Documento de Projeto de Arquitetura

PROPOSTA DE UMA ARQUITETURA DE SOFTWARE COM MICROSSERVIÇOS PARA SISTEMAS DE ALTA DEMANDA

Equipe:

Tiago Xavier de Paiva

Cristiane Yea Imamura

Sumário

1 Introdução

Este documento descreve os artefatos obtidos pela análise e projeto de uma aplicação web e mobile em conjunto de uma API, os quais serão integrados a um ecossistema de microsserviços responsáveis por realizar o rastreamento veicular. Aqui, portanto, é possível encontrar as informações importantes sobre as necessidades e requisitos de um sistema dessa natureza. Com base neles, são mostrados os padrões empregados na definição de um Projeto de Arquitetura que possa ser implementado de forma a resultar em uma solução sólida de software

2 Objetivos, Requisitos e Restrições

2.1 Objetivo do Produto

O sistema deverá ser capaz de processar uma alta quantidade de dados de forma fluida, com o uso de mensagerias para evitar eventuais custos ao banco de dados. Este também deve ser capaz de escalar suas aplicações se adaptando a picos ou baixas de dados em seus diversos fluxos.

2.2 Funções do Produto

O sistema deverá incorporar as capacidades básicas de um sistemas de rastreamento veicular, disponibilizando o posicionamento georreferenciado dos rastreadores veiculares (fornecidos pelo equipamento IOT E3+ via TCP utilizando o protocolo de comunicação *Easy Track*), permitindo o envio de comandos remotos via interface web ou mobile. Em suma o sistema de rastreamento veicular proposto implementará as seguintes funcionalidades:

- Visualização georreferenciada dos veículos em tempo real;
- Grid de situação dos veículos em tempo real;
- Histórico de posição dos veículos;
- Envio de comandos aos equipamentos;
- Monitoramento de Cercas eletrônicas;
- Alertas via Whatsapp e E-mail;

2.3 Restrições Gerais

O acesso do sistema deverá ser efetuado via aplicativo mobile ou aplicativo web através de um navegador de internet, Para o acesso cada usuário deverá ser identificado por meio de autenticação (login e senha). Somente o administrador do sistema terá a capacidade de realizar a criação de acessos para os clientes, assim como a criação de novos veículos ou equipamentos para que eles sejam aceitos e reconhecidos na integração.

2.4 Suposições e Dependências

O Servidor necessita dos seguintes requisitos mínimos:
Sistema operacional: Ubuntu Server;
Processador: AMD EPYC 7282 16 Cores ou semelhantes;
Memória RAM: 32GB ou superior;
Espaço em Disco: 200 SSD ou superior.

O Cliente necessita dos seguintes requisitos mínimos:
Dispositivo: Smartphone Android/IOS ou
Computador com acesso à internet.

2.5 Requisitos

Os requisitos do sistema são divididos em funcionais e não funcionais e eles podem ser observados no Quadro 1 e Quadro 2, respectivamente.

Quadro 1 – Requisitos Funcionais

Requisitos Funcionais	
Requisito	**Descrição**
RF_01	O sistema deverá disponibilizar ao usuário uma visualização resumida de todos os veículos integrados, em conjunto das seguintes informações (status atual, horário da última comunicação e endereço).
RF_02	O sistema deverá disponibilizar em conjunto ao resumo do RF_01 um mapa mostrando a localização de todos os veículos integrados no sistema em tempo real.
RF_03	O sistema deverá permitir com base em um período a visualização do histórico de posições de um veículo.
RF_04	O sistema deverá permitir que o usuário cadastre contatos de celular e e-mail.
RF_05	O sistema deverá permitir que o cliente cadastre cercas eletrônicas para o monitoramento da entrada e saída dos veículos de uma determinada região.
RF_06	O sistema deverá permitir que o usuário vincule veículos a cercas eletrônicas.
RF_07	O sistema deverá permitir que o usuário vincule contatos a cercas eletrônicas.
RF_08	O sistema deverá possuir diferentes níveis de usuários: usuário(cliente) e administrador. Somente administradores podem criar usuários e veículos como também realizar o seu vínculo cliente-veículos.
RF_09	O sistema deverá possuir uma integração com E-mail e WhatsApp para o envio de alertas para os contatos definidos pelo usuário.
RF_10	O sistema deverá possuir a capacidade de envio de comandos para os equipamentos aplicáveis e disponibilizar um acompanhamento de seu processamento e recebimento.

Fonte: Autoria própria.

Quadro 2 – Requisitos Não Funcionais

Requisitos Não Funcionais	
Requisito	**Descrição**
RFN_01	Os dados recebidos devem ser disponibilizados para visualização do cliente com a menor latência possível.
RFN_02	Os componentes do sistema devem lidar com as informações de forma genérica, formando fluxos que padronizam os dados recebidos e em seguida os processa via um fluxo comum.
RFN_03	O sistema deve estar preparado para lidar com um recebimento constante de dados que pode aumentar ou diminuir de volume conforme as horas do dia.
RNF_04	O sistema deve possuir mecanismos para evitar ao máximo qualquer tipo de perda de dados entre seus fluxos em casos de adversidades em serviços como banco de dados, mensagerias e afins.
RNF_05	O sistema deve utilizar prioritariamente os dados fornecidos via fluxos de mensageria, evitando ao máximo a utilização de consultas em banco de dados.

Fonte: Autoria própria.

3 Visão Geral sobre a Arquitetura

Para assegurar a confiabilidade e resiliência do sistema, será empregada a arquitetura de microsserviços onde os diferentes componentes do sistema serão repartidos em aplicações menores e de responsabilidade única, obtendo como resultado um sistema com processos desacoplados.

