerimentos de Segurança** 10](#_Toc24966156)

[**5.** **Retorno dos dados da aplicação** 11](#_Toc24966158)

[**6.** **Documentações Externas** 11](#_Toc24966159)

[**7.** **Diagrama de Caso de Uso - SUC** 11](#_Toc24966160)

[**9. Cronograma de Desenvolvimento** 12](#_Toc24966161)

**Necessidade do Negócio**

Desenvolvimento de aplicação WEB para sistematização da proposta de como organizar os dados obtidos através de equipamentos de fiscalização eletrônica de trânsito, no **Município de São Paulo**, e disponibilizá-los para utilização pela Administração Pública e pela comunidade em geral.

1. **Interdependências**

Para funcionamento da aplicação, será necessário desenvolver um Importador de Dados que será responsável por carregar as informações dos dispositivos de Radar – coletados pela Secretaria Municipal de Trânsito, e organiza-los de acordo com as especificações previstas. Os dados fornecidos deverão ser disponibilizados em formato PostgreSQL.

Na arquitetura definida para o projeto precisa-se de um ambiente Windows Server ou distribuição do tipo Linux, com instalação do Framework Spring Boot 2.2.0 para hospedagem das páginas da API, que serão desenvolvidas na tecnologia Java 11.0.5 e o banco de dados PostgreSQL 11.

1. **Solução**

A aplicação será desenvolvida em plataforma Spring Boot 2.2.0 na linguagem Java 11.0.5 e irá contemplar as seguintes funcionalidades:

1. **Interface inicial**

* Cadastramento de usuários com suas devidas permissões – comum e administrador, assim como o passo a passo para o auto cadastramento;
* Geração do token de acesso e o esclarecimento da sua utilização.

1. **API Radares**

* Busca e consulta dos radares por suas devidas localizações;
* Busca e consulta dos radares pelo seu devido enquadramento;
* Busca e consulta dos radares pelas suas devidas concessões;
* Busca e consulta dos radares de acordo com o fluxo de veículos;
* Busca e consulta dos radares para a identificação do tipo de veículo;
* Busca e consulta dos radares para número de infrações registradas;
* Busca e consulta dos radares para a acurácia de identificação das placas;
* Busca e consulta dos radares para a identificação da distribuição das velocidades por radar.

1. **API Trajetos**

* Busca e consulta dos radares para a identificação de trajetos;
* Busca e consulta dos radares para a identificação das velocidades médias por trajeto;
* Busca e consulta dos radares para a identificação das distâncias por trajeto.

1. **Exportações**

Os relatórios gerados pelas consultas de identificação citadas acima poderão ser exportados nos formatos CSV e PDF.

* 1. **Importador**

A aplicação deverá ser instalada em um servidor Windows Server ou distribuição do tipo Linux, sugerimos que seja instalada no mesmo servidor do banco de dados, com objetivo de mitigar problemas relacionados a infraestrutura de rede durante o processamento das cargas.

No mapeamento das fontes de dados, serão carregadas as informações conforme quadro abaixo.

