GERENCIAMENTO DE RESÍDUOS SÓLIDOS URBANOS – UMA SOLUÇÃO COMPUTACIONAL

Lilian Beatriz Bezerra Gomes
lbbg@discente.ifpe.edu.br
Mateus Henrique da Silva Ferraz
mhsf@discente.ifpe.edu.br
Anderson Luis Souza Moreira
anderson.moreira@recife.ifpe.edu.br

RESUMO

A gestão de resíduos sólidos urbanos (RSU) é um desafio crescente no Brasil devido ao aumento da geração de resíduos e práticas inadequadas de descarte. Apesar da coleta municipal atender a grande parte da população, a sua ineficiência causa acúmulo de resíduos nas ruas, contaminação ambiental e proliferação de insetos e doenças. A solução apresentada neste trabalho, denominada SICoL (Sistema Inteligente de Coleta de Resíduos Sólidos Urbanos), busca mitigar esta problemática através de um sistema que auxilia o cidadão durante o descarte de resíduos e o poder público durante a sua coleta e gestão. Testes comprovaram a capacidade do sistema em promover práticas sustentáveis durante o descarte e coletas mais eficientes, e reduzir impactos ambientais causados pelo acúmulo de lixo nas ruas.

Palavras-chave: Coleta; resíduos; software.

ABSTRACT

Urban solid waste (MSW) management is a growing challenge in Brazil due to the increase in waste generation and inadequate disposal practices. Although municipal collection serves a large part of the population, its inefficiency causes waste accumulation on the streets, environmental contamination and the proliferation of insects and diseases. The solution presented in this work, called SICoL (Intelligent Urban Solid Waste Collection System), seeks to mitigate this problem through a system that assists citizens during waste disposal and the government during its collection and management. Tests have proven the

Instituto Federal de Pernambuco. Campus Recife. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas. 31 de janeiro de 2025.

system's ability to promote sustainable practices during disposal and more efficient collections, and to reduce environmental impacts caused by the accumulation of garbage on the streets.

Keywords: Collection; waste; software.

---

1 INTRODUÇÃO

Atualmente, a gestão de resíduos sólidos urbanos é vista como parte central do desenvolvimento governamental de muitos países desenvolvidos e subdesenvolvidos. Nas cidades brasileiras, a crescente geração desse tipo de resíduo e as práticas de descarte estabelecidas resultaram em volumes crescentes de resíduos sólidos urbanos (RSU) acumulados. Do ponto de vista do poder público, é de sua responsabilidade a limpeza urbana e o manejo dos resíduos sólidos urbanos. Entretanto, grande parte desses resíduos são resíduos domésticos, inevitavelmente gerados de forma contínua em consequência da existência humana.

De acordo com a Associação Brasileira de Resíduos e Meio Ambiente (ABREMA, 2023), estima-se que o brasileiro tenha gerado uma média de 1,04 kg de RSU por dia em 2022. Aplicando esse valor à população brasileira divulgada pelo Censo Demográfico 2022 do Instituto Brasileiro de Geografia e Estatística (IBGE), estima-se que aproximadamente 77,1 milhões de toneladas de RSU foram geradas no país em 2022. Isso corresponde a mais de 211 mil toneladas de resíduos gerados por dia, ou cerca de 380 kg/habitante/ano.

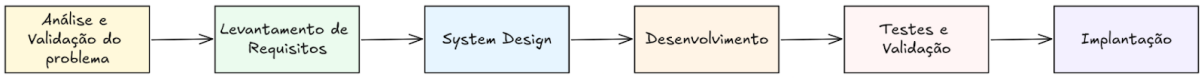
O Censo Demográfico também aponta que os serviços de coleta direta ou indireta de lixo atenderam 90,9% dos brasileiros em 2022, onde 82,5% dos moradores têm seus resíduos sólidos coletados diretamente no domicílio por serviços de limpeza (IBGE, 2022). Entratanto, a desorganização do sistema de coleta e a disposição e o tratamento inadequados do RSU contribuem para uma série de problemas que afetam diretamente a população, como a disseminação de insetos e pragas, a proliferação de doenças e a contaminação de solos, cursos d’água e lençóis freáticos.

Nesse sentido, o objetivo deste trabalho é mostrar o processo de planejamento e desenvolvimento de uma solução que busca mitigar esta problemática na origem, através de um sistema inteligente que auxilia tanto a população quanto o poder público na gestão do RSU e que funciona sob bases tecnológicas bem definidas, visando uma operação mais eficiente desde a etapa

de planejamento de coletas até a de processamento do lixo coletado nas centrais de tratamento.

2 METODOLOGIA

Figura 1 – Metodologia adotada neste trabalho



Fonte: Autores (2024)

2.1 Análise e validação do problema

Inicialmente, foi realizada uma análise do problema identificado, através de artigos, notícias e levantamentos estatísticos disponíveis na internet, com o intuito de entender o seu impacto no meio ambiente e na sociedade atual. Através desse estudo, o qual embasou a introdução do presente trabalho, foi constatada a gravidade da problemática e a falta de soluções existentes no mercado que a mitigasse significativamente, tornando viável o desenvolvimento de um sistema neste sentido.

