

Universidade da Beira Interior

Departamento de Informática



**Departamento de
Informática**

Nº 1 - 2024: Sistema Inteligente de Gestão de Resíduos

Elaborado por:

João Martins M13939
Vasco Senra M13239

Orientador:

Professor Doutor Bruno Miguel Silva

11 de setembro de 2024

Agradecimentos

A conclusão deste trabalho, bem como da grande maior parte da minha vida académica não seria possível sem a ajuda de ...

Conteúdo

| | |
|--|------------|
| Conteúdo | iii |
| Lista de Figuras | vii |
| Lista de Tabelas | ix |
| 1 Introdução | 1 |
| 1.1 Enquadramento | 1 |
| 1.2 Motivação | 2 |
| 1.3 Objetivos | 2 |
| 1.4 Organização do Documento | 3 |
| 2 Estado da Arte | 5 |
| 2.1 Introdução | 5 |
| 2.2 <i>Background: Smart Cities</i> | 5 |
| 2.3 Comparação de Projetos de Gestão de Resíduos com <i>Internet of Things</i> (IoT) | 6 |
| 2.3.1 Projeto Um | 6 |
| 2.3.2 Projeto Dois | 6 |
| 2.3.3 Projeto Três | 7 |
| 2.3.4 Escolha do Projeto para Análise | 7 |
| 2.4 Análise do Projeto | 7 |
| 2.4.1 Visão Geral do Projeto com Ideia Complementar | 7 |
| 2.4.2 Arquitetura do Sistema | 8 |
| 2.4.3 Integração de Sensores | 9 |
| 2.4.4 Modelo de Funcionamento | 11 |
| 2.4.5 Resultados e Discussões | 12 |
| 2.4.6 Conclusões do Projeto em Revisão | 13 |
| 2.5 Contraste e Contribuições do Projeto | 13 |
| 2.5.0.1 Contrastes: | 14 |
| 2.5.0.2 Contribuições: | 15 |
| 2.6 Conclusões | 15 |

| | | |
|----------|---|-----------|
| 2.7 | Projeto Mapeamento <i>Three Dimensions</i> (3D): <i>3D Surface Mapping using Ultrasonic Sensors</i> [1] | 15 |
| 2.7.1 | Visão Geral do Projeto | 15 |
| 2.7.2 | Arquitetura do Sistema | 16 |
| 2.7.3 | Modelo de Funcionamento | 16 |
| 2.7.4 | Resultados e Conclusões | 18 |
| 3 | Engenharia de Software e Arquitetura do Sistema | 21 |
| 3.1 | Introdução | 21 |
| 3.2 | Análise de Requisitos | 21 |
| 3.2.1 | Requisitos Funcionais | 21 |
| 3.2.1.1 | Requisitos Funcionais do Sistema | 22 |
| 3.2.1.2 | Requisitos Funcionais da Aplicação Móvel | 23 |
| 3.2.1.3 | Requisitos Funcionais da Dashboard | 23 |
| 3.2.2 | Requisitos Não Funcionais | 24 |
| 3.2.2.1 | Requisitos Não Funcionais do Sistema | 24 |
| 3.2.2.2 | Requisitos Não Funcionais da Aplicação Móvel | 25 |
| 3.2.2.3 | Requisitos Não Funcionais <i>Dashboard</i> | 26 |
| 3.3 | Tecnologias Utilizadas | 26 |
| 3.3.1 | Tecnologias Utilizadas no Contentor | 27 |
| 3.3.2 | Tecnologias Utilizadas no Servidor | 27 |
| 3.3.3 | Aplicação Móvel | 28 |
| 3.3.4 | <i>Dashboard</i> | 28 |
| 3.3.5 | Relatório | 28 |
| 3.4 | Casos de Uso | 29 |
| 3.4.1 | Casos de Uso da Aplicação Móvel | 29 |
| 3.4.2 | Casos de Uso da Dashboard | 30 |
| 3.5 | Diagrama de Atividades | 31 |
| 3.5.1 | Diagrama de Atividades da Aplicação Móvel | 31 |
| 3.5.2 | Diagrama de Atividades da dashboard | 33 |
| 3.6 | Arquitetura do Sistema IoT | 35 |
| 3.7 | Base de Dados | 36 |
| 3.8 | Conclusões | 37 |
| 4 | Interoperabilidade | 39 |
| 4.1 | Introdução | 39 |
| 4.2 | Interoperabilidade Semântica | 39 |
| 4.3 | Interoperabilidade Sintática | 40 |
| 4.4 | Interoperabilidade de Dispositivos | 41 |
| 4.5 | Interoperabilidade de Plataforma | 42 |
| 4.6 | Interoperabilidade de Rede | 42 |

| | | |
|----------|---|-----------|
| 4.7 | Conclusões | 43 |
| 5 | Ética e Segurança | 45 |
| 5.1 | Introdução | 45 |
| 5.2 | Código de Ética | 45 |
| 5.2.1 | Sistema IoT e Sociedade | 45 |
| 5.2.2 | Aplicação Móvel | 47 |
| 5.2.3 | <i>Dashboard</i> | 47 |
| 5.3 | Segurança da Informação | 48 |
| 5.3.1 | Ataque de Homem no Meio (<i>Man-in-the-Middle Attack</i>) | 48 |
| 5.4 | <i>Malware</i> – Criação de <i>Botnets</i> | 48 |
| 5.4.1 | <i>Denial-of-Service</i> (DOS) | 49 |
| 6 | Implementação e Demonstração | 51 |
| 6.1 | Introdução | 51 |
| 6.2 | Protótipo IoT | 51 |
| 6.2.1 | Contentor | 51 |
| 6.2.2 | Hardware | 52 |
| 6.2.3 | Software | 55 |
| 6.2.3.1 | Arduino | 55 |
| 6.2.3.2 | <i>Gateway</i> entre o dispositivo Arduino e a <i>Application Programming Interface</i> (API) do servidor | 56 |
| 6.2.3.3 | Servidor <i>Node.js</i> | 57 |
| 6.2.3.4 | <i>Trigger</i> | 58 |
| 6.3 | Clientes | 59 |
| 6.3.1 | Aplicação <i>Web - Dashboard</i> | 59 |
| 6.3.1.1 | Login | 59 |
| 6.3.1.2 | Menu | 60 |
| 6.3.2 | Aplicação Móvel | 60 |
| 6.3.2.1 | Página Inicial — Seleção da Região | 60 |
| 6.3.2.2 | Consulta dos Contentores | 61 |
| 6.3.2.3 | Sugestão Inteligente | 63 |
| 6.4 | Conclusões | 64 |
| 7 | Conclusões e Trabalho Futuro | 65 |
| 7.1 | Conclusões Principais | 65 |
| 7.2 | Trabalho Futuro | 66 |
| | Bibliografia | 67 |

Lista de Figuras

| | | |
|------|---|----|
| 2.1 | Arquitetura do projeto [2] | 9 |
| 2.2 | Diagrama de fluxo do projeto [2] | 11 |
| 2.3 | Nível original e previsto de lixo. | 14 |
| 2.4 | Esquema do <i>hardware</i> [1] | 16 |
| 2.5 | Fluxograma do programa [1] | 18 |
| 2.6 | Conversão para 2D com vista TOP-DOWN. | 19 |
| | | |
| 3.1 | Diagrama de casos de uso da aplicação móvel | 30 |
| 3.2 | Diagrama de casos de uso da Dashboard. | 31 |
| 3.3 | Diagrama de atividades da aplicação móvel | 33 |
| 3.4 | Diagram de Atividades da Dashboard. | 34 |
| 3.5 | Arquitetura do Sistema IoT | 36 |
| 3.6 | Modelo da Base de Dados | 37 |
| | | |
| 6.1 | Vista interior do contentor | 52 |
| 6.2 | Vista exterior do contentor | 52 |
| 6.3 | Servo motor e base dos sensores | 53 |
| 6.4 | Três sensores dentro do caixote para medir | 54 |
| 6.5 | Conexões Arduino e <i>Servo Motor</i> | 54 |
| 6.6 | Imagem do sistema de Login | 60 |
| 6.7 | Imagem do Menu principal | 60 |
| 6.8 | Página inicial para selecionar uma região. | 61 |
| 6.9 | Visualização do estado dos contentores. | 62 |
| 6.10 | Solução inteligente | 63 |

Lista de Tabelas

| | | |
|-----|--|----|
| 3.1 | Especificações da <i>Virtual Machine</i> | 27 |
|-----|--|----|

Acrónimos

| | |
|----------------|---|
| UBI | Universidade da Beira Interior |
| RU | Resíduos Urbanos |
| IoT | <i>Internet of Things</i> |
| SGRI | Sistema de Gestão de Resíduos Inteligente |
| RMSE | <i>Root-Mean-Square Deviation</i> |
| MAPE | <i>Mean Absolute Percentage Error</i> |
| RFID | Identificação por Radiofrequência |
| SQERT | <i>Scope, Quality, Effort, Risk, Time</i> |
| SARIMAX | <i>Seasonal AutoRegressive Integrated Moving Average with eXogenous</i> |
| JSON | <i>JavaScript Object Notation</i> |
| HTTP | <i>Hypertext Transfer Protocol</i> |
| API | <i>Application Programming Interface</i> |
| I2C | <i>Inter-Integrated Circuit</i> |
| SPI | <i>Serial Peripheral Interface</i> |
| GPIO | <i>General Purpose Input/Output</i> |
| ADCs | <i>Analog-to-Digital Converters</i> |
| TCP | <i>Transmission Control Protocol</i> |
| IP | <i>Internet Protocol</i> |
| 3D | <i>Three Dimensions</i> |
| IDE | <i>Integrated Development Environment</i> |
| GPS | <i>Global Positioning System</i> |
| LED | <i>Light Emitting Diode</i> |
| DDoS | <i>Distributed Denial-of-Service</i> |
| DOS | <i>Denial-of-Service</i> |
| API | <i>Application Programming Interface</i> |

HTTP *Hypertext Transfer Protocol*

CRUD *Create, Read, Update, Delete*

Capítulo

1

Introdução

1.1 Enquadramento

No âmbito da unidade curricular Internet das Coisas presente no Mestrado em Engenharia Informática na Universidade da Beira Interior (UBI), surge este projeto como resposta aos desafios urgentes enfrentados na gestão de resíduos, uma área de crescente preocupação devido aos impactos ambientais, riscos para a saúde pública e custos operacionais elevados associados aos métodos tradicionais de recolha.

Este sistema aproveita a tecnologia *Internet of Things* (IoT) para monitorizar e otimizar a recolha de resíduos em tempo real. Os principais componentes incluem sensores ultrassónicos instalados nos contentores de lixo para medir o nível de enchimento, um microcontrolador responsável pela recolha e processamento dos dados dos sensores, e a utilização de conexão, por Wi-Fi, para transmitir os dados para um servidor em nuvem.

O sistema é complementado por uma aplicação móvel para os residentes, permitindo-lhes localizar contentores de lixo próximos, visualizar o seu nível de enchimento e tomar decisões informadas sobre a eliminação dos seus resíduos com base na ocupação dos contentores. Além disso, a plataforma *wen* oferece às empresas de gestão de resíduos uma visão geral de todos os contentores sob a sua responsabilidade, com informações detalhadas sobre o nível de enchimento.

Com este projeto, situado para este âmbito, procura-se explorar os conceitos teóricos subjacentes à IoT, mas também aplicá-los de forma prática e inovadora para resolver problemas do mundo real.

1.2 Motivação

A gestão inadequada de resíduos é um problema premente em todo o mundo, com consequências devastadoras para o meio ambiente, a saúde pública e os custos operacionais.

Em 2022 foram produzidas em Portugal 5,323 milhões de toneladas (t) de Resíduos Urbanos (RU), mais 0,24% do que em 2021, verificando-se que a produção se manteve praticamente inalterada, quando comparada com o ano anterior. [3]

Ao nível da recolha, não se verificam diferenças significativas ao longo dos últimos anos, sendo a recolha indiferenciada o tipo de recolha preferencial para a recolha de resíduos urbanos. [3][4]

Consultando o dado estatístico presente em 1.2 é de notar que a tendência da produção do lixo em Portugal é manter-se constante ou até, nos piores dos cenários, aumentar. Por isso é necessário melhorar o nosso sistema de recolha de forma evitar que haja transbordamento de lixo pelos contentores em zonas que a produção do lixo seja maior que as capacidades dos contentores desta mesma zona.

Já com o dado estatístico presente em 1.2 é de notar que a recolha indiferenciada domina a recolha de resíduos e daí este projeto ser projeto para um contentor onde só será considerado o seu nível de enchimento.

Diante destes cenários, é evidente a necessidade urgente de adotar abordagens para lidar com a gestão de resíduos. O desenvolvimento do sistema presente neste relatório, onde se aproveita a tecnologia IoT para monitorizar e otimizar a recolha de resíduos em tempo real, surge como uma resposta a essa necessidade.

Este projeto não apenas visa abordar os problemas existentes na gestão de resíduos, mas também promover a sustentabilidade ambiental, reduzir custos operacionais e promover uma participação mais ativa da comunidade na gestão dos seus resíduos.

1.3 Objetivos

O objetivo principal deste projeto é desenvolver um Sistema de Gestão de Resíduos Inteligente (SGRI) que aproveite a tecnologia IoT para abordar as ineficiências inerentes aos métodos tradicionais de recolha de resíduos. Especificamente, o projeto visa alcançar os seguintes objetivos:

- **Recolha de Resíduos Eficiente:** Implementação de sensores ultrassônicos em contentores de resíduos para monitorizar com precisão os níveis

de enchimento em tempo real, facilitando rotas de recolha de resíduos oportunas e otimizadas.

- **Análise de Dados Aprimorada:** Utilização de um servidor na nuvem para armazenar e processar dados dos sensores.
- **Interfaces Amigáveis ao Utilizador:** Conceção de aplicação móvel intuitiva para os residentes acederem a informações sobre contentores de resíduos próximos, os seus níveis de enchimento, promovendo a participação ativa da comunidade nos esforços de gestão de resíduos. Além disso, desenvolver uma página *web* que permita que empresas de gestão de resíduos obtenham *insights* sobre padrões de produção de resíduos, otimizem horários de recolha e reduzam custos operacionais.
- **Interoperabilidade com Sistemas Existentes:** Garantir a integração do SGRI com os sistemas de gestão de resíduos existentes.
- **Precisão e Confiabilidade:** Assegurar a precisão e confiabilidade dos sensores ultrassónicos através de procedimentos matemáticos já provados no artigo da secção 2.4.1.
- **Benefícios Ambientais e Económicos:** Em última análise, o projeto visa alcançar a redução da poluição ambiental e dos riscos para a saúde pública associados às práticas ineficientes de gestão de resíduos, ao mesmo tempo que ajuda empresas de gestão de resíduos a otimizarem recolhas.

1.4 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.
2. O segundo capítulo – **Estado da Arte** – apresenta outros projetos já elaboradas na temática de SGRI com IoT, analisa o projeto mais semelhante ao que serve de propósito a este relatório, faz o contraste entre os dois, destacando as diferenças e mais-valias do último.
3. O terceiro capítulo – **Engenharia de Software** – é onde feito um levantamento dos requisitos funcionais e não funcionais do sistema, da aplicação móvel e da aplicação *web*, apresentam-se as tecnologias utilizadas

neste projeto, os casos de uso, diagrama de atividades, a arquitetura do sistema, e por fim, o modelo de dados.

4. O quarto capítulo – **Interoperabilidade** – são apresentados conceitos como interoperabilidade semântica, interoperabilidade sintática, interoperabilidade de dispositivos, interoperabilidade de plataforma e interoperabilidade de rede de forma a expor e em seguida resolver potenciais problemas do sistema.

