UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PADRÕES DE ARQUITETURA DE SOFTWARE

Documento de Arquitetura

Sicoin - Sistema de Otimização de Coleta de Resíduos em Cidades Inteligentes

João Gabriel Tavares Felix Monteiro
João Victor Rosa Couto e Silva
Karlla Loane Santos Lima
Murilo Henrique de Sousa Freua

Goiânia
2024

Histórico de Revisão

Data	Versão	Descrição	Autor(es)
30 de outubro de 2024	1.0	Criação do documento	Murilo
31 de outubro de 2024	1.0	Versão inicial	Murilo, Karlla, João Gabriel, João Victor
19 de novembro de 2024	1.1	Identificação dos aspectos	Karlla
23 de novembro de 2024	1.2	Ajustes baseados em feedback dos professores	João Victor
03 de dezembro de 2024	1.3	Atualização dos atributos de qualidade do projeto	Murilo
07 de dezembro de 2024	1.4	Refinamento de requisitos e cenários	Murilo, Karlla, João Gabriel, João Victor
10 de dezembro de 2024	1.4	Alteração de arquitetura orientada a serviços para arquitetura orientada a eventos	João Gabriel
12 de dezembro de 2024		Identificação dos concerns	João Victor
13 de dezembro de 2024	1.5	Atualização de referências e alinhamento com a ISO/IEC/IEEE 42010:2022	Karlla
14 de dezembro de 2024	1.6	Revisão e padronização do glossário	Murilo
17 de dezembro de 2024	1.7	Inclusão dos diagramas	João Gabriel
18 de dezembro de 2024	1.8	Revisão do documento final	Karlla
19 de dezembro de 2024	1.9	Finalização da execução do projeto	Murilo, Karlla, João Gabriel, João Victor

Metadados e Controle

- **Data de Início do Projeto:** 16/09/2024
- **Data de emissão do documento:** 20/12/2024
- **Status do Projeto:** finalizado
- **Autores e revisores:** João Gabriel, João Victor, Karlla, Murilo
- **Responsável pela Aprovação:** Jacson Rodrigues Barbosa e Fábio Moreira Costa.
- **Organização:** Universidade Federal de Goiás, Bacharelado em Engenharia de Software, disciplinas de Padrões de Arquitetura de Software e Software para Sistemas Ubíquos.
- **Controle de Versões:** O controle de versões é realizado por meio da plataforma GitHub, com a organização dos repositórios dividida da seguinte forma:
  - **Repositório backend**: sicoin-backend
  - **Repositório do frontend:** sicoin-frontend
  - **Repositório para aplicativo mobile:** scu-coletaplus
  - **Repositório para versionamento do documento de arquitetura e diagramas:** PadraoArqui-SCU

Sumário

1 Introdução

Este documento de arquitetura tem por objetivo descrever de forma clara e detalhada a estrutura do sistema de coleta de resíduos em cidades inteligentes, abordando seus componentes, funcionalidades e interações. Ele apresenta as decisões arquiteturais tomadas durante o desenvolvimento do sistema, alinhadas às necessidades dos stakeholders e aos objetivos do projeto, como eficiência operacional, sustentabilidade e adaptabilidade. Além disso, o documento visa garantir a rastreabilidade e a compreensão técnica do sistema, facilitando sua manutenção, evolução e integração com outros sistemas urbanos inteligentes.

1.1 Contexto do Sistema

O sistema de coleta de resíduos em cidades inteligentes foi desenvolvido para otimizar o processo de coleta com base no volume de resíduos e na geolocalização dos agentes de limpeza. A solução utiliza sensores instalados em lixeiras para monitorar, em tempo real, o nível de preenchimento de resíduos, enviando essas informações para uma camada de processamento centralizada. Além disso, os dispositivos móveis dos agentes de coleta transmitem continuamente sua localização ao sistema, permitindo o cálculo de rotas dinâmicas e otimizadas.

Com base nesses dados, o sistema sugere trajetos que evitam deslocamentos desnecessários, promovendo economia de recursos e redução de emissões de carbono. Integrado ao ecossistema de uma cidade inteligente, o sistema se adapta às condições dinâmicas do ambiente urbano, oferecendo uma abordagem flexível e orientada por dados. Os agentes de coleta utilizam dispositivos móveis para acessar as rotas atualizadas em tempo real, enquanto gestores monitoram as operações e analisam tendências por meio de uma interface web.

Por fim, é importante ressaltar que este sistema é focado na coleta eficiente de resíduos, não contemplando diretamente a gestão da destinação final dos resíduos.

1.2 Escopo do sistema

- Monitoramento em tempo real do volume das lixeiras.
- Geração automática e dinâmica de rotas otimizadas para coleta.
- Ajuste em tempo real das rotas de acordo com mudanças contextuais.
- Geração de relatórios de eficiência com dados de distância e tempo.
- Visualização de tendências por meio de dashboards gerenciais.

1.3 Alinhamento com ODS

No contexto de uma cidade inteligente, esse sistema se integra ao ecossistema urbano ao possibilitar a coleta em tempo real de dados de resíduos, alinhando-se com os Objetivos de Desenvolvimento Sustentável (ODS):

- ODS 11 - Cidades e Comunidades Sustentáveis: contribui para o desenvolvimento de cidades mais organizadas e eficientes, reduzindo desperdícios e melhorando a limpeza urbana.
- ODS 13 - Ação Contra a Mudança Global do Clima: contribui na redução das emissões de carbono ao otimizar rotas e minimizar deslocamentos desnecessários.

