

IPBeja
INSTITUTO POLITÉCNICO
DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Internet das Coisas

Sistema de monitorização e otimização da recolha de resíduos

e-Bin

Rafael Duarte Montez

Beja, 2024

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Internet das Coisas

Sistema de monitorização e otimização da recolha de resíduos

e-Bin

Rafael Duarte Montez

Orientado por :

Orientador: Doutor José Jasnau Caeiro, IPBeja

Coorientador: Doutora Maria Teresa de Abreu Luís Godinho, IPBeja

Dissertação de mestrado

2024

Resumo

Sistema de monitorização e otimização da recolha de resíduos e-Bin

A deslocação diária dos veículos de recolha de resíduos implica custos que podem ser minimizados através da utilização de contentores de resíduos urbanos inteligentes. Através da sensorização é possível criar rotas otimizadas para os contentores de resíduos urbanos, em que a capacidade máxima esteja perto de ser atingida. É apresentado um sistema baseado no paradigma arquitetónico da IoT (*Internet of Things*), concebido para reduzir os custos de deslocação. Este documento fornece uma visão geral sobre sistemas de recolha de resíduos, com principal destaque na arquitetura de *hardware* e *software* utilizada. Este projeto integra um microcontrolador de baixo custo e de baixo consumo energético para adquirir e processar os dados medidos pelos sensores. A comunicação dos dispositivos com o *gateway* de Internet é feita através da comunicação de longo alcance LoRa, o que permite que os contentores de resíduos urbanos possam estar em locais remotos. A medição do nível do enchimento dos contentores de resíduos urbanos em tempo real, permite gerar rotas com a garantia de que os contentores de resíduos urbanos selecionados para o percurso têm o seu nível de enchimento final superior a 80%. O sistema permite a visualização e análise dos dados dos contentores de resíduos urbanos, percursos e da frota através das interfaces disponíveis para os diversos tipos de utilizadores.

Palavras-chave: *poluição, recolha, contentores, resíduos, sensorização, rotas, camiões, comunicação, LoRa, nível de enchimento, visualização, análise, percursos, frota, interfaces.*

Abstract

Sistema de monitorização e otimização da recolha de resíduos e-Bin

The daily movement of waste collection vehicles involves costs that can be minimized through the use of smart bins. Through sensing, it is possible to create optimized routes for the crates, where maximum capacity is close to being reached. A system based on the IoT (Internet of Things) architectural paradigm is presented, designed to reduce travel costs. This document provides an overview of waste collection systems, with main emphasis on the hardware and software architecture used. This project integrates a low-cost, low-power microcontroller to acquire and process the data measured by the sensors. Devices communicate with the Internet gateway via long-range LoRa communication, which allows the boxes to be located in remote locations. Measuring the filling level of crates in real time allows routes to be generated with the guarantee that the crates selected for the route have a final filling level greater than 80%. The system allows the visualization and analysis of crate, route and fleet data through the interfaces available to different types of users.

Keywords: *pollution, collection, waste, bins, sensing, routes, trucks, communication, LoRa, fill level, visualization, analysis, routes, fleet, interfaces.*

Agradecimentos

Neste percurso final do Mestrado em Internet das Coisas, é o momento próprio de reconhecer as pessoas envolvidas durante os dois anos de muita aprendizagem e partilha de conhecimentos e de lhes agradecer a ajuda, apoio e acompanhamento.

Em primeiro lugar, quero agradecer aos meus orientadores, professor José Jasnau Caeiro e professora Maria Teresa Godinho pela paciência, dedicação e por todas as sugestões essenciais nesta dissertação.

Em segundo lugar, gostaria de agradecer à diretora de curso, professora Isabel Sofia Brito, que me acompanhou durante todo o percurso educativo e demonstrou sempre disponibilidade.

Agradeço também ao professor João Santos por todo o seu contributo, principalmente na parte da comunicação LoRa. Ainda quanto ao corpo docente, quero agradecer também ao professor João Martins por toda a sua ajuda principalmente numa fase inicial deste projeto.

Agradeço também aos restantes professores e colegas de curso principalmente pela troca de conhecimentos e entreaajuda.

Por fim, agradeço à minha família por todo o apoio.

Conteúdo

Resumo	i
Abstract	iii
Agradecimentos	v
Conteúdo	vii
Lista de Figuras	xi
Lista de Tabelas	xv
Lista de Listagens	xvii
1 Introdução	1
1.1 Enquadramento	2
1.2 Trabalho do autor	2
1.3 Estrutura do documento	4
2 Estado da Arte	5
2.1 Introdução	5
2.2 Cidades Inteligentes	5
2.2.1 Componentes e Características das Cidades Inteligentes	7
2.2.2 Tecnologias utilizadas na <i>Smart City</i>	7
2.2.3 Cidades inteligentes em Portugal	8
2.3 Internet das Coisas	10
2.3.1 Desafios da aplicação da IoT nas Cidades Inteligentes	10
2.3.2 Tecnologias IoT	12
2.4 Levantamento bibliográfico	12
2.4.1 Projeto IoT-TBLM	12
2.4.2 Projeto IoTfskimUm	13
2.4.3 Outros projetos	14
2.5 Projetos académicos	16

2.5.1	Smart Containers	16
2.5.2	iEcoSys - <i>Intelligent Ecologic System</i>	17
2.6	Sistemas comerciais	18
2.6.1	Bigbelly Smart Bins	18
2.6.2	SmartBin.io	19
2.6.3	Sensoneo Municipal Waste Collection	20
2.6.4	360Waste	21
2.6.5	SOMA - Gestão de resíduos sólidos urbanos	22
2.7	Análise e evolução histórica dos sistemas de gestão de resíduos	23
2.8	Conclusão	24
3	Arquitetura do Sistema	25
3.1	Introdução	25
3.2	Requisitos do sistema	26
3.2.1	Requisitos funcionais	26
3.2.2	Requisitos não funcionais	27
3.3	Diagrama de casos de uso	28
3.4	Diagrama funcional do sistema	31
3.4.1	Diagrama funcional mais aprofundado do sistema	32
3.5	Modelo de Entidade-Relacionamento	35
3.6	Análise de tarefas	36
3.7	Desenvolvimento de <i>Mockups</i>	41
3.8	Ambiente de Desenvolvimento	43
3.8.1	Linguagens de programação	43
3.9	Análise e caracterização dos sensores	45
3.9.1	Módulo GPS: NEO-6M V2	45
3.9.2	Sensor de distância: HC-SR04	46
3.9.3	Sensor de gás/fumo: MQ-2	47
3.10	Conclusão	48
4	Problema de cálculo de rotas de veículos	49
4.1	Introdução	49
4.2	Procedimento computacional de VRP	50
4.3	Pseudocódigo	51
4.4	Codificação do procedimento computacional	53
4.5	Procedimento computacional em Julia	60
4.6	Taxa de crescimento do tempo de execução	65
4.7	Conclusão	67
5	Implementação Experimental	69

5.1	Introdução	69
5.2	Plataforma de desenvolvimento de <i>hardware</i>	70
5.3	<i>Layout</i> do circuito	72
5.4	Circuito na <i>breadboard</i>	73
5.5	Fonte de alimentação	74
5.5.1	Bateria	74
5.5.2	Painel Fotovoltaico	75
5.5.3	Programação para baixo consumo energético: Modo <i>deep-sleep</i> . . .	76
5.6	Tecnologia de <i>containers</i>	77
5.7	Comunicação com o servidor	79
5.7.1	Bibliotecas do microcontrolador	80
5.7.2	Comunicações LoRaWAN	80
5.7.3	Protocolo MQTT	84
5.7.4	Programação da unidade central de controlo	85
5.8	Sistema de gestão de base de dados	92
5.9	Segurança Informática	93
5.10	Validações de formulários	95
5.11	Envio de <i>emails</i>	97
5.12	Análise dos resultados	97
5.12.1	Dispositivo e-Bin	97
5.12.2	Aplicação móvel	99
5.12.3	Plataforma <i>web</i>	108
5.13	Testes realizados	123
5.13.1	Consumo médio de corrente	123
5.13.2	<i>Endpoints</i>	124
5.13.3	Conexão SocketIO	125
5.13.4	Testes adicionais	126
5.14	Discussão Geral dos Resultados	126
5.15	Conclusão	126
6	Conclusão	129
6.1	Conclusões gerais	129
6.2	Trabalho futuro	130
	Bibliografia	133
	Anexos	141
I	<i>Mockups</i>	143
I.1	<i>Mockups</i> da plataforma <i>web</i>	143
I.2	<i>Mockups</i> da aplicação móvel	161

II Conferir <i>outputs</i> do procedimento computacional de VRP	171
III Consumo total do dispositivo	177
IV Custos relacionados	179

Lista de Figuras

1.1	Percurso dos camiões com rotas otimizadas.	3
2.1	Arquitetura de rede proposta por IoT-TBLMS.	13
2.2	Interface da plataforma <i>web</i> deste sistema.	13
2.3	Caixote Inteligente.	14
2.4	Caixote inteligente.	14
2.5	Caixote do lixo inteligente e seus componentes.	15
2.6	Caixa do projeto.	15
2.7	Interfaces da aplicação móvel do projeto.	16
2.8	Interface do projeto Smart Containers	17
2.9	Caixote do lixo inteligente iEcoSys	18
2.10	Caixote do lixo inteligente Bigbelly	18
2.11	Interfaces da Bigbelly	19
2.12	Dispositivos SmartBin.io	19
2.13	Localização dos caixotes SmartBin.io no mapa.	20
2.14	Plataforma web Sensoneo	20
2.15	Aplicação móvel Sensoneo	21
2.16	Caixote do lixo inteligente 360Waste	22
2.17	Mapa do projeto SOMA	22
3.1	Diagrama de casos de uso - Leituras.	28
3.2	Diagrama de casos de uso - Rotas.	28
3.3	Diagrama de casos de uso - Veículos.	29
3.4	Diagrama de casos de uso - Dispositivos.	29
3.5	Diagrama de casos de uso - Utilizadores.	30
3.6	Diagrama funcional do sistema.	31
3.7	Diagrama funcional mais aprofundado do sistema.	33
3.8	Modelo de Entidade-Relacionamento.	35
3.9	<i>Mockup</i> de alta fidelidade da <i>dashboard</i>	41
3.10	<i>Mockups</i> de média fidelidade da aplicação móvel.	42
3.11	<i>Mockups</i> de alta fidelidade da aplicação móvel.	42

3.12	Captura de ecrã da aplicação móvel desconectada da Internet.	43
3.13	Módulo GPS NEO-6M V2.	45
3.14	Sensor HC-SR04.	46
3.15	Modo de funcionamento do sensor HC-SR04.	47
3.16	Sensor MQ-2.	47
4.1	Gráfico da taxa de crescimento do tempo de execução do procedimento compu- tacional.	66
4.2	Gráfico da taxa de crescimento do tempo de execução.	66
5.1	LILYGO TTGO T3 LoRa32 868MHz V1.6.1.	70
5.2	Diagrama de Pinos do LILYGO TTGO T3 LoRa32 868MHz V1.6.1.	71
5.3	<i>Layout</i> do circuito eletrónico no Kicad.	72
5.4	Esquema do circuito eletrónico no Fritzing.	73
5.5	Circuito do projeto na Breadboard.	73
5.6	Bateria 18650 de 3.7V e 2500mAh.	74
5.7	Módulo de <i>Step-Up</i> MT3608.	74
5.8	Painel solar de 3W 5V.	75
5.9	Painel de controlo AAPanel.	78
5.10	ChirpStack LoRaWAN Network-Server.	79
5.11	Classes LoRaWAN na plataforma ChirpStack.	81
5.12	Diferenças entre tecnologias WAN.	82
5.13	<i>Gateway</i> MikroTik LtAP LR8 LTE6 kit.	83
5.14	RouterOS 7.5.	83
5.15	Ilustração de como é calculado o nível de enchimento.	85
5.16	Email de alerta de deteção de fumo.	91
5.17	Notificação de alerta de deteção de fumo na aplicação móvel.	91
5.18	Contagem de Notificações da aplicação móvel.	91
5.19	Firebase Cloud Messaging.	92
5.20	Exemplo de validação de formulários na plataforma <i>web</i>	95
5.21	Exemplo de validação de formulários.	96
5.22	Exemplo de validação de formulários na plataforma <i>web</i>	96
5.23	Email de registo de novo utilizador.	97
5.24	<i>Headers</i> para placa de circuito impresso.	97
5.25	Circuito do projeto numa caixa.	98
5.26	Caixote do lixo com dispositivo e-Bin.	98
5.27	Ícone da aplicação no ecrã inicial no iOS e Android respetivamente.	99
5.28	Espaço ocupado pela aplicação e-Bin no iOS e no Android respetivamente. . .	99
5.29	Capturas de ecrã da aplicação móvel no iOS.	100
5.30	Capturas de ecrã da aplicação móvel no iOS.	100

5.31	Email para reposição de <i>password</i>	101
5.32	Capturas de ecrã da aplicação móvel no iOS.	101
5.33	Capturas de ecrã da aplicação móvel no Android.	102
5.34	Capturas de ecrã da aplicação móvel no iOS.	102
5.35	Capturas de ecrã da aplicação móvel no iOS.	103
5.36	Capturas de ecrã da aplicação móvel no iOS.	104
5.37	Capturas de ecrã da aplicação móvel no iOS.	104
5.38	Capturas de ecrã da aplicação móvel no iOS.	105
5.39	Capturas de ecrã da aplicação móvel no iOS.	105
5.40	Capturas de ecrã da aplicação móvel no iOS.	106
5.41	Capturas de ecrã da aplicação móvel no iOS.	106
5.42	Escolher nova foto de perfil no iOS.	107
5.43	Captura de ecrã da página de <i>login</i> da plataforma <i>web</i>	108
5.44	Captura de ecrã da página de reposição de <i>password</i> da plataforma <i>web</i>	108
5.45	Captura de ecrã da página de nova <i>password</i> da plataforma <i>web</i>	109
5.46	Captura de ecrã da página <i>dashboard</i> da plataforma <i>web</i>	109
5.47	Captura de ecrã da página <i>dashboard</i> da plataforma <i>web</i>	110
5.48	Captura de ecrã da página <i>dashboard</i> da plataforma <i>web</i>	110
5.49	Central de notificações da plataforma <i>web</i>	111
5.50	Captura de ecrã da página de leituras da plataforma <i>web</i>	111
5.51	Gráfico das últimas 10 leituras.	112
5.52	Captura de ecrã da página das rotas da plataforma <i>web</i>	112
5.53	Captura de ecrã da página de editar rota da plataforma <i>web</i>	113
5.54	Captura de ecrã da página de detalhes da rota da plataforma <i>web</i>	113
5.55	Captura de ecrã da página de detalhes da rota da plataforma <i>web</i>	114
5.56	Captura de ecrã da página de adicionar veículo da plataforma <i>web</i>	114
5.57	Captura de ecrã da página para gerir veículos da plataforma <i>web</i>	115
5.58	Captura de ecrã de apagar veículo na plataforma <i>web</i>	115
5.59	Captura de ecrã da página de adicionar dispositivo da plataforma <i>web</i>	116
5.60	Captura de ecrã da página de adicionar dispositivo da plataforma <i>web</i>	116
5.61	Captura de ecrã da página para gerir dispositivos da plataforma <i>web</i>	117
5.62	Captura de ecrã da página de apagar dispositivo da plataforma <i>web</i>	117
5.63	Captura de ecrã da página de editar dispositivo da plataforma <i>web</i>	118
5.64	Captura de ecrã da página de adicionar utilizador da plataforma <i>web</i>	118
5.65	Captura de ecrã da página para gerir utilizadores da plataforma <i>web</i>	119
5.66	Captura de ecrã da página de editar utilizador da plataforma <i>web</i>	119
5.67	Captura de ecrã da página de editar perfil da plataforma <i>web</i>	120
5.68	Captura de ecrã da página de ajuda da plataforma <i>web</i>	120
5.69	Captura de ecrã da página <i>dashboard</i> da plataforma <i>web</i>	121

LISTA DE FIGURAS

5.70	Captura de ecrã da página de leituras da plataforma <i>web</i> .	121
5.71	Captura de ecrã da página de rotas da plataforma <i>web</i> .	122
5.72	Captura de ecrã da página de ajuda da plataforma <i>web</i> .	122
5.73	Multímetro digital KEYSIGHT 34465A.	123
5.74	Resultado dos testes com Postman .	124
5.75	Resultado dos testes com pytest .	125
I.1	<i>Mockup</i> de alta fidelidade da página de <i>password</i> .	143
I.2	<i>Mockup</i> de alta fidelidade da página de reposição de <i>password</i> .	144
I.3	<i>Mockup</i> de alta fidelidade da página de nova <i>password</i> .	145
I.4	<i>Mockup</i> de alta fidelidade da <i>dashboard</i> .	146
I.5	<i>Mockup</i> de alta fidelidade da página de leituras.	147
I.6	<i>Mockup</i> de alta fidelidade da página de rotas.	148
I.7	<i>Mockup</i> de alta fidelidade da página de detalhes da rota.	149
I.8	<i>Mockup</i> de alta fidelidade da página de editar rota.	150
I.9	<i>Mockup</i> de alta fidelidade da página de adicionar veículos.	151
I.10	<i>Mockup</i> de alta fidelidade da página para gerir veículos.	152
I.11	<i>Mockup</i> de alta fidelidade da página de editar veículo.	153
I.12	<i>Mockup</i> de alta fidelidade da página de adicionar dispositivo.	154
I.13	<i>Mockup</i> de alta fidelidade da página para gerir dispositivos.	155
I.14	<i>Mockup</i> de alta fidelidade da página de editar dispositivo.	156
I.15	<i>Mockup</i> de alta fidelidade da página de adicionar utilizador.	157
I.16	<i>Mockup</i> de alta fidelidade da página para gerir utilizadores.	158
I.17	<i>Mockup</i> de alta fidelidade da página de editar utilizador.	159
I.18	<i>Mockup</i> de alta fidelidade da página de visualizar rotas.	160
I.19	<i>Mockups</i> de média fidelidade da aplicação móvel.	161
I.20	<i>Mockups</i> de alta fidelidade da aplicação móvel.	162
I.21	<i>Mockups</i> de média fidelidade da aplicação móvel.	163
I.22	<i>Mockups</i> de alta fidelidade da aplicação móvel.	164
I.23	<i>Mockups</i> de média fidelidade da aplicação móvel.	165
I.24	<i>Mockups</i> de alta fidelidade da aplicação móvel.	166
I.25	<i>Mockups</i> de média fidelidade da aplicação móvel.	167
I.26	<i>Mockups</i> de alta fidelidade da aplicação móvel.	168
I.27	<i>Mockups</i> de média fidelidade da aplicação móvel.	169
I.28	<i>Mockups</i> de alta fidelidade da aplicação móvel.	170
II.1	Execução do procedimento computacional VRP.	174
II.2	<i>Output</i> do procedimento computacional VRP.	175

Lista de Tabelas

2.1	Vantagens e desvantagens das tecnologias de comunicação.	8
2.2	Comparação de tecnologias de comunicação.	12
5.1	Comparação de tecnologias de comunicação WAN.	81
II.1	Matriz de custos.	171
II.2	Procura de clientes.	171
II.3	Inicialização.	172
II.4	Melhores rotas.	174

Lista de Listagens

4.1	Pseudocódigo do procedimento computacional Rand-MER	52
4.2	Pseudocódigo da validação para fazer a fusão.	52
4.3	Função para calcular poupanças e ordenar.	54
4.4	Função destinada a aleatorizar as poupanças.	54
4.5	Função para validar se é possível fundir rotas.	55
4.6	Atualização dos dados para fazer a fusão de 2 rotas.	56
4.7	Processo multi-start do procedimento computacional Rand-MER	59
4.8	Código da função <code>compute savings list</code> em Julia	61
4.9	Código da função <code>biased randomization</code> em Julia	61
4.10	Código da função <code>validate merge constraints</code> em Julia	62
4.11	Código da função Rand-MER em Julia	64
4.12	Função destinada a aleatorizar as poupanças.	65
5.1	Ficheiro <code>docker-compose.yml</code>	78
5.2	Descodificar os dados JSON em Base64.	85
5.3	Lê e atualiza leituras na base de dados.	86
5.4	Emitir evento SocketIO	87
5.5	Adicionar os dados ao ficheiro CSV.	87
5.6	Verificar se os valores estão fora dos parâmetros normais e notifica os utilizadores.	91
5.7	Código para proteger os <i>endpoints</i> da aplicação móvel com <i>tokens</i> JWT em python	94
5.8	Código para proteger os <i>endpoints</i> da aplicação móvel com <i>tokens</i> JWT em python	94
5.9	Código para proteger os <i>endpoints</i> da aplicação móvel com <i>tokens</i> JWT em flutter	95
5.10	Teste com pytest	124
5.11	Teste SocketIO	126

Capítulo 1

Introdução

A poluição ambiental atinge níveis alarmantes devido ao rápido crescimento da população mundial e das atividades humanas, por causa das tecnologias relativamente recentes, e em consequência, os governos de vários países têm enfrentado dificuldades para gerir os resíduos, independentemente dos recursos humanos ou físicos [Kar+19].

Uma das manifestações da poluição é o lixo urbano, e neste contexto, foram verificados vários problemas sendo o principal a dificuldade em controlar o nível de enchimento dos caixotes do lixo numa cidade, dada a diversidade de caixotes.

A gestão de resíduos sólidos municipais continua a ser um grande problema nas áreas urbanas, causando sérios problemas de saúde e ambientais [Ram+21].