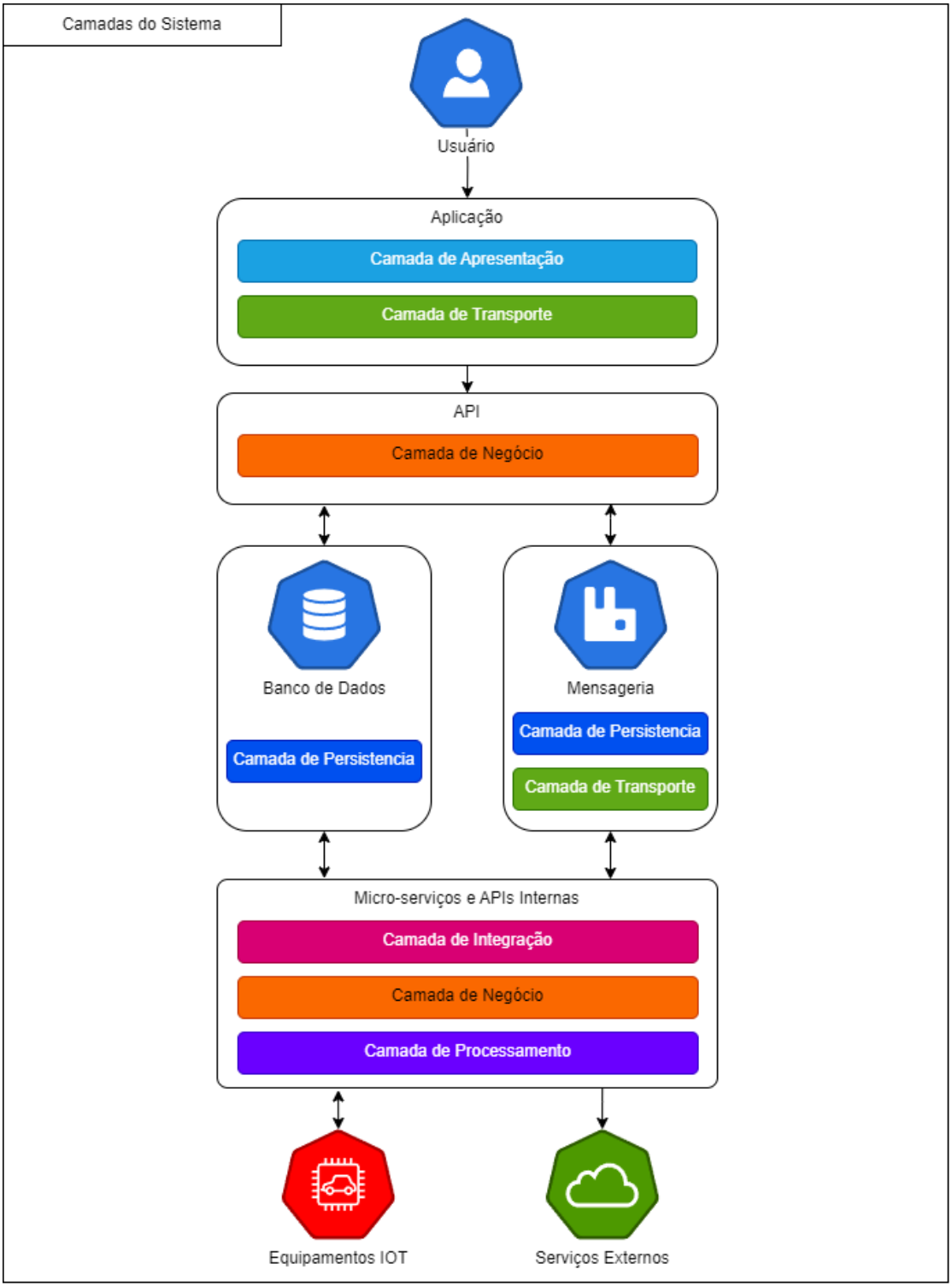
A coordenação dos fluxos de processamento será através do uso de um serviço de mensageria, onde as aplicações irão adotar os comportamentos definidos pelas arquiteturas "*pipes and filter*" e "orientada a eventos", o emprego dessas arquiteturas permitirá que o sistema distribua e filtre o processamento através de canais com isso organizando o processo em etapas e persistindo dados intermediários sem a necessidade de realizar buscas em banco de dados.

O sistema foi dividido em 5 camadas conforme descrito e demonstrado abaixo na **Figura 1**:

- **Aplicação**: Camada responsável por fornecer ao usuário a interface para a utilização do sistema possibilitando-o de exercer atividades como: o envio de comandos, monitoramento da frota de veículos e a visualização do histórico dos veículos;
- **API**: Camada que conterá as validações e regras de negócios a serem exercidas pela interface do usuário (a camada de aplicação), e realizará a ligação com o banco de dados e a mensageria;
- **Banco de Dados**: Camada que provê a persistência dos dados de forma definitiva, seu papel neste projeto será majoritariamente para o armazenamento de dados finais os quais serão somente consumidos pela API, o seu consumo por um micro-serviço deverá somente ocorrer em ocasiões em que não exista uma alternativa mais viável;

- **Mensageria**: Camada que realizará a distribuição e controle dos dados a serem processados para as aplicações, garantindo a não duplicidade e persistência desses dados transientes;
- **Microsserviços e APIs Internas**: Camada responsável pela integração com os equipamentos *IOT* e serviços externos, o processamento dos dados e a aplicação de regras de negócios.

Figura 1 – Camadas do Sistema



Fonte: Autoria própria.

4 Componentes Arquiteturais

Com base nas camadas previamente definidas na **Figura 1**, agora serão descritos os módulos do sistema e seus respectivos componentes.

4.1 Camada de Aplicação

A aplicação será dividida em dois módulos, sendo o primeiro a aplicação web e o segundo a aplicação mobile, ambas disponibilizarão ao usuário o mesmo conjunto de acessos e ferramentas, permitindo assim o consumo da mesma integração e eliminando a necessidade de ajustes especiais para cada ambiente, divergindo assim somente na apresentação de suas interfaces.

4.2 Camada de API

A API será composta de um módulo o qual irá expor *endpoints http* para consumo. Nele estarão disponíveis os seguinte serviços:

- Autenticação:
  - (POST)Realização de Login;
  - (POST)Revalidação de Token (Opção “Lembrar-me”);
- Posição:
  - (GET)Obter últimas posições dos veículos;
  - (GET)Obter última posição de um veículo;
  - (GET)Obter posições por período;
  - (GET)Obter posição de um veículo por período;
  - (WEB-SOCKET)Obter posições em tempo real;
- Veículo:
  - (GET)Obtenção de dados dos veículos;
  - (GET)Obtenção de dados de um veículo em específico;
- Comando:
  - (GET)Obter comandos disponíveis para um veículo;
  - (GET)Obter último comando enviado a um veículo;
  - (GET)Obter comandos enviados a um veículo por período;
  - (POST)Enviar comando a um veículo;
- Cerca Eletrônica:
  - (GET)Obter todas as cercas eletrônicas cadastradas;
  - (GET)Obter cerca eletrônica por identificador;
  - (POST)Cadastrar cerca eletrônica;
  - (DELETE)Excluir cerca eletrônica;
  - (PUT)Editar cerca eletrônica;
- Contato:
  - (GET)Obter todos os contatos cadastrados;
  - (GET)Obter contato por identificador;
  - (POST)Cadastrar contato;
  - (DELETE)Excluir contato;
  - (PUT)Editar contato;

O módulo da API será integrado com o banco de dados, possibilitando o acesso aos dados de posições e a realização de cadastros de cercas

eletrônicas, contatos e o rastreio de comandos enviados. A mensageria terá um contrato estabelecido com a API para o encaminhamento de comandos e o recebimento das posições parcialmente processadas.