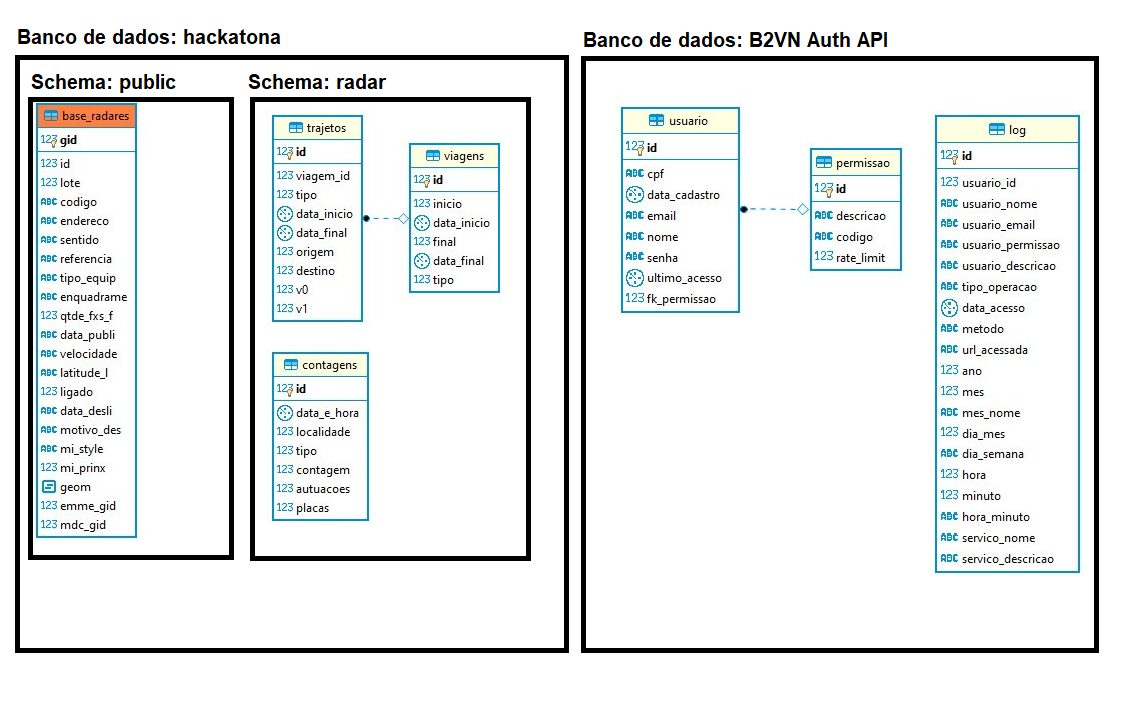
|  |  |  |
| --- | --- | --- |
| **Origem** | **Fonte** | **Formato** |
| Localização dos radares | Base PostgreSQL (Radar API) | XML, JSON, CSV, PDF |
| Enquadramento dos radares | Base PostgreSQL (Radar API) | XML, JSON |
| Zona de concessões dos radares | Base PostgreSQL (Radar API) | XML, JSON, PDF |
| Fluxo de veículos por radar | Base PostgreSQL (Radar API) | XML, JSON, PDF |
| Velocidade de veículo por radar | Base PostgreSQL (Radar API) | XML, JSON, PDF |
| Infrações por radar | Base PostgreSQL (Radar API) | XML, JSON, PDF |
| Acurácia de identificação de placa por radar | Base PostgreSQL (Radar API) | XML, JSON |
| Perfil/distribuições das velocidades por radar | Base PostgreSQL (Radar API) | XML, JSON |
| Trajetos | Base PostgreSQL (Radar API) | XML, JSON |
| Velocidade média por trajeto | Base PostgreSQL (Radar API) | XML, JSON |
| Distância do trajeto | Base PostgreSQL (Radar API) | XML, JSON |

Os arquivos obtidos e disponibilizados pelo banco de dado dos radares, serão carregados e atualizados em um período de 4 minutos. Dessa forma, devido ao altíssimo número de informações, a velocidade de resposta dependerá da performasse do servidor onde a aplicação será executada.

***Ponto de Atenção: os arquivos disponibilizados diretamente pelo banco de dados não podem ter alteração em sua estrutura.***

* 1. **Diagrama Entidade Relacionamento - DER**

O diagrama entidade relacionamento é um modelo de dados que descreve os aspectos de informação de um domínio de negócio ou seus requisitos de processo, de uma maneira abstrata e mostra as tabelas e as colunas do banco de dados.



O diagrama acima facilita ainda a comunicação entre os integrantes da equipe, pois oferece uma linguagem comum utilizada tanto pelo analista, responsável por levantar os requisitos, e os desenvolvedores, responsáveis por implementar aquilo que foi modelado.