Ao utilizar o sistema convencional de recolha de resíduos, os trabalhadores têm de se deslocar a todos os caixotes para verificar o nível de enchimento do mesmo. Este sistema dá origem a custos elevados com o veículo e desgaste do mesmo, com o combustível e o custo com os trabalhadores e ainda aumenta a poluição do ambiente.

A utilização de um caixote inteligente possibilita uma recolha otimizada dos caixotes do lixo, e assim, previne-se que estes sejam despejados quando ainda não estão cheios, evitando deslocações desnecessárias e que estes caixotes fiquem sobrelotados.

No modo de recolha tradicional, os caixotes podem transbordar, espalhando-se pela área envolvida, o que causa mau aspeto, polui o meio ambiente e causa transtornos à população. Isto deve-se aos sistemas ineficientes de recolha de resíduos, que resultam em poluição por cheiros, proliferação de insetos, aparecimento de animais que poderão originar várias doenças, com origem nos caixotes sobrelotados que não são recolhidos [Zai+20]. Estes problemas levam à vantagem da criação de um sistema de monitorização que permita gerir o nível de lixo num município para que estes possam ser recolhidos de forma eficiente [Ram+21].

As cidades geram um nível elevado de resíduos e devem ser recolhidos dentro do prazo. O planeamento da rota deve ser feito em tempo real com base no nível de enchimento atual dos caixotes [MP17].

Os caixotes inteligentes são essenciais para a gestão eficiente de resíduos nas cidades,

sendo assim possível reduzir o uso de trabalhadores, a distância geral da rota e o número de veículos, baixando os custos [MR20].

1.1 Enquadramento

Esta dissertação é dedicada ao tema da melhoria da recolha dos resíduos urbanos usando IoT (*Internet of Things*). Desenvolveu-se um projeto que permite recolher dados provenientes dos sensores implementados nos caixotes do lixo e implementar uma recolha inteligente.

O caixote inteligente evita que os camiões se desloquem todos os dias aos locais de recolha, uma vez que tal só é necessário quando a sua capacidade máxima estiver perto de ser atingida. Encontra-se também desenvolvida uma plataforma *web* e uma aplicação móvel.

A escolha deste tema deve-se à quantidade de resíduos produzidos ter vindo a aumentar constantemente sendo fundamental que se estabeleçam medidas para que estes sejam tratados de forma eficiente.

É essencial implementar caixotes do lixo inteligentes com sensores que detetem o nível de enchimento dos resíduos, de forma a melhorar os percursos dos camiões, diminuindo os custos associados, uma vez que só são recolhidos quando a sua capacidade máxima estiver perto de ser atingida.

O projeto de dissertação foi realizado com o apoio do Center for Sci-Tech Research in Earth System and Energy (CREATE)/SEPSI Laboratory - Polytechnic University of Beja, 7800-295 Beja, Portugal.

1.2 Trabalho do autor

Nesta dissertação apresenta-se um sistema de monitorização e otimização da recolha de resíduos baseado numa arquitetura IoT.

Este sistema tem como principais características: o baixo custo; o baixo consumo de energia; ser escalável; permitir a instalação em caixotes de diferentes tipos e ser um dispositivo com dimensões reduzidas.

Pretende-se tornar os contentores comuns em contentores inteligentes, através dum dispositivo colocado no interior de cada caixote. Este dispositivo deverá ser universal, para ser implementado em vários tipos de contentores, independentemente do seu tamanho ou formato.

Este dispositivo permite a monitorização de dados relevantes dos caixotes: a sua localização; o nível de enchimento; a deteção de fumo e o nível de bateria.

Existem relatos de que vários caixotes foram incendiados, por isso, para prevenir estes incidentes, este dispositivo integra um sensor que deteta a presença de gás e fumo e notifica

os utilizadores responsáveis para que possam tomar medidas de modo a evitar estragos maiores.

Este projeto deve permitir mover livremente os caixotes de um ponto da cidade para o outro e não ser necessária efetuar alteração. Isto devido à utilização do módulo GPS que permite recolher a localização em tempo real do caixote e atualizá-la no sistema, para que caso seja criada uma rota, esta tenha em conta a nova localização e assim adaptar o percurso. A utilização do módulo GPS, também permite saber se o caixote foi furtado.

Desenvolveu-se também um ecossistema interno, que pode ser adotado por qualquer município ou empresa deste ramo. Este ecossistema deve incluir uma componente física colocada no interior de cada caixote, que agrupa todos os componentes eletrónicos, uma plataforma *web* e uma aplicação móvel de fácil utilização e uma API (*Application Programming Interface*). Neste projeto a API tem como principal objetivo realizar operações de leitura e escrita na base de dados.

Este sistema conta com um sistema de permissões que mostra e oculta dados e funcionalidades dependendo do cargo que é atribuído a cada utilizador.

Desenvolveu-se um procedimento computacional de VRP (*Vehicle Routing Problem*) como parte complementar da dissertação para que sejam criadas e mostradas rotas otimizadas em que os camiões só se dirijam aos caixotes onde a sua capacidade máxima esteja perto de ser atingida. A figura 1.1 ilustra graficamente o percurso do camião.

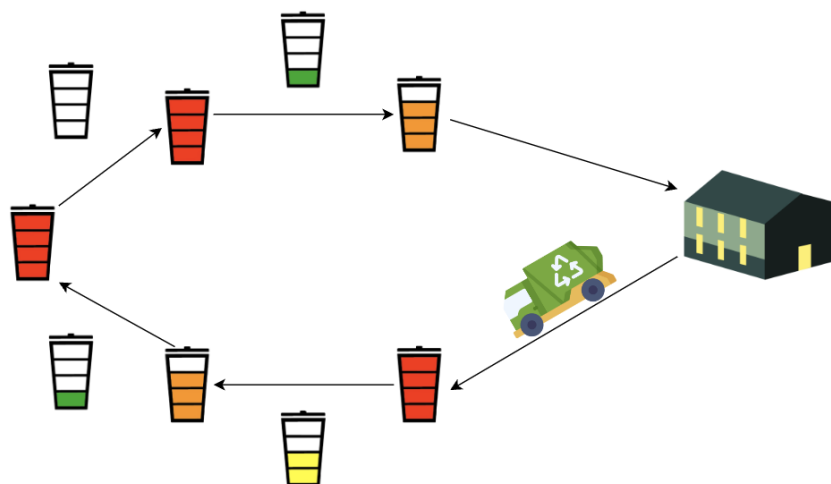


Figura 1.1: Percurso dos camiões com rotas otimizadas.

Foram desenvolvidas duas versões do VRP, uma na linguagem de programação **Python** e outra em **Julia**. O seu desenvolvimento conduziu ao estudo experimental de complexidade computacional.

1.3 Estrutura do documento

A presente dissertação de mestrado encontra-se organizada em seis capítulos, incluindo o presente capítulo da introdução e o capítulo da conclusão.

Capítulo 1 - Introdução

No capítulo da introdução é feito o enquadramento da dissertação que descreve os principais problemas que motivaram o desenvolvimento deste projeto, apresenta o trabalho do autor e por último a estrutura do documento.

Capítulo 2 - Estado da Arte

No capítulo 2 é feita uma contextualização sobre as Cidades Inteligentes e a Internet das Coisas, seguido do levantamento bibliográfico de projetos semelhantes. Ainda neste capítulo, serão apresentados alguns sistemas académicos e comerciais.

Capítulo 3 - Arquitetura do Sistema

Descreve a arquitetura de *software* proposta deste sistema baseada em IoT, e são apresentados os requisitos do sistema, diversos diagramas, a análise de tarefas, os *mockups* da plataforma *web* e da aplicação móvel e as linguagens de programação. De seguida, serão apresentados os sensores e módulos utilizados.

Capítulo 4 - *Vehicle Routing Problem*

Neste capítulo, é explicado o procedimento computacional de VRP utilizado, o pseudocódigo, o código em *Python* e em *Julia*, e por último, os gráficos da taxa de crescimento do tempo de execução.

Capítulo 5 - Implementação Experimental

O capítulo 5 incide numa vertente mais aplicada desta dissertação, ou seja, a fase da implementação experimental. Este capítulo apresenta o circuito deste projeto, a parte da alimentação e programação para baixo consumo energético, a comunicação com o servidor, sistema de gestão de base de dados, a parte da segurança informática, e de envio de *emails*. Este capítulo ainda contém a análise dos resultados, os testes realizados, bem como a discussão geral dos resultados.

Capítulo 6 - Conclusão

Este capítulo final apresenta uma síntese geral sobre os temas abordados ao longo da dissertação e propostas para trabalho futuro.

Capítulo 2

Estado da Arte

2.1 Introdução

A elaboração desta dissertação motivou o estudo e o planeamento de soluções ótimas para melhorar a recolha de resíduos.

De acordo com a literatura existente, foram desenvolvidos vários projetos para minimizar este problema.

Na Secção 2.2 serão apresentados os componentes e características das cidades inteligentes, as tecnologias mais utilizadas e alguns exemplos de cidades inteligentes em Portugal.

De seguida, na Secção 2.3 será abordado a IoT, incluindo os desafios e algumas tecnologias IoT.

Na Secção 2.4 serão apresentados sistemas que motivaram a realização deste projeto, bem como as características e funcionalidades presentes neles. Na Secção 2.5 serão apresentados alguns projetos académicos e na Secção 2.6 serão apresentados alguns sistemas comerciais. Por último na Secção 2.7 será mostrada a análise e evolução histórica destes sistemas.

2.2 Cidades Inteligentes

No final da década de 2000 e com a crise económica sentida nos últimos anos no mundo, diversas cidades tomaram uma série de medidas para diminuir custos financeiros. Estas medidas passaram pela redução de custos, a melhoria dos serviços nas comunidades e o desenvolvimento de cidades inteligentes [KL09].

O conceito de *Smart City* surgiu por volta dos anos 90 e nessa época o foco principal era compreender a forma como as TIC (Tecnologias de Informação e Comunicação) poderiam beneficiar as infraestruturas modernas no interior de uma cidade. O *California Institute for Smart Communities* iniciou o estudo nesta vertente de tentar entender de que modo as comunidades se poderiam tornar inteligentes e dessa forma as cidades fossem planeadas tendo por base a implementação das TIC [ABD15].

A *Smart City* possui diversas definições, não havendo um consenso entre os autores, uma vez que cada um define este conceito tendo em conta as suas perspetivas [Hal+00].

Segundo Hall, as cidades inteligentes são cidades que supervisionam as infraestruturas fundamentais, isto é, as estradas, os túneis, os aviões, as comunicações, a água, a energia e as pontes. As cidades inteligentes possuem uma eficiente gestão e manutenção dos seus recursos, em conjunto com o controlo de aspetos associados à segurança dos indivíduos [Hal+00].

Albino e colaboradores [ABD15], define cidade inteligente como sistemas nervosos digitais que obtêm e conectam dados de fontes heterogêneas, como sensores em hospitais, esgotos, semáforos, escolas, entre outros.

Smart City é considerado um conceito de IoT aplicado às cidades, onde diferentes tipos de sensores, comunicam através da Internet, recolhendo dados pertinentes relativamente a uma determinada cidade. Os sistemas tem como principal objetivo melhorar a gestão da cidade, fornecer informações relevantes à população em tempo real e melhorar a sustentabilidade energética e conforto das pessoas [OPB21].

Sotiris Zygiaris [Zyg12] desenvolveu um dos modelos mais estudados de cidades inteligentes, designado por *Smart City Reference Model*. Este modelo subdivide-se em sete camadas determinadas pelo desenvolvimento das cidades, assim como os seus focos. As camadas são:

1. A inovação, relacionada com as questões financeiras e tecnológicas, visando um crescimento inteligente;
2. As aplicações ligadas à tecnologia e a promoção da inteligência na vida real;
3. A integração, ou seja, transformar a cidade mais inclusiva;
4. A instrumentalização, que requer o uso de sensores e a IoT, de modo a facilitar a utilização dos dados em tempo real;
5. A interconexão que diz respeito a uma boa ligação Wi-Fi;
6. O ambiente, uma vez que é importante a cidade possuir uma pegada ecológica sustentável, tornando-a assim numa “cidade verde”;
7. A cidade que visa a identificação territorial e o planeamento inteligente.

2.2.1 Componentes e Características das Cidades Inteligentes

Existem diversos componentes de uma cidade inteligente, nomeadamente, as infraestruturas inteligentes, os edifícios inteligentes, os transportes inteligentes, a energia inteligente, a assistência médica inteligente, a educação inteligente, a tecnologia inteligente, o governo inteligente, entre outros [Moh16].

Os diversos componentes das cidades inteligentes abrangem a sustentabilidade que visa a energia e as alterações climáticas, a poluição e os resíduos, as questões sociais e de saúde, a qualidade de vida, a urbanização e a inteligência que é determinada pela melhoria de padrões económicos, sociais e ambientais das cidades e dos seus habitantes [Moh16].

Existem quatro temas fundamentais para uma *Smart City*, nomeadamente, a sociedade, a economia, o meio ambiente e o governo. A sociedade significa que a cidade é para os seus habitantes, a economia relaciona-se com o potencial de uma cidade inteligente para favorecer o crescimento económico e o crescimento de empregabilidade, o meio ambiente refere-se ao funcionamento de uma cidade inteligente na atualidade e no futuro e o governo visa a capacidade de uma cidade inteligente de administrar políticas [Moh16].

As cidades inteligentes incluem uma infraestrutura física, as TIC e os serviços. A infraestrutura física, designa-se como um elemento não inteligente da *Smart City* que se refere aos edifícios, estradas, linhas ferroviárias e sistemas de abastecimento de água. A infraestrutura das TIC é um componente inteligente fundamental na *Smart City*, uma vez que conecta todos os outros componentes. A infraestrutura de serviço é assente na infraestrutura física e pode possuir certos componentes das TIC [Moh16].

2.2.2 Tecnologias utilizadas na *Smart City*

De acordo com Schaffers e colaboradores [Sch+11], a organização dos serviços eletrónicos nos ecossistemas de inovação das cidades devem ser realizados através de tecnologias, nomeadamente, ambientes multissensoriais, aplicações que sustentem a realidade aumentada, plataformas *open-source* que facilitam a divisão de armazenamento de dados e a localização de conteúdos tendo em conta a localização do indivíduo. Estes autores salientam algumas empresas que desempenham um papel fundamental na construção e monitorização das cidades inteligentes, como a *Microsoft*, a *Cisco* e a *IBM*.

As cidades inteligentes possuem diversas tecnologias com o objetivo de monitorizar e recolher dados [Pan+13].

De acordo com a literatura existente, existem várias tecnologias de transmissão de dados para serem utilizadas numa cidade inteligente, nomeadamente *Bluetooth*, *Z-Wave*, *ZigBee*, *LoRa*, *Wi-Fi*, *WiMAX*, *LTE* e *LTE-A* [Yaq+17].

A tabela 2.1 mostra algumas vantagens e desvantagens das tecnologias de comunicação utilizadas em cidades inteligentes [Yaq+17].

Tabela 2.1: Vantagens e desvantagens das tecnologias de comunicação (*em* [Yaq+17]).

Tecnologia	Vantagens	Desvantagens
Wi-Fi	Tecnologia sem fios; O utilizador pode mover-se livremente.	Raio de serviço limitado; Está mais sujeita a interferências e perda de dados.
WiMax	Permite a comunicação sem fio de alta velocidade; Ampla área de cobertura;	Custo de instalação elevado.
LTE	Existe compatibilidade com versões anteriores e futuras; Reduz o atraso na conexão com a Internet;	Tem um custo elevado imposto pelas operadoras.
LTE-A	Permite altas taxas de dados;	Tem um custo elevado imposto pelas operadoras.
Bluetooth	Tecnologia de baixo custo; Fácil de instalar.	Só permite comunicação de curto alcance.
ZigBee	Baixo consumo de energia; Tecnologia de baixo custo.	É mais lento que os concorrentes.
Z-Wave	Mais simples que o ZigBee	Pouca mobilidade; Contém falhas de segurança.
LoRa	Baixo consumo de energia; Permite comunicação bidirecional com os sensores; É uma tecnologia aberta; É escalável; É uma tecnologia segura; É uma tecnologia de baixo custo.	Não é adequado para aplicações que exijam baixa latência.

2.2.3 Cidades inteligentes em Portugal

A tecnologia LoRa, tem sido adotada por várias cidades, como o caso de Aveiro que pretende implementar *gateways* LoRa, de modo a estabelecer a comunicação com uma plataforma *web* e duas estações para monitorizar vários parâmetros ambientais [CTM21]. Em Aveiro, a implementação de caixotes de lixo inteligentes, foi feita com base no projeto **Aveiro STEAM City**, que integra a tecnologia IoT para gestão urbana. Os caixotes de lixo inteligentes encontram-se localizados no centro da cidade e junto do *campus* universitário. Os sensores instalados permitem a recolha de dados sobre os níveis de enchimento, possibilitando otimizar as rotas e reduzir a pegada ecológica associada à recolha de lixo [Cit22].

Em Águeda, foi desenvolvido o projeto **Laboratório vivo para a Descarbonização Águeda Smart City Lab**, onde foram instalados sensores para avaliar o volume dos resíduos nos ecopontos existentes, melhorando assim a recolha e análise dos resíduos [Águ20].

Em 2019, em Esposende foi desenvolvido o projeto **Esposende Smart City**, que visa recolher dados, como os níveis de resíduos através das tecnologias de comunicação IoT [Car21].

Em Braga, também foram colocados caixotes de lixo inteligentes. O projeto foi desenvolvido pela empresa **Braval**, que instalou um conjunto de sensores em contentores subterrâneos. Este projeto permitiu reduzir a frequência das recolhas e melhorar a eficiência da gestão de resíduos. Além disso, estes caixotes contêm um sistema de prensagem, que permite armazenar mais resíduos antes de qualquer recolha [Bra22].

Em Guimarães, também foram instalados caixotes inteligentes através do programa **Guimarães Mais Verde**. Estes caixotes foram desenvolvidos em parceria com o projeto europeu **Urban-Waste**. A cidade instalou caixotes equipados com sensores que permitem a monitorização em tempo real dos níveis de enchimento [Gui21].

Na cidade do Porto, a empresa responsável pela gestão de resíduos **Porto Ambiente** implementou um sistema inteligente nos diversos caixotes da cidade. Este sistema permite planear rotas mais eficientes e reduzir os custos operacionais, contribuindo para um ambiente mais limpo [Amb22].

Em Viseu, os contentores são equipados com sensores que medem o nível de enchimento. Além disso, o sistema notifica quando os caixotes se encontram cheios, ajudando a reduzir os custos operacionais e a poluição [Vis23].

Em Sintra, a **Empresa Municipal de Ambiente de Sintra** implementou um sistema nos caixotes de lixo, com sensores de enchimento e tecnologia de prensagem e painéis solares. Esta projeto faz parte do programa **Sintra Sustentável**, que tem o objetivo de também melhorar a gestão de resíduos. O sistema é monitorizado em tempo real, o que permite otimizar as rotas dos camiões de lixo [Sin23].

Em Cascais, também foram implementados caixotes do lixo inteligentes, em colaboração com a **Cascais Ambiente**, foram instalados diversos caixotes com sensores de enchimento e painéis solares. Este sistema permite o envio de alertas quando os caixotes estão cheios, de forma a evitar recolhas desnecessárias e otimizar os percursos de gestão de resíduos [Cas21].

A Câmara Municipal de Lisboa incentiva os serviços e empresas municipais, *startups*, empresas tradicionais, universidades públicas e privadas a preparar projetos para usufruir da rede LoRa disponibilizada pelo município. Esta rede é aberta e gratuita para todos os que a pretendam utilizar. A existência desta rede permite monitorizar parâmetros ambientais, controlar o nível de enchimento dos contentores, entre outros [CSC21]. Em Lisboa, a empresa **EcoGestus** implementou diversos caixotes de lixo inteligentes, permitindo a monitorização dos contentores de lixo urbano através de sensores de enchimento [Eco22].

No Funchal, foram implementados caixotes de lixo inteligentes em áreas turísticas. Estes caixotes, além dos sensores de enchimento, que permitem reduzir as recolhas, utilizam também painéis solares. Esta iniciativa faz parte do plano de sustentabilidade **Funchal Eco** [Fun20].

2.3 Internet das Coisas

A definição de Internet das Coisas encontra-se em constante evolução, não existindo um consenso entre autores. Uma *thing* na IoT refere-se a qualquer dispositivo capaz de recolher dados e transmiti-los através da rede de forma autónoma [Mou21].

A Internet das Coisas consiste numa rede de objetos inteligentes que conseguem partilhar informações, dados e recursos e agir diante de situações e mudanças no ambiente [MRT15].

A tecnologia incorporada no dispositivo ajuda a interagir com os estados internos e o ambiente externo, o que por sua vez auxilia no processo de tomada de decisão [Mou21].

Segundo o dicionário *Oxford*, a IoT consiste na interligação através da Internet de dispositivos de computação incorporados em objetos do quotidiano, permitindo-lhes enviar e receber dados [Mou21].

2.3.1 Desafios da aplicação da IoT nas Cidades Inteligentes

As cidades inteligentes encontram-se expostas a um conjunto de fragilidades que poderão colocar toda a cidade em risco, como as falhas e *bugs*, a fiabilidade do sistema, a resiliência, problemas relacionados com os *backups* e até mesmo de segurança informática [Bai+17].

Ao implementar uma Cidade Inteligente, existem alguns desafios que podem surgir. Estes desafios passam pela segurança e a confidencialidade das informações dos utilizadores, pelos recursos financeiros e pelas dificuldades em criar e manter infraestruturas, sistemas e plataformas e pelos desafios relacionados com a implementação de grandes sistemas distribuídos como a escalabilidade e a diversidade [KZ16]. Os sistemas também devem ter um tempo de processamento aceitável.