2.2 Levantamento de requisitos

Durante esta etapa, a proposta de solução foi formalizada através da descrição em linguagem natural da dinâmica de funcionamento do sistema, e da identificação das funcionalidades e dos diferentes atores que estariam envolvidos nele. Foram identificados três tipos de usuários: gestores, coletores e cidadãos, onde cada um deles faria uso de aplicações diferentes, levando-se em consideração as funcionalidades às quais cada um teria acesso e o contexto no qual estariam inseridos.

Os usuários gestores são aqueles que fazem parte da secretaria de meio ambiente e são responsáveis pelo gerenciamento das coletas de resíduos da cidade ou do município. Através da aplicação do gestor, deverá ser possível cadastrar coletores, gerenciar as coletas a serem realizadas e as rotas a serem percorridas, além de acompanhar o deslocamento dos coletores durante as coletas. Como os gestores devem fazer uso da aplicação apenas durante o expediente de trabalho e algumas destas funcionalidades demandam telas maiores, elas serão agrupadas em uma aplicação *web*.

Os usuários coletores, por sua vez, são os funcionários subordinados à secretaria responsáveis pela coleta dos resíduos nas ruas. Na aplicação do coletor, deverá ser possível realizar login, transmitir sua localização em tempo real para os gestores, visualizar as coletas do dia atribuídas a ele, ver os

detalhes de uma coleta, ver a rota da coleta através de um mapa interativo, e iniciar e finalizar coletas. Como os coletores atuam em campo e a aplicação precisa interagir diretamente com um sistema de GPS, essas funcionalidades serão agrupadas em uma aplicação móvel, a qual possui acesso nativo à localização do usuário.

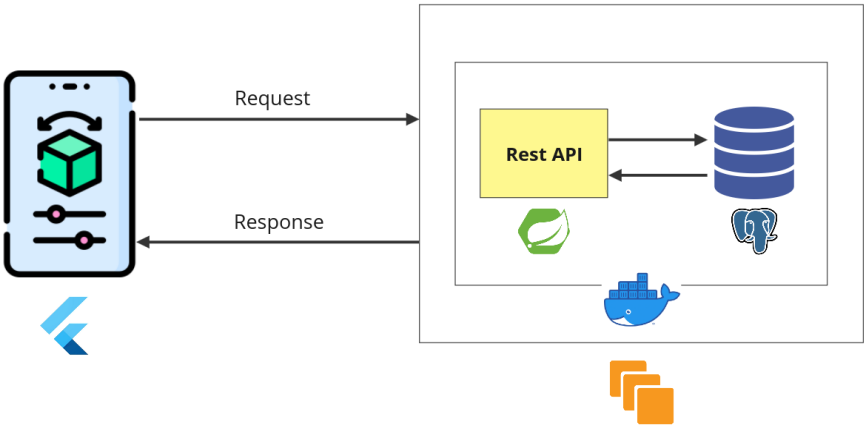
Os usuários cidadãos são as pessoas responsáveis pelo descarte do lixo e que residem na cidade onde o sistema foi implantado. Através da aplicação do cidadão, deverá ser possível se cadastrar, realizar login, cadastrar um ou mais de seus endereços residenciais, visualizar o horário estimado que o caminhão de lixo passará em cada um de seus endereços e tirar dúvidas através de um canal de comunicação com a prefeitura pelo WhatsApp. A fim de que o cidadão possa utilizar a aplicação de qualquer lugar, essas funcionalidades também serão agrupadas em uma aplicação móvel.

A partir desta visão geral do sistema, as suas funcionalidades foram detalhadas e ordenadas por nível de prioridade, onde aquelas de maior importância para o funcionamento do sistema e menor complexidade de implementação seriam as primeiras a serem implementadas. Após isso, cada uma das funcionalidades foi quebrada em atividades menores (tarefas) que foram agrupadas em ciclos de trabalho de duas ou três semanas, chamados de iterações, onde as tarefas de maior prioridade ficaram nas primeiras iterações e as de menor prioridade nas últimas.

2.3 *System Design*

Nesta etapa, o diagrama da arquitetura do sistema foi detalhadamente implementado, a partir da definição dos módulos e aplicações que fariam parte dele, de como a comunicação entre eles aconteceria e das tecnologias que seriam utilizadas para construí-los, de forma a satisfazer os requisitos e funcionalidades definidos na etapa anterior (Figura 2). Esta etapa de planejamento é extremamente importante, pois a partir dela é possível que a equipe de desenvolvimento tenha uma visão geral do funcionamento do sistema e dos seus diferentes componentes.