4.3 Camada de Banco de Dados

Esta camada irá ser composta de um único módulo, este sendo o sistema gerenciador de banco de dados, que poderá ser um banco MySQL ou PostgreSQL.

4.4 Camada de Mensageria

Também composta de um único módulo, este sendo o serviço de mensageria RabbitMQ, o qual será responsável por permitir a comunicação dos módulos da próxima camada e persistir os dados intermediários.

4.5 Camada de Microsserviços e APIs Internas

Será constituída de diversos módulos que serão compostos por microsserviços e APIs de uso Interno:

- **Módulo de Integração IOT**: Conterá os microsserviços que atuarão no início do fluxo de processamento, Esse módulo tem como responsabilidade o envio de comandos e o recebimento de dados “crus” dos equipamentos integrados, e deve fazer a persistência dos mesmos no banco de dados. Realizada a persistência um identificador único será obtido e ambas as informações serão unificadas em uma mensagem que será disparada para um canal da mensageria com um tópico identificando qual o tipo do equipamento;
- **Módulo de Pré-Processamento**: Assim como o módulo anterior esse conterá diversos microsserviços, e cada micro-serviço deste módulo será especializado em processar os dados de um equipamento específico que terá seu modelo identificado a partir do tópico designado pelo módulo de integração IOT. Os dados obtidos serão padronizados em um formato único do sistema, que irá conter informações básicas de telemetria, georreferenciamento e equipamento emissor. Após o processo de padronização os dados obtidos serão persistidos no banco de dados em conjunto do identificador de seu registro de origem, e consequentemente um novo identificador será gerado e unificado com os dados padronizados, que por fim serão passados a um outro canal da mensageria;
- **Módulo de Processamento**: Esse módulo será constituído por outros submódulos que terão suas próprias especializações no enriquecimento dos dados obtidos no módulo de pré-processamento. Através dos requisitos os seguintes submódulos foram levantados:

- **Submódulo de Endereços**: Este submódulo irá utilizar os dados de georreferenciamento obtidos no módulo de pré-processamento para consultar uma API interna (Nominatim API) que será a responsável por realizar o processo de geocodificação reversa. A obtenção dos dados postais de uma posição garante a legibilidade da localização de um veículo. Os dados obtidos serão persistidos em banco de dados em conjunto com o identificador dos dados pré-processados.
- **Submódulo de Cercas Eletrônicas**: Este submódulo terá como papel realizar a validação das entradas e saídas de um veículo em uma cerca eletrônica vinculada, utilizando dos dados de georreferenciamento obtidos no módulo de pré-processamento. A validação tem como objetivo gerar alertas para os contatos cabíveis. Quando os dados de georreferenciamento estiverem situados dentro de uma cerca, uma associação será criada, isso irá possibilitar detectar uma alteração no status de um veículo referente a determinada cerca, gerando um outro registro indicando a alteração, e assim registros serão persistidos em banco de dados, a alteração de status irá disparar em um canal uma mensagem para cada contato. Essa mensagem carregará consigo os dados do alerta (dados do pré-processamento, veículo, cerca e por fim destinatário/contato);
- **Submódulo Processador de Alertas**: Este submódulo irá desempenhar a formatação dos dados de um alerta em uma mensagem de texto que será encaminhada para um outro canal da mensageria contendo um tópico indicando qual o tipo de serviço deverá realizar o processamento daquela mensagem.
- **Submódulo de E-mail**: Este submódulo irá realizar o envio de e-mails através de uma integração com um servidor SMTP interno, persistindo um log em banco de dados sempre que o envio de um e-mail for realizado.
- **Submódulo de Whatsapp**: Este submódulo irá realizar o envio de mensagens para Whatsapp através de uma integração com uma API interna (WAHA API), persistindo um log em banco de dados sempre que o envio de uma mensagem for realizado.

5 Mecanismos Arquiteturais

Nesta fase do documento serão identificados os mecanismos do sistema em suas categorias de análise, design e implementação.

Quadro 3 – Mecanismos Arquiteturais

Mecanismos De Análise	Mecanismos De Design	Mecanismos De Implementação
Persistência	Banco de dados relacional e Mensageria	MySQL e RabbitMQ
Camada de distribuição	Adaptadores de Mensageria e Banco de dados	Pacotes oficiais
Integração com sistemas externos	Protocolo HTTP ou TCP	Acesso via API Rest ou Sockets
Front-end	Interface gráfica de interação do usuário	Implementação via Expo/React
Back-end	Microsserviços que executarão o processamento	Implementação via .NET

Fonte: Autoria própria.

6 Padrões e Estilos Arquiteturais

Como mencionado, o sistema proposto neste documento utilizará dos conceitos da arquitetura de microsserviços para guiar a elaboração e desenvolvimento de seus módulos, visando a distribuição das responsabilidades em módulos pequenos e com lógica que permita a escalabilidade horizontal da aplicação para comportar uma demanda variável com maior eficiência.