Privacidade

Devido aos dados dos utilizadores serem manipulados pela infraestrutura, pelas aplicações e pelas plataformas de uma *Smart City*, estes devem ser mantidos em segurança, para evitar que utilizadores com más intenções consigam aceder aos dados, por isso, é fundamental que os utilizadores sejam alertados de eventuais alterações [KZ16].

Segurança

Além de garantir a privacidade dos dados dos utilizadores, é importante que a infraestrutura seja segura, na medida em que, não só se deve prevenir de ataques à segurança dos sistemas, como também se deve recuperar dados. Estes ataques são, principalmente, ataques à infraestrutura de *hardware*, onde se alteram valores das leituras dos sensores e a vandalização nos equipamentos físicos das cidades [KZ16].

Gestão de dados

Um dos principais desafios é a quantidade de dados processados. Estes dados podem ser estruturados, semiestruturados ou não estruturados.

Os dados estruturados dizem respeito às informações dos indivíduos. Os dados semiestruturados relacionam-se com aos dados de leituras de sensores. Os dados não estruturados referem-se, por exemplo, às imagens de câmaras de trânsito e de segurança [KZ16].

Um dos principais desafios da gestão de dados é o armazenamento, pois a quantidade de dados produzidos numa cidade é muito elevada e por isso devem existir ferramentas que auxiliem no armazenamento e na recuperação de dados. Outro desafio, passa por conseguir um tempo de processamento aceitável [KZ16].

Escalabilidade

Um dos principais desafios na implementação de plataformas é assegurar que sejam escaláveis, uma vez que se espera que a quantidade de dados recolhidos aumente sucessivamente. Isto ocorre devido à colocação de novos dispositivos, à criação de novos serviços, aplicações e ao aumento da população [KZ16]. Os novos dispositivos devem ser retrocompatíveis e as versões devem ser guardadas ao longo dos anos.

Custo/Benefício

O desenvolvimento de infraestruturas das cidades inteligentes requer que exista algum poder económico para a construção de todos os componentes necessários, nomeadamente, a aquisição e instalação dos dispositivos pelas cidades. Estes podem ser sensores, atuadores, servidores e equipamentos de comunicação, desenvolvimento do *software*, criação de equipas para manter e gerir e a adaptação às mudanças nos processos da cidade. Os benefícios da implementação são económicos e sociais, uma vez que a qualidade de vida da população melhora, porém alguns benefícios só são visíveis para a população a longo prazo [KZ16].

Colaboração

Um desafio fundamental é incentivar as populações a utilizarem os serviços disponibilizados pelas cidades e a partilhar dados e informações sobre a sua experiência a utilizar os mesmos. A maioria dos serviços disponibilizados dependem da aderência da população [KZ16].

Interoperabilidade

A interoperabilidade consiste na capacidade de um sistema, dispositivo e redes diferentes interagirem e comunicarem entre si informações importantes. Esta desempenha um papel essencial para transmitir a conectividade entre vários dispositivos que atuam com diversas tecnologias de comunicação [Meh+17].

A interoperabilidade tornou-se um desafio devido à ausência de padrões universais, contudo para minimizar este desafio é essencial utilizar uma abordagem inteligente para fornecer conectividade a diversos dispositivos IoT [Meh+17].

2.3.2 Tecnologias IoT

Uma das razões a ter em conta na hora de escolher uma tecnologia IoT é a frequência ou banda de transmissão com que trabalham, o alcance, as taxas de dados transmitidas, bem como a topologia [Yaq+17]. A tabela 2.2 compara as tecnologias de comunicação utilizadas na *Smart City*.

Tabela 2.2: Comparação de tecnologias de comunicação (*em* [Yaq+17]).

Tecnologia de comunicação	Padrões	Frequência	Alcance	Taxas de dados	Topologia
Bluetooth	IEEE 802.15.1	2.4 GHz	1-100 m	1 Mb/s	Ponto-a-ponto
Z-Wave	-	900 MHz	100 m	9.6-100 Kb/s	Estrela, <i>cluster</i> , <i>mesh</i>
ZigBee	IEEE 802.15.4	2.4 GHz	10-20 m	250 Kb/s	<i>mesh</i>
LoRa	LoRa Alliance	868 MHz (Europa / América do Norte), 433 MHz (Ásia)	2-5 Km	290 b/s - 50 Kb/s	Estrela
Wi-Fi	IEEE 802.11	2.4 GHz, 3.6 GHz, 4.9 GHz, 5 GHz, 5.9 GHz	100 m	1-54 Mb/s	Estrela
WiMax	3GPP	3.5 GHz	50 Km	75 Mb/s	Ponto-a-multiponto, <i>mesh</i>
LTE	3GPP	2.5 GHz, 5 GHz, 10 GHz	30 Km	300 Mb/s (DL), 75 Mb/s (UL)	Estrela
LTE-A	3GPP	2.5 GHz, 5 GHz, 10 GHz, 15 GHz, 20 GHz	30 km	1Gb (DL), 500 Mb/s (UL)	Ponto-a-ponto

2.4 Levantamento bibliográfico

De acordo com a literatura existente é possível observar que existem diversos projetos semelhantes de caixotes do lixo inteligentes. Existem múltiplos projetos e sistemas comerciais para gerir os resíduos e todos têm os mesmos objetivos, porém o que diverge entre estes são maioritariamente as tecnologias utilizadas. Em seguida serão apresentados alguns projetos.

2.4.1 Projeto IoT-TBLM

O projeto IoT-TBLM (*IoT-Trash Bin Level Monitoring*) apresenta um sistema de monitorização do nível de enchimento de um caixote. Este é alimentado através de uma bateria e comunica através da tecnologia LoRa [Ram+21].

O dispositivo mede o nível não preenchido do caixote e a sua localização, processa os dados e transmite para um *gateway* a uma frequência de 915 MHz. O *gateway* permite receber informação dos dispositivos LoRa e encaminha os dados para o servidor [Ram+21].

Através de uma interface gráfica, os utilizadores podem visualizar e analisar o nível de enchimento de cada caixote, bem como a sua localização [Ram+21].

A arquitetura do projeto IoT-TBLMS, foi a base da arquitetura de rede proposta para o projeto e-Bin. Na figura 2.1 é possível observar essa arquitetura.

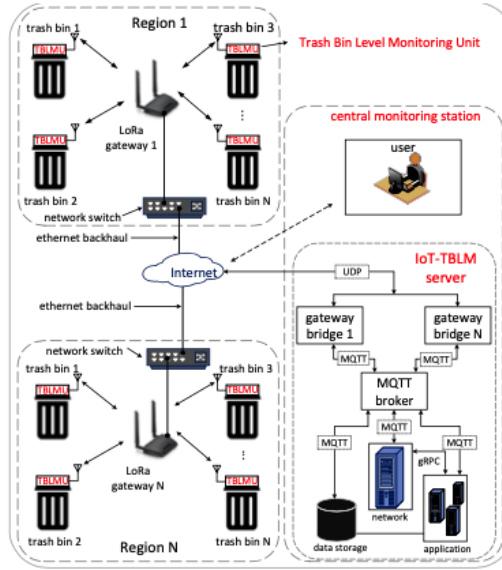


Figura 2.1: Arquitetura de rede proposta por IoT-TBLMS [Ram+21].

2.4.2 Projeto IoTfaskimUm

O projeto IoTfaskimUm, pretende superar o problema da ineficiência do sistema de recolha de resíduos. Este protótipo de um caixote inteligente baseado na IoT, pode ser utilizado nas Cidades Inteligentes [Zai+20].

Este projeto conta com um sistema de monitorização centralizado automático que fornece o nível de enchimento de um caixote, permite o planeamento ideal das rotas para recolha dos resíduos, o que reduz os tempos de recolha e economiza custos e o consumo de combustível [Zai+20].

Este projeto destaca-se pela utilização de uma plataforma *web* com uma *dashboard*, onde é visível a localização dos caixotes no mapa, como é mostrado na figura 2.2.

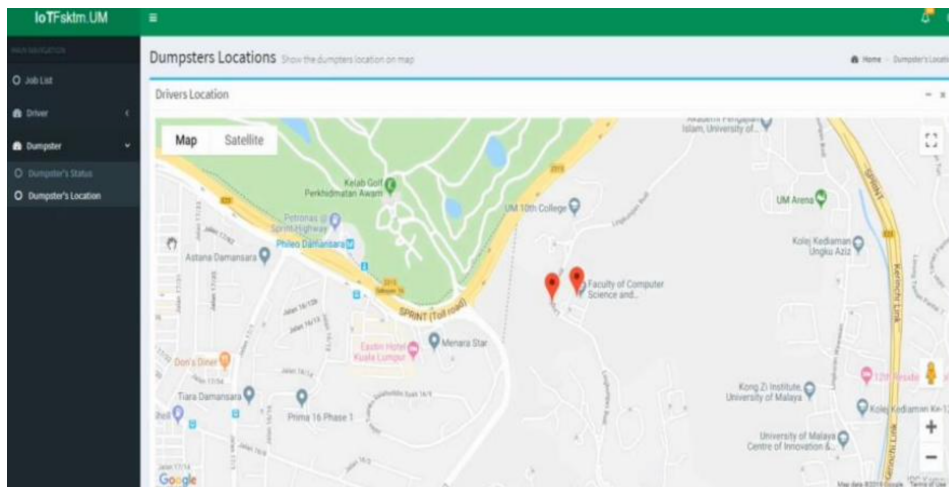


Figura 2.2: Interface da plataforma *web* deste sistema [Zai+20].

2.4.3 Outros projetos

O seguinte projeto foi desenvolvido por Addabbo e colaboradores [Add+19], utiliza sensores ultrassônicos à prova de água e tecnologia LoRa para a gestão de resíduos. Este projeto resultou na criação de uma caixa personalizada, implementada na tampa, apresentada na figura 2.3.



Figura 2.3: Caixote Inteligente [Add+19].

O sistema proposto por Haque e colaboradores [Haq+20], monitoriza os caixotes em tempo real e determina quais devem ser recolhidos em cada rota da recolha de resíduos. Este sistema diminui a distância percorrida em média 30,76%, em comparação com o sistema tradicional de recolha de resíduos e assim, reduz o custo de combustível e o trabalho humano, tornando o sistema otimizado e eficiente, permitindo a monitorização em tempo real e a navegação aprimorada. Na figura 2.4 é possível observar o caixote e o circuito deste projeto.



Figura 2.4: Caixote inteligente [Haq+20].

O projeto desenvolvido por Mishra e colaboradores [MR20], utiliza os dados de sensores de caixotes e o tráfego rodoviário em tempo real e através do algoritmo *Dijkstra*, obtém as rotas. Porém, este algoritmo não é indicado para este tipo de problemas, porque não tem em conta as capacidades dos caixotes.

O Algoritmo *Dijkstra* foi proposto em 1959 pelo cientista da computação holandês e é um algoritmo que não precisa passar por todos os nós para encontrar a rota mais curta [DZ16]. Neste projeto, a matriz de custos é obtida através da API do Google [MR20].

A otimização de rota proposta para a cidade de *Bhubaneswar*, na Índia, permitiu obter uma poupança de aproximadamente 30,28% na distância da rota [MR20]. Na figura 2.5 é possível observar o caixote deste projeto.



Figura 2.5: Caixote do lixo inteligente e seus componentes [MR20].

O seguinte projeto, desenvolvido por Oralhan e colaboradores [OOY17], pretende melhorar a qualidade das recolhas de resíduos em Kayseri, na Turquia. Este inclui sensores integrados para medir o nível de enchimento do caixote, bem como a sua temperatura e o nível de dióxido de carbono.

Neste projeto, para a otimização das rotas foi utilizado um algoritmo inspirado em colônia de formigas, em que a rota de recolha de lixo mais eficiente é enviada para os motoristas do caminhão de lixo. Este projeto utiliza *Data Mining* para prever quando o caixote do lixo pode ficar cheio. O sistema apresentou uma poupança de aproximadamente 30% nos custos da recolha de resíduos [OOY17].

A figura 2.6, mostra a caixa que foi desenvolvida para incorporar o circuito deste projeto.



Figura 2.6: Caixa do projeto [OOY17].

O projeto, desenvolvido por Wahyutama e colaboradores [WH22], conta com um **Arduino** interligado a um conjunto de sensores para medir o peso e o volume para detetar a capacidade de um caixote e um módulo GPS para obter a localização deste. Neste projeto, também foi utilizado um **Raspberry Pi** para classificar o lixo.

A utilização de um **Raspberry Pi** por caixote, permite adicionar mais funcionalidades, porém os custos são muito mais elevados do que utilizar um microcontrolador de baixo custo.

Neste projeto os dados são armazenados no SGBD (Sistema de Gestão de Bases de Dados) *Firestore Realtime Database* e a tecnologia de comunicação utilizada é a LoRa [WH22]. Também foi desenvolvida uma aplicação móvel para monitorizar em tempo real várias informações, como os detalhes da capacidade, a localização e também, o gráfico do histórico de cada caixote [WH22].

A figura 2.7 mostra as interfaces da aplicação móvel deste projeto.

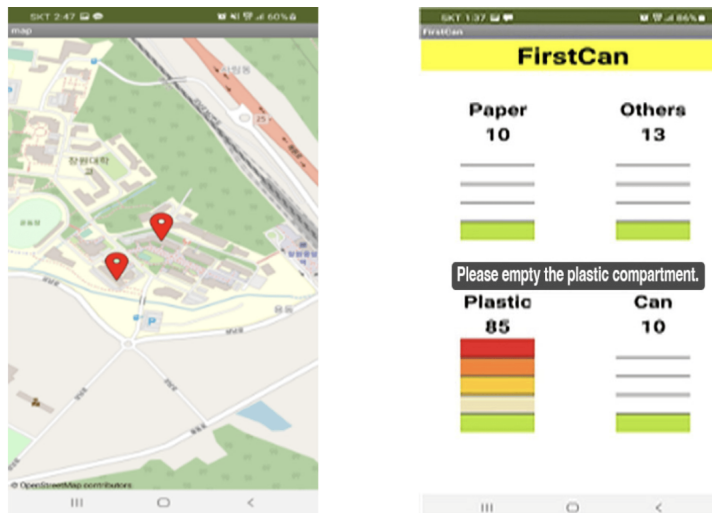


Figura 2.7: Interfaces da aplicação móvel do projeto [WH22].

2.5 Projetos académicos

2.5.1 Smart Containers

O projeto **Smart Containers**, foi desenvolvido pela Universidade de Trás-os-Montes e Alto Douro (UTAD), e tem os mesmos objetivos dos sistemas anteriores, porém focado para as recolhas de óleos alimentares e de óleos de oficinas [UTA08].

Este projeto ganhou diversos prémios, destacando-se o **Imagine Cup** da Microsoft [UTA08].

Permite monitorizar o nível de enchimento dos contentores que indica que atingiu a sua capacidade máxima e que têm de ser recolhidos pelas entidades responsáveis [UTA08].

A figura 2.8 mostra a interface do projeto Smart Containers.



Figura 2.8: Interface do projeto Smart Containers (em https://smartcontainers.utad.pt/project_en.html).

2.5.2 iEcoSys - *Intelligent Ecologic System*

O sistema iEcoSys foi desenvolvido por uma equipa do Instituto Politécnico da Guarda. Este conta com um sistema inteligente de gestão de resíduos que identifica os resíduos produzidos individualmente, através de *tags* RFID introduzidas nos sacos do lixo [Rei+15].

Ao depositar os resíduos, o centro de reciclagem identifica e pesa cada saco e os dados recolhidos são enviados para um servidor através da tecnologia de comunicação ZigBee [Rei+15].

Quando a informação chega ao servidor é inserida na base de dados para que seja possível consultar no portal iEcoSys os resíduos colocados e encomendar novos sacos [Pin14].

Os indivíduos que pretenderem usufruir deste projeto terão de adquirir sacos que possibilitem a sua identificação no sistema, no momento do depósito do lixo. O utilizador simplesmente tem de identificar o saco que se encontra a depositar próximo do ecoponto, o ecoponto com sistema de identificação e pesagem dos sacos. Seguidamente, o sistema regista na base de dados o lixo depositado, peso, data e hora [Pin14].

Os utilizadores deste projeto têm uma conta onde é depositado o valor a receber pelos resíduos, tendo em conta o peso, motivando os indivíduos a fazerem a separação do lixo. Em alternativa, poderá ser feito um desconto na fatura da água, em função da quantidade de lixo depositado [Pin14].

A figura 2.9 mostra os caixotes inteligentes iEcoSys.



Figura 2.9: Caixaote do lixo inteligente iEcoSys (em <https://empresashoje.pt/high-tech/iecosys-sistema-inteligente-para-gestao-de-residuos/>).

2.6 Sistemas comerciais

2.6.1 Bigbelly Smart Bins

A Bigbelly é uma empresa líder em soluções de gestão de resíduos. Esta utiliza caixotes do lixo inteligentes alimentados por painéis solares [Big24].

Os seus caixotes comunicam o nível de enchimento. Existem várias versões com vários tipos de tamanho [Big24].

Os caixotes são totalmente fechados, inclusive para proteger de animais [Big24].

A Bigbelly promete aos municípios menos recolhas, reduzindo o consumo de combustível e as emissões de GEE em até 70% [Big24].

A figura 2.10 mostra o caixaote do lixo inteligente Bigbelly.



Figura 2.10: Caixaote do lixo inteligente Bigbelly [Big24].

A sua interface gráfica permite uma visualização de estatísticas do volume de resíduos, reduções de recolhas e emissões de GEE. Também é possível visualizar informações em tempo real da quantidade de caixotes que precisam ser recolhidos e que têm risco de transbordar [Big24].

A Bigbelly conta com uma plataforma *web* e uma aplicação móvel. A figura 2.11 mostra as interfaces do caixote do lixo inteligente Bigbelly.



Figura 2.11: Interfaces da Bigbelly [Big24].

2.6.2 SmartBin.io

O sistema **SmartBin.io** oferece soluções de gestão de resíduos, através de sensores que comunicam dados em tempo real para otimizar as recolhas. Este sistema permite visualizar caixotes inteligentes numa plataforma *web* [Sma24].

Este sistema conta com inteligência artificial, que permite baixar ainda mais os custos [Sma24]. A figura 2.12 mostra os dispositivos **SmartBin.io**.

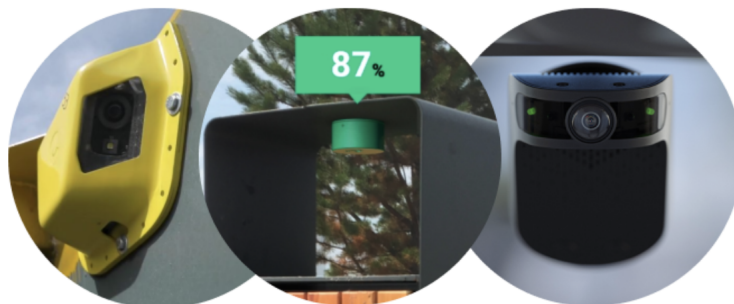


Figura 2.12: Dispositivos SmartBin.io [Sma24].

2. ESTADO DA ARTE

A figura 2.13 mostra o mapa com os caixotes SmartBin.io.

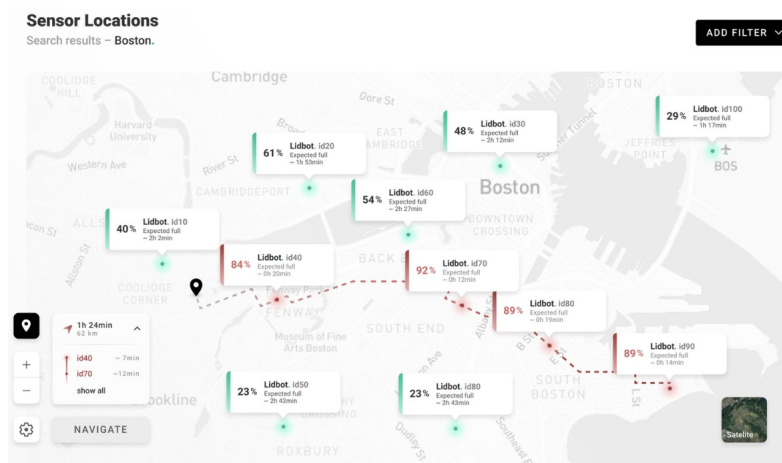


Figura 2.13: Localização dos caixotes SmartBin.io no mapa [Sma24].

2.6.3 Sensoneo Municipal Waste Collection

O sistema Sensoneo permite uma gestão inteligente de resíduos, através de sensores e de algoritmos de otimização de rotas [Sen24].

Os sensores enviam dados em tempo real dos caixotes inteligentes para um sistema de gestão de resíduos [Sen24].

Este sistema permite que se recolha apenas caixotes cheios, evitando deslocações desnecessárias, assim, também se previne que os caixotes transbordem [Sen24]. A figura 2.14 mostra a interface da plataforma web Sensoneo.