Capítulo

2

Estado da Arte

2.1 Introdução

Neste capítulo, introduz-se o tema de *Smart Cities* como a área onde este projeto se enquadra e como é perspectivada a gestão de eficiente de resíduos na mesma, realiza-se uma análise de projetos relacionados com a gestão de resíduos que recorrem à tecnologia IoT através dos seus objetivos, metodologias e resultados alcançados. Em seguida, e como complementaridade, faz-se referência e análise a um projeto útil para a elaboração da medição do nível de enchimento do contentor. De seguida, destaca-se o projeto que apresenta maior semelhança com o que é discriminado neste relatório, sendo feita uma análise mais profunda ao mesmo.

Por fim, será efetuado um contraste entre o projeto em estudo e o projeto descrito neste relatório, com o objetivo de destacar as contribuições exclusivas oferecidas pelo projeto em análise neste relatório. Dessa forma, será ressaltado o valor adicional que este traz para o campo da gestão de resíduos.

2.2 *Background: Smart Cities*

Este projeto está inserido num contexto mais amplo das *Smart Cities*.

As *Smart Cities* visam tornar a vida urbana mais confortável, segura e eficiente. Nesse sentido, as tecnologias de Internet das Coisas (IoT) desempenham um papel crucial ao possibilitar a conexão entre o mundo físico e o mundo digital. [5]

Analisando essa área com foco no tema deste projeto, observa-se que a tecnologia IoT facilita a gestão eficiente de resíduos nas cidades. Os contentores inteligentes podem reportar automaticamente os seus níveis de enchi-

mento, permitindo a criação de rotas otimizadas para a recolha de lixo e a redução dos custos operacionais.

A gestão de resíduos em cidades inteligentes tornou-se consideravelmente mais simplificada graças ao IoT. Contentores de resíduos inteligentes, equipados com sensores de níveis de enchimento, transmitem dados às autoridades de gestão de resíduos quando precisam de ser esvaziados. Isso elimina a necessidade de horários fixos de recolha, que podem resultar em viagens desnecessárias e consumo excessivo de combustível. Como resultado, não só ocorre uma redução nos custos operacionais, mas também se minimiza o impacto ambiental da recolha de resíduos. [5]

2.3 Comparação de Projetos de Gestão de Resíduos com IoT

Nesta secção, são apresentados, de forma sintetizada, três projetos que se enquadram no mesmo âmbito do projeto deste relatório, descrevendo os seus objetivos, metodologias e resultados alcançados, proporcionando uma visão do estado da arte mais recente sobre o tema.

2.3.1 Projeto Um

O projeto intitulado *An Intelligent Internet of Things (IoT) based Automatic Dry and Wet Medical Waste Segregation and Management System* [6], visa aprimorar as práticas de gestão de resíduos em instalações médicas. A sua missão é detetar, segregar e gerir eficientemente diversos tipos de resíduos médicos.

Para alcançar esse objetivo, o projeto emprega uma gama de sensores e atuadores que automatizam o processo de segregação de resíduos médicos. Além disso, utiliza um painel de controlo baseado na nuvem para monitorar o estado dos contentores e notificar as autoridades quando estiverem cheios.

Como resultado, espera-se evitar danos causados por resíduos perigosos e aprimorar significativamente as práticas de gestão de resíduos em instalações médicas.

2.3.2 Projeto Dois

O projeto *Design and Development of GIOT based Intelligent Smart Waste Management and Predictive Modelling* [2], pretende monitorar o estado de contentores inteligentes e antecipar anomalias nos dados dos sensores.

Para alcançar esse fim, o projeto emprega uma variedade de sensores, como ultrassónicos, de chama, de gás e acelerómetro, em conjunto com o mi-

crocontrolador *ESP-32* de baixa potência. Os dados provenientes dos sensores são processados pelo microcontrolador e enviados para armazenamento na nuvem, onde são analisados e utilizados para previsões por meio do modelo *Seasonal AutoRegressive Integrated Moving Average with eXogenous* (SARIMAX).

Os resultados do projeto, com um *Root-Mean-Square Deviation* (RMSE) de 0.6145 e um *Mean Absolute Percentage Error* (MAPE) de 3.20, demonstram que as previsões do modelo apresentam pouca variação relativamente aos valores reais. Gráficos que ilustram a previsão do nível do conteúdo dos contentores de lixo ao longo do tempo sustentam a eficácia do modelo de previsão proposto.

2.3.3 Projeto Três

O projeto *The Adoption of an Intelligent Waste Collection System in a Smart City* [7] visa aprimorar a eficiência na colheita de resíduos, reduzir custos e melhorar a estética dos espaços públicos nas cidades.

Para alcançar esses objetivos, o projeto utiliza tecnologias como Identificação por Radiofrequência (RFID) para comunicação entre o *software* e os contentores de lixo, permitindo acesso à localização de determinado contentor, deteção do seu nível de enchimento e controlo do odor.

O protótipo do *software* é desenvolvido utilizando a metodologia *Scope, Quality, Effort, Risk, Time* (SQERT) para visualização e apresentação dos dados, proporcionando uma representação clara das informações recolhidas.

2.3.4 Escolha do Projeto para Análise

Entre os projetos previamente apresentados, optou-se por analisar o projeto *Design and Development of GIOT based Intelligent Smart Waste Management and Predictive Modelling* [2] devido à maior semelhança com o projeto descrito neste relatório. Ambos visam monitorar lixeiras e prever o seu nível de enchimento.

2.4 Análise do Projeto

2.4.1 Visão Geral do Projeto com Ideia Complementar

O projeto analisado nesta secção propõe uma abordagem inovadora na gestão de resíduos, integrando tecnologia IoT e diversos sensores.

O sistema visa monitorar, analisar e prever os níveis de resíduos em tempo real, permitindo intervenções oportunas para manter a limpeza e higiene nas áreas urbanas.

O foco central do projeto está na conceção de um modelo operacional avançado que processe os dados dos sensores, analise os dados e utilize um modelo SARIMAX para previsões multivariadas. Esta metodologia não só aborda os desafios tradicionais da gestão de resíduos, mas também estabelece um novo padrão para sistemas inteligentes de gestão de resíduos com integração de IoT.

2.4.2 Arquitetura do Sistema

A figura representada em 2.1 ilustra a arquitetura IoT composta por quatro componentes-chave que trabalham em sinergia para permitir uma gestão eficiente de resíduos e modelação preditiva.

- **Entrada dos Sensores:** O sistema começa com a entrada de uma variedade de sensores, incluindo sensores ultrassónicos, de chama, de gás e acelerómetro, estrategicamente colocados nos contentores de resíduos para capturar dados em tempo real sobre os níveis de resíduos e parâmetros ambientais.
- **Unidade de Processamento:** Os dados dos sensores são transmitidos para uma unidade de processamento, tipicamente um microcontrolador de baixa potência como o *ESP-32*, que processa e analisa os dados recebidos para extrair informações valiosas e tendências relacionadas com os níveis de resíduos e condições do contentor.
- **Módulo IoT:** Um módulo IoT dedicado é utilizado para facilitar a comunicação e transferência de dados entre os sensores, a unidade de processamento e o sistema de armazenamento na nuvem. Este módulo garante monitorização em tempo real e registo de dados para uma gestão eficaz de resíduos.
- **Módulo de Análise de Dados e Previsão:** O último componente da arquitetura é o módulo de análise de dados e previsão, que utiliza algoritmos avançados como o modelo SARIMAX para modelação preditiva. Este módulo prevê dados anómalos dos sensores e aciona alertas ou notificações para as partes interessadas relevantes para intervenção atempada.

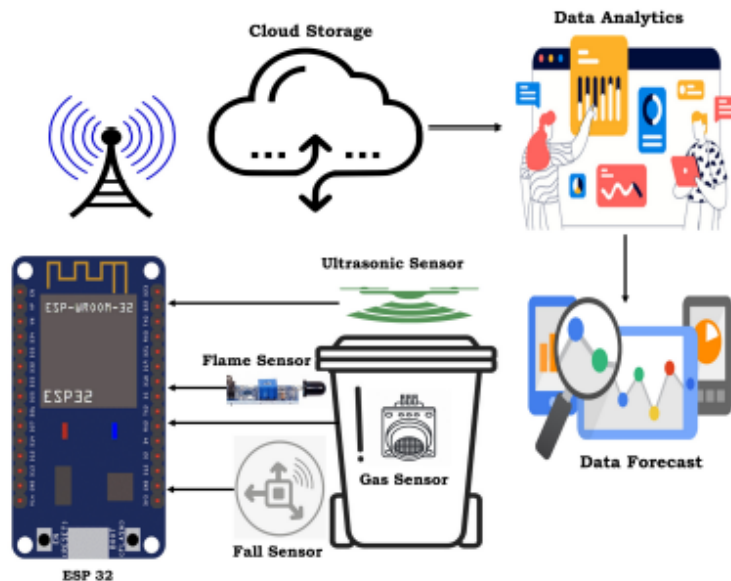


Figura 2.1: Arquitetura do projeto [2]

2.4.3 Integração de Sensores

O projeto incorpora uma variedade de sensores para permitir uma monitorização completa dos contentores de resíduos e das condições ambientais.

Os sensores são responsáveis pela captura de dados em tempo real, na análise dos níveis de resíduos e na deteção de potenciais riscos no sistema de gestão de resíduos.

Em seguida, serão mencionados e descritos os vários tipos de sensores presentes e as suas funcionalidades.

- **Sensor Ultrassónico:** O sensor ultrassónico é utilizado para medir o nível de resíduos nos contentores inteligentes. Com as suas capacidades de medição sem contacto que variam de 2cm a 400cm e uma precisão de até 3mm, este sensor fornece dados precisos sobre o nível de preenchimento dos contentores. As alterações no nível de resíduos são continuamente monitorizadas e transmitidas para a nuvem IoT para análise adicional.
- **Sensor de Chama:** Para abordar o risco de incêndios associados à decomposição dos resíduos, um sensor de chama é integrado no sistema. Este sensor deteta gases perigosos produzidos pela mistura de resíduos durante o dia, que poderiam potencialmente levar a incidentes de in-

cêndio. Ao monitorizar e detetar chamas em tempo real, o sistema pode alertar prontamente as partes interessadas e mitigar eficazmente os riscos de incêndio.

- **Sensor de Gás:** O sensor de gás é utilizado para monitorizar a presença de gases nocivos nos contentores de resíduos. Ao detetar e medir os níveis de gás, o sistema pode identificar potenciais riscos ambientais e garantir a segurança das operações de gestão de resíduos. Este sensor melhora a capacidade do sistema de manter um ambiente limpo e saudável.
- **Sensor Acelerómetro:** O sensor acelerómetro é utilizado para avaliar o movimento físico e a orientação dos contentores de lixo. Ao capturar dados sobre os movimentos e inclinações dos contentores, o sistema pode detetar quaisquer atividades ou perturbações incomuns, fornecendo informações sobre a manipulação dos contentores e potenciais problemas como sobrecarga ou manuseamento inadequado.

2.4.4 Modelo de Funcionamento

A seguinte figura 2.2 ilustra o diagrama de fluxo deste projeto.

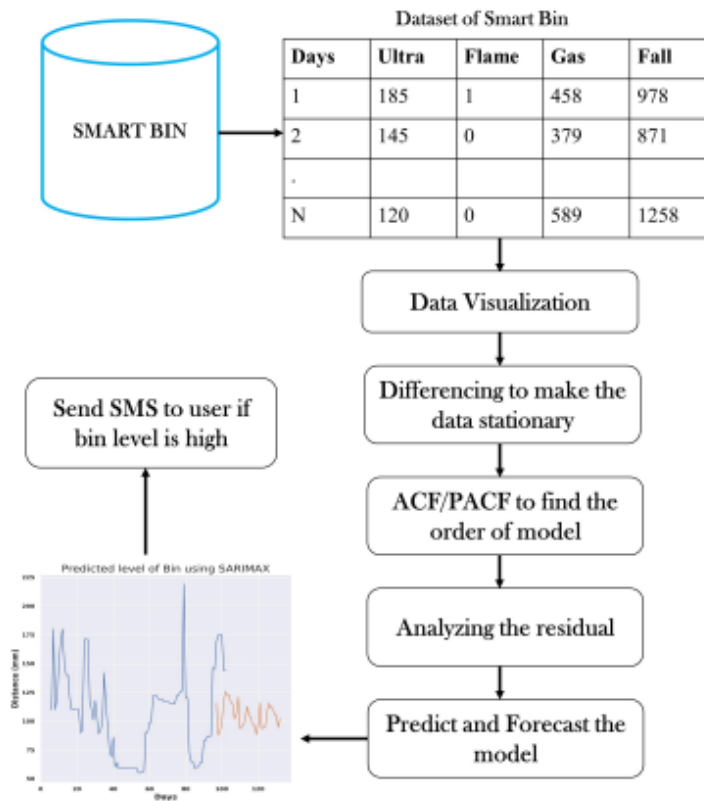


Figura 2.2: Diagrama de fluxo do projeto [2]

Após a análise deste é possível observar as seguintes funcionalidades:

- **Monitorização dos Contentores de Lixo:** O modelo de funcionamento inicia com a monitorização contínua dos contentores de lixo para avaliar o nível de enchimento e as condições ambientais. Sensores incorporados nos contentores capturam dados em tempo real sobre os níveis de resíduos, presença de gases, incidentes de chama e movimentos dos contentores.
- **Processamento e Análise de Dados:** Os dados dos sensores são transmitidos para uma unidade de processamento, ESP-32, onde são sujeitos a uma análise e processamento rigorosos. Os dados são avaliados em

relação a valores de referência predefinidos para identificar anomalias, tendências e potenciais riscos dentro dos contentores de lixo.

- **Integração IoT e Armazenamento na Nuvem:** Os dados processados são integrados de forma transparente no módulo IoT, facilitando a comunicação e transferência de dados entre os sensores, a unidade de processamento e o sistema de armazenamento na nuvem. As leituras de dados em tempo real são registadas numa plataforma de nuvem dedicada, permitindo acesso remoto e monitorização dos níveis de resíduos, parâmetros ambientais e desempenho do sistema.
- **Modelação Preditiva com SARIMAX:** O modelo de funcionamento incorpora técnicas avançadas de previsão, como o modelo SARIMAX, para prever dados anómalos dos sensores e antecipar potenciais problemas. Ao analisar dados históricos e tendências dos sensores, o sistema pode prever os níveis de resíduos, identificar padrões e acionar alertas ou notificações para as partes interessadas tomarem ações atempadas.

Concluí-se assim que este modelo combina dados dos sensores, conectividade IoT e previsão para uma gestão inteligente de resíduos. Isso dá às partes envolvidas a capacidade de tomar decisões bem fundamentadas, melhorar a utilização de recursos e garantir que a gestão de resíduos seja sustentável.

2.4.5 Resultados e Discussões

Os resultados obtidos da fase de implementação e testes do projeto, juntamente com as discussões subsequentes, forneceram *insights* sobre o desempenho do sistema, capacidades de previsão e impacto potencial nas práticas de gestão de resíduos.