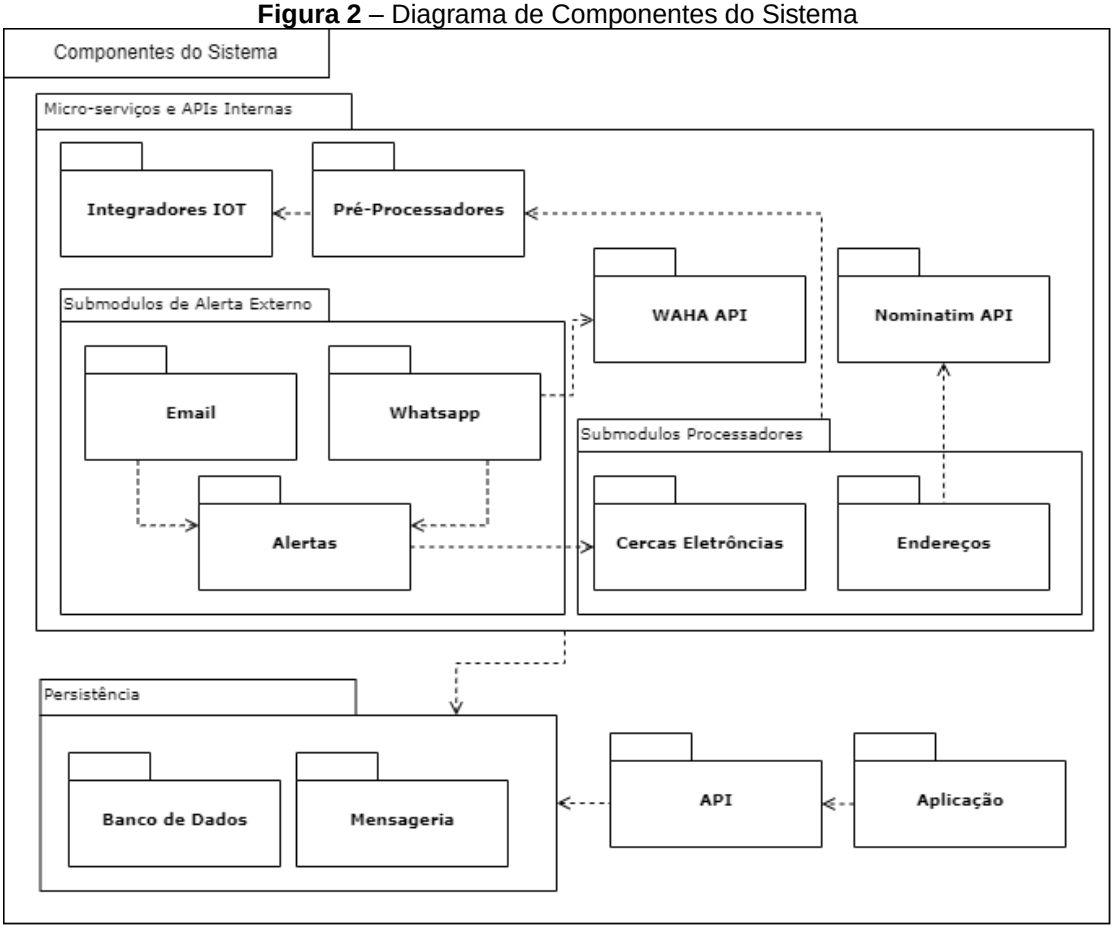
A arquitetura "*pipe and filter*" será aplicada através da mensageria, onde canais da mensageria serão os *pipes* que conectarão os módulos, os módulos por sua vez atuarão como *filters* pois irão receber um dado de um canal, o processar e se for o caso o transmitirão a um outro canal. Haverá a utilização de tópicos na mensageria, que atuarão como uma espécie de Pipe distribuidor conectado a vários *filters* e guiará o dado a seu respectivo fluxo de processamento.

A aplicação da arquitetura orientada a eventos também ocorrerá através da mensageria, na qual o recebimento de uma mensagem de um determinado módulo será responsável por desencadear todo um fluxo de processamentos.

7 Diagramas

7.1 Diagrama de Componentes

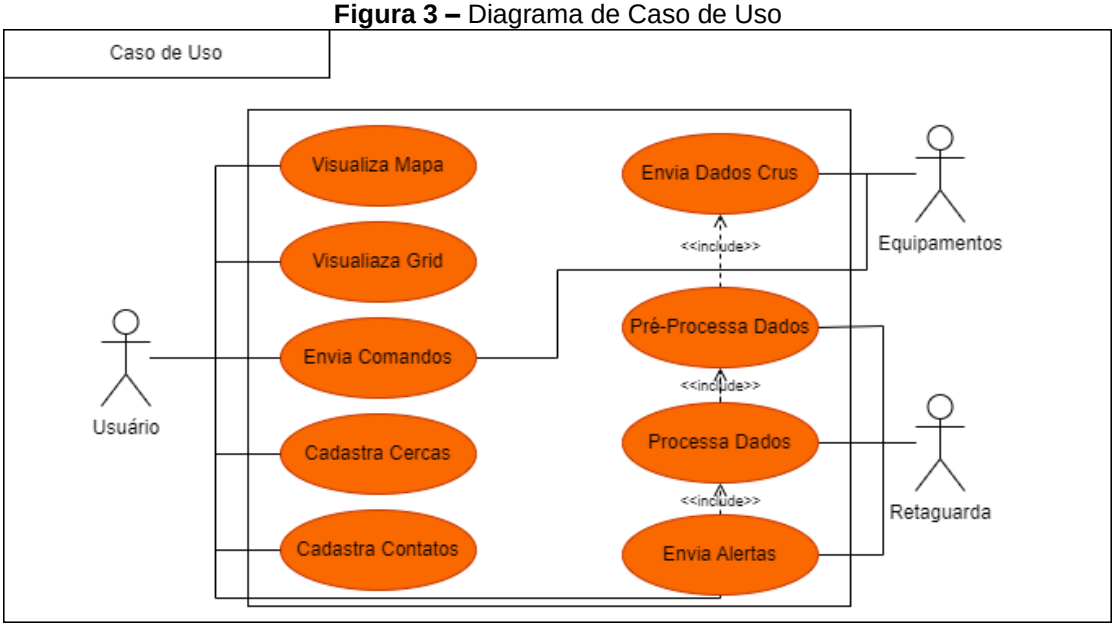
Figura 2 – Diagrama de Componentes do Sistema



Fonte: Autoria própria.

7.2 Diagrama de Caso de Uso

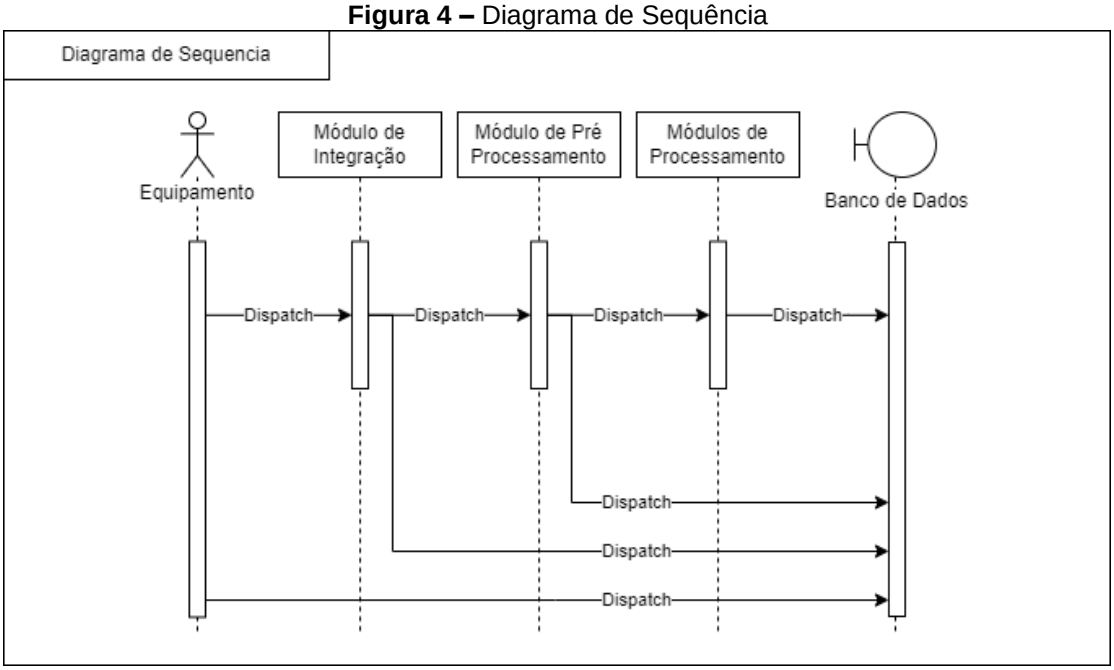
Figura 3 – Diagrama de Caso de Uso



Fonte: Autoria própria.