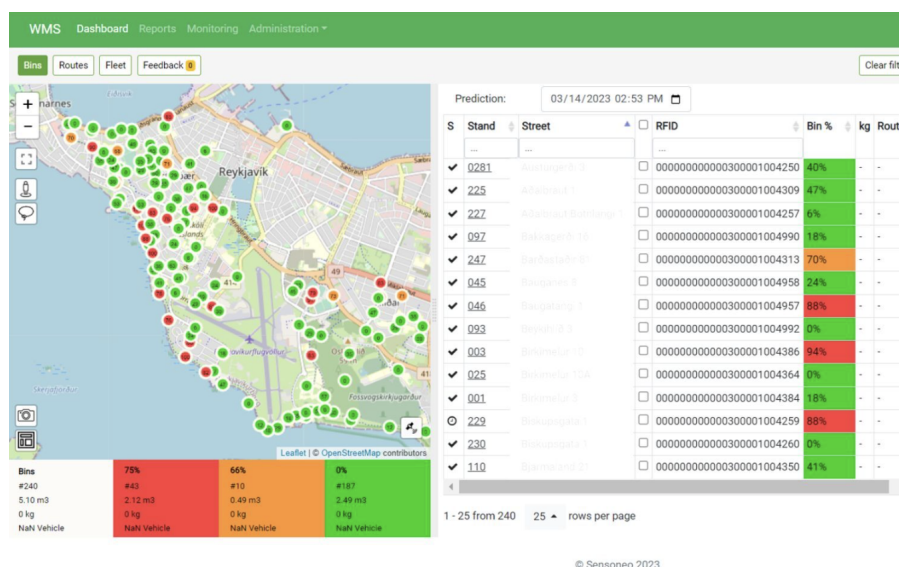


Figura 2.14: Plataforma web Sensoneo [Sen24].

Este sistema é alimentado através de baterias e permite automatizar a gestão das rotas de recolha de resíduos, otimizar as rotas, a frequência e as cargas dos veículos [Sen24]. A figura 2.15 mostra a interface da aplicação móvel Sensoneo.

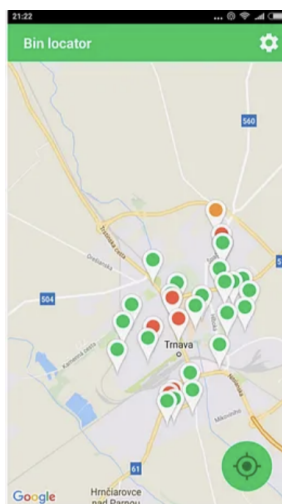


Figura 2.15: Aplicação móvel Sensoneo [Sen24].

2.6.4 360Waste

Em Portugal, também foram desenvolvidos projetos para melhorar a recolha de resíduos, como é o caso do 360Waste.

O projeto 360Waste foi desenvolvido pela empresa **EVOX Technologies**, em Castelo Branco, com o intuito de elaborar uma gestão integrada na recolha de resíduos que possibilitasse um planeamento mais competente dos percursos dos camiões que recolhem o lixo, poupando nos custos logísticos e promovendo a redução da poluição [Rod+20].

Neste projeto, para que seja possível identificar cada contentor, estes são equipados com a tecnologia LoRa e RFID (*Radio Frequency Identification*). Nos camiões também é utilizada a mesma tecnologia que possibilita obter informações sobre os contentores, nomeadamente a localização dos mesmos [Rod+20].

Os responsáveis pela recolha dos contentores do lixo utilizam um botão instalado no interior dos camiões que fornece informações sobre o nível de enchimento do contentor. Estas informações são fornecidas em tempo real para o sistema de gestão de resíduos. Este sistema também permite o envio de alertas aos responsáveis pela recolha dos contentores [Rod+20].

A figura 2.16 mostra o caixote do lixo inteligente 360Waste.



Figura 2.16: Caixaote do lixo inteligente 360Waste (em <https://jornaleconomico.sapo.pt/noticias/startup-desenvolve-sistema-gestao-inteligente-residuos-100566/>).

2.6.5 SOMA - Gestão de resíduos sólidos urbanos

A SOMA, é uma empresa portuguesa que desenvolveu um sistema que permite a gestão de frota e a gestão dos caixotes do lixo [SOM24].

O funcionário pode visualizar a rota que está a ser efetuada pelos seus camiões num mapa em tempo real. Este sistema permite também a criação de rotas mais eficientes [SOM24].

À semelhança com o projeto 360Waste, este sistema também utiliza a tecnologia RFID para proceder à identificação dos caixotes [SOM24].

A figura 3.1 mostra o mapa do caixote do lixo inteligente SOMA.



Figura 2.17: Mapa do projeto SOMA (em [SOM24]).

2.7 Análise e evolução histórica dos sistemas de gestão de resíduos

A gestão de resíduos tem evoluído significativamente ao longo dos anos, o que reflete o avanço das tecnologias e a procura por soluções mais eficientes e sustentáveis.

O projeto **SOMA** foi um dos primeiros desenvolvidos em Portugal e representou um avanço na automatização da gestão de resíduos urbanos. Este sistema não contém uma aplicação de navegação, uma vez que os motoristas apenas têm acesso à morada.

No projeto **SOMA** são utilizados dados dos níveis de enchimento anteriores, o que retira o benefício da sensorização. Com esta é possível saber em tempo real o nível de cada caixote e gerar uma rota apenas para os caixotes que estão perto de atingir o seu nível máximo de enchimento.

O projeto **Smart Containers** é o mais antigo, este apresenta uma interface mais desatualizada e *hardware* e *software* menos avançado. Este projeto é o mais limitado em termos de tecnologia de visualização e análise.

O projeto **Smart Containers** deixou de ser utilizado, pois não se adaptou às novas exigências tecnológicas e do mercado.

Com a decorrente evolução da tecnologia, surgiram novos projetos que integravam sensores mais sofisticados e comunicação em tempo real para identificar e monitorizar os caixotes. Como é o caso do projeto **360Waste**, que representou uma maior evolução na integração da tecnologia RFID e de tecnologias de comunicação para a otimização de rotas de recolha e a gestão dos caixotes.

Os projetos **IoT-TBLM** e **IoTFSkimUm** utilizam tecnologias mais recentes para uma medição mais precisa do nível de enchimento dos caixotes. Estas transmitem dados em tempo real para uma análise mais detalhada.

Com o surgimento da inteligência artificial, também surgiram novos sistemas como o **SmartBin.io** e o **Sensoneo**. Estes projetos destacam-se pela sua capacidade de otimizar a gestão de resíduos utilizando algoritmos mais avançados. Estes sistemas utilizam inteligência artificial para melhorar a eficiência das rotas.

Com a inteligência artificial os sistemas conseguem prever quando os caixotes estarão cheios e assim evitar que estes transbordem. Ao automatizar as rotas, consegue-se poupar até 70% em consumo de combustível e diminuir as emissões de gases de efeito estufa. No entanto, os projetos que dependam de dados históricos e sem integração de inteligência artificial limitam a eficácia dos sistemas, o que fará com que fiquem desatualizados.

Por último, o sistema **Bigbelly** é um dos mais recentes e tecnologicamente avançados. Este projeto foi criado por uma empresa líder no mercado que combina a utilização de painéis solares com a comunicação em tempo real para monitorizar o nível de enchimento dos caixotes autonomamente. A **Bigbelly** oferece uma interface gráfica mais detalhada e uma plataforma *web* robusta, destacando-se assim como um projeto de sucesso na área

da gestão de resíduos. Este projeto permite a visualização das rotas em tempo real e na otimização das mesmas. No entanto, a sua dependência de dados históricos e a falta da integração de uma aplicação de navegação limitaram a eficácia do sistema.

2.8 Conclusão

Existe a necessidade de implementar estratégias que possam melhorar a sustentabilidade em toda a parte do mundo, é sabido que incluir a parte da gestão de resíduos nessas estratégias é fundamental, por isso existem diversos projetos que têm como principal objetivo melhorar a gestão de resíduos através da utilização das tecnologias IoT, como a LoRaWAN.

Nos sistemas de gestão de resíduos, existem problemas como o transbordamento, tampas abertas, a concentração de resíduos no centro que leva a que estes sistemas não efetuem as leituras do nível de resíduos corretamente. Nestes casos específicos, existem soluções como a integração da inteligência artificial que pode prever o nível de enchimento destes caixotes com base em dados históricos.

Neste capítulo, foram explorados diversos projetos de caixotes do lixo inteligentes. É possível verificar estes têm objetivos em comum, no sentido em que procuram minimizar os custos da recolha do lixo e a qualidade de vida dos indivíduos, melhorando o meio ambiente.

Capítulo 3

Arquitetura do Sistema

3.1 Introdução

Os caixotes do lixo inteligentes são equipados com sensores e sistemas de processamento que incorporam tecnologias de comunicação que permitem monitorizar remotamente o seu nível de enchimento.

As principais características incluem a utilização de sensores para medir o nível de resíduos no caixote, detetar incêndios e a utilização de tecnologias IoT para transmitir dados em tempo real para uma unidade central de controlo. As tecnologias **LoRaWAN**, **Sigfox** e **NB-IoT** garantem uma comunicação de longo alcance com um baixo consumo de energia e a utilização de fontes de energia sustentáveis, através da integração de painéis solares para alimentar o sistema, de modo a torná-lo energeticamente autossuficiente.

Na Secção 3.2 serão apresentados os requisitos funcionais e não funcionais deste projeto. Na Secção 3.3 serão apresentados os diagramas de caso de uso, na Secção 3.4 será apresentado o diagrama funcional do sistema. Na Secção 3.5 será apresentado o diagrama de entidade-relacionamento. Na Secção 3.6 será apresentada a análise de tarefas que contribuíram para o desenvolvimento de *Mockups*, estes estão na Secção 3.7. Na Secção 3.8 será apresentado o ambiente de desenvolvimento e por último na Secção 3.9 a análise e caracterização dos sensores.

3.2 Requisitos do sistema

3.2.1 Requisitos funcionais

A lista de requisitos funcionais é:

- O sistema deve permitir enviar e receber leituras do nível de enchimento, detecção de fumo e localização do caixote e o nível de bateria;
- O sistema deve permitir enviar um alerta sempre que é detetado fumo, quando a localização do caixote é alterada ou existe um caixote com bateria com pouca carga;
- Os componentes devem ter características técnicas adequadas ao ambiente.
- O sistema deve ter algumas características de cibersegurança:
 - A comunicação dos dados deve ser cifrada;
 - O sistema deve permitir a autenticação de utilizadores através de *email* e *password*;
 - O sistema deve permitir atribuir permissões aos utilizadores de acordo com o seu perfil;
 - O sistema deve permitir que os utilizadores possam repor a sua *password* autonomamente;
- O sistema deve permitir:
 - Visualizar as rotas;
 - Criar rotas otimizadas tendo em conta o nível atual dos caixotes e a localização dos mesmos;
 - Editar as rotas;
 - Editar a localização do depósito;
 - Comparar a rota realizada com a rota gerada automaticamente;
- O sistema deve permitir:
 - Visualizar os veículos;
 - Criar veículos;
 - Editar veículos;
 - Apagar veículos;

- O sistema deve permitir:
 - Visualizar os dispositivos;
 - Criar dispositivos;
 - Editar dispositivos;
 - Apagar dispositivos;
- O sistema deve permitir:
 - Visualizar os utilizadores;
 - Criar utilizadores;
 - Editar utilizadores;
 - Apagar utilizadores;
 - Que cada utilizador consiga editar as suas informações pessoais.

3.2.2 Requisitos não funcionais

A lista de requisitos não funcionais é constituída pelos seguintes itens:

- A aplicação deve estar disponível *online*;
- O sistema deve estar sempre disponível - 24h/dia e 7 dias/semana;
- O sistema deve ter uma interface de fácil utilização;
- Deve existir ajuda dentro da própria interface;
- O sistema deve ser multiplataforma, pois deve funcionar no computador, *smartphone* ou *tablet*, ou seja, deve ser responsiva;
- A aplicação móvel deve funcionar para dispositivos **Android** e **iOS** de igual forma;
- O sistema deve permitir atualizar automaticamente as leituras, cada vez que é recebida sem a necessidade de atualizar a página;
- O dispositivo deve ser universal, ou seja, deve ter compatibilidade com caixotes de resíduos com diversos tamanhos e formatos.

3.3 Diagrama de casos de uso

De acordo com as funcionalidades dos utilizadores descritas nos requisitos funcionais, foram desenvolvidos diagramas de casos de uso, para que fosse possível um melhor entendimento das funções atribuídas a cada cargo. Neste caso existem 3 cargos, o administrador, a gestão e o motorista. Na figura 3.1 é possível observar que todos os utilizadores podem visualizar as leituras.

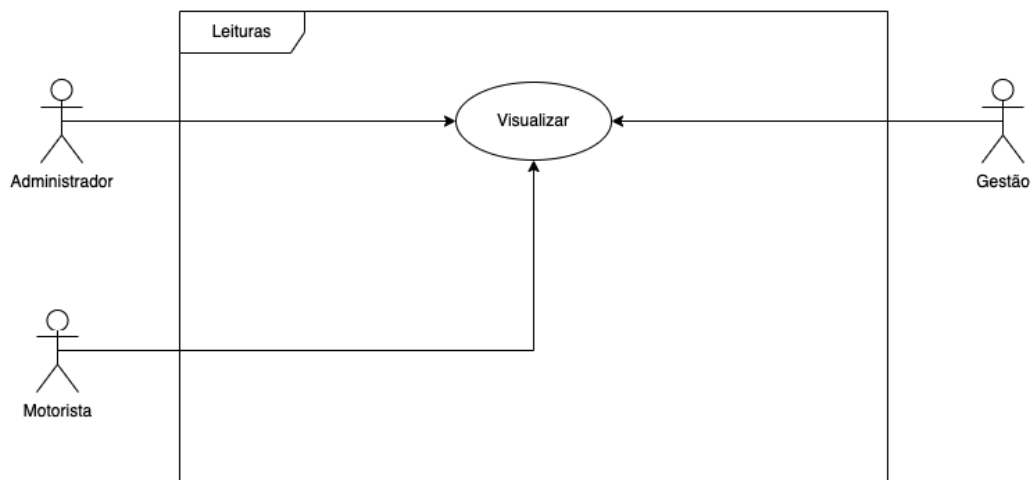


Figura 3.1: Diagrama de casos de uso - Leituras.

Na figura 3.2 é possível observar que apenas o administrador ou a gestão podem editar as rotas.

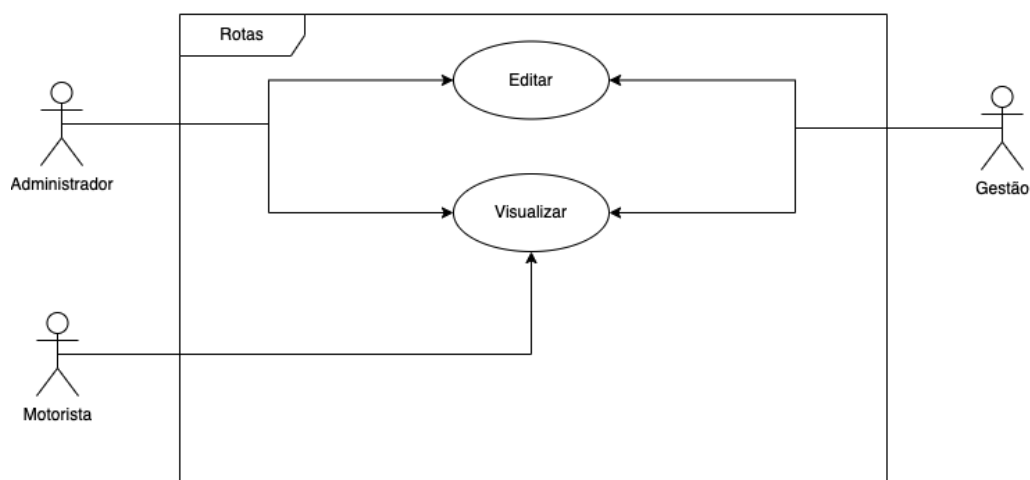


Figura 3.2: Diagrama de casos de uso - Rotas.

Na figura 3.3 é possível observar que apenas os utilizadores com cargos de administrador ou gestão podem gerir os veículos.

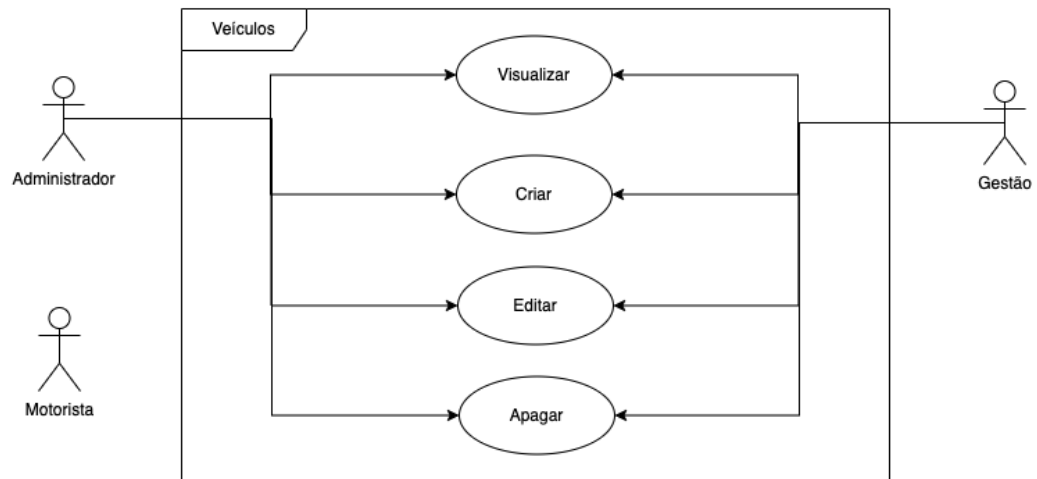


Figura 3.3: Diagrama de casos de uso - Veículos.

Na figura 3.4 é possível observar que apenas os utilizadores com cargos de administrador ou gestão podem gerir os dispositivos.

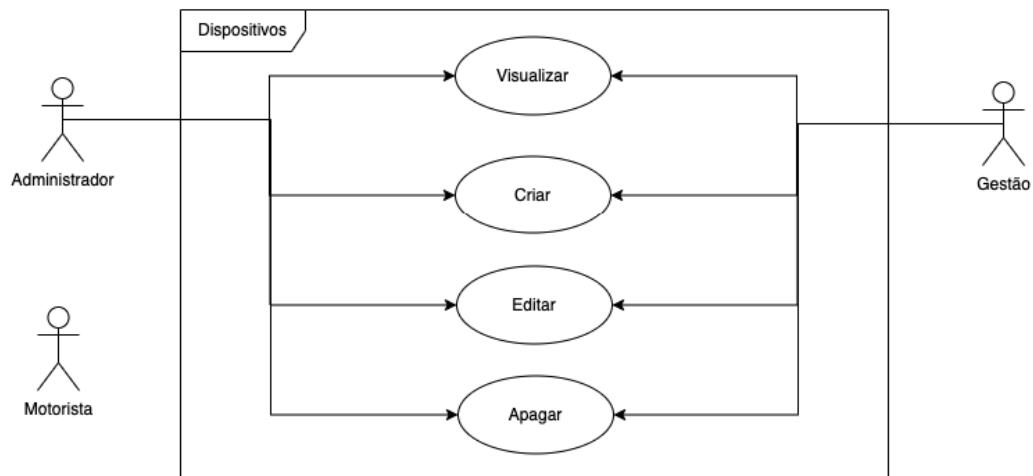


Figura 3.4: Diagrama de casos de uso - Dispositivos.

3. ARQUITETURA DO SISTEMA

Na figura 3.5 é possível observar que apenas os utilizadores com cargos de administrador ou gestão podem gerir os utilizadores.

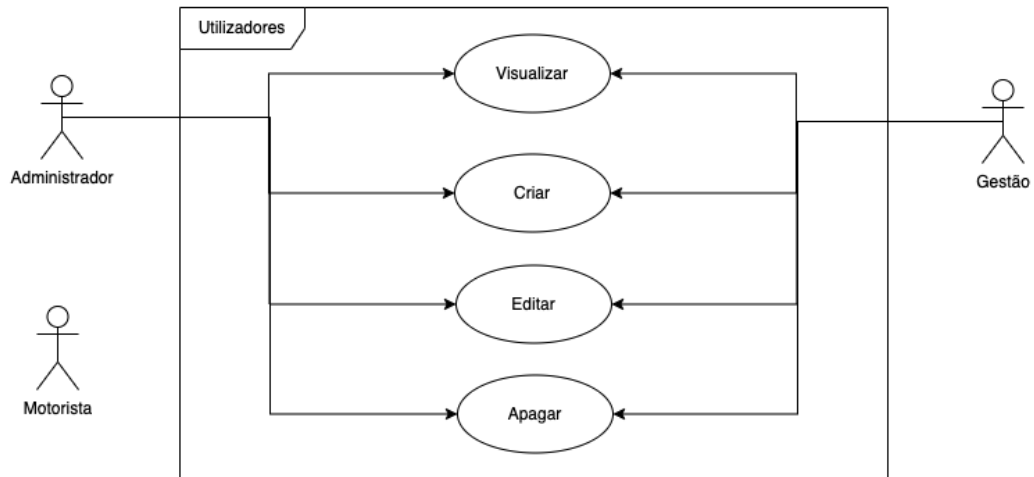


Figura 3.5: Diagrama de casos de uso - Utilizadores.

3.4 Diagrama funcional do sistema

Foram elaborados diagramas funcionais do sistema, para que fosse possível ter uma visão global de como está estruturado este projeto. O diagrama 3.6 representa o sistema e onde, e como, os dados são transmitidos.