Figura 2 – Diagrama da arquitetura do sistema



Fonte: Autores (2024)

Para o desenvolvimento do aplicativo móvel, foi utilizado o *framework* Flutter devido a sua flexibilidade que permite o desenvolvimento de aplicativos para as plataformas Android e IOS através de uma única base de código (Karasavvas, 2022). Além disso, foram integrados ao aplicativo alguns serviços externos como o Google Maps, utilizado para exibir a localização do coletor em tempo real e as rotas, e o Firebase Cloud Messaging (FCM), que permite que notificações sejam enviadas ao cidadão quando uma coleta que passa em algum de seus endereços for iniciada.

Para o desenvolvimento de uma comunicação eficiente entre os dados e o aplicativo móvel, foi desenvolvida uma REST API, uma Interface de Programação de Aplicações (API) que segue os princípios da arquitetura REST (Representational State Transfer). Esse tipo de API permite que diferentes sistemas possam se comunicar de maneira padronizada através de requisições HTTP, como *GET*, *POST*, *PUT* e *DELETE* (Red Hat, 2024).

- ***GET***: para obter dados de um recurso.
- ***POST***: para criar um novo recurso.
- ***PUT***: para atualizar um recurso existente.
- ***DELETE***: para remover um recurso.

Por ser uma abordagem amplamente utilizada e de fácil integração, a REST API foi uma escolha para o desenvolvimento da aplicação, utilizando a linguagem Java com o framework Spring Boot devido a sua robustez e por ser amplamente utilizado e testado em diversos sistemas em produção (Stack Overflow, 2024). Entre as bibliotecas utilizadas no projeto, destaca-se o Spring Data JPA, o qual é responsável pela comunicação entre a aplicação e o banco de

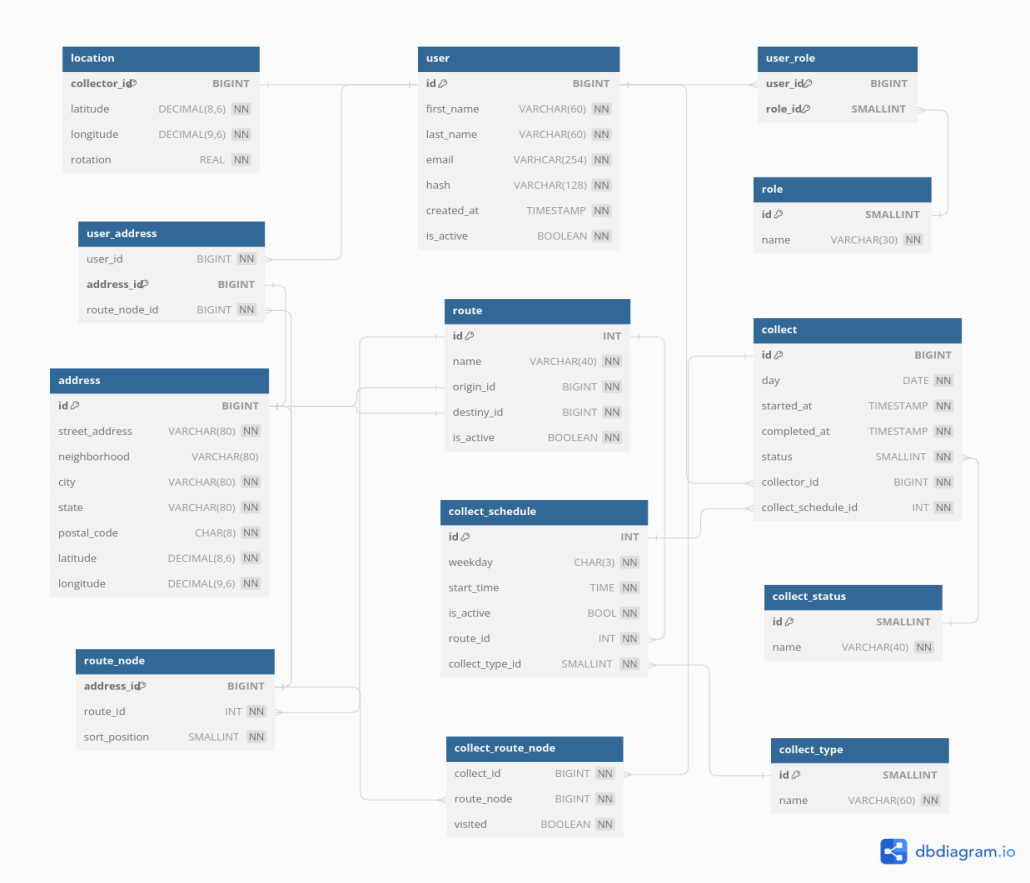
dados ao simplificar a implementação de uma camada de acesso a dados (Spring, 2025).