1.4 Glossário

API (Application Programming Interface): conjunto de definições e protocolos que permite a interação entre diferentes sistemas ou aplicações, facilitando a comunicação e o uso de funcionalidades de maneira padronizada.
API REST (Representational State Transfer): estilo arquitetural de APIs que utiliza padrões do protocolo HTTP, permitindo comunicação eficiente entre cliente e servidor.
HTTP (HyperText Transfer Protocol): protocolo de comunicação usado para transferir informações na web, como páginas, dados e APIs.
IoT (Internet of Things): rede de objetos físicos incorporados a sensores, software e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos e sistemas pela internet.
MQTT (Message Queuing Telemetry Transport): protocolo leve de comunicação para IoT, projetado para dispositivos com recursos limitados ou redes instáveis, garantindo a troca eficiente de mensagens.
MQTT com QoS 1 (Quality of Service): nível de garantia do MQTT onde cada mensagem será entregue pelo menos uma vez, com possibilidade de duplicação. Ideal para garantir confiabilidade em sistemas de IoT.
Stakeholders: pessoas, grupos ou entidades que têm interesse direto ou indireto em um projeto, como usuários, gestores, desenvolvedores, reguladores e a sociedade em geral.
Sensores IoT: dispositivos conectados que coletam informações do ambiente, como temperatura, pressão ou nível de resíduos, e transmitem os dados para sistemas centralizados.
WebSocket: protocolo de comunicação que permite conexão bidirecional contínua entre cliente e servidor, usado para notificações em tempo real.
Flutter: framework para desenvolvimento de aplicativos móveis, compatível com Android e iOS, permitindo a criação de interfaces de usuário responsivas.

SQLite: banco de dados relacional leve e embutido, ideal para aplicativos móveis que precisam armazenar dados localmente.
AWS (Amazon Web Services): plataforma de serviços de computação em nuvem que oferece infraestrutura escalável, como servidores, bancos de dados e armazenamento.

2 História de Usuários e Cenários

HU1 - Geração automática de rotas otimizadas

Como agente de coleta,
Quero que o sistema gere automaticamente uma rota com base na minha localização e no volume das lixeiras,
Para garantir uma coleta eficiente e sem desperdício de recursos.

Cenário 1.1: Geração de rota no início do turno

Dado que iniciei meu turno através do aplicativo,
E o sistema detectou lixeiras com necessidade de coleta,
Quando o sistema processar as informações,
Então uma rota otimizada é enviada automaticamente para o meu dispositivo, priorizando as lixeiras mais cheias.

Cenário 1.2: Inclusão de lixeira cheia durante a rota

Dado que estou executando uma rota ativa,
E uma nova lixeira atinge a 70% de sua capacidade,
Quando o sistema receber essa alteração,
Então ele recalculará a rota e a enviará para o meu dispositivo, com a lixeira incluída no novo trajeto.

Cenário 1.3 (Negativo): Falha no envio da rota

Dado que iniciei meu turno,
E o sistema não conseguiu processar as informações devido à falta de conexão,
Quando eu tentar atualizar a rota,
Então o sistema exibirá uma mensagem de erro e sugerirá uma rota alternativa baseada nos últimos dados disponíveis.

HU2 - Notificação de mudanças na rota em tempo real

Como agente de coleta,
Quero ser notificado automaticamente sobre mudanças na rota,
Para garantir que eu atenda às novas demandas de coleta de lixo.

Cenário 2.1: Lixeira cheia identificada durante a coleta

Dado que estou seguindo a rota ativa,
E uma lixeira próxima atinge 70% de sua capacidade,
Quando o sistema detectar a situação e recalcular a rota,
Então eu recebo uma notificação com a rota atualizada.

Cenário 2.2 (Negativo): Notificação atrasada ou não recebida

Dado que estou executando uma rota ativa,
E uma lixeira atinge 80% de sua capacidade,
Quando o sistema recalcula a rota mas não consegue enviar a notificação,
Então eu continuo com a rota anterior e o sistema registra a falha para supervisão.

HU3 - Selecionar o caminhão que realizará a coleta

Como agente de coleta,
Quero selecionar o caminhão que será utilizado para realizar a coleta,
Para que o sistema registre e associe o veículo às rotas designadas.

Cenário 3.1: Seleção bem-sucedida do caminhão

Dado que iniciei meu turno através do aplicativo,
E o sistema exibe a lista de caminhões disponíveis,
Quando eu seleciono o caminhão que será utilizado,
Então o sistema registra a associação do caminhão com as rotas designadas para o turno.

Cenário 3.2 (Negativo): Falha na seleção do caminhão

Dado que iniciei meu turno pelo aplicativo,
E o sistema não consegue acessar os dados de disponibilidade dos caminhões devido a um erro de conexão,
Quando eu tento selecionar o caminhão,

Então o sistema exibe uma mensagem informando a falha e sugere tentar novamente ou contatar o suporte.

HU4 - Iniciar coleta no aplicativo

Como agente de coleta,
Quero iniciar meu turno pelo aplicativo,
Para que o sistema registre minha localização e gere a rota de coleta automaticamente.

Cenário 4.1: Início de turno pelo aplicativo

Dado que estou com o aplicativo aberto,
E o sistema já identificou que existem lixeiras a serem coletadas,
Quando eu inicio a coleta no aplicativo,
Então o sistema registra minha localização inicial e gera uma rota otimizada para coleta.

Cenário 4.2 (Negativo): Falha no registro de localização

Dado que estou tentando iniciar meu turno pelo aplicativo,
E o sistema não consegue acessar minha localização,
Quando eu tento registrar a coleta,
Então ele exibe uma mensagem de erro e solicita que eu ative o GPS para prosseguir.