7.3 Diagrama de Sequência

Figura 4 – Diagrama de Sequência

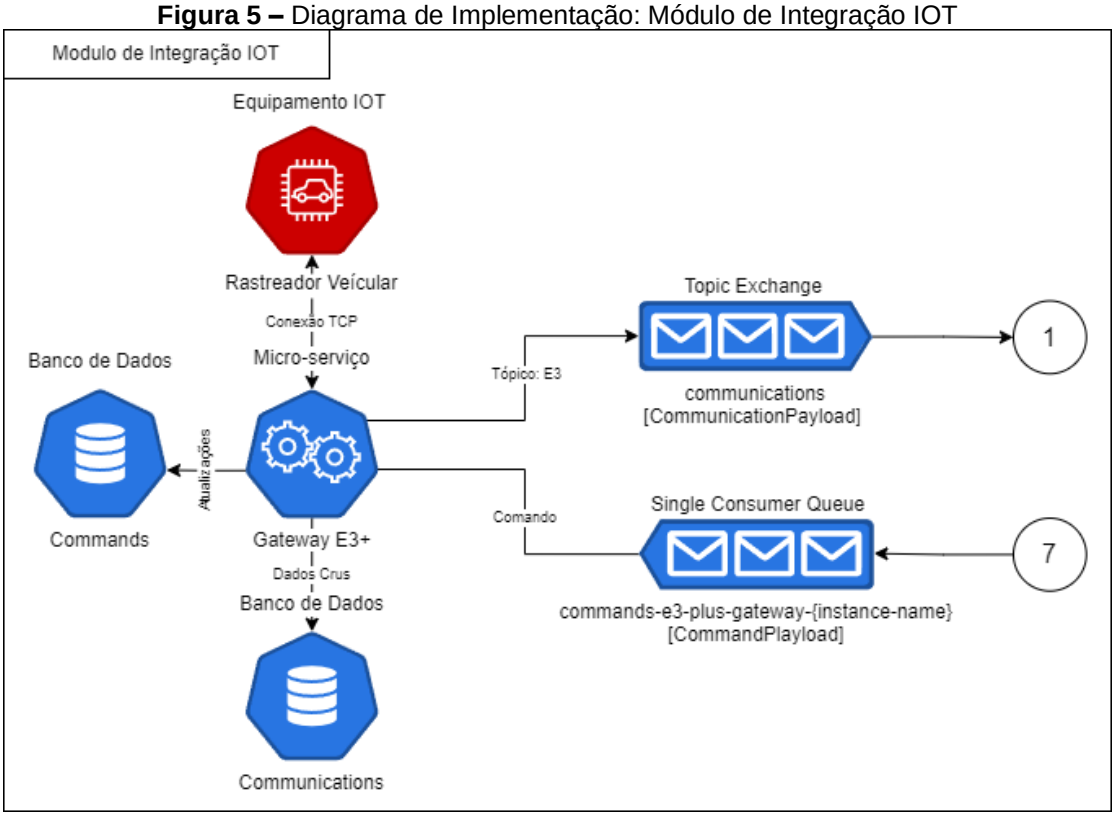


Fonte: Autoria própria.

7.4 Diagrama de Implementação

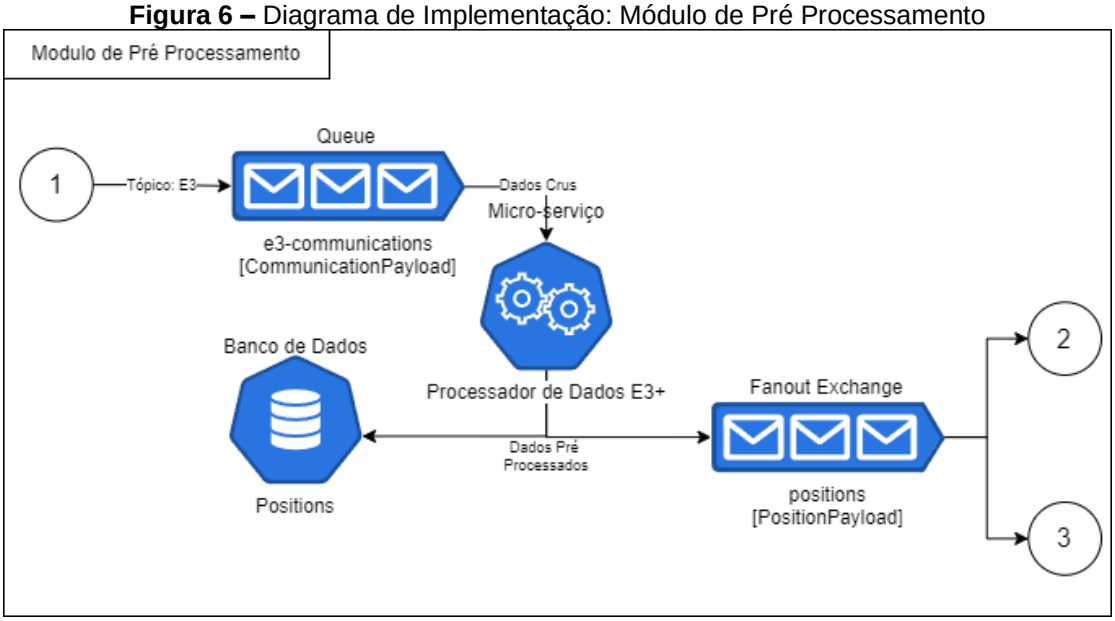
Esta etapa define a organização interna dos módulos do sistema.