* 1. **Portal WEB**

Essa seção descreve as telas e as funcionalidades visuais do sistema.

* + 1. **Login**



Para acesso ao Portal, o usuário deverá informar seu e-mail e senha. Caso tenha esquecido a senha, será possível recuperar sua senha através do link - **Esqueci minha senha**

Após o processo de login no Portal, o usuário terá acesso a página principal. Na parte superior da página está localizado o menu de acesso as funcionalidades. Mais à direita, está localizada o acesso aos dados do usuário e a opção de saída do Portal (logout).

Caso o usuário tenha a permissão de administrador, será apresentado o título “administrador” no menu de acesso.

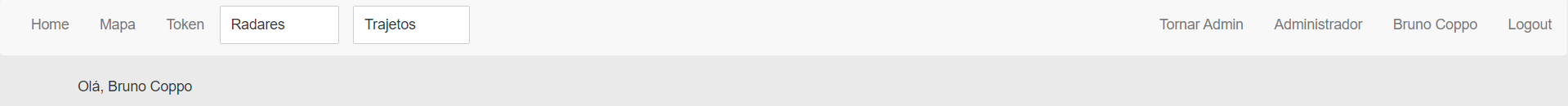
O Portal está subdividido em 2 módulos, sendo eles:

1. Módulo de Autorização, onde estão concentradas as configurações de acessos;
2. Módulo de Consulta, onde estão concentradas as buscas e consultas a partir radares para identificação de parâmetros, principal funcionalidade deste projeto.
   * 1. **Usuários**

Buscou-se desenvolver um padrão de usabilidade nas páginas do Portal que proporcione uma boa experiência aos usuários.

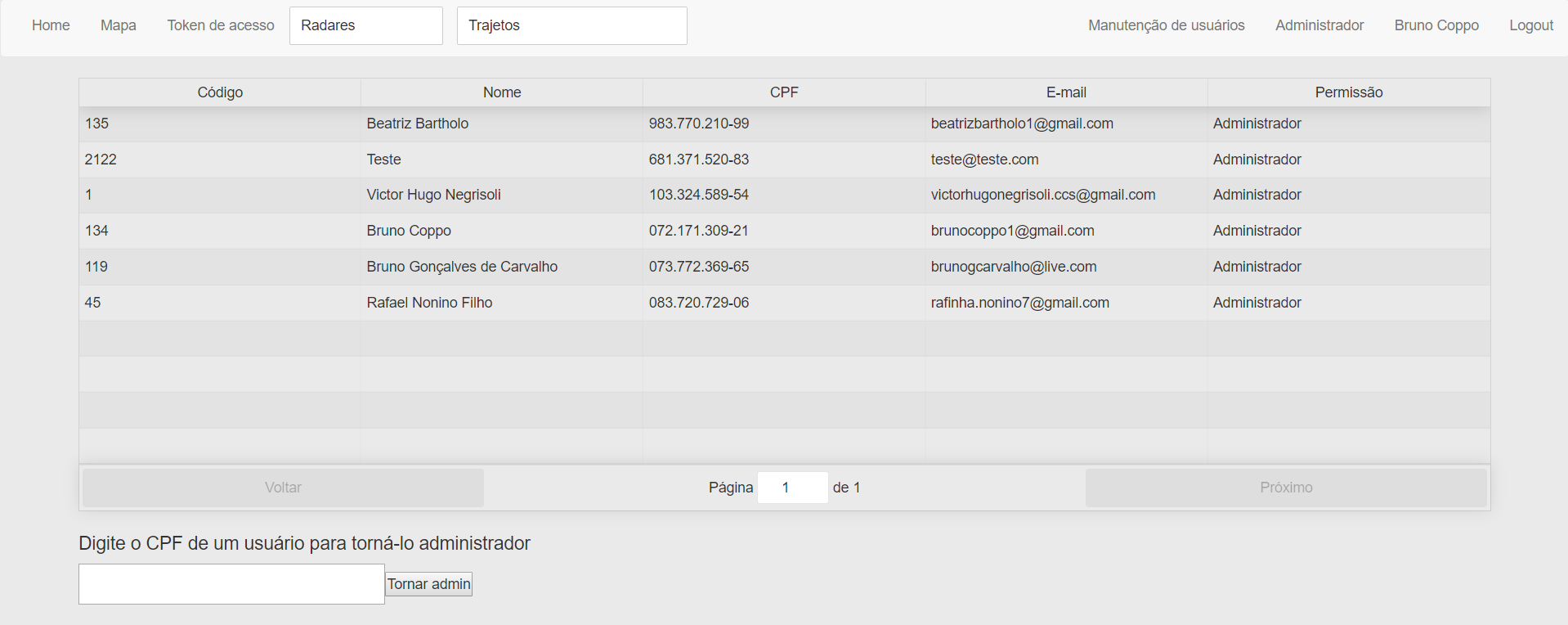
Dessa forma, para facilitar a utilização da página, desenvolveu-se um auto cadastramento de usuários. Com tal, inseriu-se a funcionalidade de recuperação de senha e a identificação de duplos cadastros, caso ocorra.