HU5 - Gerar relatórios de eficiência da coleta

Como supervisor,
Quero ter acesso a relatórios de coleta de lixo,
Para avaliar o desempenho ecológico e operacional das rotas sugeridas.

Cenário 5.1: Geração de relatório de coleta

Dado que estou acessando o portal do sistema,
Quando eu solicitar o relatório no sistema,
Então ele apresentará dados como distância percorrida, tempo gasto e a emissão de carbono durante a coleta.

Cenário 5.2 (Negativo): Falha na geração de relatório

Dado que estou solicitando um relatório,
E o sistema não consegue processar os dados devido a um erro no servidor,
Quando eu tento gerar o relatório,
Então o sistema exibe uma mensagem informando a falha e sugere uma nova tentativa em poucos minutos.

HU6 - Visualizar tendências de geração de resíduos

Como supervisor,
Quero visualizar gráficos de tendência sobre a geração de resíduos por região,
Para identificar áreas com maior frequência de coleta e sazonalidades.

Cenário 5.1: Gráfico de tendência por região

Dado que o sistema possui dados históricos das coletas,
E cada lixeira está associada a uma localização específica,
Quando eu acesso o dashboard de tendências,
Então eu visualizo um gráfico com regiões que geram mais resíduos e o sistema sugere ajustes na frequência de coleta.

Cenário 5.2 (Negativo): Dados insuficientes para análise

Dado que o sistema possui poucos dados históricos de coleta,
Quando eu acesso o dashboard de tendências,
Então o sistema exibe um aviso informando que os dados disponíveis podem não refletir tendências confiáveis.

HU7 - Visualizar mapa com o estado atual das lixeiras

Como gestor,
Quero visualizar um mapa da região com o estado atual das lixeiras,
Para monitorar a capacidade de cada lixeira e planejar coletas futuras.

Cenário 7.1: Visualização de mapa atualizado

Dado que estou acessando o portal do sistema,
E o sistema possui dados atualizados dos sensores,
Quando eu seleciono a opção de visualizar o mapa,

Então ele exibe um mapa da região com o estado atual de cada lixeira, indicando aquelas próximas à capacidade máxima.

Cenário 7.2 (Negativo): Dados insuficientes para análise

Dado que estou tentando acessar o mapa,
E o sistema não consegue carregar os dados devido a uma falha de conexão,
Quando eu solicito a visualização,
Então o sistema exibe uma mensagem de erro informando o problema e sugere uma nova tentativa mais tarde.

HU8 - Visualizar histórico de coleta de uma lixeira

Como gestor,
Quero visualizar o histórico de coleta de uma lixeira específica,
Para analisar padrões de uso e identificar necessidades de ajustes na frequência de coleta.

Cenário 8.1: Consulta de histórico de coleta

Dado que estou acessando o portal do sistema,
E selecionei uma lixeira específica,
Quando eu solicito o histórico de coleta,
Então o sistema exibe os registros de coleta daquela lixeira, incluindo datas, horários e volumes coletados.

Cenário 8.2 (Negativo): Dados insuficientes para análise

Dado que estou tentando consultar o histórico de uma lixeira,
E o sistema não possui dados registrados para aquela lixeira,
Quando eu solicito o histórico,
Então o sistema exibe uma mensagem informando que não há dados disponíveis para análise.

3 Stakeholders e Perspectivas

3.1 Identificação dos Stakeholders

Os stakeholders fundamentais ao sistema foram identificados com base nas suas preocupações e na relevância para a arquitetura do projeto:

- Agentes de coleta: necessitam de rotas ajustadas automaticamente para que a coleta seja otimizada em tempo real e requerem uma interface móvel intuitiva.
- Gestores públicos: desejam uma visão consolidada e histórica das coletas para análise de desempenho e tendências. Esperam que o sistema contribua com métricas de economia e indicadores de sustentabilidade.
- Cidadãos: esperam que a coleta seja realizada de forma eficaz para manter a limpeza da cidade.
- Desenvolvedores do sistema: responsáveis pela implementação e manutenção técnica.
- Arquiteto de software: responsável por garantir a integridade técnica, desempenho e escalabilidade do sistema.

3.2 Identificação das Perspectivas e Preocupações (*concerns*)

As perspectivas representam as diferentes visões que os stakeholders possuem sobre o sistema, refletindo suas necessidades, expectativas e a forma como interagem ou impactam a arquitetura. Já as preocupações (concerns) são os aspectos específicos considerados relevantes para a arquitetura, como desempenho, eficiência, escalabilidade, operabilidade e sustentabilidade. De acordo com a norma ISO/IEC/IEEE 42010:2022, cada concern deve ser associado ao stakeholder que o detém, de forma a garantir que a arquitetura atenda às suas expectativas e requisitos.

Na tabela 1, identificamos as seguintes perspectivas e preocupações associadas aos principais stakeholders do sistema.

Stakeholder	**Perspectiva**	**Preocupação**
Agentes de coleta	Operacional	- As rotas são atualizadas em tempo real para evitar atrasos e ineficiência? - A interface móvel reflete a visão atualizada do ambiente? - A interface móvel é intuitiva e fácil de usar durante a operação?
Gestores públicos	Estratégica e organizacional	- O sistema fornece métricas claras de eficiência e sustentabilidade? - As informações históricas permitem análise de desempenho e planejamento futuro?
Cidadãos	Social	- A coleta de resíduos está sendo realizada de forma eficaz para manter a cidade limpa? - O sistema contribui para o bem-estar da comunidade ao minimizar acúmulo de lixo?
Desenvolvedores	Tecnológica	- A arquitetura do sistema facilita a manutenção, escalabilidade e evolução futura? - Os componentes são fáceis de implementar e testar?
Arquiteto de software	Tecnológica	- A arquitetura é tecnicamente viável e sustentável a longo prazo? - A solução atende aos requisitos de desempenho, escalabilidade e integração entre os componentes?