O banco de dados utilizado foi o PostgreSQL, por se tratar de um dos bancos de dados relacionais mais robustos e utilizados no mercado (Stack Overflow, 2024), além de ser fácil de configurar e bem documentado, facilitando a sua utilização. Neste sistema, optou-se por um banco de dados relacional (SQL) com uma estrutura normalizada com o intuito de garantir a integridade das informações, remover a necessidade de dados duplicados através do uso de relacionamentos entre tabelas e, consequentemente, reduzir os requisitos de armazenamento de dados. Em troca, operações de leituras são mais lentas quando *join queries* são necessárias para agrupar informações relacionadas armazenadas em diferentes tabelas, quando comparadas a operações de leitura em bancos de dados não-relacionais (NoSQL) com estruturas de-normalizadas.

Entretanto, como a base de usuários por cliente (cidade ou município) não era gigante e o uso da plataforma pelos cidadãos, os quais representam a maior parte dos usuários, aconteceria de forma pontual, *join queries* não trariam um impacto significante à performance do sistema. Além disso, no contexto do sistema, performance de leitura não era um atributo inegociável, o que faz com que os benefícios do uso de um banco de dados SQL se sobressaiam aos benefícios do uso de um banco de dados NoSQL.

Antes de implementar o *schema* do banco de dados, o seu diagrama de entidade-relacionamento (ERD) foi modelado, com o objetivo de facilitar a visualização das estruturas das tabelas e o relacionamento entre elas (Figura 3). O *schema* de um banco de dados é a descrição em linguagem formal, definida pelo banco de dados escolhido, da sua estrutura e das tabelas necessárias para armazenar os dados produzidos pelo uso das funcionalidades do sistema. Para o design do ERD foi utilizado o dbdiagram, uma ferramenta online, gratuita e colaborativa para modelagem de ERDs utilizando DBML (Database Markup Language).

Figura 3 – ERD do sistema modelado no dbdiagram



Fonte: Autores (2024)

Para facilitar o processo de desenvolvimento e de *deploy* do *back-end* do sistema em uma máquina virtual, foi utilizado o Docker, uma plataforma de código aberto que permite criar, implantar e executar aplicações de forma isolada em contêineres. Esses contêineres são pacotes leves e portáteis que incluem tudo o que a aplicação precisa para funcionar, como código, bibliotecas e dependências, garantindo que ela funcione de forma consistente em qualquer ambiente, seja em desenvolvimento, teste ou produção. (Itau Tech, 2024).

Dessa forma, através da conteinerização em conjunto dos serviços da REST API e do banco de dados PostgreSQL, é possível garantir uma execução estável e consistente de ambos os serviços, independentemente do ambiente em que estão sendo executados, simplificando a configuração, portabilidade e escalabilidade do sistema. Esta configuração é realizada através do uso do Docker Compose, um arquivo de configuração do Docker que permite configurar múltiplos serviços de um sistema, executá-los e gerenciá-los de maneira coordenada através da linha de comando.

A máquina virtual utilizada para fazer o *host* do sistema de produção foi adquirida através do serviço EC2 (Elastic Compute Cloud) da AWS (Amazon Web Services), o qual é amplamente utilizado em soluções corporativas (Stack Overflow, 2024) ao prover infraestrutura escalável a preços acessíveis e com ótimo desempenho.

2.4 Desenvolvimento

Apesar das etapas da metodologia estarem apresentadas em formato de sequência, o processo de desenvolvimento do sistema ocorreu em múltiplas iterações, como destrinchado na etapa de levantamento de requisitos. Durante cada iteração, as suas tarefas foram desenvolvidas, testadas e validadas com o cliente, e, a depender do seu *feedback*, implantadas em produção. Através da metodologia de desenvolvimento adotada, foi possível:

- **Reduzir o ciclo de *feedback* do cliente:** A cada iteração, uma parte do sistema era entregue ao cliente, diferentemente do que acontece em uma metodologia em cascata onde o cliente apenas tem acesso ao sistema ao final do projeto. A abordagem adotada aumenta a satisfação e confiança do cliente no projeto e na equipe.
- **Flexibilizar mudanças no *backlog* e no *design* do sistema:** Como o projeto era entregue em partes, mudanças nos requisitos solicitadas pelo cliente eram menos custosas de serem implementadas e menos propensas a falhas, pois o sistema ainda não se encontrava em produção e a quantidade de LoC (linhas de código), e consequentemente do tamanho da área de impacto de uma mudança, era menor.
- **Detectar falhas e inconsistências prematuramente:** Cada parte desenvolvida do sistema era testada, então falhas eram detectadas rapidamente. Em contraste, na metodologia de desenvolvimento em cascata, a etapa de testes ocorre após a etapa de desenvolvimento.
- **Aprimorar o processo de desenvolvimento:** Através de ciclos de desenvolvimento, é possível revisar, ao final de cada iteração, os procedimentos que deram certo e os que não surtiram o efeito esperado, permitindo mudanças e flexibilizações para as próximas iterações.

2.5 Testes e Validação

Nesta etapa, as funcionalidades que foram desenvolvidas na etapa de desenvolvimento são devidamente testadas, usando testes de ponta a ponta (*end-to-end*) e testes de aceitação com o cliente para validar se aquilo que foi implementado condiz com o que foi definido na etapa de levantamento de

requisitos. Os *feedbacks* obtidos são então coletados e, caso correções sejam necessárias, estas são implementadas nas iterações seguintes.