Figura 5 – Diagrama de Implementação: Módulo de Integração IOT



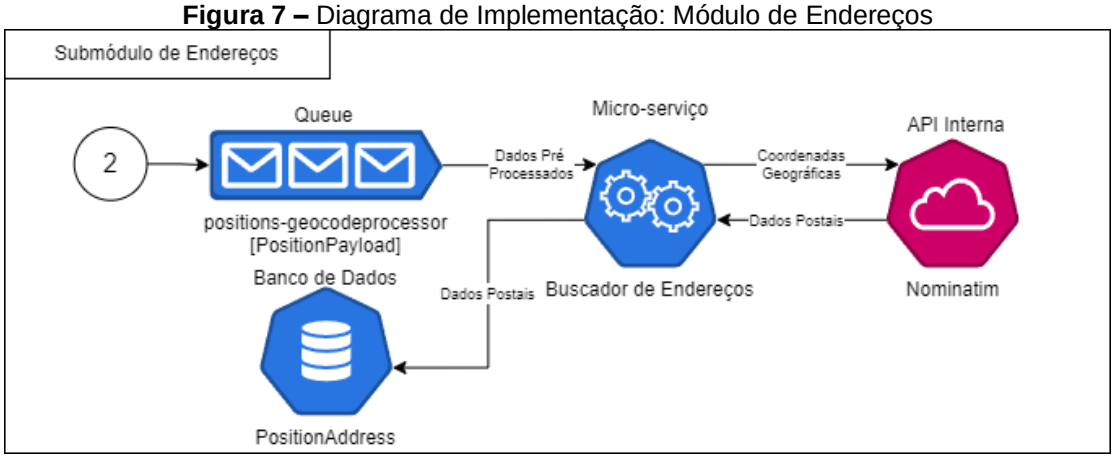
Fonte: Autoria própria.

Figura 6 – Diagrama de Implementação: Módulo de Pré Processamento



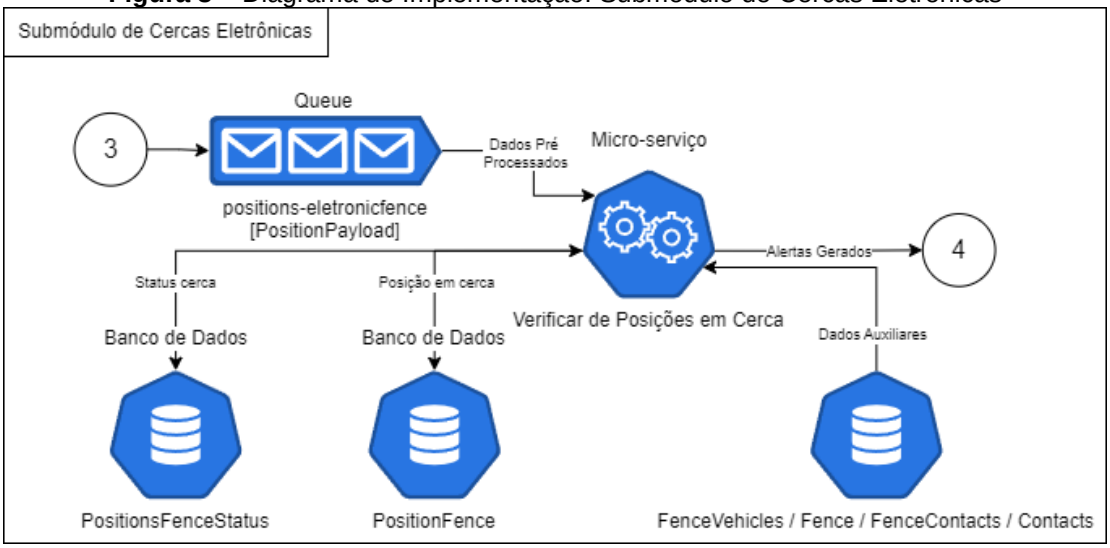
Fonte: Autoria própria.

Figura 7 – Diagrama de Implementação: Módulo de Endereços



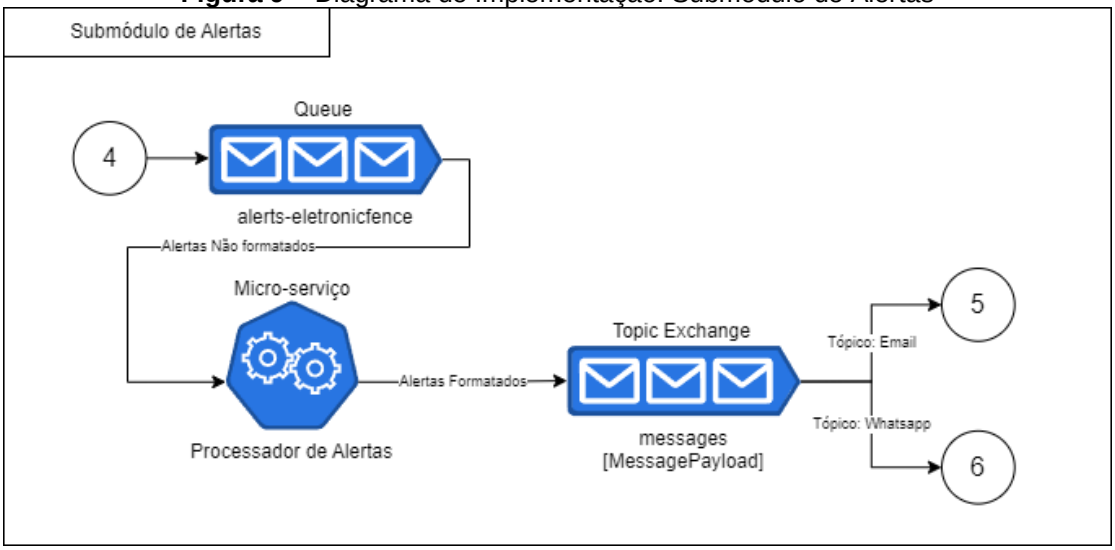
Fonte: Autoria própria.

Figura 8 – Diagrama de Implementação: Submódulo de Cercas Eletrônicas



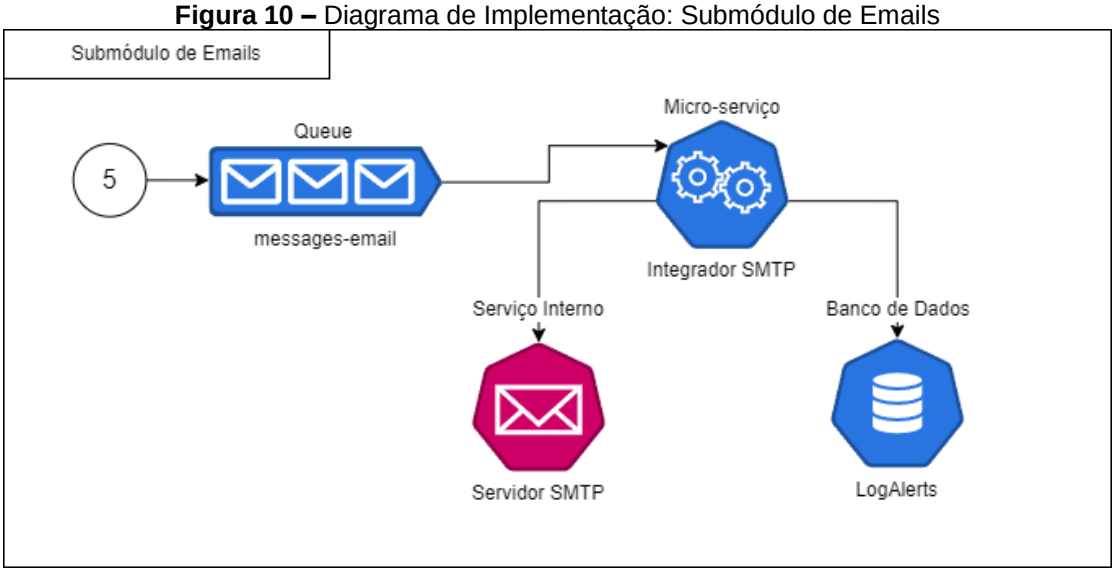
Fonte: Autoria própria.