Tabela 1: *Perspectivas e preocupações dos stakeholders*

4 Identificação dos Aspectos

Conforme a norma ISO/IEC/IEEE 42010:2022, os aspectos representam diferentes dimensões ou características relevantes para a arquitetura do sistema, tais como os aspectos estruturais, comportamentais, funcionais e programáticos. Cada aspecto está diretamente associado às preocupações (concerns) dos stakeholders, servindo para garantir que a arquitetura atenda adequadamente os requisitos identificados.

Na tabela 2, apresentamos os aspectos considerados relevantes para a arquitetura do sistema e suas respectivas associações com as preocupações dos stakeholders.

Aspecto	Descrição	Concern Associado
Estrutural	Organização dos componentes do sistema e suas interações.	Viabilidade técnica, integração entre componentes.
Comportamental	Comportamento dinâmico do sistema em resposta às mudanças de contexto.	Atualização em tempo real de rotas e ajustes automáticos.
Funcional	Funcionalidades principais oferecidas pelo sistema.	Geração de rotas otimizadas, dashboards de eficiência.
Programático	Questões relacionadas à implementação, desempenho e escalabilidade.	Desempenho do sistema, escalabilidade e manutenibilidade.
Sustentabilidade	Impacto ambiental e eficiência no uso de recursos públicos.	Redução de deslocamentos, otimização de rotas e economia de recursos.

Tabela 2: *Descrição dos aspectos*

5 Atributos de Qualidade

Os atributos de qualidade garantem que o sistema atenda aos requisitos identificados, considerando as preocupações dos stakeholders. Para o desenvolvimento deste projeto, incluímos consideradores referentes aos critérios de computação ubíqua necessários para sistemas distribuídos, adaptáveis e em tempo real.

5.1 Desempenho

- Descrição: garantia de resposta rápida e processamento eficiente dos dados do sistema.
- Métricas/Justificativas:
  - Tempo de resposta: Inferior a 2 segundos para geração de rotas.
  - Latência: máxima de 200 milissegundos no processamento e transmissão dos dados dos sensores IoT

5.2 Disponibilidade

- Descrição: garantia de funcionamento contínuo e confiável dos serviços críticos do sistema.
- Métricas/Justificativas:
  - Uptime: 99,9% para monitoramento e geração de rotas.
  - Replicação: balanceamento de carga em servidores.

5.3 Segurança

- Descrição: proteção dos dados durante a transmissão e armazenamento.
- Métricas/Justificativas:
  - Autenticação: uso de JWT (Json Web Token).
  - Criptografia: end-to-end para proteção de dados sensíveis.
  - Autorização: permissões de acesso baseadas em papéis de usuário.

5.4 Escalabilidade

- Descrição: capacidade de expandir os serviços e componentes do sistema conforme aumento da demanda.
- Métricas/Justificativas:
  - Horizontal: expansão de serviços de processamento.
  - Vertical: aumento de recursos nos bancos de dados.

5.5 Interoperabilidade

- Descrição: capacidade de integração entre sensores IoT, aplicativos móveis e servidores de processamento.
- Métricas/Justificativas:
  - Protocolos: uso de MQTT (IoT) e HTTP/REST (APIs).
  - Plataformas: compatibilidade com Android/iOS.

5.6 Integridade

- Descrição: garantia de que os dados capturados e transmitidos são precisos e completos.
- Métricas/Justificativas:
  - Validação: checksums e validação de integridade dos dados.
  - Entrega confiável: uso do MQTT com QoS 1 (at least once) para garantir a entrega das mensagens

6 Viewpoints da Arquitetura do Sistema

Conforme a norma ISO/IEC/IEEE 42010:2022, os viewpoints da arquitetura representam diferentes perspectivas utilizadas para descrever o sistema. Cada viewpoint define o foco, as preocupações abordadas e as regras de representação, servindo como base para a criação das visões (views). Esses pontos de vista garantem que a arquitetura atenda às necessidades dos stakeholders e cubra os aspectos essenciais do sistema, como lógica, dados, implantação e componentes.

6.1 Ponto de vista arquitetural geral

- Propósito: fornecer uma visão global do sistema, destacando os componentes principais e as interações entre eles.
- Preocupações abordadas: estrutura geral do sistema e atendimento aos atributos de qualidade.
- Visão associada: Diagrama de alto nível da arquitetura.

6.2 Ponto de vista lógico

- Propósito: representar a estrutura lógica do sistema, com foco nos componentes de software, suas responsabilidades e interações.
- Preocupações abordadas: modularidade, organização do código e relações entre os componentes.
- Visão associada: diagrama de classes do sistema.

6.3 Ponto de vista de comportamental

- Propósito: representar o comportamento dinâmico do sistema, destacando o fluxo de interações entre os componentes para realizar determinadas funcionalidades.
- Preocupações abordadas: sequência de operações, troca de mensagens entre componentes e lógica de execução das funcionalidades.
- Visão associada: diagrama de Sequência detalhando o fluxo de mensagens entre sensores, aplicativos e servidores durante a geração de rotas ou outra funcionalidade central.

6.4 Ponto de vista de físico

- Propósito: representar a distribuição física dos componentes do sistema, com foco na infraestrutura e implantação.
- Preocupações abordadas: alocação de componentes em servidores, dispositivos IoT e aplicativos móveis.
- Visão associada: diagrama de Implantação com sensores, servidores e dispositivos móveis.