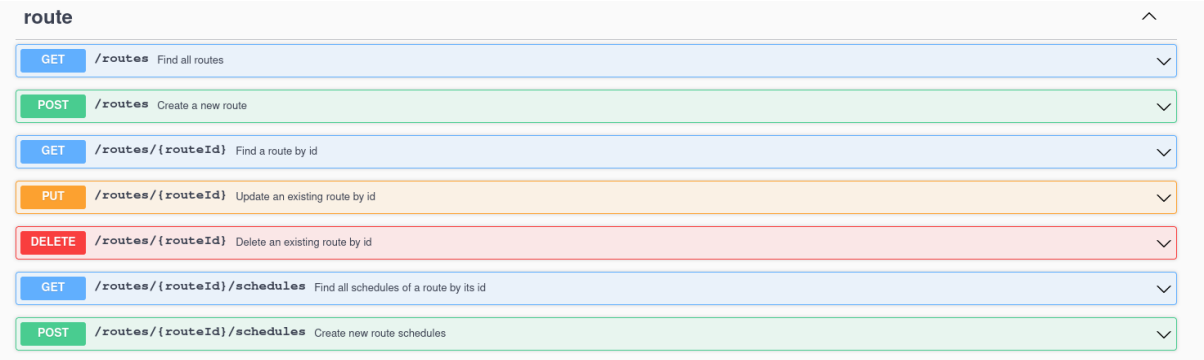
2.6 Implantação

Nesta etapa, as melhorias que foram implementadas e validadas pelo cliente durante todo o processo de desenvolvimento são colocadas em produção. A partir dos *feedbacks* gerados pelos usuários das aplicações, atualizações consentidas pelo cliente são implementadas. Este ciclo de manutenção se repete até o fim do contrato com a prefeitura.

3 RESULTADOS E ANÁLISE

Inicialmente, a REST API foi desenvolvida para permitir a comunicação entre as aplicações clientes e o banco de dados. Ela foi desenvolvida com Java e Spring Boot e todos os recursos implementados foram documentados utilizando a especificação OpenAPI 3.0 (Figura 4), permitindo que aqueles responsáveis pelo desenvolvimento dos aplicativos conseguissem fazer uma integração perfeita com os serviços necessários.

Figura 4 – Documentação dos endpoints de rota usando OpenAPI 3.0



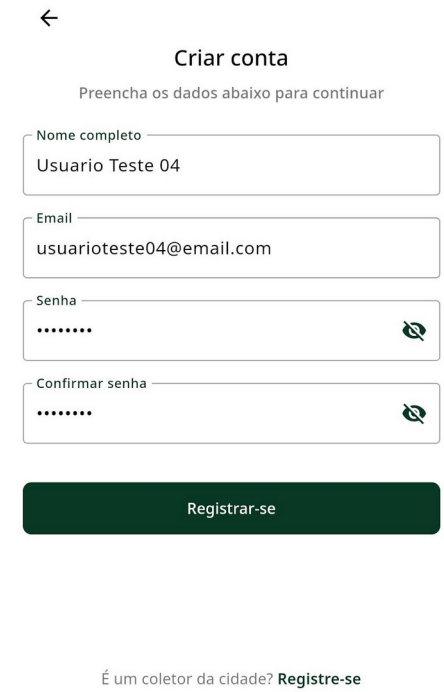
Fonte: Autores (2024)

O sistema subdivide-se em duas aplicações com grupos de funcionalidades diferentes, sendo uma delas destinada ao coletor de lixo e outra ao usuário cidadão, estando segregadas através do processo de autenticação (Figura 5). Os dois tipos de usuário se comunicam entre si através da troca de dados com a API. Tanto usuários coletores quanto usuários cidadãos devem estar devidamente cadastrados no sistema para que possam ter acesso às funcionalidades (Figura 6).

Figura 5 – Tela de login do SICoL

Fonte: Autores (2024)

Figura 6 – Tela de cadastro do SICoL



Fonte: Autores (2024)

Além disso, como o usuário coletor tem acesso privilegiado a algumas funcionalidades do sistema, o seu cadastro não deve estar liberado para todas as pessoas. Caso um usuário queira cadastrar-se como coletor da cidade, ele deve passar por uma checagem de segurança (Figura 7). Essa checagem consiste na validação de um token de segurança que apenas as pessoas responsáveis pela coleta da cidade possuem acesso.

Figura 7 – Tela de segurança para cadastro de coletor

Fonte: Autores (2024)

Assim que um cidadão se cadastra na aplicação, uma tela de mapa é exibida para que ele possa selecionar o seu endereço na cidade (Figura 8). Ao confirmar, ele é redirecionado para uma tela para confirmar os dados de endereço que foram automaticamente preenchidos (Figura 9). Caso exista alguma incongruência nos dados, ele pode alterá-los diretamente ou retornar ao mapa para selecionar outra localização. Quando o cadastro é feito, a API automaticamente associa o endereço cadastrado à rota de coleta mais próxima, através da aplicação da fórmula Haversine entre as coordenadas do endereço cadastrado e os pontos que compõem as rotas mais próximas.