Os usuários com a permissão de administrador podem, além das funcionalidades básicas, consultar os usuários cadastrados no sistema e alterar a permissão dos mesmos para administrador.



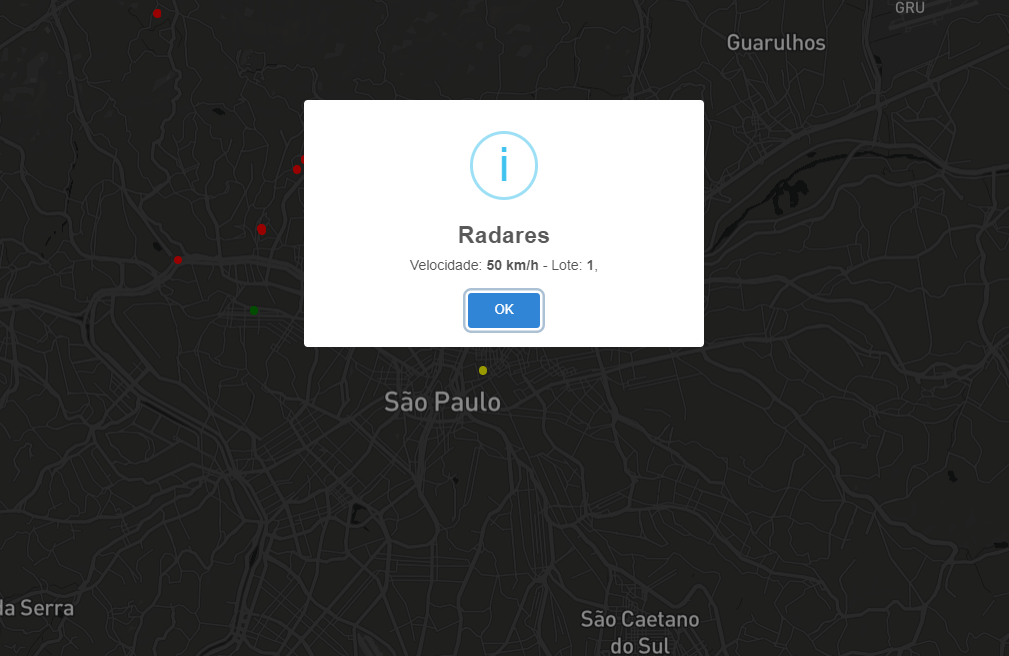
O Portal terá 2 perfis de acesso:

1. Administrador: acesso às funcionalidades do Portal e configurações gerais da conta dos usuários cadastrados;
2. Usuário comum: acesso às funcionalidades do Portal.