7 Visão da Arquitetura

A visão da arquitetura apresenta as diferentes representações do sistema derivadas dos pontos de vista definidos. Essas visões descrevem os aspectos essenciais da arquitetura, abordando as preocupações dos stakeholders e os atributos de qualidade identificados.

7.1 Visão da Arquitetura Geral

Esta visão, derivada do ponto de vista arquitetural geral, fornece uma visão global do sistema. O diagrama associado (figura 1)apresenta as principais camadas e componentes do sistema, destacando suas interações. O diagrama de alto nível ilustra os principais módulos, como sensores IoT, processamento em nuvem, aplicativos móveis e cliente web, além dos protocolos utilizados para comunicação.

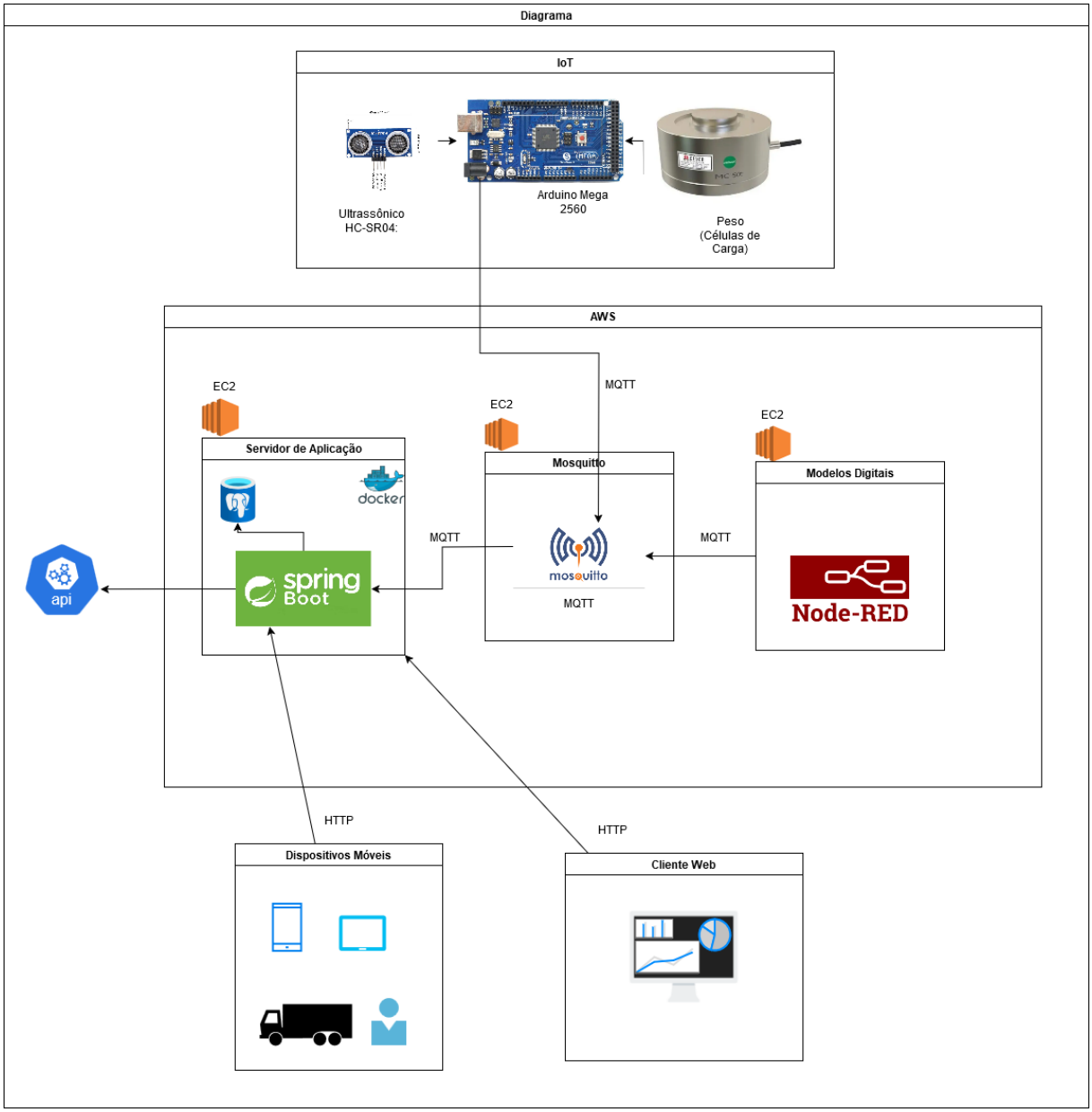

Figura 1: *Visão geral da arquitetura e implantação do sistema*

A arquitetura do sistema foi projetada para integrar dispositivos IoT, processamento em nuvem e interfaces de usuário móveis e web, garantindo escalabilidade, reatividade e conectividade em tempo real. Ela é composta por três principais camadas: IoT, servidor de aplicação e interfaces de usuário, todas interligadas por meio do protocolo MQTT e hospedadas em serviços na AWS.

7.1.1 Descrição dos componentes

Dispositivos IoT

Esta camada compreende os sensores instalados nas lixeiras e os dispositivos móveis dos agentes de coleta. Os sensores monitoram continuamente o nível de resíduos e enviam essas

informações em tempo real, enquanto os dispositivos móveis enviam dados de da geolocalização para o sistema.