A fórmula de Haversine é uma maneira muito precisa de calcular distâncias entre dois pontos na superfície de uma esfera usando a latitude e a longitude dos dois pontos. A fórmula de Haversine é uma reformulação da lei esférica dos

cossenos, mas a formulação em termos de Haversines é mais útil para ângulos e distâncias pequenas (Kettle, 2017).

Figura 8 – Tela de mapa para seleção de endereço

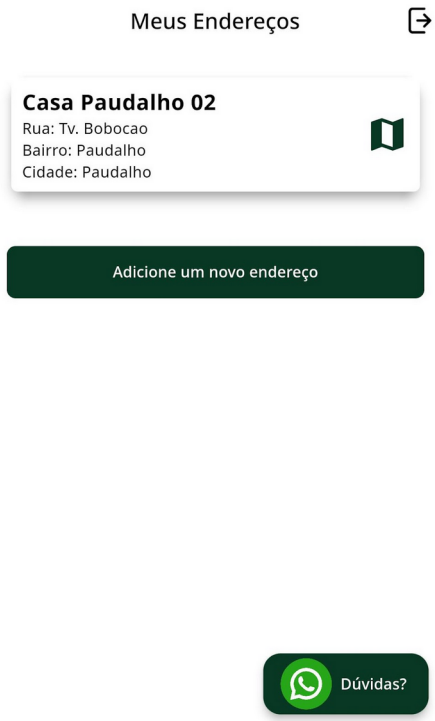
Fonte: Autores (2024)

Figura 9 – Tela de formulário para cadastro de endereço

Fonte: Autores (2024)

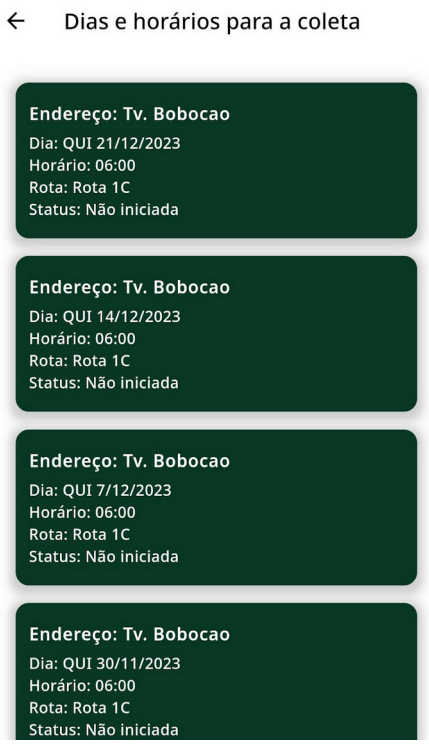
Após o cadastro do seu endereço, o usuário cidadão é redirecionado para a tela inicial da sua interface (Figura 10). Nela estão listados todos os seus endereços cadastrados, sendo possível adicionar um novo ou remover um existente. Ao clicar em um destes endereços, é possível ver a lista de coletas que serão realizadas na rota associada, juntamente com as informações da coleta (Figura 11). Além disso, no canto inferior direito da tela inicial, tem um botão que ao ser clicado redireciona o usuário ao canal de apoio do WhatsApp da prefeitura, e no canto superior direito tem um botão para finalizar a sessão.

Figura 10 – Tela inicial do usuário cidadão



Fonte: Autores (2024)

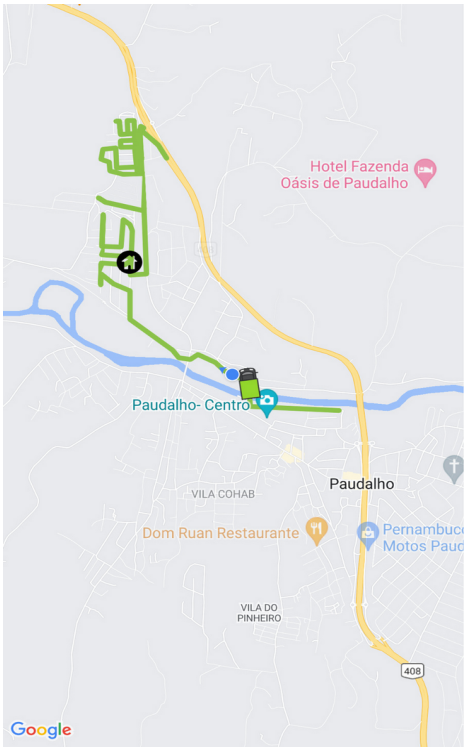
Figura 11 – Tela de coletas do endereço



Fonte: Autores (2024)

Caso o usuário clique em uma coleta que tenha sido iniciada pelo coletor, ele é redirecionado para a tela de mapa que mostrará o seu endereço, a rota que o caminhão de coleta fará e a localização do caminhão atualizada em tempo real (Figura 12).