**Mapa**

O processo de mapeamento possui o objetivo de facilitar a visualização dos mesmos, podendo filtrar os radares de acordo com o lote. Ao clicar, é informado os dados dos dispositivos como, por exemplo, a velocidade máxima permitida e o lote em questão.



* + 1. **Token de Acesso**

Um token de acesso é uma assinatura única e específica que está vinculada a um usuário ou sessão ativa em um servidor de aplicação. Existem alguns padrões de tokens de acesso disponibilizadas em serviços web, tais como o padrão JSON Web Token ou JWT.

Quando as requisições são feitas a um servidor, são executadas e então morrem, matando sua presença no servidor, e, para controlar quem pode ou não acessar um endpoint em um servidor, é necessária alguma assinatura que permita ao servidor saber quem está requisitando.

O padrão Bearer token utiliza uma hash que identifica um usuário ativo na aplicação, enviando essa autorização pelo header da requisição. Esse header é identificado pelo Spring Security e pelo protocolo OAuth2.0 e direciona o acesso do usuário, que pode ser um status 200, de requisição aceita, um 401, de sem autorização ou de token expirada, um status 403, de rota proibida, entre outros status de resposta de métodos HTTP.

* + 1. **Radares**

O menu Radares possui as seguintes subdivisões:

* Consulta dos radares por suas devidas localizações;
* Consulta dos radares pelo seu devido enquadramento;
* Consulta dos radares pelas suas devidas concessões;
* Consulta dos radares de acordo com o fluxo de veículos;
* Consulta dos radares para a identificação do tipo de veículo;
* Consulta dos radares para a identificação de infrações;
* Consulta dos radares para a acurácia de identificação das placas;
* Consulta dos radares para a identificação da distribuição das velocidades por radar.

A partir do acesso às subdivisões, o sistema retornará os dados, paginados, para as devidas consultas. Dessa forma, o usuário poderá analisar e filtrar, de forma clara, os parâmetros indicados no banco de dados.

* + 1. **Trajetos**

O menu Trajetos possui as seguintes subdivisões:

* Consulta dos radares para a identificação de trajetos;
* Consulta dos radares para a identificação das velocidades médias por trajeto;
* Consulta dos radares para a identificação das distâncias por trajeto.

A partir do acesso às subdivisões, o sistema retornará os dados, paginados, para as devidas consultas. Dessa forma, o usuário poderá analisar e filtrar, de forma clara, os parâmetros indicados no banco de dados.

* + 1. **Proposta de Análise Comparativa de Dados**

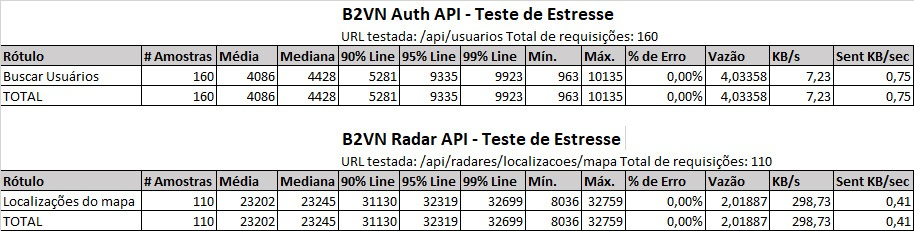
Desenvolveu-se um exemplo de análise entre dados para exemplificar o possível uso dos dados aqui organizados e estruturados. Para isso relacionou-se dados de acurácia (eficiência de registros de veículos de um determinado radar) **x** concessão do radar (lote de compra/concessão que o radar faz parte). Como resultado têm-se um relatório por gráfico, onde o administrador público pode ponderar a eficiência de cada lote/origem dos radares.

Da mesma forma, propõe-se análises do tipo: fluxo de veículos **x** estudo de emissão de gases/veículo **x** estudo de microclima; fluxo de veículos **x** chamados 156.

1. **Requerimentos de Desempenho – Teste de Estresse**

O objetivo dos testes de stress é a análise da arquitetura desenvolvida, verificando se a solução desenvolvida pode realmente responder à demanda.