- **Avaliação de Métricas de Desempenho:** O projeto avaliou as métricas de desempenho do sistema proposto, incluindo RMSE e MAPE. Estas métricas servem como indicadores da precisão das previsões e da discrepância do modelo preditivo relativamente aos valores reais. Os valores obtidos de RMSE e MAPE, de 0.6145 e 3.20, respetivamente, demonstram a capacidade do sistema de prever os níveis de resíduos com desvios mínimos.
- **Análise de Precisão e Desvio das Previsões:** Os resultados da modelação preditiva usando o algoritmo SARIMAX foram analisados para avaliar a precisão e confiabilidade do processo de previsão. Ao comparar os valores previstos com os dados reais dos sensores, foi avaliada a capacidade do sistema de prever leituras anormais dos sensores e enviar

notificações atempadas às partes interessadas. A discussão centrou-se na eficácia do sistema em antecipar as flutuações nos níveis de resíduos e potenciais riscos. A figura 2.3 mostra como o nível real de enchimento de um contentor se compara com o nível previsto. Foi usado os dados dos primeiros 95 dias para representar o que realmente aconteceu, enquanto os dados dos próximos 7 dias são utilizados para prever o que acontecerá nos 102 dias totais.

- **Eficiência Operacional e Gestão de Recursos:** As discussões giraram em torno da eficiência operacional e dos benefícios de gestão de recursos oferecidos pelo sistema inteligente de gestão de resíduos. Ao possibilitar a monitorização em tempo real, análise de dados e modelação preditiva, o sistema melhora as práticas de gestão de resíduos, otimiza a alocação de recursos e promove estratégias sustentáveis de tratamento de resíduos. As implicações dessas melhorias operacionais nos processos globais de gestão de resíduos foram examinadas minuciosamente.
- **Impacto Ambiental e Considerações de Sustentabilidade:** Os resultados e discussões do projeto também abordaram o impacto ambiental e as considerações de sustentabilidade do sistema implementado. Ao promover a limpeza, gestão eficiente de resíduos e deteção proativa de riscos, o sistema inteligente de gestão de resíduos contribui para criar um ambiente verde e seguro. As discussões destacaram o papel do sistema na promoção da sustentabilidade ambiental e iniciativas de redução de resíduos.

2.4.6 Conclusões do Projeto em Revisão

Em suma, este projeto integra tecnologias IoT, sensores avançados e algoritmos preditivos, oferecendo uma abordagem inovadora para a gestão de resíduos. Ao combinar capacidades de monitorização inteligente com técnicas de modelação preditiva como o SARIMAX obtém-se um sistema eficiente, que visa a sustentabilidade e a responsabilidade ambiental.

2.5 Contraste e Contribuições do Projeto

O projeto delineado neste relatório, centrado no desenvolvimento de um SGRI que recorre à tecnologia IoT, apresenta vários elementos contrastantes e contribuições únicas quando comparado com o projeto analisado na secção 2.4, *Design and Development of GIOT based Intelligent Smart Waste Management and Predictive Modelling* [2].

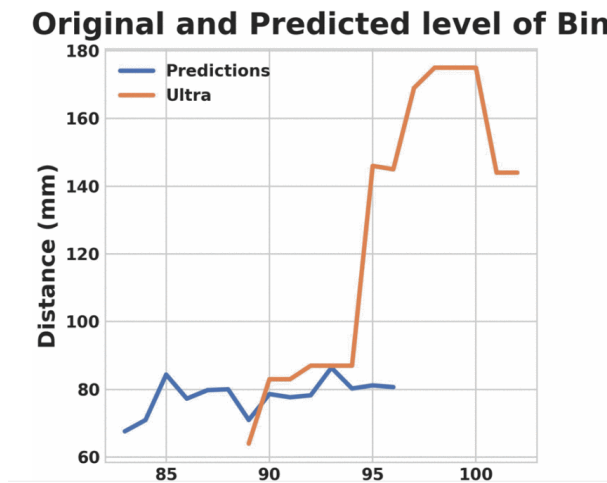


Figura 2.3: Nível original e previsto de lixo.

2.5.0.1 Contrastes:

- **Scope e Foco:**

- O projeto *Design and Development of GIOT based Intelligent Smart Waste Management and Predictive Modelling* enfatiza a modelação preditiva e a previsão em tempo real dos níveis de resíduos, enquanto o projeto que serve o propósito deste relatório coloca uma ênfase mais forte na monitorização em tempo real e otimização dos processos de recolha de resíduos usando tecnologia IoT.

- **Componentes e Funcionalidade:**

- O projeto descrito neste relatório incorpora uma aplicação móvel para os residentes e um *web dashboard* para as empresas de gestão de resíduos, melhorando o envolvimento do utilizador e a eficiência operacional. Em contraste, o outro projeto pode focar-se mais no processamento de dados dos sensores e modelos de previsão.

- **Interação do Utilizador:**

- A inclusão de uma aplicação móvel para os residentes acederem a informações em tempo real e tomarem decisões informadas sobre o descarte de resíduos diferencia este projeto em termos de promoção da participação ativa da comunidade e transparência nas práticas de gestão de resíduos.

2.5.0.2 Contribuições:

- **Envolvimento do Utilizador Melhorado:** Ao disponibilizar uma aplicação móvel aos residentes para interagirem com os contentores de resíduos e tomarem decisões de descarte com base nos níveis de ocupação, este projeto contribui para fomentar um sentido de envolvimento e responsabilidade comunitária nas práticas de gestão de resíduos.
- **Eficiência Operacional:** A integração de um *dashboard* para empresas de gestão de resíduos monitorizarem e otimizarem as operações de recolha com base em dados de sensores em tempo real contribui para melhorar a eficiência operacional, levando a poupanças de custos e uma melhor alocação de recursos.
- **Transparência e Acessibilidade:** O foco do projeto no desenvolvimento de interfaces de utilizador intuitivas tanto para residentes como para empresas de gestão de resíduos contribui para uma maior transparência, acessibilidade e facilidade de utilização nos processos de gestão de resíduos, alinhando-se com o objetivo de promover práticas eficientes de gestão de resíduos urbanos.

2.6 Conclusões

Neste capítulo foi proporcionada uma visão abrangente sobre os projetos de gestão de resíduos com IoT e realizou-se uma análise detalhada do projeto "*Design and Development of GIOT based Intelligent Smart Waste Management and Predictive Modelling*"[2]. Através desta revisão, foi possível identificar pontos de contraste e contribuições significativas do projeto descrito neste relatório em relação ao projeto referenciado em 2.4.

Nos capítulos seguintes serão apresentados os requisitos, bem como as tecnologias usadas para os desenvolver.

2.7 Projeto Mapeamento *Three Dimensions* (3D): *3D Surface Mapping using Ultrasonic Sensors*[1]

2.7.1 Visão Geral do Projeto

O mapeamento de contorno de superfície é uma técnica gráfica utilizada para representar superfícies tridimensionais. O artigo *3D Surface Mapping using*

Ultrasonic Sensors[1] apresenta um protótipo para obter gráficos de contorno de superfície usando um robô equipado com três sensores ultrassônicos.

Os sensores operam como um *scanner* ultrassônico 3D, explorando o ambiente e mapeando-o num plano 3D. Os sensores do robô, juntamente com o *Servo Motor*, dão *scan* os arredores dentro de uma faixa de 180 graus, obtendo informações em coordenadas cilíndricas. Esses dados são então transferidos para um computador via porta *serial Arduino* e salvos num arquivo de texto usando o *Processing*.

O *Matlab* lê o ficheiro de texto, converte as coordenadas para forma retangular e as representa-as graficamente plano 3D para obter um gráfico de dispersão dos arredores no mesmo espaço dimensional. Um processo chamado Reconstrução converte o gráfico de dispersão obtido num mapa de contorno de superfície dos objetos. Este sistema permite efetivamente representação gráfica de superfícies planares em 3D.

2.7.2 Arquitetura do Sistema

O sistema conta com três sensores ultrassônicos montados num único suporte verticalmente, um acima do outro, com espaçamento igual entre eles. Este suporte inteiro é montado num único *Servo Motor*, como mostrado na figura 6.3. O *Servo Motor* pode varrer um ângulo de 0 a 180 graus. Os sensores ultrassônicos, juntamente com um *Servo Motor*, são conectados a uma placa *Arduino Uno R3*.

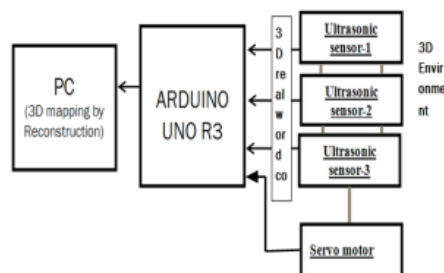


Figura 2.4: Esquema do *hardware* [1]

2.7.3 Modelo de Funcionamento

Um sensor ultrassônico mede a distância usando a diferença de tempo entre o envio do sinal e a recepção do mesmo. A distância, assim, medida por cada sensor ultrassônico corresponde a:

- r (em cm) - distância em centímetros.
- Φ (em graus) - ângulo varrido pelo *Servo Motor*.
- z é a altura de cada sensor ultrassônico medida a partir da base do suporte

Assim, cada sensor ultrassônico em cada etapa fornece um conjunto de coordenadas do mundo real em 3D no formato de coordenadas cilíndricas (r , Φ , z). O *Servo Motor* varre de 0 a 180 graus em 12 etapas com um tamanho de passo de 15 graus. Em cada etapa, os três sensores ultrassônicos efetuam *scan* ao ambiente 3D sendo obtidos três conjuntos de coordenadas cilíndricas do mundo real a partir dos três sensores ultrassônicos. Obtém-se, no total, um conjunto de 36 coordenadas cilíndricas do mundo real. Os sensores ultrassônicos, juntamente com o *Servo Motor*, são conectados à placa *Arduino*.

Um programa realizado no *Integrated Development Environment* (IDE) do *Arduino* (cujo fluxograma é mostrado na figura 2.5) controla o movimento do *servo motor* e recebe os dados na forma de coordenadas do mundo real em 3D dos sensores através da porta COM. Portanto, os dados referentes ao ambiente 3D na forma de coordenadas são transferidos para o *laptop*. O conjunto dessas coordenadas é transferido da janela serial para um ficheiro de texto usando o *Processing*.

O ficheiro de texto é então lido por um programa feito no *Matlab* que converte as coordenadas cilíndricas (r , Φ , z) em coordenadas retangulares, isto é, (x , y , z), e esse conjunto de 36 coordenadas retangulares do mundo real em 3D é reconstruído para obter o mapa de contorno de superfície do ambiente 3D obtido pelo *scan*.

As coordenadas do ficheiro de texto são convertidas para as coordenadas retangulares e lidas na forma de matriz pelo *Matlab*. Esses pontos são representados graficamente no plano tridimensional para produzir um gráfico de dispersão do ambiente ao qual foi dado *scan*. Uma vez que os pontos são representados graficamente, as superfícies planares juntam os pontos adjacentes para produzir o mapa de superfície.

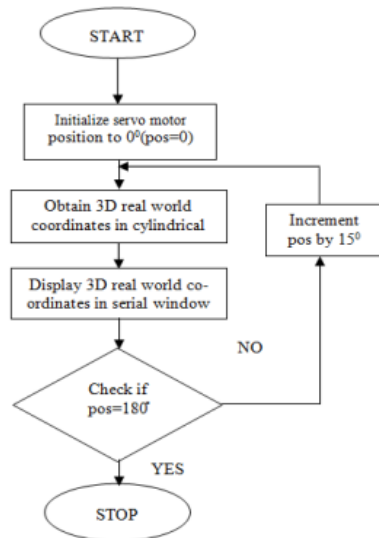


Figura 2.5: Fluxograma do programa [1]

2.7.4 Resultados e Conclusões

Através da experiência realizada pelos autores do projeto [1] obtiveram-se os seguintes planos:

- um plano 3D;
- representação 2D da vista *TOP-DOWN* do mesmo plano imagem 2.6;

Pareceu-nos satisfatório o resultado obtido pelo projeto mencionado acima, pelo qual após análise detalhada realizado por nós foi decidido incorporar o projeto apresentado e adaptá-lo para atender aos objetivos específicos do nosso trabalho exposto neste relatório.

Apesar da proposta de mapeamento tridimensional oferecer uma base sólida para nossa abordagem, faremos algumas adaptações para atender às necessidades do nosso projeto, nomeadamente em vez da vista *Top-Down* queremos converter para uma vista lateral para melhor entendimento do nível de enchimento e forma do mesmo.

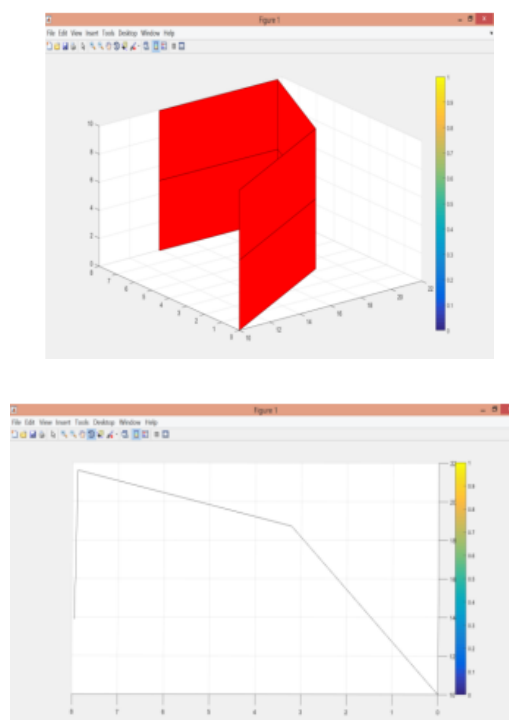


Figura 2.6: Conversão para 2D com vista TOP-DOWN.

Capítulo

3

Engenharia de Software e Arquitetura do Sistema

3.1 Introdução

Neste capítulo, vamos explorar os requisitos funcionais e não funcionais deste projeto, juntamente com as tecnologias empregadas. Em relação às tecnologias utilizadas, faremos uma análise detalhada da aplicação móvel, do painel de controle (dashboard) e do sistema em geral, utilizando diagramas de casos de uso e diagramas de atividades.

Além disso, iremos discutir a estrutura da base de dados e a metodologia para armazenamento dos dados.

3.2 Análise de Requisitos

A fase de análise de requisitos é realizada para compreender as necessidades funcionais e não funcionais de um sistema. Envolve a identificação e documentação das funcionalidades e características necessárias para cumprir os objetivos do projeto. Esta fase irá identificar os requisitos funcionais e não funcionais da aplicação móvel e web do SGRI.

3.2.1 Requisitos Funcionais

Os requisitos funcionais do Sistema de Gestão Inteligente de Resíduos (SGRI) englobam operações fundamentais para a monitorização eficiente de caixotes de lixo, disponibilização de dados em tempo real e otimização das ope-

rações de recolha, distribuídos entre os requisitos da aplicação móvel e da Dashboard.

3.2.1.1 Requisitos Funcionais do Sistema

1. **RF01:** O sistema deve determinar o nível de enchimento dos contentores de lixo utilizando sensores ultrassónicos.
2. **RF02:** O sistema deve monitorizar a localização de cada contentor de lixo através de um módulo *Global Positioning System* (GPS).
3. **RF03:** O sistema deve enviar os dados dos sensores para um servidor na nuvem através de *Wi-Fi*.
4. **RF04:** O sistema deve apresentar o nível de enchimento num ecrã *Light Emitting Diode* (LED) oito por oito com diferentes cores: verde para "Baixo", amarelo para "Moderado" e vermelho para "Alto".
5. **RF05:** A aplicação móvel deve permitir aos utilizadores localizar contentores de lixo próximos, visualizar os seus níveis de enchimento e seleccionar onde depositar o seu lixo com base na ocupação, fornecer assistência inteligente, sugerindo a melhor opção de descarte com base na localização do utilizador e nos níveis de enchimento dos contentores de lixo próximos.
6. **RF06:** O *dashboard* para empresas de gestão de resíduos deve fornecer uma visão geral de todos os contentores de lixo sob a responsabilidade da empresa, com informações detalhadas sobre o nível de enchimento, localização, e gráficos que ilustram estatísticas pertinentes à empresa. Deve também permitir a gestão dos contentores de lixo, incluindo a adição, edição ou eliminação de contentores.
7. **RF07:** O sistema deve determinar o nível de enchimento dos contentores de lixo a cada 15 minutos para fornecer dados em tempo real.
8. **RF08:** O sistema deve utilizar um *Serve Motor* para permitir o movimento de varrimento dos sensores ultrassónicos, obtendo coordenadas 3D a partir de diferentes ângulos para fins de mapeamento.
9. **RF09:** O sistema deve utilizar um controlador de carga para regular a tensão e a corrente provenientes dos painéis solares.
10. **RF10:** O sistema deve utilizar um conversor DC-DC para converter a tensão da bateria na tensão necessária para os sensores, microcontrolador e módulo GPS.