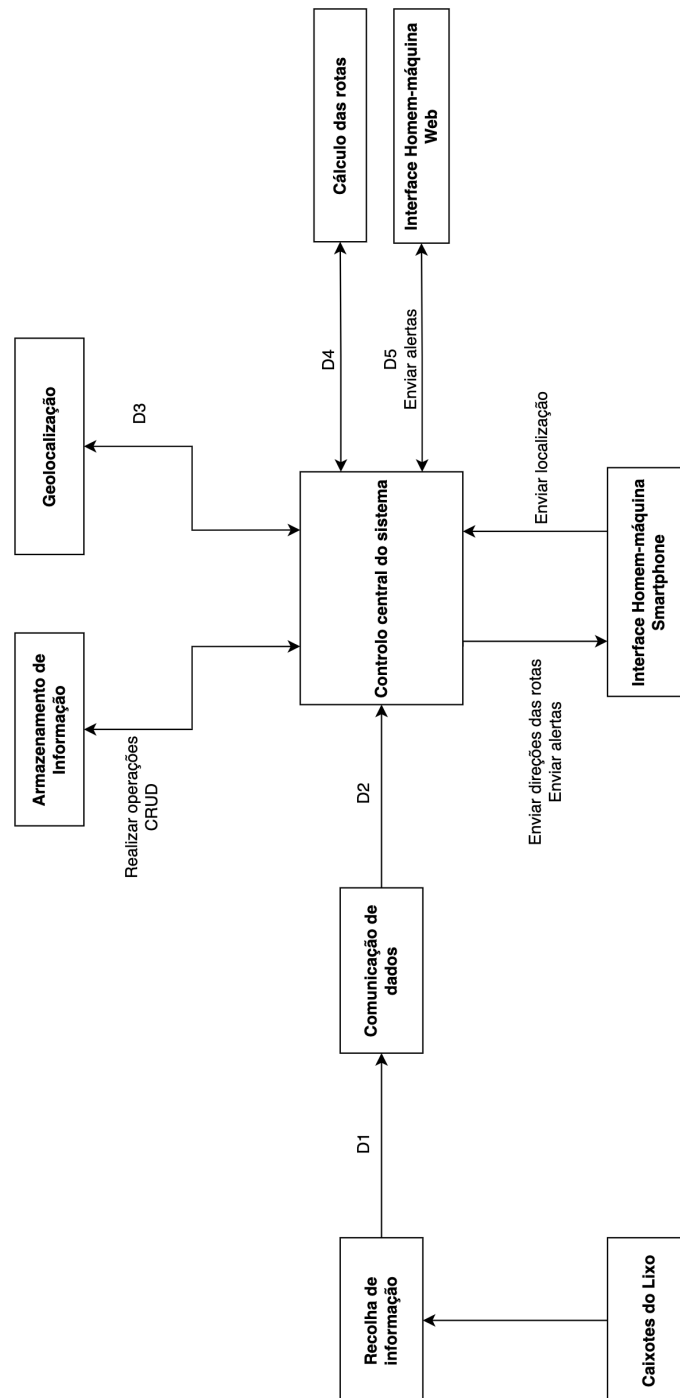


Figura 3.6: Diagrama funcional do sistema.

Existem diversos tipos de dados que são transmitidos pelo sistema, estes foram denominados por **D1**, **D2**, **D3** e **D4**.

Os dados **D1**, são os dados recolhidos pelos sensores dos caixotes, ou seja, a distância, deteção de fumo, as coordenadas GPS e o nível de bateria, apresentados na Secção 3.9.

Conforme indicado anteriormente nos requisitos não funcionais, os dispositivos colocados no interior do caixote são universais, ou seja, são compatíveis com vários caixotes dependendo do seu tamanho ou formato. Deste modo, é enviada a distância para que seja calculado o nível de enchimento na unidade central de controlo, tendo em conta a dimensão de cada caixote indicada pelo utilizador no momento do registo do mesmo. Desta forma, é possível generalizar e não ter que programar cada microcontrolador para as dimensões específicas de cada caixote.

Os dados **D2** são os dados dos sensores, mas processados pelo *gateway* LoRaWAN, para que sejam publicados através do protocolo MQTT (*Message Queuing Telemetry Transport*), para que o cliente, ou seja, a unidade central de controlo possa aceder.

Os dados **D3** são os dados geográficos, ou seja, a matriz de custos e as direções das rotas.

Os dados **D4** são os dados necessários para executar o procedimento computacional de VRP, que tem como dados de entrada a capacidade, a procura dos clientes e a matriz de custos. Os dados de saída são as rotas otimizadas.

Os dados **D5** são os dados mostrados nas interfaces para o utilizador final. Os dados são relativos aos utilizadores, aos veículos, aos dispositivos, às leituras e às rotas.

3.4.1 Diagrama funcional mais aprofundado do sistema

A figura 3.7 apresenta um diagrama funcional mais aprofundado do sistema. Este diagrama apresenta os protocolos de comunicação utilizados. Conforme referido na introdução deste capítulo, a tecnologia IoT adotada é a LoRaWAN que é utilizada para a comunicação entre os dispositivos de cada caixote, o *gateway* LoRaWAN transmite as leituras indicadas nos requisitos funcionais, estas são cifradas e em formato JSON.

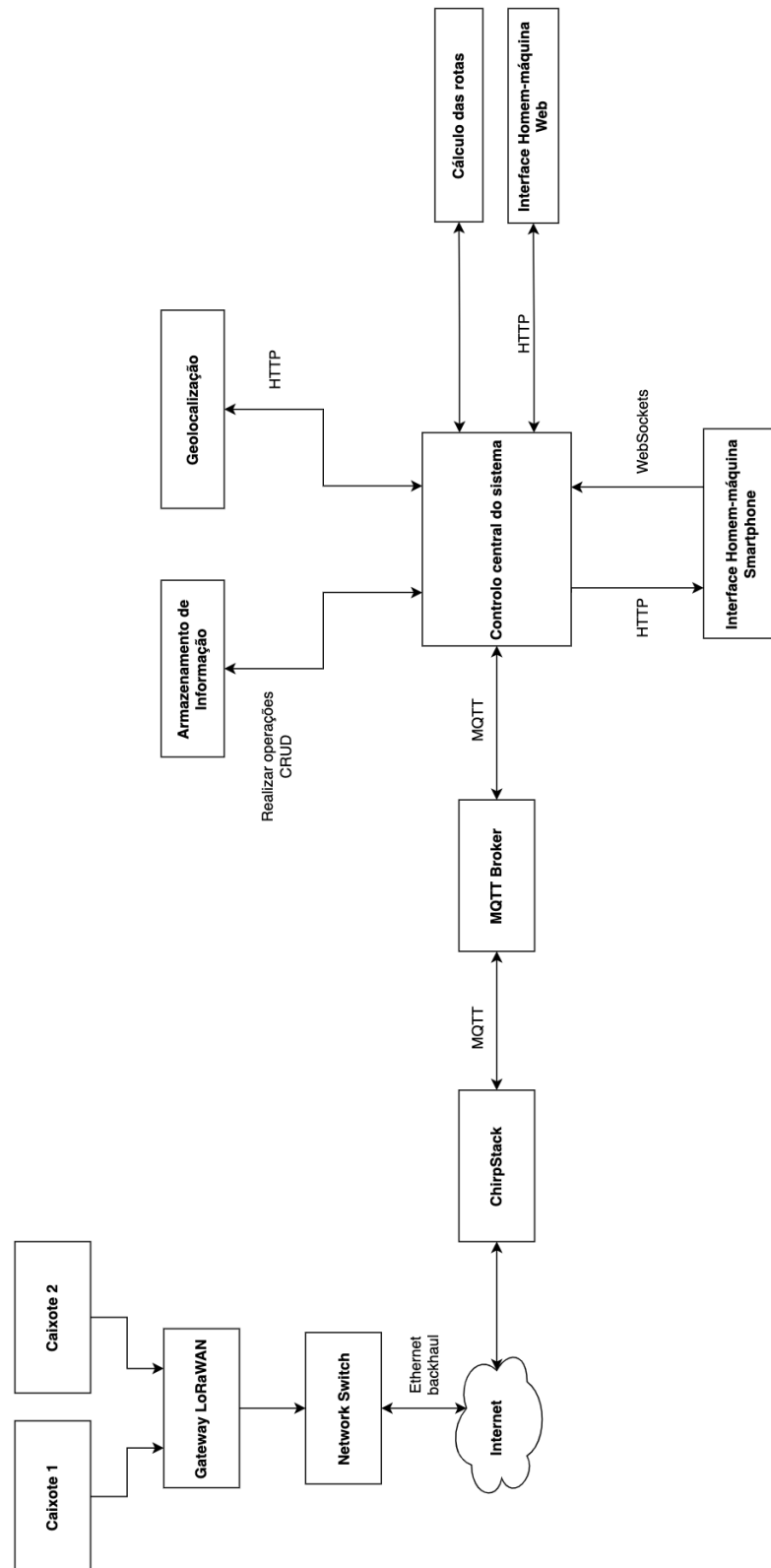


Figura 3.7: Diagrama funcional mais aprofundado do sistema.

No diagrama da figura 3.7, o *gateway* está conectado à rede interna, neste caso do município. O *gateway* processa os dados e a plataforma LoRaWAN ChirpStack publica-os no *Broker* MQTT, através do protocolo de comunicação MQTT.

A plataforma ChirpStack apresenta um servidor de rede *open source* utilizado para configurar redes LoRaWAN públicas ou privadas. Esta fornece uma interface *web* para a gestão de *gateways*, dispositivos, integração com *clouds*, bases de dados, entre outros [Chi24].

O cliente MQTT da unidade central de controlo é o Eclipse Paho MQTT. Este conecta-se e fica à escuta (subscrive o tópico) para caso uma leitura seja publicada ele a obtenha. A unidade central de controlo integra uma API que está ligada à base de dados relacional onde se irá realizar as operações CRUD (*Create*, *Read*, *Update* e *Delete*). Estas operações são efetuadas para atender aos requisitos funcionais mencionados anteriormente.

A unidade central de controlo também comunica através de API's externas para obter informação geográfica, como obter a matriz de custos e as direções das rotas para o camião. Esta também comunica com uma API para obter o preço do combustível. As requisições são efetuadas através do protocolo HTTP, tal como nas restantes interfaces referidas nos requisitos não funcionais, a aplicação móvel no iOS, no Android e plataforma *web*. As interfaces referem-se às páginas criadas para atender as necessidades dos utilizadores deste projeto apresentadas nos requisitos funcionais.

Este projeto utiliza tecnologia WebSockets, como é possível verificar no diagrama da figura I.15. Esta tecnologia permite enviar em tempo real informações de como a rota está sendo executada, a sua localização e informações como o ângulo para onde se move e a velocidade a que dirige.

3.5 Modelo de Entidade-Relacionamento

Na parte conceitual, o modelo de Entidade-Relacionamento está diretamente relacionado com a interface homem-máquina referida no diagrama funcional, apresentado anteriormente na Secção 3.4.

A figura 3.8 apresenta o modelo de Entidade-Relacionamento da base de dados relacional deste projeto, que se encontra na terceira forma normal.

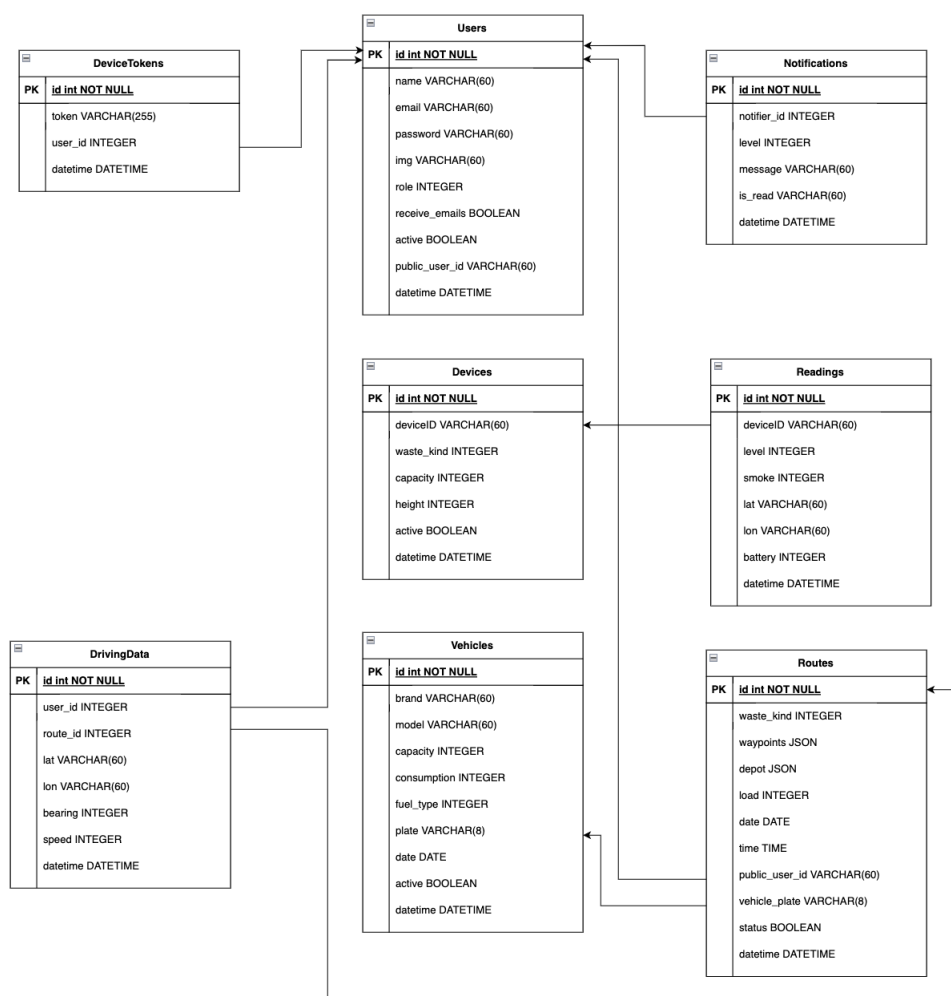


Figura 3.8: Modelo de Entidade-Relacionamento.

3.6 Análise de tarefas

As principais tarefas que os utilizadores executam nas interfaces foram analisadas com base nos requisitos definidos inicialmente.

Tarefa 1: Autenticação do utilizador

1. Abrir a aplicação ou o site;
2. Introduzir o *email* e a *password*.

Plano: Realizar passo 1 e enquanto não for inserido um *email* e uma *password* válidos, realizar o passo 2.

Tarefa 2: Recuperar senha:

1. Introduzir o *email* para recuperação;
2. Clicar no link recebido no *email*;
3. Introduzir a nova *password*;
4. Realizar autenticação;

Plano: Realizar passo 1 e enquanto não for inserido um *email* existente, realizar o passo 2.

Tarefa 3: Edita o seu perfil:

1. Abrir a aplicação ou o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de editar perfil;
4. Alterar os dados pessoais ou foto de perfil;
5. Submeter alterações.

Tarefa 4: Consultar as leituras:

1. Abrir a aplicação ou o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de leituras;

4. Visualizar as leituras.

Tarefa 5: Visualizar as rotas:

1. Abrir a aplicação ou o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página das rotas;
4. Clicar sobre a rota pretendida.

Tarefa 6: Realizar uma rota:

1. Abrir a aplicação;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Seleccionar a rota a realizar;
4. Iniciar a rota.

Tarefa 7: Administrador ou Gestão edita rotas:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar rotas;
4. Seleccionar uma rota;
5. Alterar os dados;
6. Submeter alterações.

Tarefa 8: Administrador ou Gestão adiciona novo veículo:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de adicionar veículos;
4. Introduzir dados do novo veículo;

5. Submeter o formulário.

Tarefa 9: Administrador ou Gestão visualiza todos os veículos:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar veículos.

Tarefa 10: Administrador ou Gestão edita veículos:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar veículos;
4. Seleccionar um veículo;
5. Alterar os dados;
6. Submeter alterações.

Tarefa 11: Administrador ou Gestão apaga veículo:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar veículos;
4. Seleccionar apagar veículo;
5. Confirmar eliminação.

Tarefa 12: Administrador ou Gestão adiciona novo dispositivo:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de adicionar dispositivos;
4. Introduzir dados do novo dispositivo;

5. Submeter o formulário.

Plano: Caso exista um veículo registado com essa matrícula, realizar passo 4.

Tarefa 13: Administrador ou Gestão visualiza todos os dispositivos:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar dispositivos.

Tarefa 14: Administrador ou Gestão edita dispositivos:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar dispositivos;
4. Selecionar um dispositivo;
5. Alterar os dados;
6. Submeter alterações.

Tarefa 15: Administrador ou Gestão apaga dispositivo:

1. Abrir o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar dispositivos;
4. Selecionar apagar dispositivo;
5. Confirmar eliminação.

Tarefa 16: Administrador ou Gestão adiciona novo utilizador:

1. Abrir a aplicação ou o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de adicionar utilizadores;

3. ARQUITETURA DO SISTEMA

4. Introduzir dados do novo utilizador;
5. Submeter o formulário.

Plano: Caso exista um utilizador registado com esse *email*, realizar passo 4.

Tarefa 17: Administrador ou Gestão visualiza todos os utilizadores:

1. Abrir a aplicação ou o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar utilizadores.

Tarefa 18: Administrador ou Gestão edita utilizador:

1. Abrir a aplicação ou o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar utilizadores;
4. Selecionar um utilizador.
5. Alterar os dados;
6. Submeter alterações.

Tarefa 19: Administrador ou Gestão apaga utilizador:

1. Abrir a aplicação ou o site;
2. Autenticar-se, caso não tenha sessão iniciada;
3. Aceder à página de visualizar utilizadores;
4. Selecionar apagar utilizador;
5. Confirmar eliminação.

3.7 Desenvolvimento de *Mockups*

Foram desenvolvidos *mockups* com base na análise de tarefas e nos diagramas de casos de uso.

Os *mockups* ajudam a consolidar ideias e facilitam na hora da sua implementação. Os *mockups* da plataforma *web* são de alta fidelidade, uma vez que, apresentam uma interface muito semelhante ao produto final.

De modo a estruturar a aplicação móvel, foram criados *mockups* de média e alta fidelidade, estes permitiram ter uma ideia do *layout* para melhorar a usabilidade da aplicação. Posteriormente, foram desenvolvidos os *mockups* de alta fidelidade, que permitiram ter uma melhor visão do *design* final da aplicação antes da sua conceção.

A ferramenta utilizada para desenvolver os *mockups* de média fidelidade foi a **Balsamiq**, enquanto para desenvolver os *mockups* de alta fidelidade foi utilizado o **Figma**. Em seguida serão apresentados alguns destes, sendo que é possível encontrar o restante no Anexo I.

A figura 3.9 apresenta um exemplo de um *mockup* de alta fidelidade da plataforma *web*.

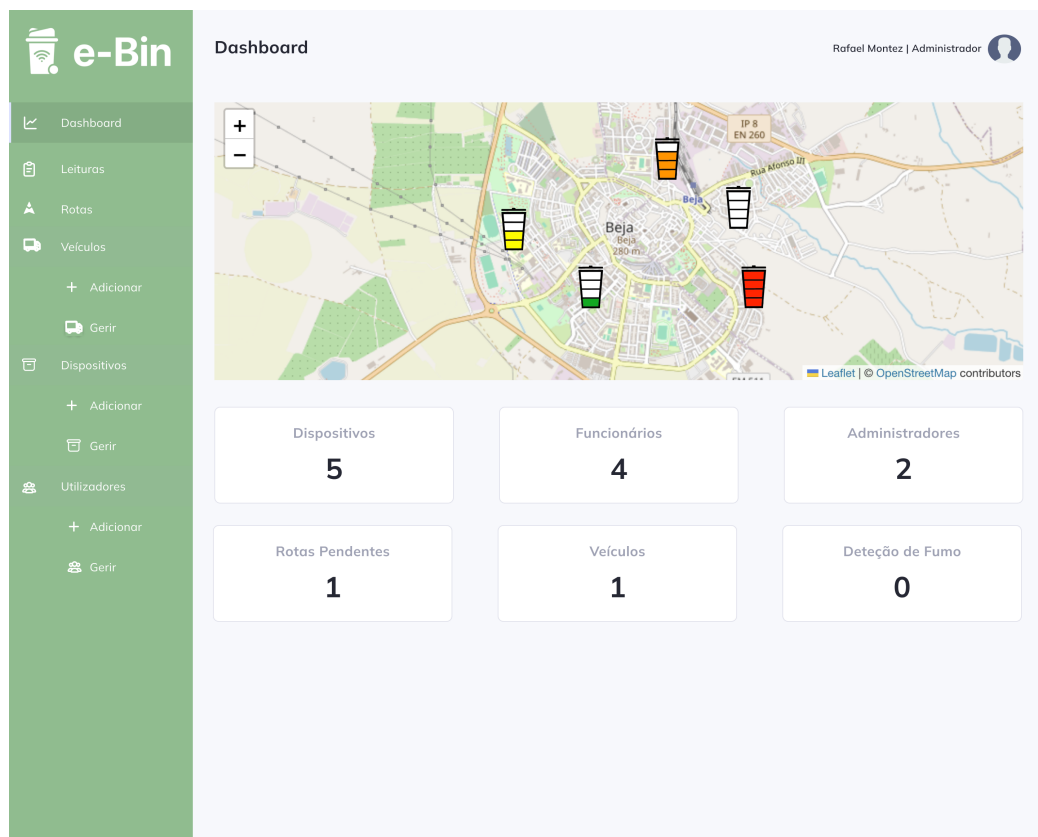


Figura 3.9: *Mockup* de alta fidelidade da *dashboard*.

3. ARQUITETURA DO SISTEMA

A figura 3.10 apresenta exemplos de *mockups* de média fidelidade da aplicação móvel.