Para tal teste na aplicação desenvolvida, fora utilizado o Software JMeter



1. **Requerimentos de Segurança**

Toda a segurança e autenticação do projeto foi desenvolvida através do framework Spring Security, um subprojeto dentro do Spring Framework que disponibiliza serviços de autenticação e autorização. O protocolo de autenticação utilizado é o OAuth2.0, fazendo também o uso do padrão Bearer para disponibilização de tokens de acesso à API.

1. **Retorno dos dados da aplicação**

Os dados da aplicação estão sendo retornados em JSON e, para minimizar os erros de compatibilização, os mesmos estão sendo retornados, também, em XML.

1. **Documentações Externas**

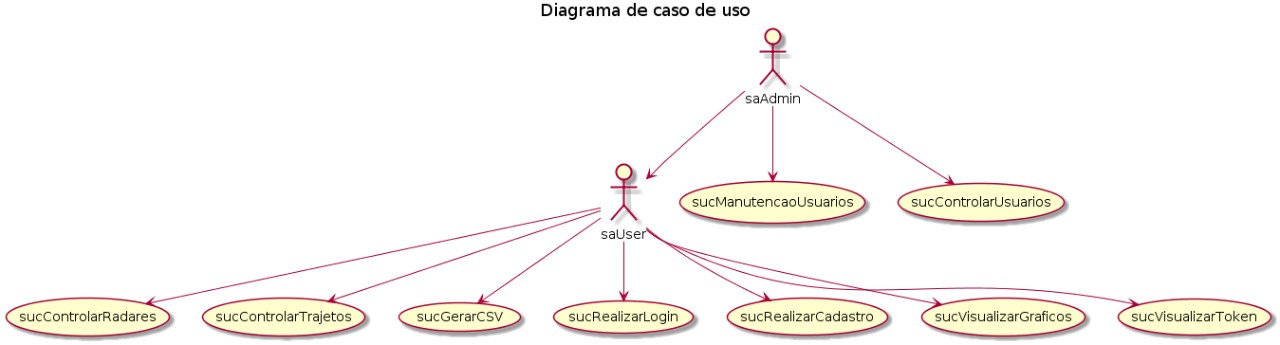
A documentação da API está toda em Swagger.

Para análise, consulte: [**http://localhost:8081/swagger-ui.html**](http://localhost:8081/swagger-ui.html)

1. **Diagrama de Caso de Uso - SUC**

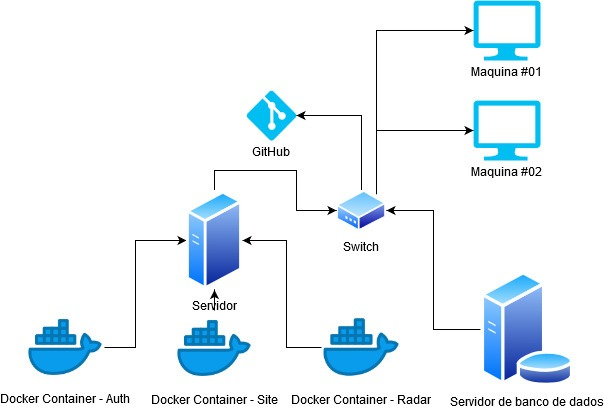
O primeiro diagrama da UML que será visto é denominado diagrama de caso

de uso, esse diagrama descreve as funcionalidades propostas pelo projeto e é uma excelente ferramenta para o levantamento dos requisitos funcionais do sistema.



**Diagrama de Implantação**

O diagrama de implantação é o diagrama estrutural responsável por estabelecer a relação entre os recursos de infraestrutura e artefatos do sistema, como pode



**9. Cronograma de Desenvolvimento**

O cronograma de desenvolvimento formulado para a entrega do produto final em 30 dias está apresentado no anexo desse documento.