3.2.1.2 Requisitos Funcionais da Aplicação Móvel

A aplicação móvel serve como a principal interface para os **residentes** interagirem com o SGRI. Os seguintes requisitos fornecem funcionalidades aos utilizadores para acederem a informações em tempo real sobre os caixotes do lixo.

1. **RF01:** A aplicação deve solicitar ao utilizador que escolha a sua região atual numa lista de regiões disponíveis quando a aplicação for aberta pela primeira vez.
2. **RF02:** Ao selecionar a região, a aplicação deve exibir uma lista de contentores na região escolhida na página principal.
3. **RF03:** Cada entrada de contentor na lista deve incluir a percentagem de enchimento e a distância entre a localização do contentor e a localização atual do utilizador.
4. **RF04:** A página principal deve incluir um ícone de ajuda para sugerir a melhor opção de descarte com base na localização do utilizador e nos níveis de preenchimento dos contentores do lixo próximos.
5. **RF05:** O menu principal deve incluir um ícone ou opção para alterar a região para outra.
6. **RF06:** A aplicação deve solicitar ao utilizador que escolha uma região diferente na lista disponível quando a opção de alterar a região for selecionada.
7. **RF07:** A aplicação deve lembrar da última região selecionada e navegar automaticamente o utilizador para a última região selecionada quando a aplicação for aberta novamente.

3.2.1.3 Requisitos Funcionais da Dashboard

o *dashboard* serve como a principal interface para a empresa de recolha de lixo interagir com o SGRI. Os seguintes requisitos fornecem funcionalidades aos utilizadores para acederem a informações em tempo real sobre os caixotes do lixo.

- **RF1: Controlo sobre o Sistema**
 - Adicionar/Remover um novo contentor de lixo ao sistema.
 - Alterar informações dos contentores de lixo.

- Adicionar/Remover administradores do sistema.
- Editar utilizadores do sistema.
- **RF2: Visualização em Tempo Real do Estado dos Contentores de Lixo**
 - Mostrar o nível de enchimento dos contentores de lixo em tempo real numa interface.
 - Atualizar dinamicamente o estado dos contentores à medida que os dados dos sensores mudam.
- **RF3: Detalhes dos Contentores de Lixo**
 - Fornecer informações detalhadas sobre cada contentor de lixo, incluindo localização e nível de enchimento.
 - Mostrar dados pertinentes para análise dos administradores.
- **RF4: Notificações**
 - Enviar notificações *push* aos funcionários sobre atualizações críticas, como contentores próximos atingindo a capacidade máxima ou alterações na localização de determinado contentor.

3.2.2 Requisitos Não Funcionais

Os requisitos não funcionais definem o desempenho, usabilidade e fiabilidade do sistema para além das funcionalidades principais, garantindo uma operação eficiente e uma experiência de utilizador sem problemas.

3.2.2.1 Requisitos Não Funcionais do Sistema

1. **RNF01:** O sistema deve ser alimentado por energia solar para reduzir os custos operacionais e ser sustentável.
2. **RNF02:** O sistema deve ser capaz de operar em todas as condições climáticas.
3. **RNF03:** O sistema deve ser capaz de operar continuamente, com a energia armazenada durante o dia usada para alimentar o sistema à noite.
4. **RNF04:** O sistema deve ser capaz de transmitir dados para um servidor em nuvem via *Wi-Fi*.
5. **RNF05:** O sistema deve ser capaz de exibir o nível de preenchimento numa matriz LED oito por oito em diferentes cores.

6. **RNF06:** O aplicativo móvel deve ser compatível com os sistemas operacionais *Android* e *iOS*.
7. **RNF07:** O *dashboard* deve ser compatível com os navegadores mais recentes, incluindo Google Chrome, Mozilla Firefox, Safari e Microsoft Edge.
8. **RNF08:** O sistema deve garantir a privacidade dos utilizadores, não partilhando as suas informações pessoais sem consentimento.
9. **RNF09:** O sistema deve ser resistente a vandalismo e roubo.
10. **RNF10:** O sistema deve ser fácil de instalar e manter.

3.2.2.2 Requisitos Não Funcionais da Aplicação Móvel

1. **RNF01:** A aplicação deve ter uma interface amigável ao utilizador, facilitando a navegação e compreensão para os utilizadores.
2. **RNF02:** A aplicação deve ser responsiva e rápida, garantindo uma experiência de utilização suave.
3. **RNF03:** A aplicação deve ser fiável, fornecendo informações precisas e atualizadas sobre os níveis de preenchimento e localizações dos contentores de resíduos.
4. **RNF04:** A aplicação deve ser segura, protegendo os dados e a privacidade dos utilizadores.
5. **RNF05:** A aplicação deve ser compatível com diferentes sistemas operativos móveis como o *iOS* e o *Android*.
6. **RNF06:** A aplicação deve ser escalável, permitindo a adição de mais regiões e contentores no futuro.
7. **RNF07:** A aplicação deve ser mantível, permitindo atualizações e correções de erros de forma fácil.
8. **RNF08:** A aplicação deve fornecer mensagens de erro claras e úteis para orientar os utilizadores.
9. **RNF09:** A aplicação deve ser capaz de lidar com vários utilizadores ao mesmo tempo, sem degradação do desempenho.
10. **RNF13:** A aplicação deve ser capaz de lembrar a configuração do utilizador, isto é, a última região selecionada.

11. **RNF14:** A aplicação deve fornecer uma apresentação visualmente apelativa e intuitiva dos níveis de preenchimento e localizações dos contentores de resíduos.

3.2.2.3 Requisitos Não Funcionais *Dashboard*

Os requisitos não funcionais específicos do *dashboard* focam-se em aspetos como o desempenho, a usabilidade e a segurança.

- **RNF1: Desempenho**

- O *dashboard* deve ter um tempo de resposta mínimo ao obter e exibir dados em tempo real dos caixotes do lixo.
- Esta deve conseguir lidar eficientemente com pedidos de utilizadores concorrentes sem degradação significativa de desempenho.

- **RNF2: Fiabilidade**

- Esta deve ter mecanismos para lidar com erros de forma ágil e recuperar de falhas inesperadas sem perda de dados.

- **RNF3: Segurança**

- O *dashboard* deve empregar protocolos de encriptação para garantir a segurança da transmissão de dados entre o *dashboard* e o servidor na nuvem.
- O *dashboard* deve empregar protocolos de segurança para garantir apenas o acesso a pessoal autorizado a aplicação.

- **RNF4: Escalabilidade**

- A arquitetura do *dashboard* deve ser escalável para acomodar potenciais aumentos na base de utilizadores e volume de dados ao longo do tempo.
- Deve ser capaz de lidar com funcionalidades adicionais ou melhorias sem alterações arquitetónicas significativas.

3.3 Tecnologias Utilizadas

Nesta secção serão apresentadas as tecnologias utilizadas, no sistema, na aplicação móvel, no *dashboard* e neste documento, respetivamente. A secção 3.6, elabora mais aprofundadamente os usos destas tecnologias e como elas se interligam.

3.3.1 Tecnologias Utilizadas no Contentor

Nos contentores do lixo, onde o sistema será aplicado, é onde se irão situar os diversos sensores e componentes do mesmo.

- Material aplicado nos contentores:
 - Três sensores ultrassónicos (*HC-SR04*);
 - Um *ESP32*;
 - Um *Servo Motor SG90*;
 - Uma fonte de alimentação, gerada por:
 - * Um painel solar;
 - * Controlador de carga;
 - * Bateria;
 - * Conversor DC-DC;
 - Um Módulo GPS;
 - Uma Matriz LED 8x8;

As tecnologias utilizadas para o envolvimento deste material como um todo foram:

- *Python*;
- *Arduino IDE*;

3.3.2 Tecnologias Utilizadas no Servidor

O servidor é o responsável por armazenar os dados vindos dos contentores do lixo.

Este servidor *web* e a correspondente base de dados serão *hosted* numa *virtual machine* criada na plataforma *oceanos* [?].

Na tabela 3.1 estão representada as especificações da *virtual machine*.

Tabela 3.1: Especificações da *Virtual Machine*

| Componente | Valor |
|------------------------|----------------|
| CPUs | 4 |
| RAM (MB) | 4096 |
| Disco do Sistema (GB) | 40 |
| Nome da Imagem | Ubuntu Desktop |
| Tamanho da Imagem (GB) | 5.11 GB |

As tecnologias utilizadas no servidor e na base de dados foram:

- *Node.js*;
- *Express.js*;
- *MySQL*;
- *CSS*;
- *HTML*;
- JavaScript Object Notation (*JSON*);
- *APIs RESTful*;

3.3.3 Aplicação Móvel

s1 A aplicação terá as seguintes tecnologias envolvidas no seu desenvolvimento:

- *Android Studio IDE*;
- *Java*;

3.3.4 Dashboard

A dashboard sera desenvolvida em springboot e sera baseada numa interface web propria com sistema de login, e uma interface com os contentores, as suas informações e graficos pertinentes para analise.

- *Servidor Web (Apache2)*;
- *okeanus*

3.3.5 Relatório

O relatório foi escrito utilizando *LaTeX*. Para facilitar a colaboração em tempo real, utilizou-se a plataforma *Overleaf*. O Overleaf é uma ferramenta baseada em *web* que permite que múltiplos autores trabalhem no mesmo documento simultaneamente, agilizando assim o processo de escrita e desenvolvimento do relatório.

3.4 Casos de Uso

O diagrama de caso de uso resume os detalhes dos utilizadores do seu sistema (também conhecidos como atores) e as interações entre eles. [8]

Esta secção será dividida em duas sub-secções, a primeira onde se irá apresentar o diagrama de casos de uso para a aplicação móvel e na segunda o diagrama de casos de uso para a aplicação *web (dashboard)*

3.4.1 Casos de Uso da Aplicação Móvel

Na figura 3.1 está ilustrado o ator da aplicação móvel (o residente), e as opções que pode tomar ao iniciar a aplicação. Com este diagrama é possível observar que o utilizador pode seleccionar uma região, mudar a mesma, visualizar a lista de contentores dessa mesma região, pedir ajuda para tomar a sua decisão de onde depositar o lixo e fechar aplicação, tendo em consideração que quando a voltar a abrir, a aplicação já não lhe peça para escolher novamente.

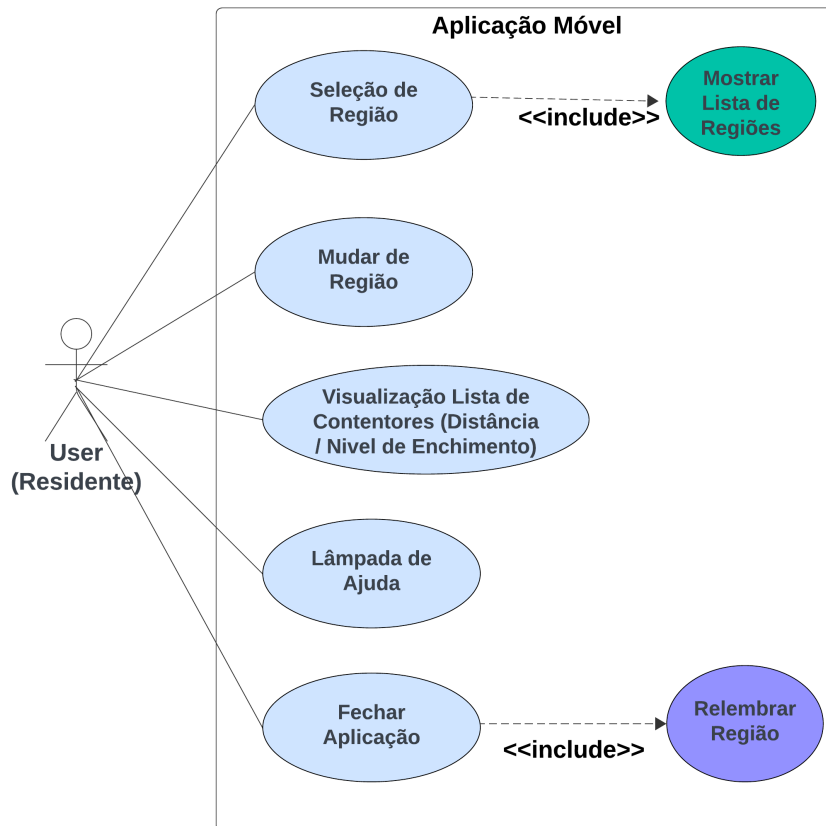


Figura 3.1: Diagrama de casos de uso da aplicação móvel

3.4.2 Casos de Uso da Dashboard

A dashboard inicia-se numa página de login, o utilizador introduz as suas credenciais e dependendo do tipo de utilizador (admin ou funcionario) será redirecionado para outra página web.

O utilizador admin pode adicionar, remover e editar caixotes de lixo e regiões, para além disso pode também adicionar e/ou remover utilizadores do sistema. Por último, este pode ainda verificar vários tipos de gráficos e dados para análise interna da empresa e ou emétricas de qualidade da mesma. O utilizador normal (funcionario) após o login apenas vai para a página onde só consegue ver o nível de enchimento dos caixotes do lixo.

No final a dashboard comunica com a base de dados para guardar as alterações feitas pelo administrador.

Segue abaixo o diagrama de casos de uso da dashboard 3.4.

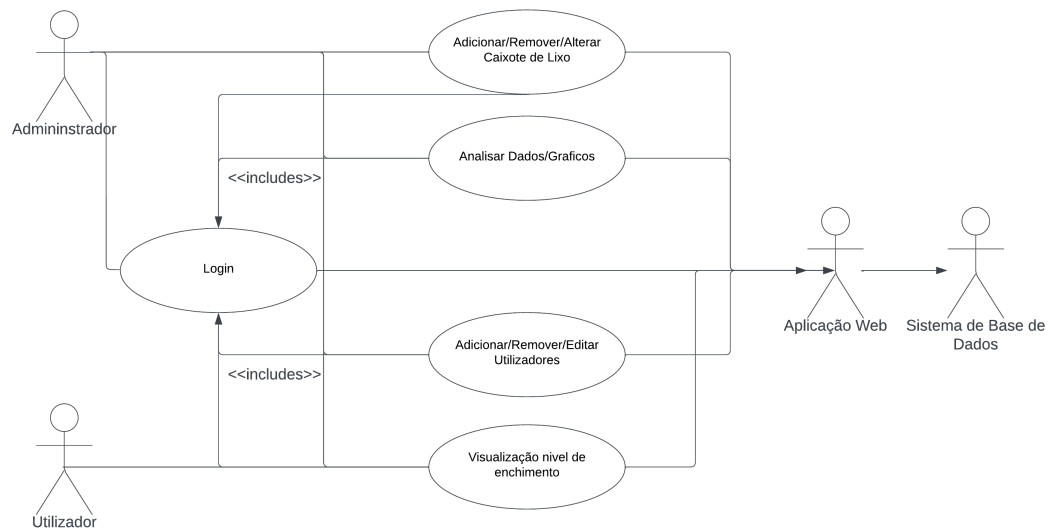


Figura 3.2: Diagrama de casos de uso da Dashboard.