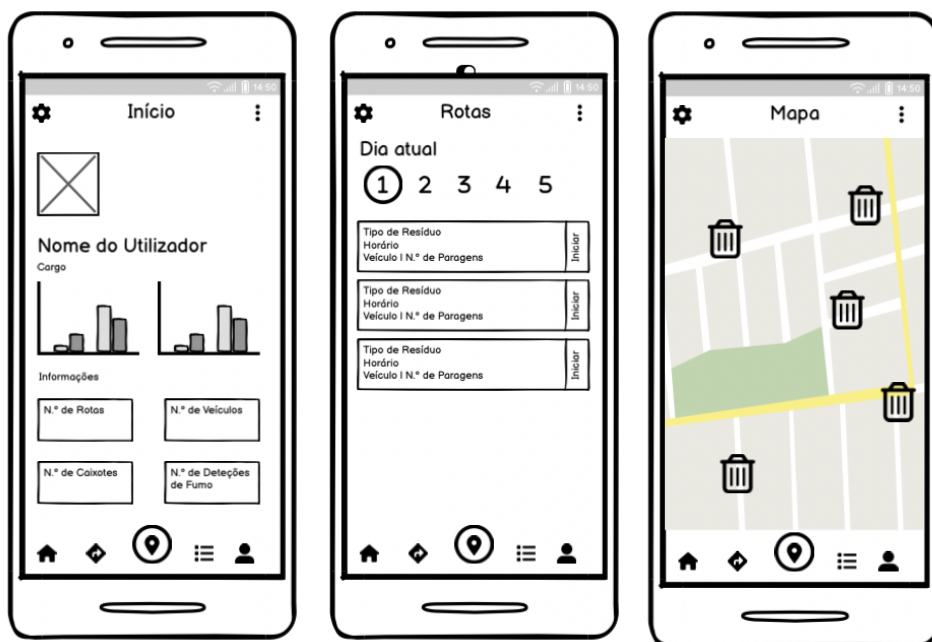


Figura 3.10: *Mockups* de média fidelidade da aplicação móvel.

A figura 3.10 apresenta exemplos de *mockups* de alta fidelidade da aplicação móvel.

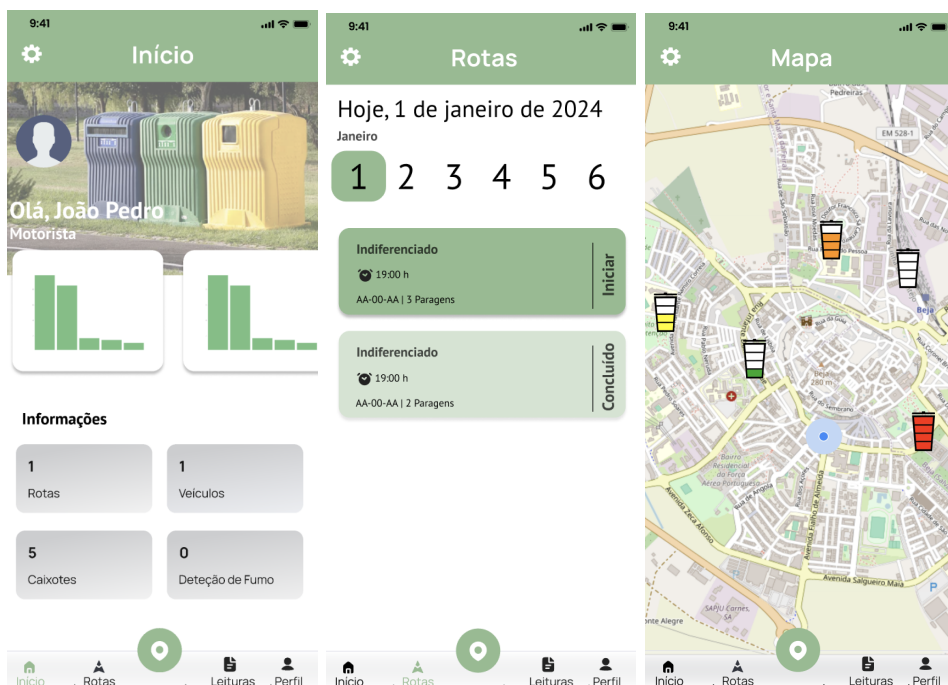


Figura 3.11: *Mockups* de alta fidelidade da aplicação móvel.

3.8 Ambiente de Desenvolvimento

Nesta secção será discutido o ambiente de desenvolvimento utilizado para elaborar o sistema com as características e funcionalidades apresentadas anteriormente.

O projeto foi desenvolvido utilizando *containers Docker* (ver mais informações na Secção 5.6). Os *containers* estão a ser executados num Servidor HP ProLiant ML110 G6, que conta com um processador Intel Xeon X3450, 16 GB de memória RAM DDR3, um disco SSD de 256 GB com o Ubuntu Server 22.04 e 2 discos rígidos de 500GB em RAID 1 para armazenar os *backups* e garantir redundância.

O código do projeto foi desenvolvido com o editor **Visual Studio Code** na versão 1.92. No controlo de versões foi utilizado o **Git** na versão 2.39.0.

De modo a simular a aplicação móvel foi utilizado o simulador do **Android Studio Giraffe** e do **Xcode** na versão 15.3, com o simulador de um iPhone com o iOS 17 .

O código do microcontrolador foi produzido através da IDE do **Arduino** na versão 2.1.0.

O código de uma alternativa do procedimento computacional de VRP foi desenvolvido na linguagem de programação **Julia** na versão 1.10.

3.8.1 Linguagens de programação

Na fase de implementação da plataforma *web*, foram utilizados recursos e linguagens de programação orientadas para o desenvolvimento de aplicações *web*. As interfaces gráficas da plataforma *web* foram programadas utilizando **HTML**, **CSS** e **JavaScript**, enquanto a aplicação móvel foi programada em **Dart** com a *framework Flutter*. Estas ferramentas foram escolhidas uma vez que são muito utilizadas.

No caso da aplicação móvel foi projetada para ser acedida em ambiente *web*, por isso caso o utilizador tente aceder *offline*, será exibido um aviso, como mostra a figura 3.12.



Figura 3.12: Captura de ecrã da aplicação móvel desconectada da Internet.

O *frontend* da plataforma *web*, utiliza o template **Volt** que é baseado em **Bootstrap** 5, uma vez que é responsivo, ou seja, permite uma melhor experiência para os dispositivos móveis. Na plataforma *web* alguns elementos são comuns a todas as páginas após um *login* bem-sucedido, como é o caso da central de notificações e da barra de pesquisa. Esta permite ao utilizador aceder a qualquer página rapidamente. As notificações enviadas pelo servidor são mostradas nessa central de notificações e podem ser ocultas quando visualizadas.

O mapa utilizado para desenvolver este projeto foi o **OpenStreetMap Leaflet**, juntamente com a API de informação geográfica **OpenRouteService**, que permite obter a matriz de custos e as direções. Escolheu-se utilizar este ecossistema criado pela comunidade e de livre utilização [Ope24].

A programação do lado do servidor é feita na linguagem de programação **Python** utilizando a *microframework* **Flask** devido à sua simplicidade e bom desempenho [Dee23]. Esta não contém vários módulos agregados ao contrário da *framework* **Django**, portanto as dependências são instaladas conforme a sua necessidade, tornando o projeto mais leve. O **Flask** também está muito bem documentado [Dee23].

Ao utilizar o **Flask** é possível conjugar extensões desenvolvidas para funcionarem perfeitamente com este *Framework Web*, como é o caso do **Flask SQLAlchemy**, o **Flask-Mail**, o **Flask Login**, entre outros utilizados neste projeto. O **Flask SQLAlchemy** é um *Object-Relational Mapping* (ORM) que permite fazer ligações a bases de dados diversas. Um ORM facilita as operações CRUD, uma vez que, não é necessário escrever a *query* manualmente, tornando o sistema mais seguro, e dificultando ataques tais como **SQL-Injection**. Ao utilizar o **Flask SQLAlchemy** é possível migrar para outra base de dados relacional como o **PostgreSQL** ou o **SQLite**, sem ser necessário fazer grandes alterações ao código do projeto.

O microcontrolador foi programado na linguagem de programação do **Arduino**, que é baseada em **C++** com algumas particularidades específicas do microcontrolador. Escolheu-se trabalhar com esta plataforma, uma vez que existe muita documentação.

O código do microcontrolador tem como principal função obter os dados dos sensores e enviar estes no formato JSON para o *gateway* **LoRaWAN**.

Verificou-se que o tempo de execução do procedimento computacional desenvolvido em **Python** era demorado. Por isso desenvolveu-se uma alternativa na linguagem de programação **Julia**, em que o tempo de execução é significativamente mais baixo (ver mais informações na Secção 4.5).

3.9 Análise e caracterização dos sensores

Os componentes foram escolhidos tendo em conta vários aspetos como: o custo/benefício, visto que ao proceder à instalação de caixotes inteligentes num município os componentes serão instalados em quantidades significativas; a robustez aos fatores ambientais, para que possam operar em condições atmosféricas reais; o consumo de corrente de cada componente; a dimensão, uma vez que ao utilizar componentes grandes, se ocuparia muito espaço; a precisão e a exatidão das leituras dos sensores, pois é importante que as medições sejam consistentes e próximas do valor real.

Neste protótipo foram utilizados 3 sensores, o HC-SR04, para obter dados relativos ao nível de enchimento, o sensor MQ-2 para detetar a presença de fumo e módulo GPS NEO-6M para obter a localização de cada caixote.

Na Secção 5.7.1 são mencionadas as bibliotecas utilizadas para ler e enviar dados do microcontrolador. Na Secção 5.13.1, são exibidas as medições do consumo de corrente destes sensores. Os custos relativos aos sensores e componentes utilizados podem ser encontrados mais detalhadamente no Anexo IV.

3.9.1 Módulo GPS: NEO-6M V2

O módulo escolhido para obter a localização dos caixotes é o NEO-6M V2. Este módulo GPS obtém a sua localização na Terra e fornece a latitude e a longitude da sua posição [Ful22].

O módulo GPS NEO-6M V2 é de baixo custo (aproximadamente 10 EUR), consome pouca corrente (45 mA) e funciona no intervalo de temperaturas entre -40°C a 85°C. Este módulo é autónomo e tem um bom desempenho [Ful22].

Este módulo GPS inclui uma pequena bateria e uma EEPROM integrada para guardar as configurações, também contém uma antena externa de cerâmica ligada à placa, como é possível verificar na figura 3.13 [Ful22].



Figura 3.13: Módulo GPS NEO-6M V2 [Ful22].

O módulo GPS GPS NEO-6M V2 pode localizar até 22 satélites em 50 canais, é alimentado com 5V e consome 45 mA de corrente [Ful22].

A potência é calculada através da lei de Ohm ($P = V \times I$) da seguinte forma:

$$P = 5V \times 45mA = 225W$$

A tecnologia de comunicação com o microcontrolador é feita via UART (Serial), em que foi utilizado apenas o pino TX do módulo além dos pinos de alimentação. Neste caso existe só transmissão de dados do módulo para o microcontrolador, ou seja, as coordenadas.

A utilização de um módulo GPS dentro de cada caixote permite obter a sua localização aproximada e através desta gerar rotas para o local exato do caixote. Também permite detetar se o caixote foi retirado do seu local e localizá-lo.

Neste projeto, é possível mover livremente os caixotes de lugar e as rotas são criadas de acordo com a posição atual do caixote, sem a necessidade de alterar nenhuma configuração no microcontrolador.

3.9.2 Sensor de distância: HC-SR04

O sensor escolhido para medir a altura dos resíduos nos caixotes é o HC-SR04. Este sensor é o mais popular entre os sensores ultrassônicos para este tipo de projetos, devido ao seu baixo custo (aproximadamente 5 EUR), baixo consumo de corrente, dimensões reduzidas (45mm x 20mm x 15mm e 10g de peso) [Mor14] e funciona no intervalo de temperaturas entre -15°C a 70°C [PiB24].

O sensor HC-SR04 é alimentado com 5V e consome 15mA de corrente, funciona até uma distância máxima de 4m, com uma margem de erro de 0,3 cm [Mor14]. A distância máxima suportada é suficiente para obter a altura dos resíduos nos caixotes. A figura 3.14 mostra um sensor HC-SR04.



Figura 3.14: Sensor HC-SR04 [Mor14].

A potência é calculada através da lei de Ohm ($P = V \times I$) da seguinte forma:

$$P = 5V \times 15mA = 75mW$$

O sensor ultrassônico HC-SR04, opera numa frequência acima da audição humana, cerca de 40 kHz. O sensor envia uma onda e escuta-a em seguida, a onda sonora bate no objeto e volta. O sensor acompanha o tempo entre o envio da onda sonora e do seu retorno e calcula a distância a que o objeto se encontra. Este sensor utiliza a medida do tempo de voo [Mor14].

Na figura 3.15 é possível verificar o modo de funcionamento do sensor HC-SR04.

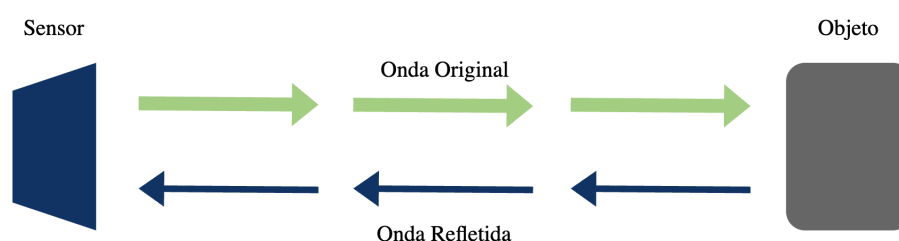


Figura 3.15: Modo de funcionamento do sensor HC-SR04.

3.9.3 Sensor de gás/fumo: MQ-2

O sensor utilizado para detetar a presença de fumo e gás nos caixotes é o MQ-2. Este é de baixo custo (aproximadamente 5 EUR), contém uma ampla deteção de gases e permite detetar a presença de fumo [Zhe18]. Funciona no intervalo de temperaturas entre -20°C a 50°C [Pol24].

O sensor de gás MQ-2 tem alta sensibilidade ao gás propano e ao fumo, porém, pode detetar gás natural e outros gases inflamáveis. Quando existe a presença de um gás inflamável, a condutividade do sensor aumenta junto com o aumento da concentração do gás [Zhe18].

Este sensor pode ter diversas aplicações, por exemplo pode ser integrado num sistema de alarme [Zhe18]. A figura 3.16 mostra um sensor MQ-2.



Figura 3.16: Sensor MQ-2 (em <https://www.botnroll.com/pt/biometricos/939-mq-2-gas-sensor-brick.html>).

O sensor MQ-2 é alimentado com 5V, porém tem um elevado consumo de corrente, cerca de 150mA, o que motivou o estudo de soluções para diminuir esse consumo neste projeto (ver mais informações na Secção 5.5.3).

A potência é calculada através da lei de Ohm ($P = V \times I$) da seguinte forma:

$$P = 5V \times 150mA = 750mW$$

3.10 Conclusão

Neste capítulo, foi exposta a arquitetura do sistema para desenvolver o caixote inteligente e-Bin.

O capítulo descreve como se projetou um sistema utilizando uma arquitetura IoT. Uma eventual melhoria no contexto desta arquitetura é a criação de redundância nos pontos críticos.

Os diagramas e a análise de tarefas permitiram visualizar e estruturar o desenvolvimento do sistema, permitindo a criação de *mockups* que facilitaram a compreensão e a validação do design proposto.

A análise dos sensores revelou a importância de escolher componentes que garantam a robustez e a eficiência da comunicação no sistema, atendendo às características do ambiente.

Em última instância, a arquitetura desenvolvida vê-se que não só orienta o desenvolvimento e a implementação do sistema, como também assegura que este seja escalável e capaz de adaptar-se a necessidades futuras.

Capítulo 4

Problema de cálculo de rotas de veículos

4.1 Introdução

O *Vehicle Routing Problem* (VRP) é um problema bastante estudado. Este é aplicado em diversos setores, desde o comércio à saúde. O VRP tem diversas aplicações, nomeadamente a recolha de resíduos [Cac+14]. Neste projeto, foi desenvolvido um procedimento computacional de VRP para que sejam geradas rotas que tenham em conta apenas os caixotes em que a sua capacidade máxima esteja perto de ser atingida.

A criação de rotas para os veículos de recolha de resíduos urbanos é baseada num procedimento computacional híbrido aleatório originado na solução tradicional da heurística de *Clarke and Wright*, designado **Rand-MER** [Cac+14]. Este tem em conta veículos homogéneos, ou seja, que tenham a mesma capacidade, multiviagens e tempo máximo de condução contínua. Em Portugal, o tempo de condução de um motorista de veículos pesados é regulado pela legislação nacional e europeia (Regulamento (CE) n.º 561/2006), que define os limites máximos de condução e os períodos de descanso obrigatórios. Esta lei não permite que um motorista conduza mais de 4 horas e 30 minutos seguidas [Uni22].

Este problema matemático, representa uma generalização do VRP comum que considera uma frota homogénea, ou seja, uma frota fixa de veículos com a mesma capacidade. Permite que esses veículos realizem múltiplas viagens num período finito de tempo, ou seja o período de trabalho [Cac+14].

Na Secção 4.2 será apresentada a explicação do procedimento computacional de VRP utilizado, seguido do pseudocódigo na Secção 4.3 e do código na linguagem de programação *Python* na Secção 4.4 e em *Julia* na Secção 4.5. Por último, na Secção 4.6, são apresentados os gráficos com resultados experimentais da taxa de crescimento do tempo de execução.

4.2 Procedimento computacional de VRP

O presente projeto utiliza uma heurística aleatória baseada em poupanças para VRP com múltiplas viagens. Este procedimento computacional combina a heurística de *Clarke and Wright* e uma memória temporária da melhor rota encontrada. Para que este procedimento computacional funcione num cenário de VRP heterogêneo, são necessárias algumas alterações para obter rotas ótimas, como a utilização de métodos de procura local: o *2-Opt* e uma técnica de divisão avançada [Cac+14].

Neste caso, uma rota é viável se a procura total de clientes visitados por um veículo não ultrapassar a sua capacidade. O custo total de uma solução é baseado na distância total, juntamente com o custo fixo de utilização de um veículo. No algoritmo de Caceres Cruz e colaboradores [Cac+14] tenta-se atribuir as rotas aos veículos menores, pois estes têm um custo de utilização menor, neste caso isso não é tido em conta, uma vez que todos os veículos têm a mesma capacidade.

Antes da execução do procedimento computacional são feitas algumas verificações, como por exemplo, se a maior procura dos clientes não excede a capacidade máxima do camião.

Este procedimento computacional utiliza uma aleatorização tendenciosa de uma heurística determinística, que permite transformá-lo num procedimento computacional probabilístico *multi-start* [Cac+14]. Neste projeto, manteve-se as características metaheurísticas desejáveis, como a precisão, a velocidade, a simplicidade e a flexibilidade [Cac+14].

A heurística de *Clarke and Wright* é uma primeira aproximação, para obter uma melhor qualidade de rotas, deve ser adotado um algoritmo mais avançado.

O procedimento computacional tem como parâmetros de entrada, os custos de mover um nó para outro, as procuras de clientes, a capacidade máxima do veículo, o tempo máximo e o número de iterações do processo *multi-start*. Este é executado uma vez por dia, tendo em conta o nível de resíduos de cada caixote em tempo real. Todos os motoristas ativos são notificados ao serem geradas novas rotas.

Primeiramente, é criada uma solução inicial: o procedimento computacional calcula as poupanças associadas a cada aresta que conecta dois clientes diferentes, ou seja, as poupanças associadas a uma determinada aresta são calculadas baseadas na sua redução de custos (distâncias e/ou tempos) [Cac+14].

As arestas são armazenadas numa lista que é classificada da maior para a menor poupança e inicia-se um processo de inicialização múltipla. Este processo é executado repetidamente até que uma condição final baseada em iterações seja alcançada. A cada iteração deste processo *multi-start* são executadas várias etapas [Cac+14].

É, assim, obtida uma nova lista de poupanças, aleatorizando a lista original. Ao aleatorizar a lista de poupanças da heurística de *Clarke and Wright*, a heurística determinística descrita na próxima etapa é transformada num procedimento computacional probabilístico. Portanto, cada combinação de arestas tem uma certa probabilidade de ser selecio-

nada. A fase de construção visa selecionar um conjunto diferente de arestas. Este tipo de procedimento computacional ajuda a explorar o espaço de soluções viáveis de forma diversificada. Isto permite obter diferentes resultados em cada iteração do procedimento *multi-start* [Cac+14].

Ao utilizar uma distribuição de probabilidade tendenciosa, neste caso a geométrica, a maior parte da lógica por trás da heurística clássica é mantida, ou seja, as arestas com poupanças mais altas terão uma maior probabilidade de serem selecionadas do que as com poupanças mais baixas. Na sua implementação, as poupanças negativas são logo descartadas, uma vez que não têm qualquer vantagem em serem utilizadas.

Inicia-se um processo iterativo até que a lista de poupanças fique vazia, no qual a aresta no topo da lista aleatória é obtida, esta aresta irá ligar duas rotas diferentes (**Rota A** e **Rota B**). Para que haja fusão, as duas rotas precisam estar diretamente ligadas ao depósito, e também as restrições de capacidade e do tempo máximo da rota são validadas [Cac+14].

Os *outputs* do procedimento computacional de VRP foram conferidos com dados conhecidos anteriormente, como se pode consultar no Anexo II.

4.3 Pseudocódigo

O procedimento computacional tem como parâmetros de entrada: os custos; a capacidade dos resíduos; a capacidade máxima do camião; o tempo máximo de cada rota e o número máximo de iterações do processo *multi-start*.

Inicialmente é calculada a lista de poupanças presente na heurística de *Clarke and Wright*, e também é calculada uma solução inicial que corresponde a ir do depósito a cada caixote e regressar. Inicia-se o processo *multi-start* em que a solução atual é a inicial. Na linha 5 do pseudocódigo da listagem 4.1, o critério de paragem é o número máximo de iterações. Este é definido antes de executar o procedimento computacional.