Figura 12 – Tela de acompanhamento de coleta



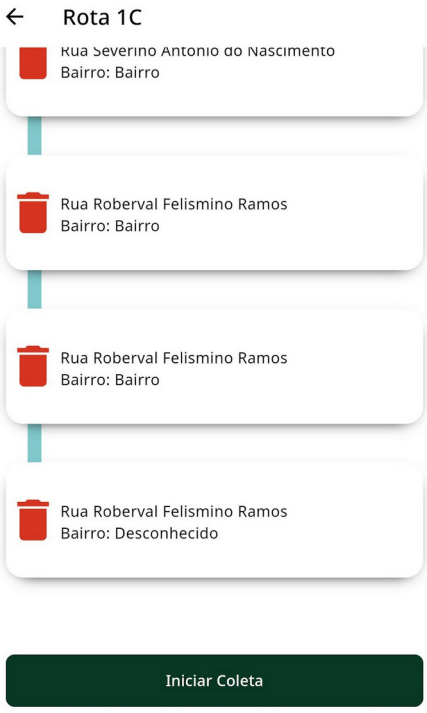
Fonte: Autores (2024)

A tela inicial da interface de coletor é a tela de coletas (Figura 13), que é acessada logo após o processo de autenticação. Nela é possível visualizar uma lista com as coletas do dia. Em cada uma delas estão detalhadas as informações necessárias para a execução da coleta, como a rota associada, o dia e horário e qual tipo de resíduo será coletado. É importante observar também que existe uma segregação entre coletas já realizadas e coletas a serem realizadas, de forma que as que já foram realizadas não podem ser iniciadas novamente. Ao clicar em uma das coletas a serem realizadas, o coletor é redirecionado à tela que mostra todo o caminho da rota em formato de linha do tempo onde ao final está disposto um botão para dar início à coleta (Figura 14). Além disso, também é possível encerrar a sessão clicando no botão posicionado no canto superior direito da tela inicial.

Figura 13 – Tela inicial do usuário coletor

Fonte: Autores (2024)

Figura 14 – Tela de detalhes da coleta



Fonte: Autores (2024)

Ao iniciar uma das coletas, uma requisição é enviada para o *endpoint* da API responsável por agrupar todos os usuários cidadãos que possuem endereços associados a rota que será percorrida na coleta iniciada, e dispara uma notificação para os seus dispositivos utilizando o FCM (Figura 15). O intuito desta funcionalidade é estimular o cidadão a colocar o lixo para fora de casa apenas no horário em que seu recolhimento será realizado, evitando acúmulo de lixo nas ruas e todos os malefícios que esta condição causa.

Figura 15 – Notificação de início de coleta

Fonte: Autores (2024)

Os resultados apresentados neste trabalho foram baseados em testes realizados em ambientes controlados, para verificar o funcionamento das funcionalidades implementadas, e também na cidade de Paudalho, junto a Secretaria de Meio Ambiente do município, para verificar o efeito prático do uso do sistema pelos gestores, coletores e cidadãos. Foi observada uma diminuição do lixo acumulado nas ruas, impulsionada principalmente pelo fácil acesso da população a agenda de coletas em suas residências e pelas notificações em tempo real do melhor horário para fazer a separação e o descarte do lixo.

A análise também focou na precisão e eficiência do sistema em monitorar o cumprimento das rotas pelos caminhões, em busca de rotas mais eficientes que atendessem a todas as residências dos setores do município em que o sistema estava implantado. Durante os testes, foram utilizados indicadores de desempenho, como variação de movimento, otimização das rotas e proximidade do caminhão de coleta.

- **Variação de movimento:** Mostra a variação de movimento dos carros. Dessa forma, utilizando esse indicador, é possível identificar possíveis problemas como falta de atualização ou envio de dados de localização incorretos.
- **Otimização das rotas:** Monitora o número de vezes que as rotas de coleta foram ajustadas para otimização, com base na demanda e localização dos endereços cadastrados pelos usuários do sistema.
- **Proximidade do Caminhão de Coleta:** Monitora a proximidade do caminhão de coleta em relação ao endereço do usuário e envia uma notificação quando o veículo estiver prestes a chegar na localização programada para a coleta. Esse recurso visa otimizar o tempo do usuário para realizar o despejo adequado dos resíduos.

4 CONSIDERAÇÕES FINAIS

A versatilidade do sistema de monitoramento desenvolvido neste projeto destaca-se por sua capacidade de atender a diversas necessidades que envolve a coleta de resíduos nas grandes cidades, proporcionando uma gestão mais eficiente por parte dos usuários e responsáveis. Além disso, o *software* desenvolvido facilita o processo complexo de monitoramento e coleta, criando um fluxo de interação mais inteligente e eficiente entre cidadãos e municípios. Este projeto demonstra os benefícios de um cuidado adequado e imediato com as cidades, alinhando-se aos Objetivos de Desenvolvimento Sustentável (ODS) 11 e 12 da ONU, que visam garantir cidades e comunidades sustentáveis e promover o consumo e descarte responsáveis para todos (ONU, 2024).