3.5 Diagrama de Atividades

Um diagrama de atividade fornece uma visualização do comportamento de um sistema descrevendo a sequência de ações num processo.

Os diagramas de atividades são semelhantes a fluxogramas porque mostram o fluxo entre as ações numa atividade; no entanto, os diagramas de atividades também podem mostrar fluxos paralelos ou simultâneos e fluxos alternativos. [9]

Nesta secção serão apresentados os diagramas de atividades da aplicação móvel e da aplicação *web*, respetivamente.

3.5.1 Diagrama de Atividades da Aplicação Móvel

Na figura 3.3 está ilustrada o fluxo de atividades da aplicação móvel usada pelos residentes para selecionar e visualizar contentores de diferentes regiões. Segue uma breve explicação deste fluxo:

- **Abertura da Aplicação:**
 - O utilizador inicia o processo abrindo a aplicação móvel.
- **Verificação da Primeira Abertura:**

- A aplicação verifica se é a primeira vez que o utilizador a está a abrir.
- Se sim, a aplicação pede ao utilizador para escolher uma região.
- Se não, a aplicação recupera a última região seleccionada das preferências partilhadas e carrega os contentores correspondentes.

- **Seleção de Região:**

- O utilizador escolhe uma região entre as opções disponíveis.

- **Visualização de Contentores:**

- A aplicação guarda a região seleccionada nas preferências partilhadas.
 - Os contentores correspondentes à região seleccionada são recuperados da base de dados.
 - A lista de contentores, juntamente com os níveis de preenchimento e distâncias, é apresentada ao utilizador.
 - Um ícone de lâmpada é exibido para indicar a melhor opção de eliminação com base na localização do utilizador e nos níveis de preenchimento dos contentores próximos.

- **Pedido de Mudança de Região:**

- Se o utilizador desejar mudar de região, ele pode solicitar essa alteração.

- **Escolha de Nova Região:**

- A aplicação pede ao utilizador para escolher uma região diferente.
 - A nova região seleccionada é guardada nas preferências partilhadas.
 - Os contentores correspondentes à nova região são recuperados da base de dados e apresentados ao utilizador.

- **Encerramento da Aplicação:**

- Quando o utilizador fecha a aplicação, a última região seleccionada é guardada nas preferências partilhadas para ser recuperada na próxima vez que a aplicação for aberta.

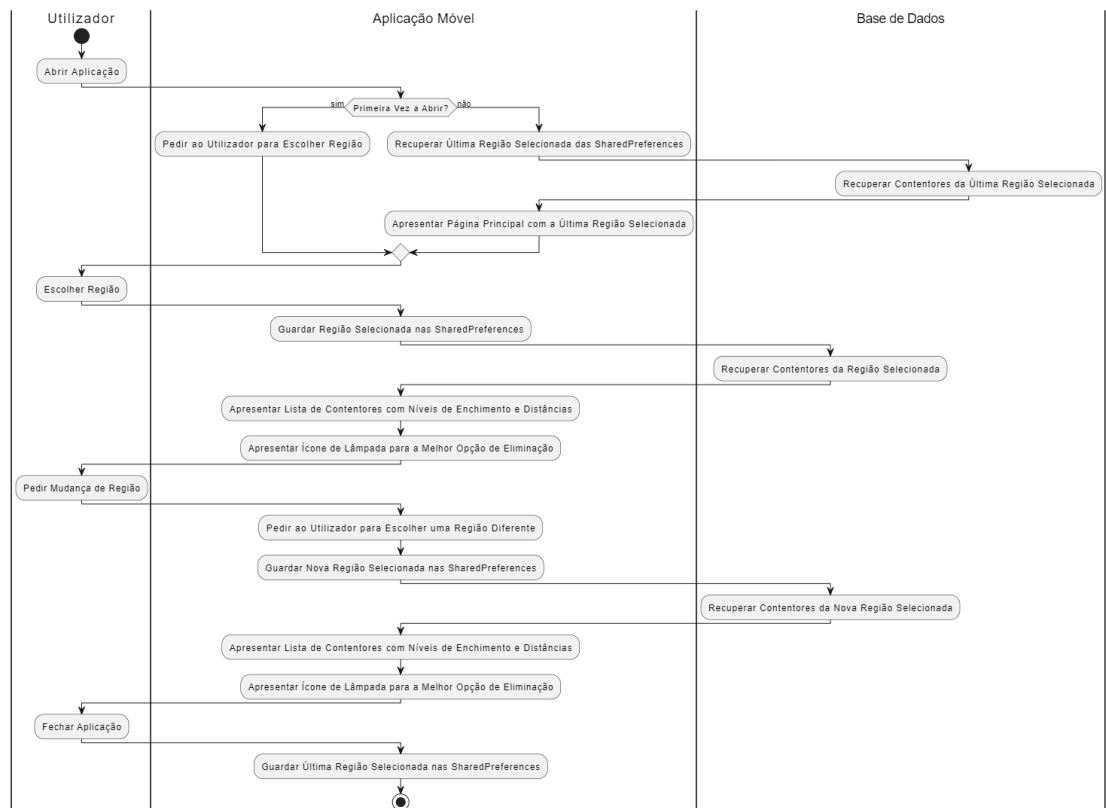


Figura 3.3: Diagrama de atividades da aplicação móvel

3.5.2 Diagrama de Atividades da dashboard

Na figura 3.4 está ilustrada o fluxo de atividades da dashboard usada pelos funcionarios da empresa. Segue uma breve explicação deste fluxo:

- **Abertura da Aplicação:**
 - O utilizador abre a dashboard através do navegador.
- **Login:**
 - O utilizador faz login.
 - A dashboard verifica o tipo de utilizador.
 - Se administrador, a dashboard abre em modo de administrador.
 - Se funcionario, a dashboard abre em modo limitado.
- **Funcionario:**

- O funcionario pode visualizar o nivel de enchimentos dos caixotes de lixo.

- **Administrador:**

- Pode ver graficos e dados para analise de varios aspetos.
- Pode editar informações e/ou adicionar/remover contentores de lixo, regiões e utilizadores.

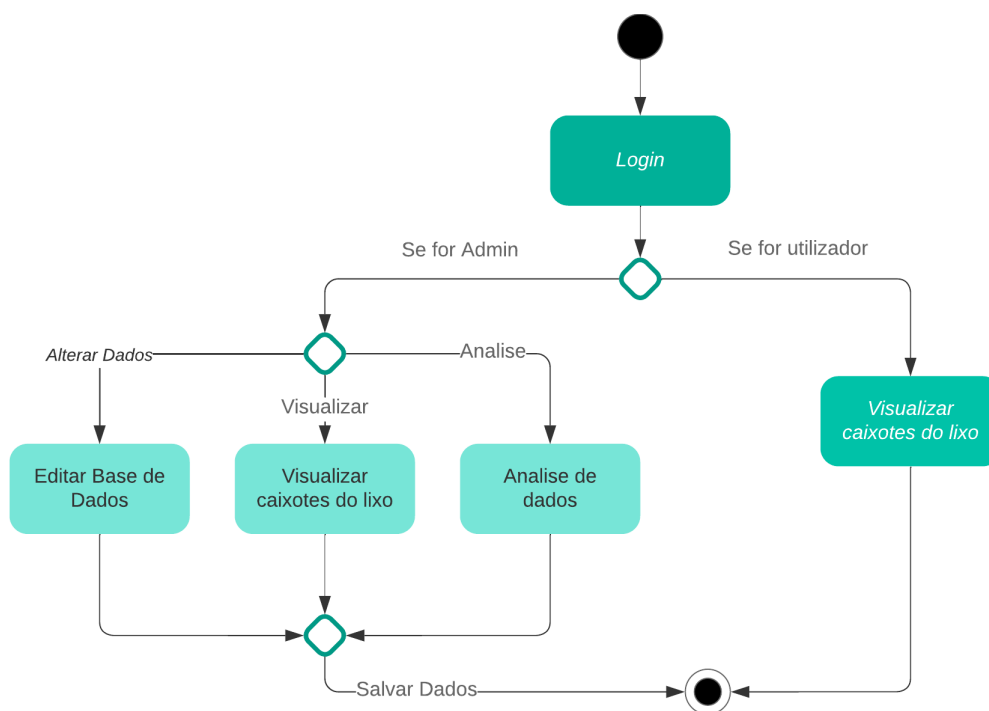


Figura 3.4: Diagram de Atividades da Dashboard.

3.6 Arquitetura do Sistema IoT

A arquitetura do sistema descreve como os diferentes componentes se relacionam para formar um todo.

A figura 3.5 ilustra a estrutura das diferentes componentes utilizadas e que compõem a arquitetura do sistema a ser implementado.

A leitura desta arquitetura deve começar no **Sensor Node**, onde será feita a recolha dos dados. Neste nodo estão três sensores ultrassônicos com um *servo motor*, ligados a um microcontrolador, *ESP32* com disponibilidade para fazer ligações sem fios.

Para armazenar e fornecer energia de forma renovável aos componentes do nodo anterior, vem o **Energy Node**. Este é constituído por painéis solares que capturam a energia solar, um controlador de carga para transportar o fluxo de eletricidade dos painéis solares para a bateria, uma bateria onde é armazenada a eletricidade gerada pelos painéis solares e o conversor CC/CC que converte a tensão da bateria para a tensão necessária para alimentar o nodo anterior.

Através do *Gateway* é feita a ligação do *ESP32* com o **servidor remoto** que contém uma aplicação *Node.js* desenvolvida em *Express.js*, e é apoiada por uma *REST API*, que permite ao microcontrolador enviar os dados recolhidos e receber informação que exibirá na Matriz LED 8x8. Esta aplicação será também responsável por comunicar com a base de dados através do *Cloud Firestore*. O **Laptop** irá comunicar com a base de dados, nos dois sentidos, na consulta de dados e na entrega de novos dados que surgirão de novo processamento (computacionalmente exigente para um microcontrolador).

No **UI Node**, a aplicação móvel irá somente fazer consultas aos dados armazenados. Já a aplicação web (*dashboard*) poderá consultar mas também fazer alterações aos mesmos.

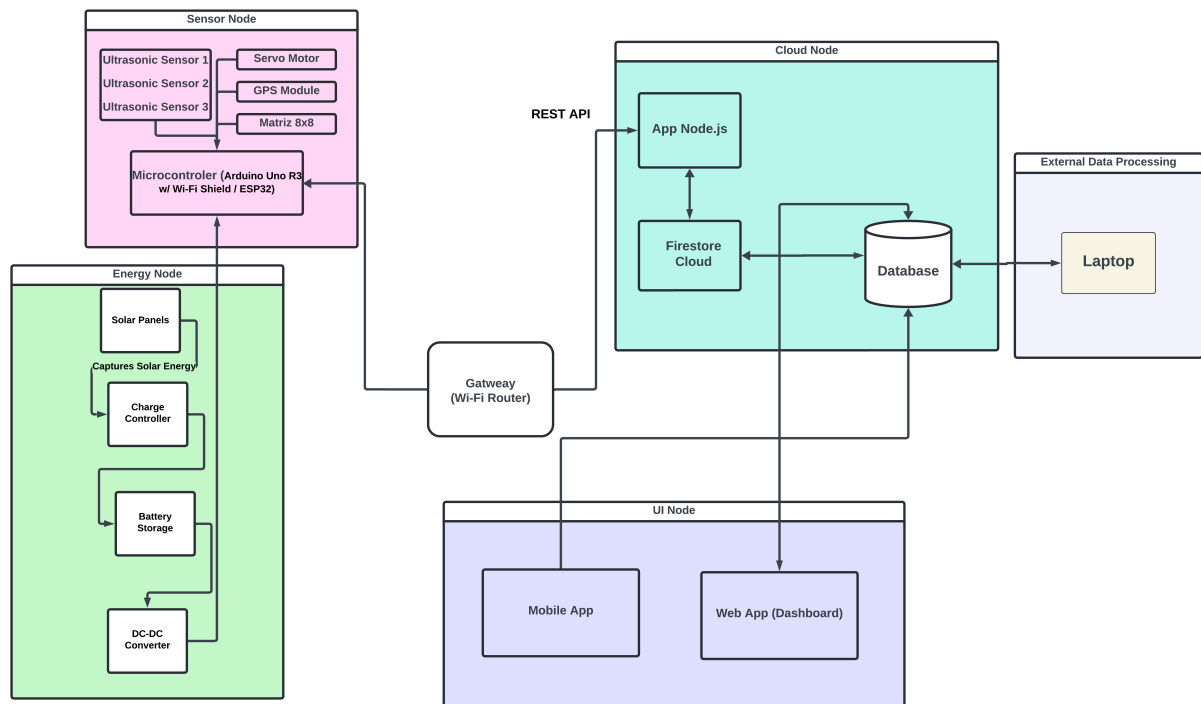


Figura 3.5: Arquitetura do Sistema IoT

3.7 Base de Dados

A base de dados implementada neste projeto é relacional. A figura 3.6 representa o modelo proposto. Este é composto por cinco tabelas: *Companie*, *Regions*, *Containers*, *SensorData*, e *Final_Stats*.

1. **Companie:** Esta tabela armazena informações sobre a empresa de gestão de resíduos. Cada funcionário é identificada por um *ID_Employee* único. Outros detalhes incluem o email da funcionário, password e role.
2. **Regions:** Esta tabela contém informações sobre as diferentes regiões onde o sistema de gestão de resíduos opera. Cada região é identificada por um *ID_Região* único e possui um *nome_região*.
3. **Containers:** Esta tabela contém informações sobre os contentores de resíduos. Cada contentor é identificado por um *ID_Container* único e está associado a um *id_region*. A localização geográfica de cada contentor é armazenada como latitude e longitude.

4. **SensorData:** Esta tabela armazena os dados recolhidos dos sensores instalados nos contentores. Cada registo de dados é identificado por um ID_Record único e está associado a um ID_Container. Os dados incluem medições de três sensores (s1_r, s1_o, s2_r, s2_o, s3_r, s3_o) e um timestamp. Os sensores ultrassónicos registam os dados em coordenadas cilíndricas: r (em cm), Φ (em graus) - ângulo varrido pelo servo motor dentro de uma faixa de 180° , incrementando a posição de 15° em 15° . O z que é a altura de cada sensor ultrassónico medida a partir do fundo do suporte, como esta é constante optou-se por não incluir este como atributo da tabela.
5. **Final_Stats:** Esta tabela armazena os resultados processados a partir dos dados dos sensores. Cada resultado é identificado por um ID_Result único e está associado a um ID_Container. Os resultados incluem o fill_level do contentor e um timestamp.