Componentes:

- Sensores nas lixeiras
- Dispositivos móveis dos agentes

Servidor de Aplicação

O servidor de aplicação é o núcleo do sistema e está hospedado na AWS. Ele desempenha as seguintes funções principais:

- Processamento de dados: recebe dados das lixeiras IoT e os processa em tempo real, armazenando as informações em um banco de dados relacional PostgreSQL.
- Gêmeo digital: mantém o estado atualizado das lixeiras como uma representação digital que reflete o estado físico atual.
- Integração com Node-RED: O servidor interage com o Node-RED via MQTT, que simula os sensores IoT e os eventos de coleta.
- Geração de rotas: Comunica-se com a API do Google Maps para calcular e otimizar rotas de coleta, considerando o estado das lixeiras e a localização dos agentes.

Broker MQTT

O broker MQTT é o componente central que gerencia a comunicação entre os dispositivos IoT, o servidor de aplicação e o Node-RED. Ele funciona como um intermediário, recebendo mensagens de publicadores (os dispositivos IoT simulados no Node-RED) e as encaminhando para os assinantes (o servidor de aplicação e outros módulos).

O broker MQTT é hospedado na infraestrutura AWS para garantir alta disponibilidade e baixa latência na transmissão de mensagens.

Camada de Aplicação

Nessa camada, estão os aplicativos e interfaces de usuário que permitem o acesso e a interação com o sistema. O aplicativo móvel, usado pelos agentes, exibe rotas em tempo real e notifica os agentes sobre ajustes dinâmicos. O portal web permite que supervisores monitorem a operação, acompanhem as rotas, analisem tendências e visualizem relatórios de desempenho.

Componentes:

- Dispositivos móveis (smartphones/tablets) - aplicativo móvel.
- Desktop/Laptop - portal web.

Node-RED

O Node-RED é uma ferramenta visual utilizada para simular os dispositivos IoT no sistema. Ele simula o comportamento dos sensores físicos e permite testar a lógica do sistema antes da integração com dispositivos reais.

O Node-RED gera dados simulados de lixeiras, como nível de preenchimento e peso acumulado, e envia essas informações para o broker MQTT. Os fluxos no Node-RED foram configurados para representar múltiplas lixeiras, cada uma com tópicos e identificadores únicos, permitindo uma simulação precisa e escalável.

Interface externa: API do Google Maps

Essa camada representa a integração do sistema com a API do Google Maps, que é utilizada para a geração e visualização de rotas otimizadas. Ela fornece dados de geolocalização, cálculo de rotas e visualizações em mapas, permitindo que os agentes de coleta acompanhem suas rotas em tempo real e que os supervisores tenham uma visão geográfica detalhada.

Hospedagem e Serviços na AWS

Tanto o servidor de aplicação, quanto o broker MQTT e o Node-RED estão hospedados na AWS, garantindo alta disponibilidade e escalabilidade. A nuvem AWS permite que o sistema gerencie a carga de processamento em tempo real e suporte a comunicação entre os componentes.

7.1.2 Comunicação entre as Camadas

1. Os dispositivos IoT simulados no Node-RED enviam dados para o servidor de aplicação via MQTT.
2. O servidor processa os dados, atualiza o gêmeo digital e interage com a API do Google Maps para gerar rotas otimizadas.
3. As rotas e notificações são enviadas para os dispositivos móveis e o portal web via APIs REST.
4. Supervisores e agentes interagem com as interfaces de usuário para visualizar dados e realizar ações no sistema.

7.2 Visão Lógica

Baseada no ponto de vista lógico, esta visão detalha a estrutura lógica do sistema. O diagrama, na figura 2 associado representa as classes principais e seus relacionamentos, destacando as responsabilidades e interações.

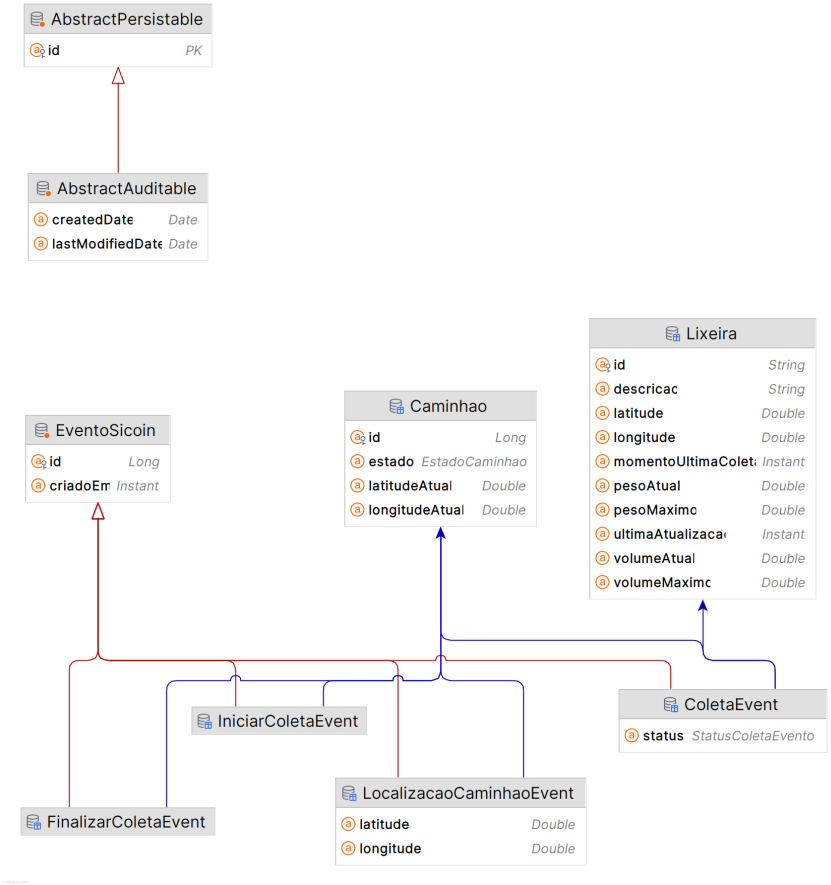

Figura 2: *Diagrama de entidades*

7.3 Visão Comportamental

Esta visão, baseada no ponto de vista comportamental, descreve o comportamento dinâmico do sistema. O diagrama associado apresenta a sequência de interações entre os componentes para realizar funcionalidades específicas. O diagrama de sequência ilustra o fluxo de mensagens entre sensores IoT, servidores e aplicativos.