Para trabalhos futuros, a aplicação de técnicas de aprendizagem de máquina e inteligência artificial no processamento e análise dos dados coletados promete ampliar ainda mais a eficácia do sistema. O desenvolvimento de uma aplicação *web* para os gestores das cidades realizarem o gerenciamento independente de coletores, rotas e coletas, e a integração dessas tecnologias pode melhorar a precisão do monitoramento, além de permitir a previsão de locais que são mais propícios a sofrer com o descarte impróprio de resíduos com base em padrões de comportamento dos usuários. Assim, o sistema poderá fornecer *insights* mais profundos e personalizados promovendo uma

resposta mais rápida e eficaz na coleta de resíduos sólidos urbanos, contribuindo para o bem estar das populações e cidades.

Em conclusão, este projeto explorou o impacto positivo do desenvolvimento do *software* de monitoramento de resíduos sólidos urbanos (RSU) no contexto da sustentabilidade das cidades, destacando suas implicações econômicas, sociais e ambientais. Ao abordar os benefícios, como a melhoria na eficiência da coleta de resíduos e o fortalecimento do engajamento entre cidadãos e gestores, assim como os desafios relacionados ao uso de dados e novas tecnologias, foi possível compreender a complexidade desse sistema. Além disso, com a integração de tecnologias de aprendizagem de máquina e inteligência artificial, vislumbra-se um futuro em que o monitoramento e a coleta de resíduos possam ser ainda mais eficientes, possibilitando análises preditivas que identifiquem áreas com potencial para descartes inadequados. Por fim, este projeto contribui para uma visão mais ampla sobre o papel da tecnologia no desenvolvimento sustentável e ressalta a necessidade de uma abordagem colaborativa e integrada para lidar com os desafios da gestão de resíduos nas cidades modernas.

REFERÊNCIAS

ABREMA – Associação Brasileira de Empresas de Tratamento de Resíduos e Emissões. **Panorama dos Resíduos Sólidos do Brasil 2023**. Disponível em: https://www.abrema.org.br/download/92323/?tmstv=1709314789. Acesso em: 30 set. 2024.

IBGE. **Censo 2022**: rede de esgoto alcança 62,5% da população, mas desigualdades regionais e por cor e raça persistem. **Agência de notícias**, 23 fev. 2024. Disponível em: https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/39237-censo-2022-rede-de-esgoto-alcanca-62-5-da-populacao-mas-desigualdades-regionais-e-por-cor-e-raca-persistem. Acesso em: 29 jan. 2025.

ITAU TECH. **Docker**: uma ferramenta para potencializar o desenvolvimento de softwares. **Medium**. 13 out. 2022. Disponível em: https://medium.com/itautech/docker-uma-ferramenta-para-potencializar-o-desenvolvimento-de-softwares-534eabf18a4d. Acesso em: 14 nov. 2024.

KARASAVVAS, Theodoros. **Why Flutter is the most popular cross-platform mobile SDK**. **Stack Overflow**, 21 fev. 2022. Disponível em: https://stackoverflow.blog/2022/02/21/why-flutter-is-the-most-popular-cross-platform-mobile-sdk/. Acesso em: 14 nov. 2024.

KETTLE, Simon. **Distance on a sphere**: The Haversine Formula. **Esri Community**, 10 mai. 2017. Disponível em: https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128. Acesso em: 14 nov. 2024.

ONU. **Objetivos de Desenvolvimento Sustentável**. Disponível em: https://brasil.un.org/pt-br/sdgs. Acesso em: 30 set. 2024.

RED HAT. **O que é uma API REST?**. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em: 14 nov. 2024.

SPRING. **Spring Data JPA:** Overview. Disponível em: https://spring.io/projects/spring-data-jpa#overview. Acesso em: 30 jan. 2025.

STACK OVERFLOW. **2024 Stack Overflow Developer Survey**: Cloud Platforms. Disponível em: https://survey.stackoverflow.co/2024/technology#1-cloud-platforms. Acesso em: 14 nov. 2024.

STACK OVERFLOW. **2024 Stack Overflow Developer Survey**: Databases. Disponível em: https://survey.stackoverflow.co/2024/technology#1-databases. Acesso em: 14 nov. 2024.

STACK OVERFLOW. **2024 Stack Overflow Developer Survey**: Programming, scripting and markup languages. Disponível em: https://survey.stackoverflow.co/2024/technology#1-programming-scripting-and-markup-languages. Acesso em: 14 nov. 2024.

STACK OVERFLOW. **2024 Stack Overflow Developer Survey**: Web Frameworks and Technologies. Disponível em: https://survey.stackoverflow.co/2024/technology#1-web-frameworks-and-technologies. Acesso em: 14 nov. 2024.