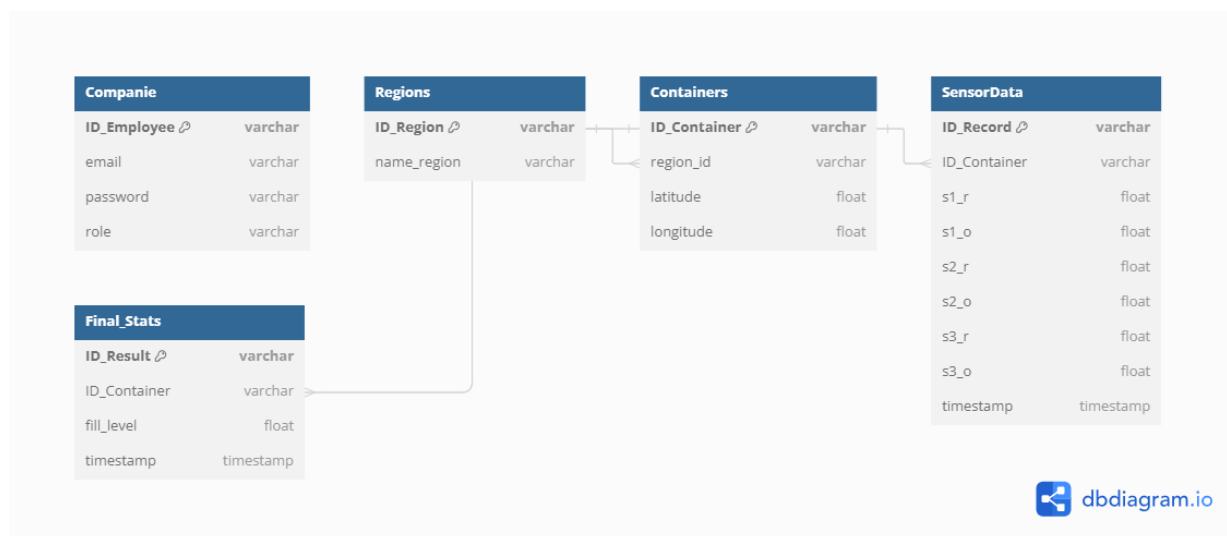


Figura 3.6: Modelo da Base de Dados

3.8 Conclusões

Neste capítulo, discutiu-se as tecnologias implementadas, juntamente com a estratégia para fazê-lo. Detalhou-se o projeto, referindo os métodos de implementação previstos, os requisitos funcionais e não funcionais do sistema, e as componentes do *dashboard* e da aplicação móvel. Utilizou-se diagramas de

atividades, casos de uso e uma representação da arquitetura do sistema para oferecer uma visão mais completa do projeto.

Capítulo

4

Interoperabilidade

4.1 Introdução

Este capítulo dedica-se à discussão da interoperabilidade dentro do contexto do sistema descrito. Serão expostos os desafios relacionados a esta temática e a devida proposta de resolução.

4.2 Interoperabilidade Semântica

No contexto do SGRI, a interoperabilidade semântica assegura que todos os componentes do sistema compreendam e interpretem os dados da mesma forma. Isto é crucial para que o sistema funcione corretamente e de forma eficiente.

- **Modelo de Dados Comum:** Os sensores ultrassônicos, o microcontrolador, o servidor na nuvem, a aplicação móvel e o *dashboard* necessitam de concordar num modelo de dados comum para os dados do sensor. Para isso foi decidido que cada leitura do sensor consiste nos valores das medições respetivas e num carimbo de data/hora, com o carimbo de data/hora num formato padrão como o ISO 8601.
- **Unidades Padrão:** É importante que todos os componentes usem as mesmas unidades de medida. Neste caso, o nível de preenchimento será compreendido por todos os componentes como uma percentagem, onde 100% indica um contentor cheio e 0% indica um contentor vazio.

- **Terminologia Consistente:** Todos os componentes usam a mesma terminologia para evitar confusão. A aplicação móvel refere-se a "contentores de resíduos", e o *dashboard* não deve referir-se a eles como "caixotes do lixo".
- **Validação de Dados:** A validação de dados foi implementada em várias etapas para garantir que os dados estejam conforme o modelo acordado. O microcontrolador avalia os dados do sensor antes de os enviar para o servidor na nuvem, e o servidor valida os dados antes de os armazenar.

4.3 Interoperabilidade Sintática

No SGRI, a interoperabilidade sintática garante que todos os componentes do sistema possam trocar dados num formato que todos eles compreendam. Isto é crucial para o funcionamento sem falhas do sistema.

- **Formato de Dados:** Os dados transmitidos do microcontrolador para o servidor na nuvem, e do servidor na nuvem para a aplicação móvel e o *dashboard*, precisam de estar num formato que estes componentes consigam entender. Para isso, foi escolhido um formato de mensagem padronizado, o JSON, devido ao seu amplo suporte e facilidade de uso. Por exemplo, uma leitura de determinado sensor é representada como um objeto JSON como {"sensor_id": "1", "s1_r": 1.5, "s1_o": 2.3, "s2_r": 0.8, "s2_o": 1.2, "s3_r": 1.1, "s3_o": 0.5, "timestamp": "2022-01-01T12:00:00Z"}
- **Protocolos de Comunicação:** O microcontrolador e o servidor na nuvem precisam de usar um protocolo de comunicação comum para trocar dados. O *Hypertext Transfer Protocol* (HTTP) foi a escolha para isso, com o microcontrolador a fazer pedidos HTTP *POST* para enviar dados para o servidor.
- **Design da Application Programming Interface (API):** O servidor na nuvem fornece uma API bem definida para a aplicação móvel e o *dashboard* obterem dados. Definiram-se *endpoints* específicos (como /contentores para obter uma lista de todos os contentores, ou /contentores/:id para obter detalhes sobre um contentor específico), e especificar o formato das respostas.
- **Comunicação com o Painel:** O microcontrolador envia dados para o painel num formato que o painel possa compreender. Isto é, um si-

nal simples que corresponde a cada cor (0 para "Muito Baixo", 1 para "Baixo", 2 para "Moderado", 3 para "Alto"). O microcontrolador e o painel terão que usar um protocolo de comunicação comum. Para isso utiliza-se um protocolo com fios (como *Inter-Integrated Circuit* (I2C) ou *Serial Peripheral Interface* (SPI)) se o painel estiver fisicamente ligado ao microcontrolador, ou um protocolo sem fios (como Bluetooth ou Wi-Fi) se o painel for separado.

- **Gestão de Erros:** O sistema deve ter uma forma consistente de representar e lidar com erros. Por exemplo, se a aplicação móvel solicitar dados para um contentor que não existe, o servidor responde com um código de estado 404 e um objeto JSON como {"error": "Contentor não encontrado"}.

4.4 Interoperabilidade de Dispositivos

A interoperabilidade de dispositivos refere-se à capacidade de diferentes dispositivos de *hardware* num sistema trabalharem juntos de forma harmoniosa. No contexto do SGRI, isso envolve os seguintes componentes:

- **Sensores Ultrassónicos:** Estes sensores são instalados nos contentores de resíduos para medir o nível de enchimento. Eles precisam de ser compatíveis com o microcontrolador, o que significa que devem ser capazes de comunicar usando um protocolo suportado pelo microcontrolador.
- **Microcontrolador:** Este dispositivo recolhe e processa dados dos sensores ultrassónicos. Deve ter as interfaces necessárias (como pinos *General Purpose Input/Output* (GPIO) para sensores digitais, ou *Analog-to-Digital Converters* (ADCs) para sensores analógicos) para se ligar aos sensores. Também precisa de ter capacidades *Wi-Fi* para transmitir dados para o servidor na nuvem.
- **Servidor na Nuvem:** Este não é um dispositivo físico, mas precisa de ser capaz de receber e processar dados do microcontrolador. Para isso, executa-se um servidor *web* que pode aceitar pedidos HTTP.
- **Aplicação Móvel:** Esta é executada nos *smartphones* dos residentes. Precisa de ser capaz de comunicar com o servidor na nuvem para recuperar dados. O que envolve fazer pedidos HTTP à API do servidor.

- **Painel:** Este dispositivo é adicionado ao contentor de resíduos para indicar o nível de enchimento. Para isso, recebe sinais do microcontrolador e muda de texto de acordo.

4.5 Interoperabilidade de Plataforma

A interoperabilidade de plataforma consiste na capacidade de sistemas IoT heterogêneos, com sistemas operativos diferentes e/ou domínios diferentes comunicarem entre si. No contexto do SGRI, isso envolve os seguintes aspectos:

- **Firmware do Microcontrolador:** O microcontrolador executa *firmware* que recolhe dados dos sensores ultrassónicos e os transmite para o servidor na nuvem. Este firmware é compatível com o *hardware* do microcontrolador e utiliza as suas capacidades de *Wi-Fi*.
- **Software do Servidor na Nuvem:** O servidor na nuvem executa *software* que recebe dados do microcontrolador, armazena-os e analisa-os. Este *software* precisa de ser compatível com o sistema operativo do servidor e é capaz de lidar com pedidos HTTP.
- **Aplicação Móvel:** A aplicação móvel executa nos sistemas operativos móveis *Android* e *iOS*. Faz pedidos HTTP à API do servidor na nuvem e mostra os dados recebidos aos utilizadores.
- **Dashboard:** O *dashboard* é uma aplicação *web* que funciona em vários navegadores *web* (como *Chrome*, *Firefox* e *Safari*). É capaz de fazer pedidos HTTP à API do servidor na nuvem e de mostrar os dados recebidos aos utilizadores.
- **Firmware do Painel:** O painel executa *firmware* que recebe sinais do microcontrolador e muda o texto em conformidade. Este *firmware* precisa de ser compatível com o hardware do painel.

4.6 Interoperabilidade de Rede

A interoperabilidade de rede refere-se à capacidade de sistemas conseguirem comunicar entre si através de tipologias de redes diferentes. No contexto do SGRI, isso envolve os seguintes componentes:

- **Conectividade do Microcontrolador:** O microcontrolador utiliza *Wi-Fi* para transmitir dados para o servidor na nuvem. Precisa de ser capaz de se ligar a vários tipos de redes *Wi-Fi* (como redes domésticas, redes públicas ou redes dedicadas de IoT) e lidar com diferentes protocolos de segurança de rede.
- **Conectividade do Servidor na Nuvem:** O servidor na nuvem precisa de ser acessível através da *internet*, para poder receber dados do microcontrolador e fornecer dados à aplicação móvel e ao *dashboard*. Isso envolve lidar com vários protocolos de rede (como *Transmission Control Protocol (TCP)/Internet Protocol (IP)* e HTTP), e pode também envolver lidar com *firewalls* e outras medidas de segurança de rede.
- **Conectividade da Aplicação Móvel:** A aplicação móvel precisa de ser capaz de se ligar ao servidor na nuvem através de vários tipos de redes (como *Wi-Fi*, 4G ou 5G). Deve ser capaz de lidar com diferentes condições de rede (como variações na intensidade do sinal ou congestionamento da rede) e continuar a funcionar mesmo quando a rede não está disponível.
- **Conectividade do Painel Web:** O painel web, tal como a aplicação móvel, precisa de ser capaz de se ligar ao servidor na nuvem através de vários tipos de redes. Deve ser projetado para lidar com diferentes condições de rede e continuar a funcionar mesmo quando a rede não está disponível.

4.7 Conclusões

Neste capítulo, foi discutido a importância de garantir a comunicação eficaz e a integração harmoniosa entre os diversos dispositivos e sistemas do SGRI.

Foram abordados conceitos como interoperabilidade semântica, interoperabilidade sintática, interoperabilidade de dispositivos, interoperabilidade de plataforma e interoperabilidade de rede de forma a expor e em seguida resolver potenciais problemas do sistema proposta em cada área específica da temática em questão.

Capítulo

5

Ética e Segurança

5.1 Introdução

A ética é o domínio da filosofia responsável pela investigação dos princípios que orientam o comportamento humano. Ou seja, que tem por objeto o juízo de apreciação que distingue o bem e o mal, o comportamento correto e o incorreto. [10] O presente capítulo aborda o código de ética, descrito na 5.2, bem como a segurança da informação tratada pelo sistema, na secção 5.3

5.2 Código de Ética

Nesta secção serão expostos os princípios fundamentais para garantir o uso ético e responsável do sistema IoT proposto.

5.2.1 Sistema IoT e Sociedade

Consentimento Informado

1. **Informação ao Utilizador:** Informar as câmaras municipais e as empresas de gestão de resíduos sobre os tipos de dados recolhidos, tais como níveis de enchimento, localização dos contentores e como os sensores monitorizam os caixotes. Obter o consentimento tanto das câmaras municipais como das empresas de gestão de resíduos para a recolha e utilização destes dados.
2. **Atualizações do Sistema:** Notificar os utilizadores sempre que o sistema sofrer atualizações ou alterações, garantindo a sua compreensão e concordância com estas atualizações.

3. **Recomendações Iniciadas pelos Utilizadores:** Permitir que os residentes e as empresas de gestão de resíduos sugiram atualizações ou melhorias ao sistema, dando ênfase ao *feedback* dos utilizadores.

Privacidade

1. **Acesso aos Dados:** Restringir o acesso aos dados recolhidos apenas ao pessoal autorizado, como os funcionários de gestão de resíduos e os administradores do sistema. Garantir que os residentes possam visualizar dados relevantes relativos aos seus contentores do lixo.
2. **Integridade dos Dados:** Proteger os dados contra modificações maliciosas. Manter a integridade dos dados desde a recolha até ao processamento e armazenamento.
3. **Não Manipulação de Dados:** Garantir que os administradores do sistema não possam manipular os dados recolhidos pelos sensores.

Confiabilidade e Resiliência

1. **Testes do Sistema:** Realizar testes extensivos do sistema, incluindo testes às leituras dos sensores ultrassónicas, e ao valor de enchimento obtido pelo sistema.

Transparência

1. **Comunicação Clara:** Explicar de forma clara e aberta todas as práticas e protocolos aos utilizadores. Fornecer informações detalhadas sobre como o sistema funciona, como os dados são recolhidos e utilizados, e as medidas de segurança em vigor.
2. **Código Open Source:** Tornar o código do sistema disponível como *open source*, permitindo que qualquer parte interessada reveja e verifique as funcionalidades e as operações do sistema.

Segurança Física

1. **Não Interferência com o Ambiente:** Garantir que os sensores e outros componentes instalados nos contentores de resíduos, alimentados por energia solar, não tenham um impacto negativo nos contentores ou no seu meio envolvente.

2. **Responsabilidade por Danos:** Estabelecer uma responsabilidade clara por quaisquer danos causados pelos componentes do sistema, garantindo que a responsabilidade possa ser atribuída com precisão.

5.2.2 Aplicação Móvel

No contexto de uma aplicação móvel voltada para a gestão de resíduos e o meio ambiente, os princípios éticos focados foram:

1. **Privacidade e Proteção de Dados:** Garantir a proteção e confidencialidade dos dados dos utilizadores é essencial para manter a confiança dos residentes. Embora a aplicação não exija login, recolhe as coordenadas GPS dos utilizadores e armazena-as. É crucial respeitar as leis de privacidade em vigor e utilizar esses dados exclusivamente para melhorar a funcionalidade da aplicação. A transparência sobre como as coordenadas GPS são recolhidas e utilizadas é fundamental.
2. **Sustentabilidade:** Promover práticas de descarte sustentável é essencial para o objetivo principal da aplicação. A aplicação deve fornecer informações precisas e atualizadas sobre a localização e o nível de ocupação das lixeiras para facilitar este processo. Além disso, a aplicação inclui um ícone de lâmpada na página principal para sugerir a melhor opção de descarte, com base na localização do utilizador e nos níveis de preenchimento das lixeiras próximas.
3. **Usabilidade:** Desenvolver uma interface intuitiva e acessível. Terá que fornecer uma navegação simples e eficiente de forma a permitir a inclusão de residentes de todo o espectro de idades.

5.2.3 Dashboard

No contexto de uma dashboard voltada para a gestão de resíduos e o meio ambiente, os princípios éticos focados foram:

1. **Acesso Autorizado:** Restringir o acesso a dashobard apenas ao pessoal autorizado, como os funcionários de gestão de resíduos e os administradores do sistema.
2. **Integridade dos Dados:** Proteger os dados contra modificações maliciosas. Manter a integridade dos dados desde a recolha até ao processamento e armazenamento.