Na figura 3, apresentamos o diagrama de sequência representando o comportamento do sistema e interação dos componentes para gerar rotas de coletas.

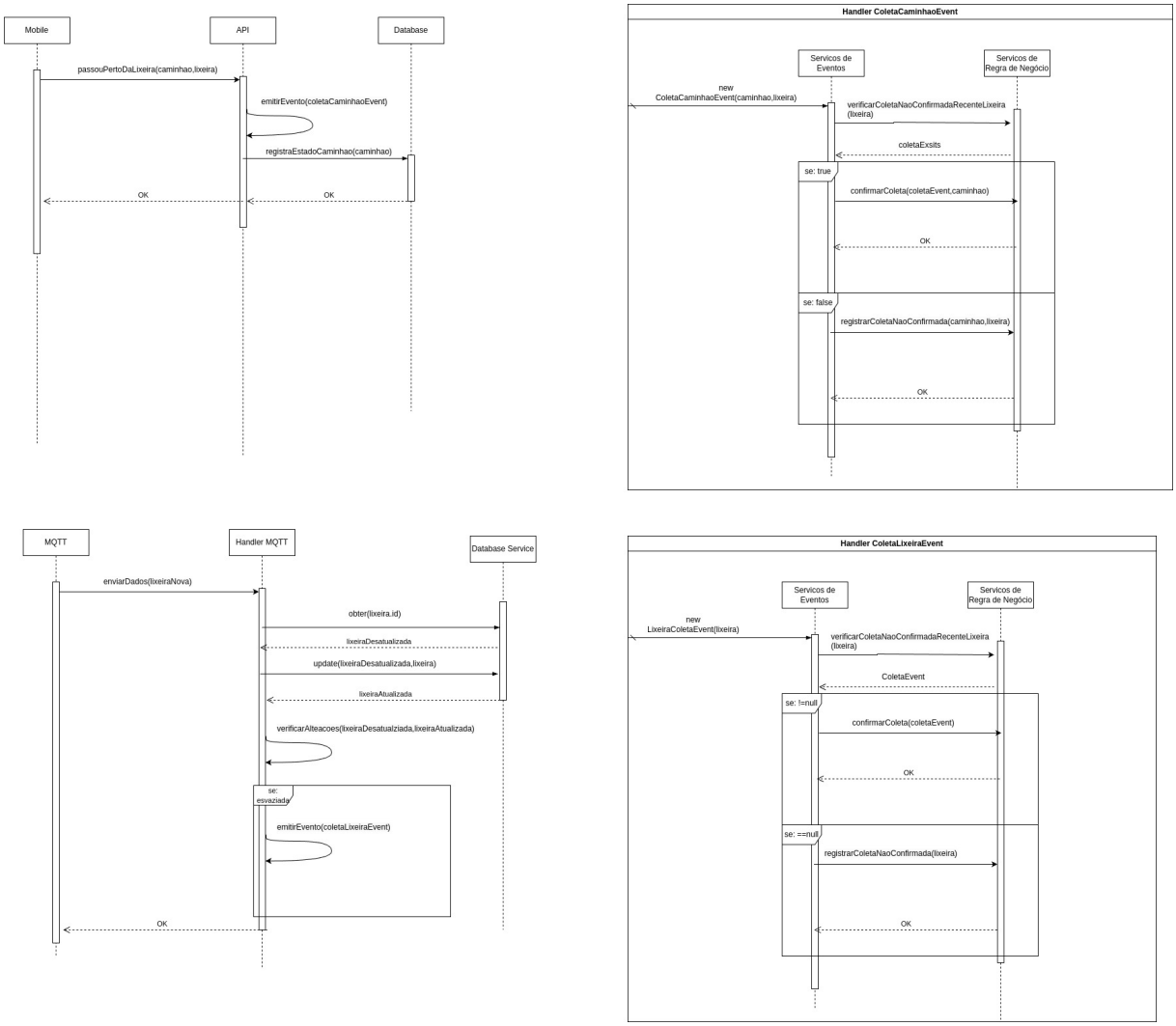

Figura 4: *Diagrama de eventos do sistema*

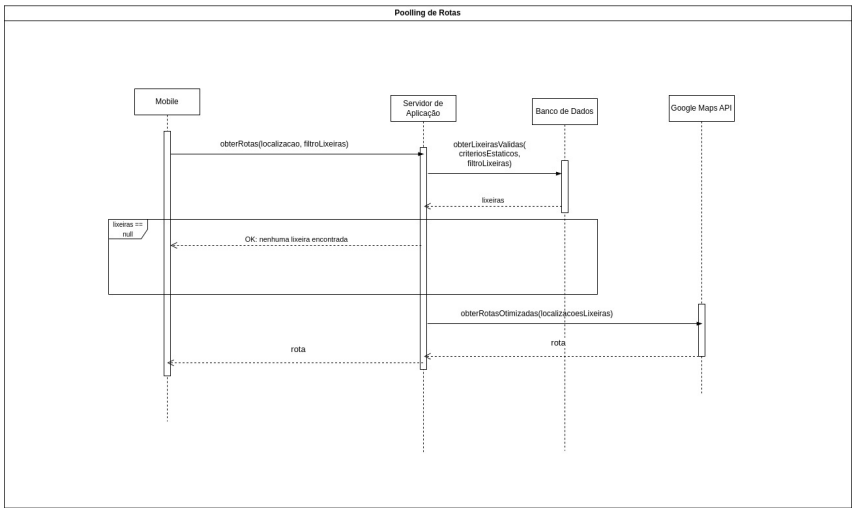

Figura 3: *Geração de rotas de coletas*

Na figura 4, apresentamos o diagrama de eventos do sistema.