Figura 9 – Diagrama de Implementação: Submódulo de Alertas



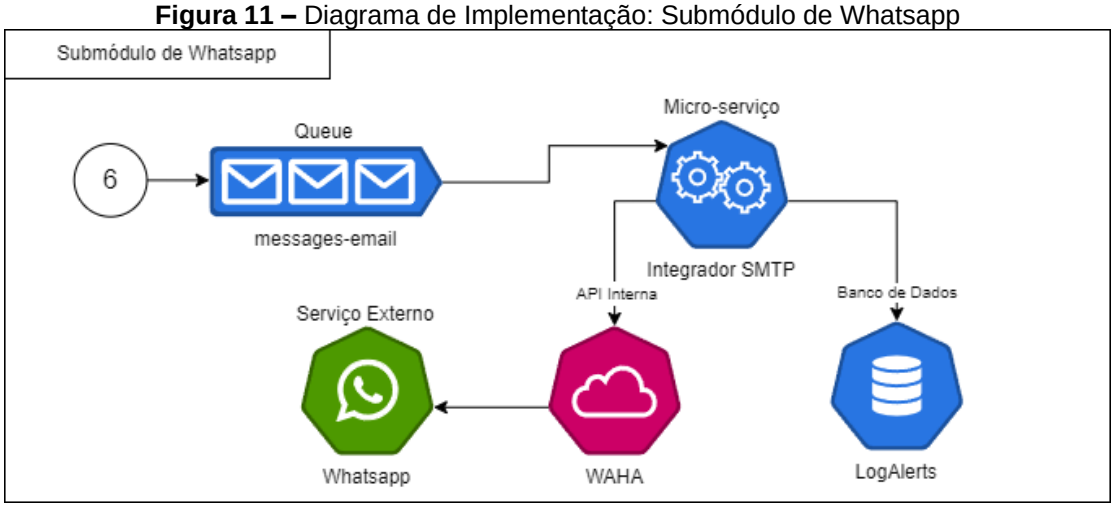
Fonte: Autoria própria.

Figura 10 – Diagrama de Implementação: Submódulo de Emails



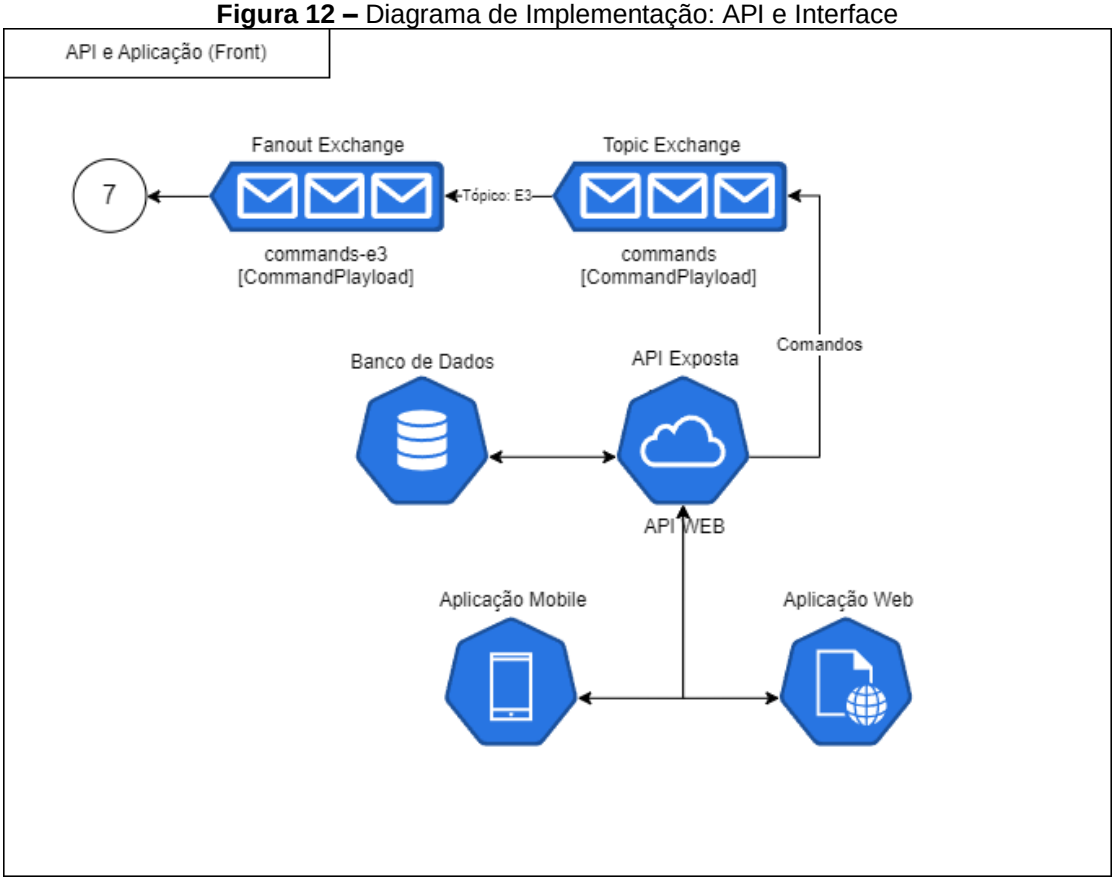
Fonte: Autoria própria.

Figura 11 – Diagrama de Implementação: Submódulo de Whatsapp



Fonte: Autoria própria.