3. **Não Manipulação de Dados:** Garantir que os administradores do sistema não possam manipular os dados recolhidos pelos sensores, mas possam analisar e notar irregularidades.

5.3 Segurança da Informação

Esta secção cogita apresentar uma análise das potenciais vulnerabilidades de segurança do nosso sistema e descrever as medidas implementadas para garantir a sua proteção contra essas ameaças. É essencial identificar e compreender as possíveis ameaças que o sistema pode enfrentar, para assim implementar as medidas preventivas e corretivas adequadas.

5.3.1 Ataque de Homem no Meio (*Man-in-the-Middle Attack*)

Um ataque de homem no meio é um ataque cibernético em que um ator malicioso se coloca no meio de duas partes, tipicamente um utilizador e uma aplicação, para interceptar as suas comunicações e trocas de dados, utilizando-as para fins maliciosos.

Ao posicionar-se secretamente entre o utilizador e um sistema de confiança, como um *site* ou aplicação, um cibercriminoso pode facilmente obter dados sensíveis. O utilizador assume que, por exemplo, está a interagir exclusivamente com um *site* de confiança e, sem saber, cede credenciais de *login*, informações financeiras ou outros dados comprometedores.

De forma a combater este ataque, as comunicações dos vários sensores, enviadas através de uma estrutura de dados do tipo JSON devem ser cifrados com *AES-128-CBC*, com uma chave secreta previamente partilhada de forma segura, resolvendo assim um ataque de homem no meio.

5.4 Malware – Criação de Botnets

Uma *botnet* é uma rede de computadores infetados que podem ser controlados remotamente e forçados a enviar *spam*, espalhar *malware* ou preparar ataques de *Distributed Denial-of-Service* (DDoS) sem o consentimento dos proprietários do dispositivo.

Os cibercriminosos costumam visar os dispositivos IoT ao construir *botnets* porque eles têm segurança fraca. Por exemplo, grandes *botnets* de IoT foram construídos tentando fazer *login* usando credenciais padrão ou explorando vulnerabilidades não corrigidas no *software* dos dispositivos IoT. Estas brechas de segurança nos dispositivos IoT permitem que um atacante aceda ao dispositivo e execute o *malware* do *botnet*. Uma vez que o malware do

botnet está instalado num dispositivo IoT, o dispositivo pode ser controlado remotamente para executar as ordens do atacante. Por exemplo, um *bot* de IoT pode ser instruído a participar num ataque DDoS contra um endereço específico.

No sistema proposto neste relatório, será empregue uma rede independente, fornecida, por exemplo, pela câmara municipal, onde só os dispositivos do respetivo sistema tenham acesso. Esta rede tem como objetivo separar os dispositivos IoT, de outras redes, com níveis de segurança menos apropriados. O administrador desta rede independente deverá ter em consideração o que esta acarreta e deve protegê-la o melhor possível de acessos indevidos.

5.4.1 *Denial-of-Service (DOS)*

Um DOS é um tipo de ataque cibernético em que um ator malicioso tem por objetivo tornar um computador ou outro dispositivo indisponível para os utilizadores a que se destinam, interrompendo o funcionamento normal do dispositivo. Estes ataques normalmente funcionam sobrecarregando ou inundando uma máquina visada com solicitações até que o tráfego normal não possa ser processado, resultando em negação de serviço para utilizadores adicionais. O mesmo caracteriza-se pelo uso de um único computador para lançar o ataque, já o DDoS é um tipo de ataque DOS que se origina em muitas fontes distribuídas, tais como um ataque DDoS de *botnet*. [11]

De forma a prevenir este ataque no nosso sistema, serão implementadas *firewalls*. Além disso, é importante manter o sistema atualizado com *patches* de segurança e adotar boas práticas de segurança aquando a elaboração do mesmo.

Capítulo

6

Implementação e Demonstração

6.1 Introdução

O presente capítulo aborda os pormenores mais relevantes da implementação do sistema. Na secção 6.2 é feita uma breve demonstração da implementação real do sistema, incluindo o contentor, o *hardware* e *software* nele envolvido. A secção 6.3 descreve a interface acessível à empresa de gestão de resíduos 6.3.1 e a secção 6.3.2 à pessoa comum que deseja depositar o seu lixo.

6.2 Protótipo IoT

6.2.1 Contentor

O nosso contentor foi simulado utilizando uma caixa de cartão, originalmente uma embalagem de computador. A parte traseira da caixa possui dois furos: um destinado à fixação do servomotor e outro para a passagem de todos os cabos e fios necessários para as conexões entre o *Arduino*, a *breadboard* e o protótipo IoT. A caixa também conta com uma tampa única, que atualmente serve como uma simulação das tampas dos contentores. Em versões futuras, essa tampa poderá ser equipada com um mecanismo de tranca automática. Quando o nível crítico de enchimento for atingido, a tampa fecha automaticamente para evitar danos aos materiais, sobrecarregamento dos contentores e acumulação de lixo do lado de fora, contribuindo assim para a saúde pública. Segue abaixo, ilustrado em 6.7 e em 6.2, duas imagens do mesmo já com o dispositivo IoT instalado nele.



Figura 6.1: Vista interior do contentor



Figura 6.2: Vista exterior do contentor

6.2.2 Hardware

Para a realização do protótipo em termos de *hardware*, foram utilizadas os seguintes componentes: Segue abaixo as ilustrações em 6.3, 6.4, 6.5, o *hardware* e como foram montados os devidos elementos.

1. **Bread board 830:** Uma pequena placa de ensaio usada para prototipagem de circuitos eletrônicos sem a necessidade de solda, facilitando a conexão e desconexão dos componentes e dos fios.
2. **ARDUINO UNO R3:** Uma plataforma de prototipagem eletrônica baseada em *hardware* e *software* livres.
3. **LAN Shield acoplado ao Arduino:** Um módulo de expansão que se conecta ao Arduino, permitindo que ele se comunique com redes Ethernet, permitindo o envio de dados para a *cloud*.
4. **Servo Motor (Servo Motor SG90):** Um motor de precisão que pode ser controlado para se mover a posições específicas. No nosso protótipo, o *servo motor* é usado para rodar os sensores para captar todo o conteúdo.
5. **Sensor Distância HC-SR04:** Utilizamos três sensores ultrassônicos no protótipo para medir a distância até o lixo no contentor de forma a determinar o nível de enchimento.



Figura 6.3: Servo motor e base dos sensores



Figura 6.4: Três sensores dentro do caixote para medir

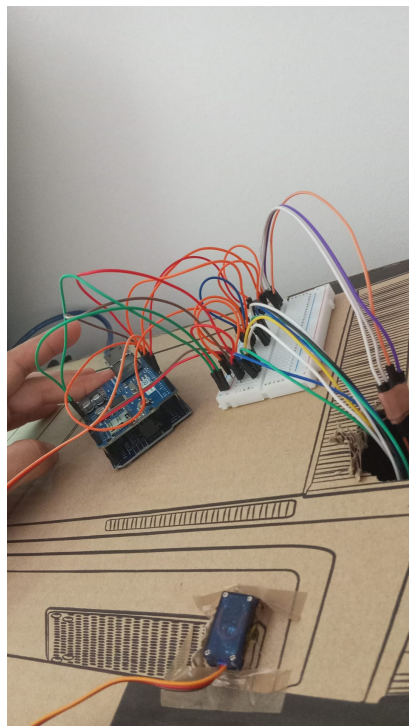


Figura 6.5: Conexões Arduino e *Servo Motor*

6.2.3 Software

No *software*, temos quatro categorias:

1. **Arduino:** O código escrito para o microcontrolador Arduino Uno é responsável por controlar todos os componentes de *hardware* conectados. Isso inclui a leitura dos sensores, o controle do *servo motor* e a comunicação com o *LAN Shield* para enviar dados.
2. **Gateway entre o dispositivo Arduino e a API do servidor:** Esta categoria refere-se ao envio de dados recolhidos pelos sensores do Arduino para um servidor remoto através do método *POST*. Utilizando o LAN Shield, os dados são transmitidos através da rede Ethernet.
3. **Servidor Node.js** Este serve como o servidor de *backend*, responsável por lidar com pedidos de API, processar dados e interagir com a base de dados.
 - **API:** A API é o intermediário que recebe os dados enviados pelo Arduino, envia para o servidor *node.js* e os disponibiliza para outras aplicações. Esta é utilizada para armazenar os dados na base de dados, e fornece *endpoints* para que a aplicação móvel e o *dashboard* consigam aceder às informações em tempo real.
4. **Trigger:** Esta categoria envolve o cálculo do nível de enchimento baseado nos valores recolhidos pelos sensores.

6.2.3.1 Arduino

O código do Arduino foi elaborado de forma a que três sensores ultrassónicos conectados a um *servo motor*, posam varrer um ângulo de 0° a 180° de forma obtermos uma representação mais completo do nível de enchimento do contentor.

No trecho de código ilustrado em 6.1 é possível observar que o ângulo do servo motor irá ser incrementado de quinze em quinze graus e cada incremento segue-se uma leitura dos valores dos três sensores ultrassónicos.

```
void loop() {  
  //Obrigar o servo a iniciar/voltar a posicao inicial  
  int posInicial = 0;  
  int posFinal = 180;  
  // Move o servo do posInicial para posFinal em incrementos de 15 graus  
  for(int pos = posInicial; pos <= posFinal; pos += 15) {  
    servo.write(pos);  
  }  
}
```

```
// Realiza a leitura dos tres sensores ultrassonicos
float distance1 = lerSensor(trigPin1 , echoPin1);
float distance2 = lerSensor(trigPin2 , echoPin2);
float distance3 = lerSensor(trigPin3 , echoPin3);

printResults(pos, distance1 , distance2 , distance3 ,id);
wait(1);
}
```

Excerto de Código 6.1: Código para controlar o servo motor

6.2.3.2 Gateway entre o dispositivo Arduino e a API do servidor

Este código em Node.js atua como um intermediário entre um dispositivo Arduino e um servidor.

1. **Leitura de dados do Arduino:** O código cria um *parser* que lê dados da porta serial onde o Arduino está conectado. Os dados são lidos linha por linha, com cada linha sendo dividida em partes usando uma expressão regular.
2. **Processamento de dados:** Quando os dados são recebidos do Arduino, o código tenta fazer o *match* dos dados com uma expressão regular. Se os dados correspondem à expressão regular, eles são divididos em partes e convertidos em números, se necessário. Esses dados são então armazenados num objeto *extractedData*.
3. **Bufferização de dados:** O objeto *extractedData* é então adicionado a um *buffer* (*bufferedData*). Se a posição for diferente de zero e o *buffer* estiver vazio, o *buffer* é limpo. Se o *buffer* atingir um certo tamanho (*MAX_RECORDS*), os dados são enviados para a API e o *buffer* é limpo.
4. **Envio de dados para a API:** A função *sendDataToAPI* é usada para enviar os dados para a API. Ela configura as opções para a solicitação HTTP, incluindo o *hostname*, a porta e o caminho da API, e o método HTTP (POST).
5. **Tratamento de erros:** Se os dados recebidos do Arduino não correspondem à expressão regular, um erro é registrado e o *buffer* é limpo.

Em resumo, este código lê dados de um Arduino, processa esses dados, armazena-os num *buffer* até que haja dados suficientes para enviar, e então envia esses dados para uma API. Ele também lida com erros que podem ocorrer durante esse processo.

6.2.3.3 Servidor *Node.js*

Este servidor *Node.js* é um servidor de aplicação *web* que lida com solicitações HTTP enviadas pelo cliente, processa essas solicitações e retorna uma resposta. Ele é construído usando a *framework Express.js* e conecta-se a uma base de dados *MySQL* para armazenar e recuperar dados.

Configuração do servidor: O servidor está configurado para rodar na porta 3000. Ele usa o middleware `bodyParser.json()` para analisar o corpo das solicitações recebidas em formato JSON.

Conexão com a base de dados: O servidor cria uma conexão com a base de dados *MySQL* usando as credenciais fornecidas como variáveis de ambiente. Ele tenta-se conectar e emite um erro se a conexão falhar.

Rotas: O servidor define várias rotas para lidar com operações *Create, Read, Update, Delete* (CRUD) nas tabelas *Companie, Containers, Regions, SensorData, Final_Stats* da base de dados.

Exemplo de Rotas para a tabela *SensorData* :

- A rota GET `/api/sensorData` é usada para buscar todos os registros da tabela *SensorData*. Quando uma solicitação GET é enviada para `/api/sensorData`, o servidor executa a consulta SQL `SELECT * FROM SensorData` e retorna os resultados.
- A rota POST `/api/sensorData` é usada para criar um novo registro na tabela *SensorData*. Quando uma solicitação POST é enviada para `/api/sensorData` com um corpo de solicitação contendo os dados do novo registro, o servidor executa a consulta SQL `INSERT INTO SensorData SET ?` e retorna uma mensagem de confirmação.
- A rota PUT `/api/sensorData/:id` é usada para atualizar um registro existente na tabela *SensorData*. Quando uma solicitação PUT é enviada para `/api/sensorData/:id` com um corpo de solicitação contendo os novos dados do registro e um parâmetro de rota especificando o ID do registro a ser atualizado, o servidor executa a consulta SQL `UPDATE SensorData SET ... WHERE ID_Record = ?` e retorna uma mensagem de confirmação.

- A rota DELETE `/api/sensorData/:id` é usada para excluir um registro existente na tabela `SensorData`. Quando uma solicitação DELETE é enviada para `/api/sensorData/:id` com um parâmetro de rota especificando o ID do registro a ser excluído, o servidor executa a consulta SQL `DELETE FROM SensorData WHERE ID_Record = ?` e retorna uma mensagem de confirmação.

6.2.3.4 *Trigger*

O *trigger* utilizado para calcular a nível de enchimento do contentor é ilustrado no trecho de código 6.2. Neste considerou-se a altura do caixote como 20 centímetros.

Este calcula o nível de enchimento de um contentor de lixo cada vez que novos dados do sensor são adicionados à tabela `SensorData`.