7.4 Visão Física

A visão física, derivada do ponto de vista físico, representa a distribuição dos componentes do sistema em termos de infraestrutura. Este diagrama pode ser visualizado em conjunto com o diagrama da figura 1: Visão geral da arquitetura e implantação do sistema. Ele mostra a alocação de sensores, servidores e dispositivos móveis. O diagrama de implantação detalha a localização dos sensores IoT, infraestrutura em nuvem e aplicativos móveis.

8 Decisões Arquiteturais

DA-01 - Node-RED

- **Decisão**

  Utilizar o Node-RED para simulação de sensores IoT.

- **Descrição**

  O Node-RED será empregado para criar fluxos que simulem sensores instalados nas lixeiras, enviando dados como nível de resíduos e status em tempo real.

- **Justificativa**

  Uma vez que este trabalho consiste na construção de um protótipo funcional em um curto período de tempo, com limitações para aquisição de sensores físicos e toda a complexidade envolvida na configuração e integração com um broker de mensagens para captura de dados, o Node-RED foi escolhido por sua flexibilidade, facilidade de uso e integração com o backend, além de possibilitar a simulação eficiente de dispositivos IoT sem a necessidade de hardware físico.

DA-02 - Java Spring

- **Decisão**

  Utilizar o framework Java Spring para desenvolvimento do backend.

- **Descrição**

  O backend será implementado com Java Spring para gerenciar APIs REST, processar dados dos sensores e aplicar regras de negócio.

- **Justificativa**

  Escolhido pela robustez, suporte a microserviços e pela ampla adoção no mercado, garantindo escalabilidade e manutenção facilitada.

DA-03 - PostgreSQL

- **Decisão**

  Adotar o PostgreSQL como banco de dados relacional.

- **Descrição**

  O PostgreSQL será utilizado para persistir dados históricos, gerenciar informações sobre sensores, rotas e relatórios analíticos.

- **Justificativa**

  Escolhido por seu suporte avançado a dados relacionais, escalabilidade e capacidade de lidar com consultas complexas.

DA-04 - AWS

- **Decisão**

  Utilizar a AWS para hospedagem e infraestrutura de backend.

- **Descrição**

  Os serviços de backend serão implantados na AWS para garantir alta disponibilidade, balanceamento de carga e escalabilidade.

- **Justificativa**

  Escolhido por sua confiabilidade, variedade de serviços escaláveis e suporte a arquiteturas modernas.

DA-05 - MQTT

- **Decisão**

  Adotar o protocolo MQTT para comunicação entre sensores e backend.

- **Descrição**

  O MQTT será utilizado para transmitir dados dos sensores IoT de maneira eficiente, com baixa latência e suporte a redes instáveis.

- **Justificativa**

  Escolhido por seu baixo consumo de recursos, suporte a QoS e adequação a sistemas IoT.

DA-06 - React

- **Decisão**

  Utilizar o React para o desenvolvimento do frontend.

- **Descrição**

  O portal web será desenvolvido em React para oferecer uma interface interativa e responsiva aos supervisores do sistema.

- **Justificativa**

  Escolhido por sua eficiência, suporte a SPA (Single Page Applications) e comunidade ampla para suporte e bibliotecas.

DA-07 - React

- **Decisão**

  Utilizar Flutter para o aplicativo móvel e SQLite para banco de dados local.

- **Descrição**

  O aplicativo móvel será desenvolvido em Flutter, enquanto o SQLite será usado para armazenar dados offline no dispositivo.

- **Justificativa**

  Flutter foi escolhido por ser multiplataforma (Android/iOS) e SQLite por sua eficiência em armazenamento local e suporte a operações offline.

DA-08 - Processamento em Tempo Real

- **Decisão**

  Implementar processamento em tempo real no backend.

- **Descrição**

  Dados enviados pelos sensores serão processados imediatamente para gerar rotas otimizadas e notificações dinâmicas.

- **Justificativa**

  Necessário para atender ao requisito de tempo real, garantindo atualização contínua das informações e rápida resposta às mudanças.

DA-09 - API do Google Maps

- **Decisão**

  Integrar o sistema com a API do Google Maps para geração de rotas.

- **Descrição**

  A API será utilizada para calcular e exibir rotas otimizadas, além de fornecer dados de geolocalização e visualização em mapas.

- **Justificativa**

  Escolhido por sua confiabilidade, suporte robusto a mapas e rotas, e compatibilidade com aplicações web e móveis.

Referências

[1] Amazon EC2: Secure and resizable compute capacity in the cloud. https://aws.amazon.com/ec2/. Acessado em: 18 de dezembro de 2024.

[2] Node-RED: Low-code programming for event-driven applications. https://nodered.org/. Acessado em: 18 de dezembro de 2024.

[3] Objetivo 11: Tornar as cidades e os assentamentos humanos inclusivos, seguros, resilientes e sustentáveis. https://brasil.un.org/pt-br/sdgs/11. Acessado em: 18 de dezembro de 2024.

[4] Objetivo 13: Ação contra a mudança global do clima. https://brasil.un.org/pt-br/sdgs/13. Acessado em: 18 de dezembro de 2024.