A lista de poupanças é aleatorizada com o pressuposto de que as poupanças maiores têm maior probabilidade de aparecerem primeiro. Inicia-se um ciclo em que se vai retirando o primeiro elemento dessas poupanças e se atribui essas rotas à **Rota A** e à **Rota B**.

Verifica-se se a fusão é válida, ou seja, as duas rotas precisam estar diretamente ligadas ao depósito, e cumprir com as restrições de capacidade e do tempo máximo da rota.

Uma rota é melhor se tiver um custo mais baixo que a melhor rota até ao momento, e caso seja encontrada numa determinada iteração, esta é guardada. Como se trata de um procedimento computacional *multi-start*, este será iterado múltiplas vezes, e numa das vezes deverá encontrar a melhor solução.

4. PROBLEMA DE CÁLCULO DE ROTAS DE VEÍCULOS

Na listagem 4.1 é possível observar o pseudocódigo do procedimento computacional Rand-MER.

Listing 4.1 Pseudocódigo do procedimento computacional Rand-MER.

```
1: procedure VRP(nodes, costs, vehicles)
2:   Initialize savingsList  $\leftarrow$  COMPUTESAVINGSLIST(nodes, costs)
3:   Initialize initialSol  $\leftarrow$  CONSTRUCTINITIALSOL(nodes, costs)
4:   Initialize bestSol  $\leftarrow$  initialSol
5:   while stopping criteria not satisfied do ▷ Defined by MaxIter parameter
6:     currentSol  $\leftarrow$  initialSol ▷ Reset base solution
7:     randomSavingsList  $\leftarrow$  BIASEDRANDOMIZATION(savingsList)
8:     while randomSavingsList is not empty do
9:       savingEdge  $\leftarrow$  SELECTNEXTEDGE(randomSavingsList)
10:      routeA  $\leftarrow$  GETSTARTINGROUTE(savingEdge)
11:      routeB  $\leftarrow$  GETCLOSINGROUTE(savingEdge)
12:      mergeIsValid  $\leftarrow$  VALIDATEMERGECONSTRAINTS(savingEdge, routeA, routeB, vehicles, currentSol)
13:      if mergeIsValid then
14:        unifiedRoute  $\leftarrow$  UNIFYROUTES(routeA, routeB, savingEdge)
15:      end if
16:    end while
17:    if currentSol outperforms bestSol then
18:      bestSol  $\leftarrow$  currentSol
19:    end if
20:  end while
21:  return bestSol
22: end procedure
```

Na listagem 4.2 é possível observar o pseudocódigo das restrições para efetuar a fusão de 2 rotas.

Listing 4.2 Pseudocódigo da validação para fazer a fusão.

```
1: procedure VALIDATEMERGECONSTRAINTS(edge, routeA, routeB, vehicles, solution)
2:   if nodes of edge are internal in routeA or routeB then
3:     return false
4:   end if
5:   if cost(routeA) + cost(routeB) - saving(edge) > MaxRouteLength then
6:     return false
7:   end if
8:   if capacity(vehicle) < load(routeA) + load(routeB) then
9:     return false
10:  end if
11:  return true
12: end procedure
```

4.4 Codificação do procedimento computacional

A codificação do procedimento computacional foi desenvolvida na linguagem de programação Python, devido à sua facilidade de leitura.

No momento da criação das rotas, esta aplicação irá sempre buscar uma nova matriz de custos para ter em conta as condições atuais, por exemplo uma rua mudar de sentido, ou até mesmo os caixotes mudarem de posição. A matriz de custos é calculada tendo em conta que se trata de veículos pesados e que existem ruas onde estes não podem passar.

De modo a evitar utilizar os ciclos *for*, pois estes reduzem o desempenho, são utilizados vetores de rastreamento, que são estruturas de dados simples. Estes adicionam a um vetor os índices de cada lista utilizada, e assim, é possível obter um melhor desempenho do procedimento computacional, sendo executado em menos tempo. Caso exista uma fusão, os vetores de rastreamento são atualizados.

Os 4 vetores de rastreamento utilizados foram:

- fR: Que indica qual é o primeiro cliente de cada rota;
- lR: Que indica qual é o último cliente de cada rota;
- dR: Que indica qual é a procura dos clientes incluídas em cada rota;
- cr: Que indica qual o custo para cada rota.

No código da listagem 4.3, é possível observar o cálculo das poupanças, ou seja $S_{ij} = C_{i0} + C_{0j} - C_{ij}$ e a sua ordenação através da função *sorted*.

```

1  class Vrp():
2      '''
3      A Savings-based Randomized Heuristic for VRP with Multi-trips
4
5      Input data:
6          :param costs: List # The costs of moving from one node to another in sec
7          :param q: List # Individual demands
8          :param max_capacity: Int # Vehicle max capacity
9          :param max_time: Int # Max time in sec
10         :param max_ms_iter: Int # Maximum number of iterations of the multi-start
11         process
12     '''
13     def __init__(self, costs, q, max_capacity, max_time, max_ms_iter):
14         self.costs = costs
15         self.q = q
16         self.max_capacity = max_capacity
17         self.max_time = max_time
18         self.max_ms_iter = max_ms_iter
19         self.fR = [] # Firsts elements
20         self.lR = [] # Lasts elements

```

```

21     self.dR = [] # Demands
22     self.cR = [] # Costs
23     self.idx_a = 0 # Index Route A
24     self.idx_b = 0 # Index Route B
25
26     def compute_savings_list(self):
27         routes = [[i] for i in range(1, len(self.q))]
28
29         #  $S_{ij} = C_{i0} + C_{0j} - C_{ij}$ 
30         savings = [
31             [i, j, (self.costs[i][0] + self.costs[0][j]) - self.costs[i][j]]
32             for i in range(1, len(routes) + 1)
33             for j in range(1, len(routes) + 1)
34             if i != j
35         ]
36
37         return sorted(savings, key=itemgetter(2), reverse=True)

```

Listing 4.3: Função para calcular poupanças e ordenar.

O código da listagem 4.4, permite baralhar as poupanças aleatoriamente, de acordo com os pesos, ou seja, as poupanças maiores têm mais probabilidade de aparecerem primeiro.

```

1  def biased_randomization(self, savings_list):
2      x = 0
3      saving_idx = 2
4
5      # Delete negative numbers
6      pos_savings_list = list(filter(lambda item: item[saving_idx] > 0, savings_list))
7
8      x = sum(item[saving_idx] for item in pos_savings_list)
9
10     weights = [item[saving_idx] / x for item in pos_savings_list]
11     #print(sum(weights)) # 1
12
13     selected_indices = np.random.choice(len(pos_savings_list),
14                                         size=len(pos_savings_list), replace=False, p=weights)
15
16     return [pos_savings_list[i] for i in selected_indices]

```

Listing 4.4: Função destinada a aleatorizar as poupanças.

Ao executar o procedimento computacional, este vai retirar o primeiro elemento da lista aleatória através da chamada de função (`pop(0)`) e atribuir o **Rota A** ao primeiro nó e atribuir o **Rota B** ao segundo nó.

No código da listagem 4.5, é possível observar as 4 condições para executar a fusão, ou seja, verificar se **Rota A** não é o último ou se **Rota B** não é o primeiro, se o índice de **Rota A** não é igual ao índice de **Rota B**, se a carga não ultrapassa a capacidade máxima e se a fusão excede o tempo máximo permitido, ou seja as 4 horas e 30 minutos.

```

1  def validate_merge_constraints(self, route_a, route_b, saving_edge):
2      # 1º Condition - Check if Route A is not last or if Route B is not first
3      if not route_a in self.lR or not route_b in self.fR:
4          return False # No merge
5
6      # 2º Condition - Check if the index of Route A is not the same as the
7      # index of Route B
8      self.idx_a = self.lR.index(route_a)
9      self.idx_b = self.fR.index(route_b)
10
11     if self.idx_a == self.idx_b:
12         return False # No merge
13
14     # 3º Condition - Check that demands does not exceed maximum capacity
15     sum_pD = self.dR[self.idx_a] + self.dR[self.idx_b]
16     if sum_pD > self.max_capacity:
17         return False # No merge
18
19     # 4º Condition - Check if the merging exceeds the max time
20     merging_cost = self.cR[self.idx_a] + self.cR[self.idx_b] - saving_edge[2]
21
22     if merging_cost > self.max_time:
23         return False # No merge
24
25     return True # Merge

```

Listing 4.5: Função para validar se é possível fundir rotas.

No código da listagem 4.6, é possível verificar o que acontece caso a fusão seja válida, ou seja, executa a fusão, atualiza os vetores de rastreamento e a solução atual. Os elementos da posição da lista são substituídos por *None*, para que esta esteja ocupada, de modo a não serem desassociados, posteriormente são removidos com recurso à função *filter*.

```

1  # Check merging conditions
2  merge_is_valid = self.validate_merge_constraints(route_a, route_b, saving_edge)
3

```

```
4  if merge_is_valid:
5      # Update current sol
6      x = current_sol[self.idx_a]
7      y = current_sol[self.idx_b]
8      z = x + y # Unified Route
9      current_sol.append(z)
10
11     current_sol[self.idx_a] = None
12     current_sol[self.idx_b] = None
13     current_sol = self.remove_None(current_sol)
14
15     # Update Firsts and Lasts
16     self.fR.append(self.fR[self.idx_a])
17     self.lR.append(self.fR[self.idx_b])
18
19     self.fR[self.idx_a] = None
20     self.fR[self.idx_b] = None
21
22     self.lR[self.idx_a] = None
23     self.lR[self.idx_b] = None
24
25     self.fR = self.remove_None(self.fR)
26     self.lR = self.remove_None(self.lR)
27
28     # Update Demands
29     sum_pD = self.dR[self.idx_a] + self.dR[self.idx_b]
30
31     self.dR.append(sum_pD)
32
33     self.dR[self.idx_a] = None
34     self.dR[self.idx_b] = None
35     self.dR = self.remove_None(self.dR)
36
37     # Update Costs
38     merging_cost = self.cR[self.idx_a] + self.cR[self.idx_b] - saving_edge[2]
39
40     self.cR.append(merging_cost)
41
42     self.cR[self.idx_a] = None
43     self.cR[self.idx_b] = None
44     self.cR = self.remove_None(self.cR)
```

Listing 4.6: Atualização dos dados para fazer a fusão de 2 rotas.

Como se trata de um procedimento computacional *multi-start*, este será iterado múltiplas vezes, numa das vezes deverá encontrar a melhor solução. Caso seja encontrada uma rota com menor custo que a melhor até ao momento, essa passa a ser a melhor.

A listagem 4.7 mostra o código principal do procedimento computacional Rand-MER.

```

1  def rand_mer(self):
2      # Create Clarke & Wright's savings sorted list
3      savings_list = self.compute_savings_list()
4
5      print('\n== == Savings == ==')
6      for i in range(len(savings_list)):
7          print(savings_list[i][0], '--', savings_list[i][1], ' ', savings_list[i][2])
8
9      # Build initial dummy solution
10     initial_sol = [[i] for i in range(1, len(self.costs))]
11
12     best_sol = initial_sol
13
14     self.cR = [self.costs[0][i+1] + self.costs[i+1][0] for i in range(len(self.costs)
15 - 1)]
16
17     best_sol_cost = sum(self.cR)
18
19     print('\n== == Multi-start process == ==')
20
21     # Multi-start process with a stopping criteria
22     while self.max_ms_iter > 0:
23         # Reset the base solution to the dummy solution
24         initial_sol = [[i] for i in range(1, len(self.costs))]
25         current_sol = initial_sol
26
27         self.fR = [i for i in range(1, len(self.costs))]
28
29         self.lR = [i for i in range(1, len(
30 self.costs))]
31
32         self.dR = [self.q[idx] for idx in range(len(self.q)) if idx != 0]
33
34         self.cR = [self.costs[0][i+1] + self.costs[i+1][0] for i in range(len(
35 self.costs) - 1)]
36
37
38         # Biased randomized selection of savings
39         random_savings_list = self.biased_randomization(savings_list.copy())
40
41         # Execute the route merging process
42         while random_savings_list:

```

```
43         saving_edge = random_savings_list.pop(0)
44
45         route_a = saving_edge[0]
46         route_b = saving_edge[1]
47
48         # Check merging conditions
49         merge_is_valid = self.validate_merge_constraints(route_a, route_b,
50 saving_edge)
51
52         if merge_is_valid:
53             # Update current sol
54             x = current_sol[self.idx_a]
55             y = current_sol[self.idx_b]
56             z = x + y # Unified Route
57
58             current_sol.append(z)
59
60             current_sol[self.idx_a] = None
61             current_sol[self.idx_b] = None
62             current_sol = self.remove_None(current_sol)
63
64             # Update Firsts and Lasts
65             self.fR.append(self.fR[self.idx_a])
66             self.lR.append(self.fR[self.idx_b])
67
68             self.fR[self.idx_a] = None
69             self.fR[self.idx_b] = None
70
71             self.lR[self.idx_a] = None
72             self.lR[self.idx_b] = None
73
74             self.fR = self.remove_None(self.fR)
75             self.lR = self.remove_None(self.lR)
76
77             # Update Demands
78             sum_pD = self.dR[self.idx_a] + self.dR[self.idx_b]
79
80             self.dR.append(sum_pD)
81
82             self.dR[self.idx_a] = None
83             self.dR[self.idx_b] = None
84             self.dR = self.remove_None(self.dR)
85
86             # Update Costs
87             merging_cost = self.cR[self.idx_a] + self.cR[self.idx_b] -
88 saving_edge[2]
89
```

```

90         self.cR.append(merging_cost)
91
92         self.cR[self.idx_a] = None
93         self.cR[self.idx_b] = None
94         self.cR = self.remove_None(self.cR)
95
96         current_sol_cost = sum(self.cR)
97
98         if current_sol_cost < best_sol_cost:
99             print(f"a better sol found: {current_sol}")
100             best_sol = current_sol # Update Best Solution
101             best_sol_cost = current_sol_cost # Update Best Solution Cost
102             pass
103         pass
104     pass
105
106     self.max_ms_iter -= 1
107     pass
108
109     # Add 0 at the start and end of each route
110     best_sol = [[0] + route + [0] for route in best_sol]
111
112     print('\n== == == Result == == ==')
113     print(f'route: {best_sol} total cost: {round(best_sol_cost, 3)}')
114
115     return best_sol

```

Listing 4.7: Processo multi-start do procedimento computacional Rand-MER.

4.5 Procedimento computacional em Julia

Os algoritmos desenvolvidos em `Python` tendem a ter um tempo de execução maior quando comparados com implementações semelhantes feitas em linguagens modernas como `Julia` [Jul24]. A linguagem de programação `Julia` é projetada para ser uma linguagem otimizada para um maior desempenho numérico, por isso, `Julia` normalmente oferece uma vantagem significativa em termos de tempo de execução.

Verificou-se o tempo de execução do procedimento computacional em `Julia` e em `Python`. Concluiu-se que em `Julia`, o tempo de execução é significativamente mais baixo. Caso o projeto venha a escalar para grandes cidades como Lisboa ou Porto, este necessita de ser executado rapidamente para muitos caixotes, e por isso é apresentada a mesma abordagem, mas em `Julia`.

No código da listagem 4.8, é possível observar o cálculo das poupanças, ou seja $S_{ij} = C_{i0} + C_{0j} - C_{ij}$ a sua ordenação através da função `sort`.

```
1  mutable struct Vrp
2      costs::Matrix{Float64} # The costs of moving from one node to another in sec
3      q::Vector{Float64} # Individual demands
4      max_capacity::Int # Vehicle max capacity
5      max_time::Int # Max time in sec
6      max_ms_iter::Int # Maximum number of iterations of the multi-start process
7      fR::Vector{Union{Int, Nothing}} # Firsts elements
8      lR::Vector{Union{Int, Nothing}} # Lasts elements
9      dR::Vector{Union{Int, Nothing}} # Demands
10     cR::Vector{Union{Int, Nothing}} # Costs
11     idx_a::Int # Index Route A
12     idx_b::Int # Index Route B
13 end
14
15 function compute_savings_list(vrp::Vrp)
16     n = length(vrp.q) - 1
17     savings = []
18     #  $S_{ij} = C_{i0} + C_{0j} - C_{ij}$ 
19     for i in 1:n
20         for j in 1:n
21             if i != j
22                 saving = vrp.costs[i + 1, 1] + vrp.costs[1, j + 1] -
23                     vrp.costs[i + 1, j + 1]
24                 push!(savings, [i, j, saving])
25             end
26         end
27     end
28
29     return sort(savings, by=x->x[3], rev=true)
30 end
```


Listing 4.8: Código da função `compute savings list` em Julia.

O código da listagem 4.9, permite baralhar as poupanças aleatoriamente, de acordo com os pesos, onde as poupanças maiores têm mais probabilidade de aparecerem primeiro.

```

1  function biased_randomization(vrp::Vrp, savings_list)
2      x = 0
3      saving_idx = 3
4
5      # Delete negative numbers
6      pos_savings_list = filter(item -> item[saving_idx] > 0, savings_list)
7
8      x = sum(item[saving_idx] for item in pos_savings_list)
9
10     weights = [item[saving_idx] / x for item in pos_savings_list]
11
12     selected_indices = sample(1:length(pos_savings_list),
13                               Weights(weights),
14                               length(pos_savings_list),
15                               replace=false)
16
17     return [pos_savings_list[i] for i in selected_indices]
18 end

```

Listing 4.9: Código da função `biased randomization` em Julia.

No código da listagem 4.10, é possível observar as 4 condições para executar a fusão.

```

1  function validate_merge_constraints(vrp::Vrp, route_a, route_b, saving_edge)
2      # 1º Condition - Check if Route A is not last or if Route B is not first
3      if !(route_a in vrp.lR) || !(route_b in vrp.fR)
4          return false # No merge
5      end
6
7      # 2º Condition - Check if the index of Route A is not the same as the index
8      of Route B
9      vrp.idx_a = findfirst(isequal(route_a), vrp.lR)
10     vrp.idx_b = findfirst(isequal(route_b), vrp.fR)
11
12     if vrp.idx_a == vrp.idx_b
13         return false # No merge
14     end
15

```

```

16     # 3º Condition - Check that demands does not exceed maximum capacity
17     sum_pD = vrp.dR[vrp.idx_a] + vrp.dR[vrp.idx_b]
18     if sum_pD > vrp.max_capacity
19         return false # No merge
20     end
21
22     # 4º Condition - Check if the merging exceeds the max time
23     merging_cost = vrp.cR[vrp.idx_a] + vrp.cR[vrp.idx_b] - saving_edge[3]
24
25     if merging_cost > vrp.max_time
26         return false # No merge
27     end
28
29     return true # Merge
30 end

```

Listing 4.10: Código da função validate merge constraints em Julia.

A listagem 4.11 mostra a função Rand-MER, responsável por chamar todas as outras, executar a fusão, atualizar os vetores de rastreamento e a solução atual.

```

1  function rand_mer(vrp::Vrp)
2      # Create Clarke & Wright's savings sorted list
3      savings_list = compute_savings_list(vrp)
4
5      # Build initial dummy solution
6      initial_sol = [Int64[i] for i in 1:(length(vrp.q) - 1)]
7
8      best_sol = initial_sol
9
10     vrp.cR = [vrp.costs[1, i + 1] + vrp.costs[i + 1, 1] for i in 1:length(vrp.q)
11               - 1]
12
13     best_sol_cost = sum(vrp.cR)
14
15     # Multi-start process with a stopping criteria
16     while vrp.max_ms_iter > 0
17         # Reset the base solution to the dummy solution
18         current_sol = deepcopy(initial_sol)
19         vrp.fR = collect(1:(length(vrp.q) - 1))
20         vrp.lR = collect(1:(length(vrp.q) - 1))
21         vrp.dR = [vrp.q[i + 1] for i in 1:(length(vrp.q) - 1)]
22         vrp.cR = [vrp.costs[1, i + 1] + vrp.costs[i + 1, 1]
23                 for i in 1:length(vrp.q)-1]
24
25         # Biased randomized selection of savings

```

```

26     random_savings_list = biased_randomization(vrp, savings_list)
27     # Execute the route merging process
28     while !isempty(random_savings_list)
29         saving_edge = popfirst!(random_savings_list)
30
31         route_a = saving_edge[1]
32         route_b = saving_edge[2]
33
34         # Check merging conditions
35         merge_is_valid = validate_merge_constraints(vrp, route_a, route_b,
36             saving_edge)
37
38         if merge_is_valid
39             # Update current sol
40             x = current_sol[vrp.idx_a]
41             y = current_sol[vrp.idx_b]
42             z = vcat(x, y) # Unified Route
43             push!(current_sol, z)
44
45             current_sol[vrp.idx_a] = []
46             current_sol[vrp.idx_b] = []
47             current_sol = filter(!isempty, current_sol)
48
49             # Update Firsts and Lasts
50             push!(vrp.fR, vrp.fR[vrp.idx_a])
51             push!(vrp.lR, vrp.fR[vrp.idx_a])
52
53             vrp.fR[vrp.idx_a] = nothing
54             vrp.fR[vrp.idx_b] = nothing
55
56             vrp.lR[vrp.idx_a] = nothing
57             vrp.lR[vrp.idx_b] = nothing
58
59             vrp.fR = filter(x -> !isnothing(x), vrp.fR)
60             vrp.lR = filter(x -> !isnothing(x), vrp.lR)
61
62             # Update Demands
63             sum_pD = vrp.dR[vrp.idx_a] + vrp.dR[vrp.idx_b]
64
65             push!(vrp.dR, sum_pD)
66
67             vrp.dR[vrp.idx_a] = nothing
68             vrp.dR[vrp.idx_b] = nothing
69             vrp.dR = filter(x -> !isnothing(x), vrp.dR)
70
71             # Update Costs
72             merging_cost = vrp.cR[vrp.idx_a] + vrp.cR[vrp.idx_b]

```

```

73         - saving_edge[3]
74
75         push!(vrp.cR, merging_cost)
76
77         vrp.cR[vrp.idx_a] = nothing
78         vrp.cR[vrp.idx_b] = nothing
79         vrp.cR = filter(x -> !isnothing(x), vrp.cR)
80
81         current_sol_cost = sum(vrp.cR)
82
83         if current_sol_cost < best_sol_cost
84             best_sol = current_sol # Update Best Solution
85             best_sol_cost = current_sol_cost # Update Best Solution Cost
86         end
87     end
88 end
89 vrp.max_ms_iter -= 1
90 end
91
92 # Add 0 at the start and end of each route
93 best_sol = [vcat([0], route, [0]) for route in best_sol]
94
95 println("route: ", best_sol, " total cost: ", round(best_sol_cost, digits=3))
96 end

```

Listing 4.11: Código da função Rand-MER em Julia.