1. **Ativação do Trigger:** O *trigger* é ativado quando novos dados do sensor são adicionados. Só prossegue se o sensor tiver concluído uma varredura completa do contentor (indicado por uma *position* de 180).
2. **Recuperação de Dados:** O *trigger* busca as últimas 10 medições de distância (*r*) na tabela `SensorData` para o contentor atual (`ID_Container`).
3. **Cálculo:** O *trigger* calcula a média dessas 10 medições, subtraindo esta média da altura total do contentor para obter a altura do lixo. Esta altura é então convertida para uma percentagem da altura total do contentor para obter o nível de enchimento.
4. **Atualização das Estatísticas:** O nível de enchimento calculado, juntamente com o ID do contentor e a hora atual, é adicionado à tabela `final_stats` para monitorização e análise.
5. **Conclusão:** Se o sensor não tiver concluído uma varredura completa, o *trigger* não faz nada e espera pelos próximos dados a serem adicionados.

```
DELIMITER //
CREATE TRIGGER after_sensor_insert
AFTER INSERT ON SensorData
FOR EACH ROW
BEGIN
    DECLARE i INT DEFAULT 0;
    DECLARE r FLOAT;
    DECLARE pos FLOAT;
    DECLARE bin_height FLOAT DEFAULT 20;
```



```
DECLARE total_distance FLOAT DEFAULT 0;
DECLARE fill_level FLOAT;
DECLARE cur CURSOR FOR SELECT s1_r, s2_r, s3_r, position FROM
    SensorData WHERE ID_Container = NEW.ID_Container ORDER BY id DESC
    LIMIT 10;

IF NEW.position = 180 THEN
    OPEN cur;
    FETCH cur INTO r, pos;
    WHILE i < 10 DO
        SET total_distance = total_distance + r;
        SET i = i + 1;
        FETCH cur INTO r, pos;
    END WHILE;
    CLOSE cur;
    SET fill_level = ((bin_height - (total_distance / i)) / bin_height)
        * 100;
    INSERT INTO final_stats (ID_Container, fill_level, timestamp) VALUES
        (NEW.ID_Container, fill_level, NOW());
END IF;
END; //
DELIMITER ;
```

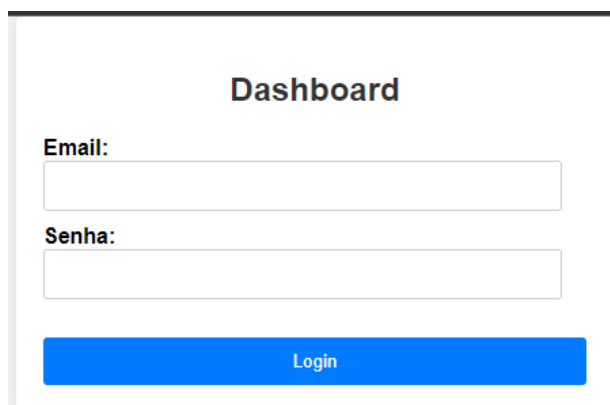
Excerto de Código 6.2: Trigger associado ao nível de enchimento

6.3 Clientes

6.3.1 Aplicação Web - Dashboard

6.3.1.1 Login

O login deveria ser feito de forma segura e cifrada, no entanto, o mesmo não foi implementado por falta de tempo, as comunicações cliente-servidos deveriam ser cifradas (o que não são) gerando aí uma vulnerabilidade considerável. Ignorando isso, o login é simples com um campo email e password, ao fazer login verifica-se os dados estão corretos através de mecanismos próprios e depois consoante a resposta reencaminha o cliente para o seu próprio menu (funcionário/administrador). A cada login é gerado um token com validade de doze horas que garante que não existem tentativas de burlar o sistema de login. Após um login bem-sucedido como mencionado acima, o utilizador será redirecionado para uma página onde tem todas as opções que precisa para monitorizar e analisar os contentores.

A login form titled "Dashboard" in bold. It contains two input fields: "Email:" and "Senha:". Below the fields is a blue button labeled "Login".

Dashboard

Email:

Senha:

Login

Figura 6.6: Imagem do sistema de Login

6.3.1.2 Menu

No menu tem várias opções, dentre delas, adicionar e remover funcionários e/ou contentores, e analisar o nível de enchimento deles no momento atual e/ou por datas. Apesar de estar construída a base de todo o sistema, o mesmo não foi implementado, faltando criar a interface gráfica para mostrar/controlar esses dados e as chamadas das mesmas com a API já desenvolvida.

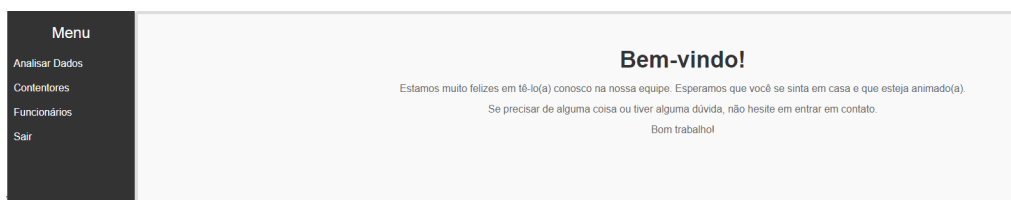


Figura 6.7: Imagem do Menu principal

6.3.2 Aplicação Móvel

O foco desta secção incide na demonstração da aplicação móvel desenvolvida. Sendo assim será demonstrado detalhadamente todas as funcionalidades oferecidas.

6.3.2.1 Página Inicial — Seleção da Região

A página inicial da aplicação está ilustrada na figura 6.8. Nesta página é possível escolher a região pretendida num conjunto de regiões disponíveis na base de dados.

Ao escolher uma região, será pedida permissão ao utilizador para recolher a sua localização de forma calcular a distância deste ao contentor, como está explicado em 6.3.2.2

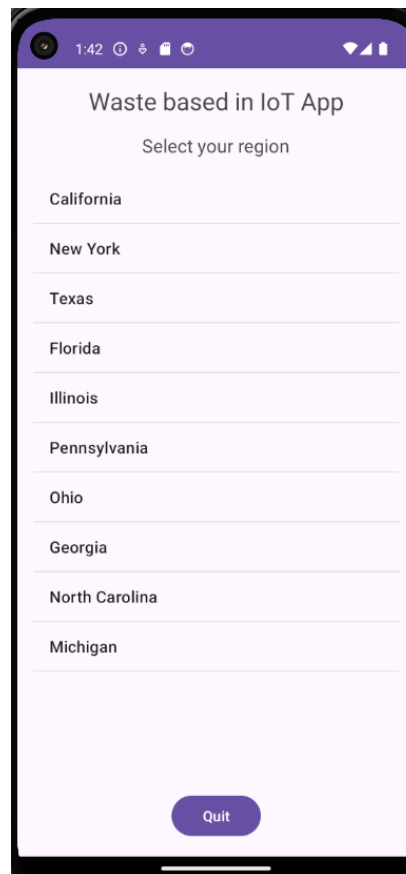


Figura 6.8: Página inicial para seleccionar uma região.

6.3.2.2 Consulta dos Contentores

A página ilustrada na figura 6.9 mostra respetivos contentores de determinada região, a distância destes ao utilizador, e o seu nível de enchimento baseado na última medição que foi recolhida dos sensores.

No botão "*Intelligent Suggestion*" será dada a melhor opção ao utilizador tendo em conta a distância deste ao contentor e o nível de enchimento do mesmo.

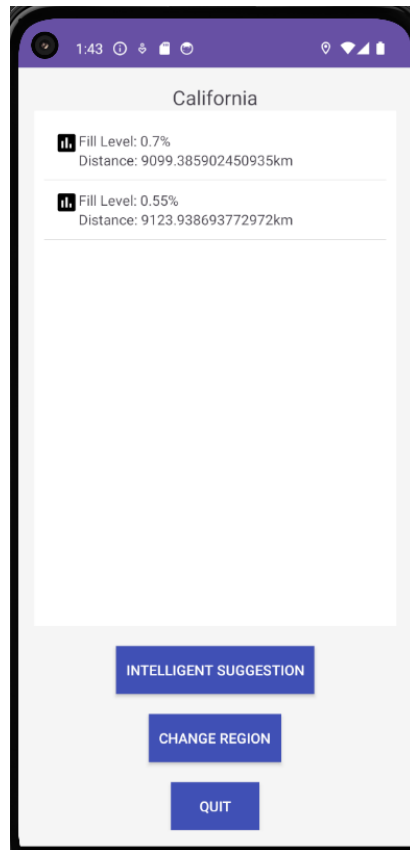


Figura 6.9: Visualização do estado dos contentores.

Com as coordenadas do utilizador recolhidas, faz-se o cálculo da distância deste ao contentor através de, ilustrado no trecho 6.3:

```
private double calculateDistance(double userLat, double userLong, double
    containerLat, double containerLong) {
    // Convert degrees to radians
    userLat = Math.toRadians(userLat);
    userLong = Math.toRadians(userLong);
    containerLat = Math.toRadians(containerLat);
    containerLong = Math.toRadians(containerLong);

    // Calculate the differences
    double dLat = containerLat - userLat;
    double dLong = containerLong - userLong;

    // Calculate the distance
    double a = Math.pow(Math.sin(dLat / 2), 2)
        + Math.cos(userLat) * Math.cos(containerLat)
        * Math.pow(Math.sin(dLong / 2), 2);
```

```
double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

// Convert the distance from radians to kilometers
double distance = 6371 * c;

return distance;
}
```

Excerto de Código 6.3: Cálculo da distância do contentor ao utilizador

6.3.2.3 Sugestão Inteligente

A 6.10 mostra como a solução inteligente é mostrada ao utilizador.

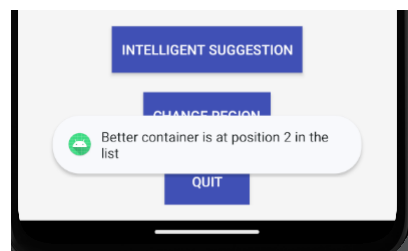


Figura 6.10: Solução inteligente

O critério para seleccionar um contentor dos demais, é ilustrado em 6.4. Daí observa-se que um nível de enchimento mais baixo e uma distância mais curta resultam numa pontuação mais alta.

```
double score = 1.0 / (container.getFillLevel() *
    Math.pow(container.getDistance(), 2));
Log.d("ViewDecision", "Container: " + containers.
    indexOf(container) + " Score: " + score);

if (score > maxScore) {
    maxScore = score;
    betterContainer = container;
}
}
```

Excerto de Código 6.4: Escolha do contentor

6.4 Conclusões

Neste capítulo forneceu-se então uma visão geral do nosso sistema IoT nas suas diversas componentes e como foram feitas as ligações entre as mesmas. Foi também apresentado as interfaces para o utilizador normal (residente) e para um funcionário de uma determinada empresa de gestão de resíduos, sendo que a primeira em formato móvel e a segundo com uma vertente *web* e de análise.

Foi também exposto o *trigger* que nos dá informação sobre o nível de enchimento do contentor que é objetivo primordial deste projeto.

Capítulo

7

Conclusões e Trabalho Futuro

7.1 Conclusões Principais

Este projeto contribuiu significativamente para o desenvolvimento das nossas competências em conceitos de IoT, além de nos permitir compreender os desafios e vantagens associados a esta tecnologia. Embora o protótipo final não tenha atingido completamente as nossas expectativas em termos de robustez e desenvolvimento, devido à escassez de materiais e tempo, ele ainda serve como uma base sólida para demonstrar as capacidades do dashboard e da aplicação móvel.

Reconhecemos que gostaríamos de ter tido mais tempo e recursos para aprimorar o projeto, especialmente no que diz respeito à segurança, funcionalidades e fiabilidade. Um dos pontos que ficaram pendentes foi a implementação completa do mapeamento 3D. Este recurso foi concebido como uma ideia promissora, mas não foi totalmente implementado, pois faltou a conversão de 3D para 2D e os testes necessários nesse paradigma. Em vez disso, implementámos um *trigger* que não possui a mesma precisão do mapeamento planeado.

No entanto, apesar das limitações, o trabalho realizado até agora representa um passo importante na direção certa e oferece uma plataforma para futuras melhorias e desenvolvimentos. Consideramos bastante satisfatório o resultado final, que serve como um bom ponto de partida para que, no futuro, possamos aprimorar ou continuar este projeto, seja por nós ou por outros interessados.

7.2 Trabalho Futuro

Durante o desenvolvimento, implementámos diversas funcionalidades essenciais para a operação e demonstração básica do sistema. No entanto, devido ao tempo limitado e à carga de trabalho, algumas melhorias e implementações adicionais não foram realizadas.

Entre as melhorias futuras, destaca-se a necessidade de implementar a criptografia nas comunicações entre os sensores nos contentores e a *cloud*. Essa medida aumentaria a segurança dos dados transmitidos, protegendo informações sensíveis contra intercetações e ataques cibernéticos. Além disso, é fundamental melhorar o *design* do dashboard para torná-lo mais intuitivo e visualmente atraente, facilitando a interpretação dos dados pelos utilizadores.

Outra implementação essencial seria a integração de um sistema de GPS de alta precisão para monitorização e localização dos contentores. Isso melhoraria a rastreabilidade dos contentores, otimizando as rotas de recolha. Também é crucial adaptar e testar o protótipo em cenários reais, considerando diferentes condições ambientais e operacionais, para garantir que o sistema funcione de maneira eficaz em diversas situações e ambientes, identificando e resolvendo problemas práticos que possam surgir.

Para melhorar a aplicação web destinada aos residentes, devemos implementar funcionalidades adicionais que aumentem a interação e a usabilidade da plataforma. Além disso, a implementação de um fecho automático nos contentores que seja ativado quando os níveis de lixo atingirem um ponto crítico pode prevenir a sobrecarga e evitar problemas de higiene e saúde pública.

Finalmente, uma implementação futura interessante seria a adição de um visor nos contentores, permitindo uma interface direta para as pessoas que recolhem o lixo e para os cidadãos, fornecendo informações em tempo real sobre o estado do contentor e instruções sobre o uso adequado. Estes aspetos não foram implementados no projeto atual, mas seriam melhorias significativas para a continuação e o aperfeiçoamento do nosso projeto.

Bibliografia

- [1] Bharath R Student, Anusha S Raj, A Sushmitha. 3D Surface Mapping using Ultrasonic Sensors, 2018. [Online] <https://ijisrt.com/wp-content/uploads/2018/08/3D-Surface-Mapping-using-Ultrasonic-Sensors-1.pdf>. Último acesso a 9 de Abril de 2024.
- [2] N. Bharathiraja, T. Deepa, S. Hariprasad, and Arun Chokkalingam. Design and Development of GIOT based Intelligent Smart Waste Management and Predictive Modelling, 2022. [Online] <https://ieeexplore.ieee.org/document/9777210>. Último acesso a 2 de Abril de 2024.
- [3] Agência Portuguesa do Ambiente. Relatório anual de resíduos urbanos 2022. *Relatório Anual Resíduos Urbanos*, page 11, 2023.
- [4] Agência Portuguesa do Ambiente. Dados sobre Resíduos Urbanos, 2023. [Online] <https://apambiente.pt/residuos/dados-sobre-residuos-urbanos>. Último acesso a 1 de Abril de 2024.
- [5] Dmitriy Malets. Internet of Things (IoT) and Its Role in the Smart Cities of the Future, 2023. [Online] <https://www.linkedin.com/pulse/internet-things-iot-its-role-smart-cities-future-dmitriy-malets/>. Último acesso a 8 de Abril de 2024.
- [6] K. Belsare and M. Singh. An Intelligent Internet of Things (IoT) based Automatic Dry and Wet Medical Waste Segregation and Management System, 2022. [Online] <https://ieeexplore.ieee.org/document/10010913>. Último acesso a 2 de Abril de 2024.
- [7] Oluwasegun Julius Aroba, Thuthukani Xulu, Nonsikelelo N. Msani, Thuso T. Mohlakoana, Experience E. Ndlovu, and Simphiwe M. Mthethwa. The Adoption of an Intelligent Waste Collection System in a Smart City, 2023. [Online] <https://ieeexplore.ieee.org/document/10082750>. Último acesso a 2 de Abril de 2024.

- [8] Lucidchart. Diagrama de caso de uso UML: O que é, como fazer e exemplos, 2022. [Online] <https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml>. Último acesso a 12 de Abril de 2024.
- [9] IBM. Diagrama de Atividades, 2021. [Online] <https://www.ibm.com/docs/pt-br/rational-soft-arch/9.7.0?topic=diagrams-activity>. Último acesso a 12 de Abril de 2024.
- [10] CARLOS CARAPETO FÁTIMA FONSECA. In *ÉTICA E DEONTOLOGIA - Manual de Formação*, page 8. Dossier, 2019.
- [11] CloudFare. O que é um ataque de negação de serviço?, 2023. [Online] <https://www.cloudflare.com/pt-br/learning/ddos/glossary/denial-of-service/>. Último acesso a 17 de Abril de 2024.