Figura 12 – Diagrama de Implementação: API e Interface

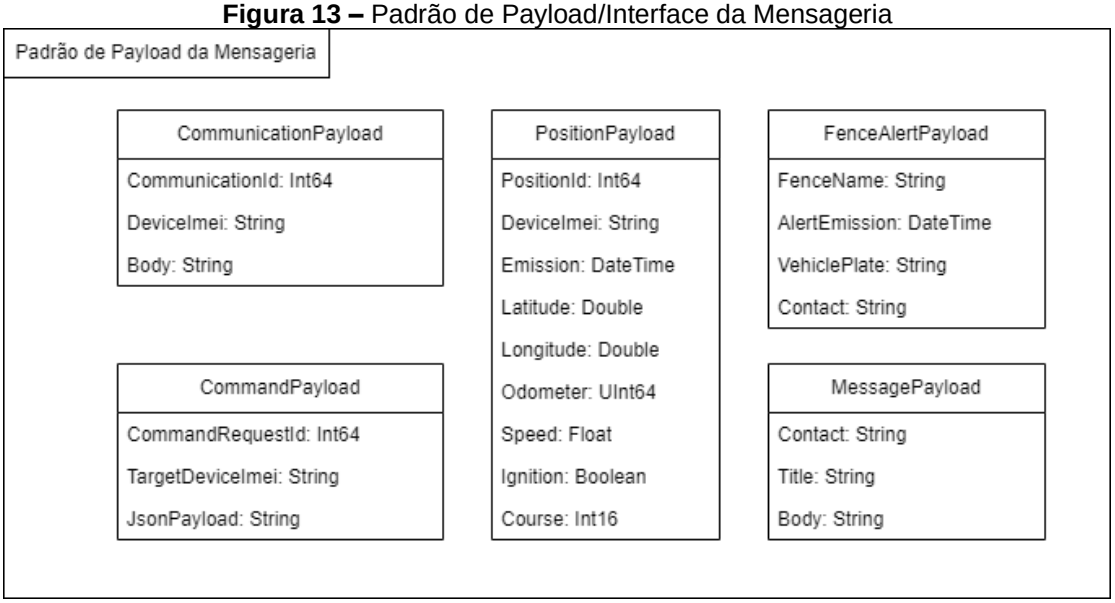


Fonte: Autoria própria.

8 Interface de Componentes

Esta etapa define o formato usado em cada elemento da mensageria.

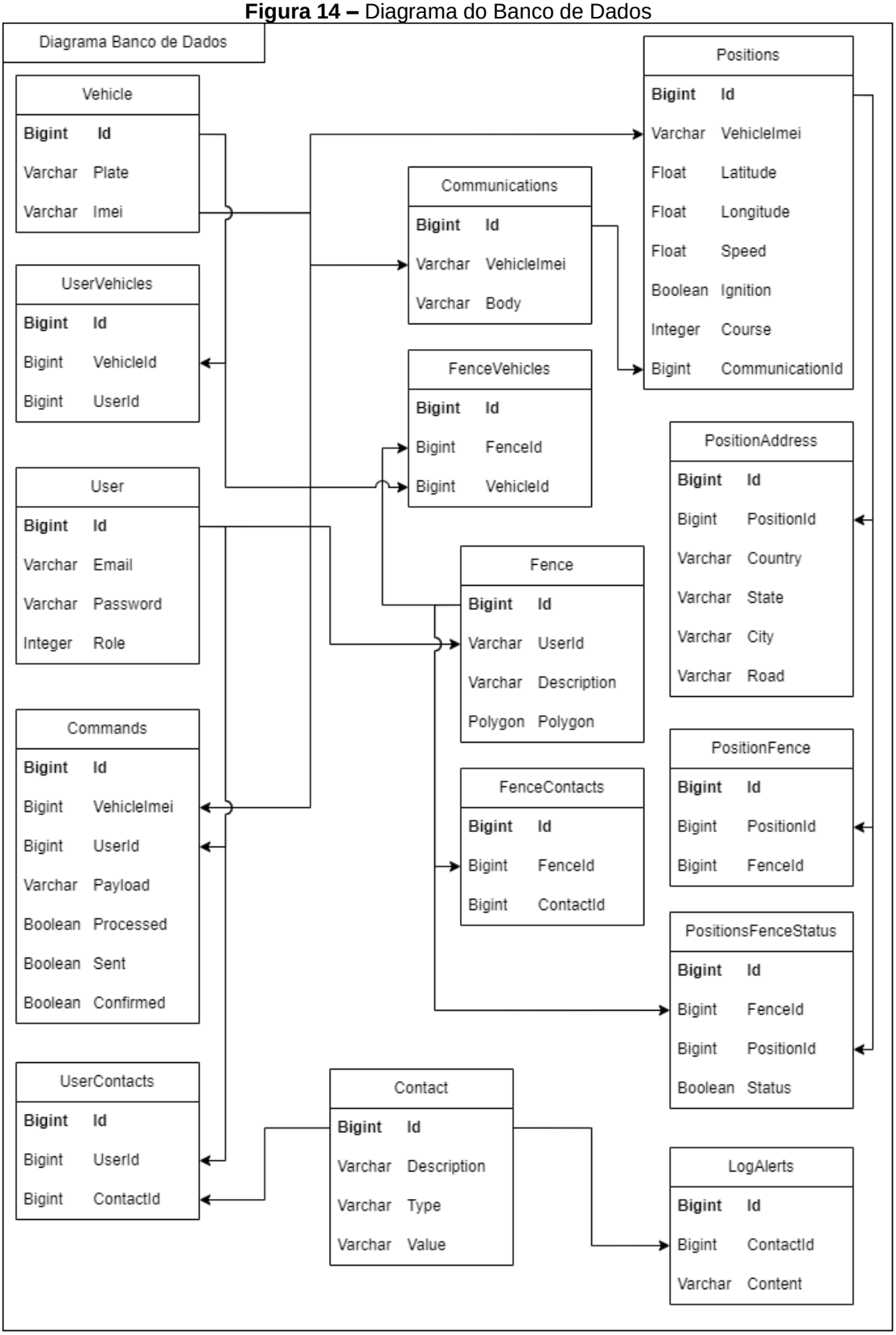
Figura 13 – Padrão de Payload/Interface da Mensageria



Fonte: Autoria própria.

9 Estrutura do Banco de Dados

Figura 14 – Diagrama do Banco de Dados



Fonte: Autoria própria.

10 Riscos e Mitigações

10.1 Falhas nas aplicações

Na ocasião em que exceções sejam detectadas dentro das aplicações um log deve ser gerado e lançado, o dado deverá ser rejeitado na mensageria, a qual deverá por sua vez possuir uma política de 3 tentativas antes do encaminhamento do dado a uma DQL (DeadLetter Queue).

10.2 Falhas em módulos de integração IOT

Caso ocorra alguma falha em persistir o dado em quaisquer ambiente, o dado deverá ser persistido em disco para processamento posterior quando a situação estiver normalizada.