4.6 Taxa de crescimento do tempo de execução

Nesta dissertação foi analisada a taxa de crescimento do tempo de execução, pois esta dá uma indicação de como pode escalar o procedimento computacional. Para cada conjunto de elementos foram feitas 30 execuções do procedimento computacional e obtida a mediana do tempo de execução, uma vez que, a mediana é mais adequada a sistemas irregulares do que a média. Assim, é possível verificar o crescimento do procedimento computacional, de modo a saber se este é eficiente ou não.

Através da análise do código do procedimento computacional na listagem 4.1, conclui-se que o procedimento computacional cresce exponencialmente em relação ao tamanho da entrada.

A função da listagem 4.12 é destinada realizar uma amostragem sem reposição de uma lista de índices baseada num conjunto de pesos. Esta função é onde se verifica mais ganho de todo o procedimento computacional.

O valor mediano do tempo de execução do procedimento computacional para 100 elementos aleatórios na linguagem de programação em **Python** é de 6.91400933265686 segundos e em **Julia** é de 1.843022963 segundos. Para 120 elementos aleatórios, em **Python** é de 11.586083889007568 segundos e em **Julia** é de 2.341676891 segundos.

$$\frac{11.586083889007568}{2.341676891} \approx 4.95$$

O tempo de execução em **Python** é aproximadamente 5 vezes maior do que em **Julia**.

```
1 selected_indices = np.random.choice(len(pos_savings_list),
2 size=len(pos_savings_list), replace=False, p=weights)
```

Listing 4.12: Função destinada a aleatorizar as poupanças.

O procedimento computacional foi executado numa máquina com o sistema operativo Ubuntu na versão 22.04, um processador AMD EPYC 7453 com 28 *Cores*, 4 placas gráficas NVIDIA A100 SXM4 de 80 GB e 2 TB de memória RAM.

Foram criados gráficos da taxa de crescimento do tempo de execução em que o eixo do x corresponde ao número de elementos e o eixo do y ao tempo de execução em segundos, estes foi desenhados com o **gnuplot**. Os dados de entrada são para dados aleatórios distribuídos, assim os caixotes vão estar uniformemente espalhados. Para melhorar deve-se utilizar dados reais de *datasets* de VRP [Día06].

4. PROBLEMA DE CÁLCULO DE ROTAS DE VEÍCULOS

A figura 4.1 apresenta o gráfico da taxa de crescimento do tempo de execução do procedimento computacional.

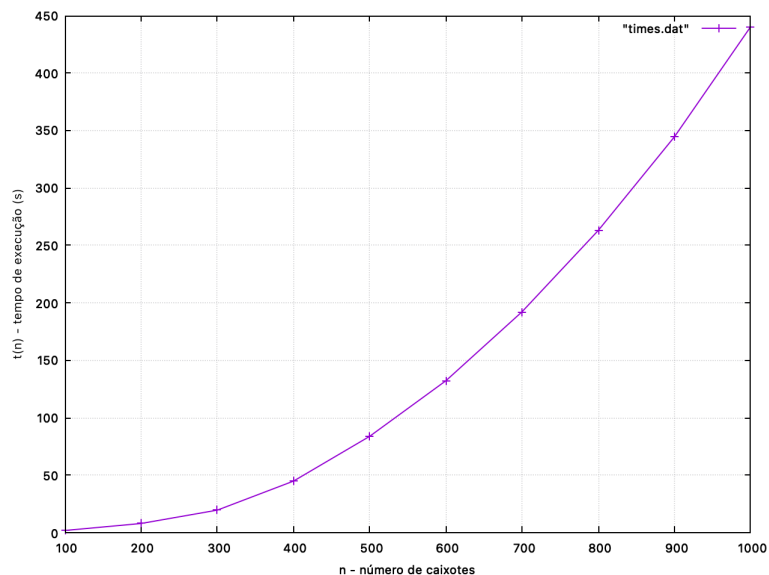


Figura 4.1: Gráfico da taxa de crescimento do tempo de execução do procedimento computacional.

A figura 4.2 apresenta o gráfico da taxa de crescimento do tempo de execução da função destinada a realizar uma amostragem sem reposição de uma lista de índices baseada num conjunto de pesos, esta influencia o crescimento exponencial deste procedimento computacional.

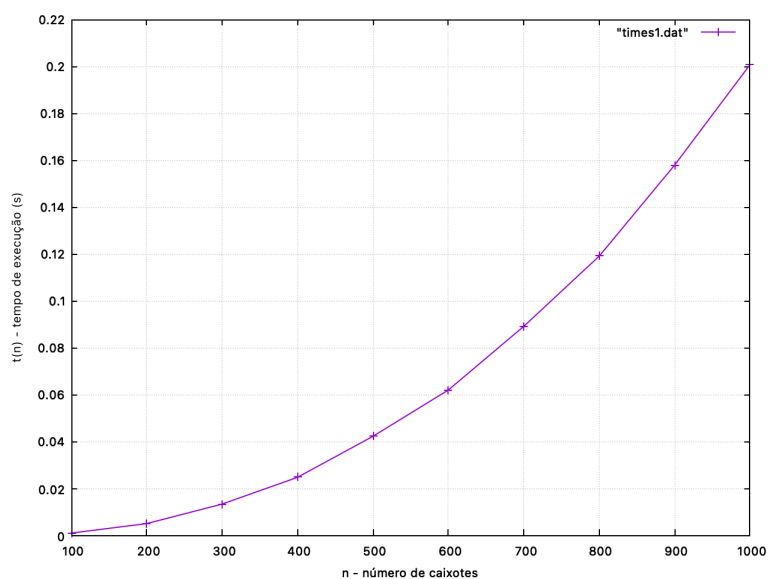


Figura 4.2: Gráfico da taxa de crescimento do tempo de execução.

4.7 Conclusão

O procedimento computacional **Rand-MER** implementado neste projeto, oferece uma abordagem eficaz e flexível para VRP na recolha de resíduos, utilizando caixotes do lixo inteligentes.

Alguns aspetos iniciais do desempenho computacional devem ser melhorados futuramente, uma vez que, foram obtidos com dados gerados aleatoriamente. De modo a garantir uma noção real do crescimento do procedimento computacional deve-se utilizar dados reais, através de *datasets* de VRP [Día06].

A abordagem escolhida, aplicada à otimização das rotas, contribui para ações mais sustentáveis e eficientes para cidades mais pequenas. Porém, caso o número de caixotes venha a aumentar substancialmente, é necessário utilizar um algoritmo mais eficiente.

Capítulo 5

Implementação Experimental

5.1 Introdução

O tema deste capítulo é a implementação experimental do caixote do lixo inteligente **e-Bin**. Apresenta-se a parte de *hardware* e de *software* utilizado para desenvolver este protótipo. Na parte de *hardware*, na Secção 5.2, é apresentada a plataforma de desenvolvimento de *hardware* escolhida para este projeto.

O desenvolvimento deste sistema envolve várias etapas fundamentais, como a construção do circuito eletrónico na Secção 5.3 e a implementação de componentes na *breadboard* na Secção 5.4. Na Secção 5.5 é apresentada a parte da alimentação deste projeto.

Na parte de *software*, na Secção 5.6 será apresentada a tecnologia de *containers* adotada. Além disso, na Secção 5.7 será abordada a tecnologia de comunicação LoRa. A LoRa é a tecnologia de longa distância mais utilizada em IoT e fornece uma variedade de configurações de comunicação de camada física, uma alta taxa de transferência, é confiável e apresenta um elevado alcance da comunicação [All24].

Na Secção 5.8 será apresentado o sistema de gestão de base de dados escolhido para armazenar os dados. A segurança informática é uma prioridade, por isso foram implementadas um conjunto de medidas apresentadas na Secção 5.9.

Na Secção 5.10 mostram-se as validações de formulários e na Secção 5.11 o envio de *emails*.

Na Secção 5.12 é apresentada a análise dos resultados obtidos e na Secção 5.13 são apresentados os testes realizados. Assim, é possível obter uma visão sobre o desempenho e a eficácia do caixote do lixo inteligente em condições reais e extremas de operação.

Por último, na Secção 5.14 é apresentada a discussão geral dos resultados experimentais.

5.2 Plataforma de desenvolvimento de *hardware*

A plataforma de desenvolvimento de *hardware* escolhida foi o LILYGO TTGO T3 LoRa32 V1.6.1, esta integra o chip ESP32-WROVER na versão 3 e um *chip* LoRa SX1276 da Semtech.

Foi escolhida esta plataforma, devido ao seu ótimo custo/benefício e relativamente baixo consumo energético, o que a distingue de outras versões anteriores da LILYGO TTGO que apresentam um maior consumo de energia, ou até mesmo de outras placas como o Arduino UNO.

O ESP32 é um *chip* que integra Wi-Fi e Bluetooth de 2.4 GHz, este foi projetado para obter a melhor potência e desempenho e conta com uma ampla variedade de aplicações [Sys19]. Neste projeto as tecnologias Wi-Fi e Bluetooth encontram-se desativadas, de modo a obter uma maior poupança de energia.

As principais características do LILYGO TTGO T3 LoRa32 868MHz V1.6.1, são o facto de conter um módulo LoRa de longo alcance, com baixo consumo energético e suporte para uma antena externa. Tem também alimentação através de bateria de lítio e permite o carregamento por USB [LIL24]. O intervalo de temperatura de funcionamento deste microcontrolador é entre -40°C a 85°C [Ein21]. A figura 5.1 mostra o microcontrolador utilizado.



Figura 5.1: LILYGO TTGO T3 LoRa32 868MHz V1.6.1 [LIL24].

O *chip* LoRa integrado no TTGO é o SX1276 de 868MHz e apresenta um baixo consumo a comunicar, pelo que não se justificou utilizar um módulo externo, como por exemplo o RFM95.

Foi instalada uma antena externa na entrada específica do microcontrolador, de forma a melhorar o ganho, para que exista uma maior capacidade de captar e transmitir sinais mais eficientemente.

A figura 5.2 mostra o diagrama de pinos do microcontrolador.

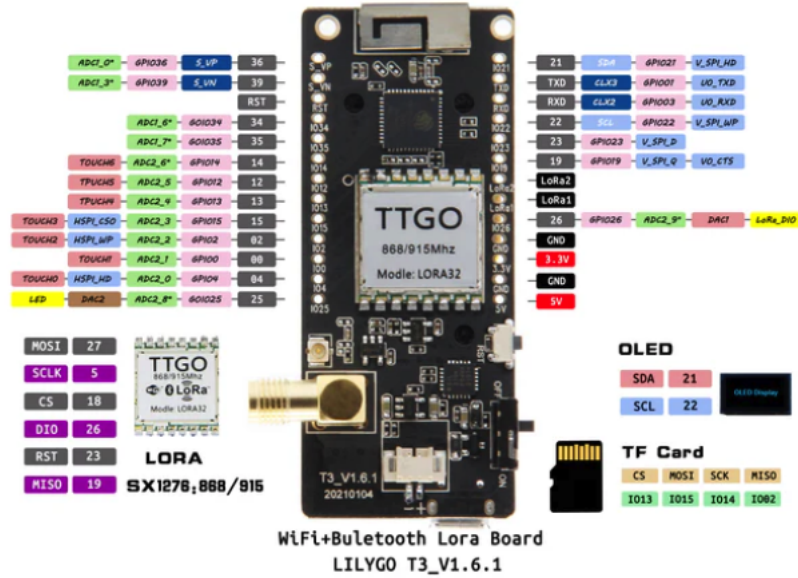


Figura 5.2: Diagrama de Pinos do LILYGO TTGO T3 LoRa32 868MHz V1.6.1 [LIL24].

Este módulo pode ter diversas aplicações, como por exemplo, a leitura automática de contadores, a automação residencial, sistemas de alarme, sistemas de irrigação de longo alcance, entre outros [Cor15].

5.3 Layout do circuito

A figura 5.3 mostra o *layout* do circuito eletrônico deste protótipo desenhado com o Kicad. Esta ferramenta avançada é muito utilizada em projetos de eletrônica, pois é *open-source* e tem uma comunidade muito ativa. Esta ferramenta permite criar esquemas de circuitos, *layouts* da placa de circuito impresso, simular circuitos e visualizá-los em 3D. [1Bi24].

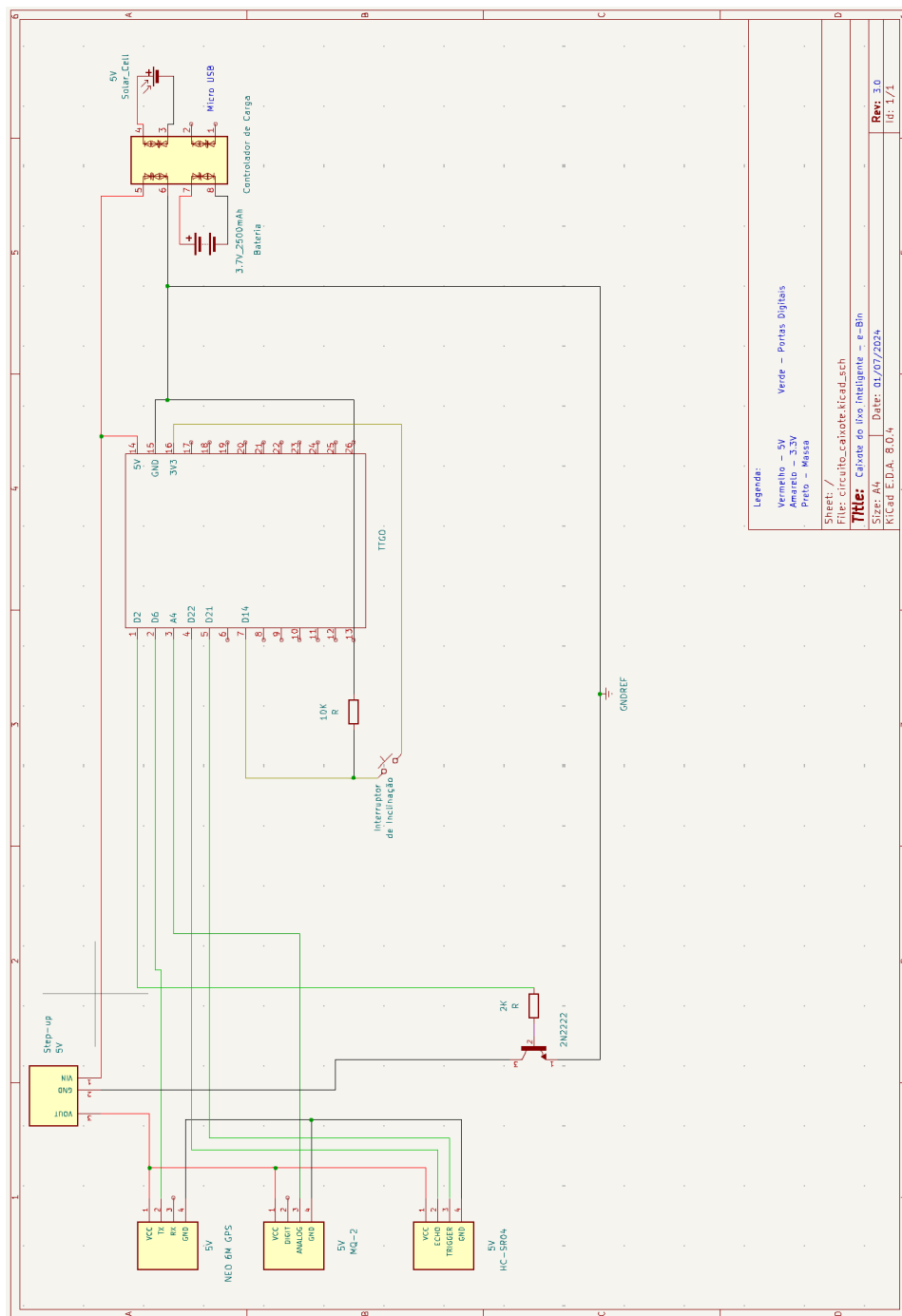


Figura 5.3: Layout do circuito eletrônico no Kicad.

A figura 5.4 mostra o esquema do circuito eletrônico deste protótipo desenvolvido com o **Fritzing**. Esta ferramenta *open-source* permite a criação de circuitos numa interface bastante intuitiva, ao contrário do Kicad é mais utilizado por iniciantes e permite a visualização realista dos componentes. Permite, também, a criação da placa de circuito impresso [Fri20].

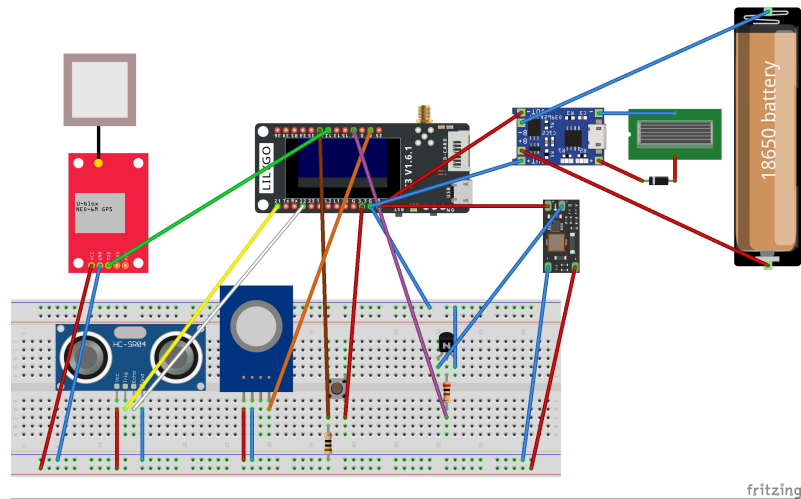


Figura 5.4: Esquema do circuito eletrônico no Fritzing.

5.4 Circuito na *breadboard*

Na figura 5.5 é possível observar o circuito implementado na *breadboard*, alimentado através de uma bateria.

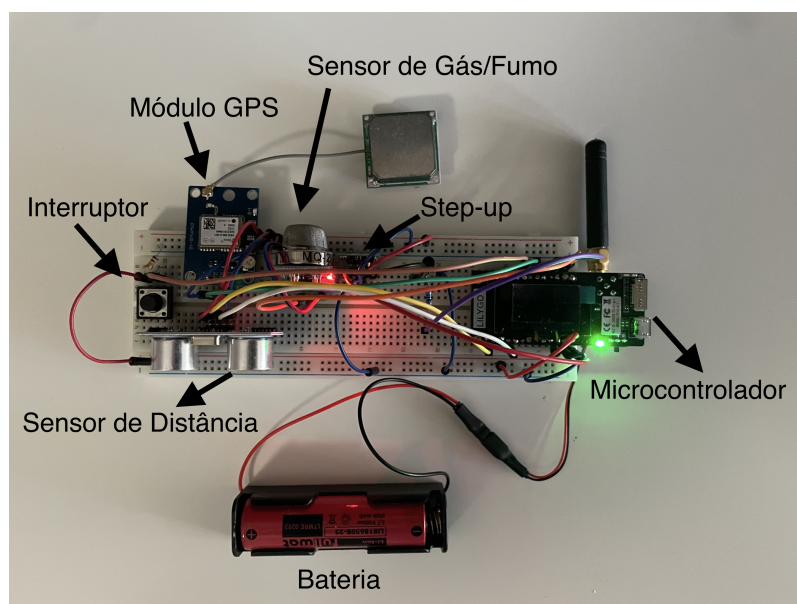


Figura 5.5: Circuito do projeto na *Breadboard*.

5.5 Fonte de alimentação

O dispositivo é alimentado através de uma bateria de lítio e um painel fotovoltaico, e portanto, é autónomo da rede elétrica.

5.5.1 Bateria

A bateria de lítio escolhida é de 3.7 V e tem capacidade de 2500 mAh. Mantém uma corrente de 2.5A por uma hora.

A figura 5.6 mostra uma bateria idêntica à que foi utilizada neste projeto.



Figura 5.6: Bateria 18650 de 3.7V e 2500mAh (em <https://www.botnroll.com/pt/18650/3983-pilha-li-ion-18650-3-7v-2500mah-18x65mm-26a-max.html>).

Os sensores escolhidos funcionam a 5V, porém a tensão da bateria é 3.7V, podendo ir até 4.2V com uma carga completa. Para elevar essa tensão para 5V foi utilizado um módulo de *Step-Up*. Este módulo é um circuito que aumenta a tensão de entrada para uma tensão de saída maior, é útil para alimentar sensores e outros componentes que operam a tensões mais elevadas do que a tensão de entrada. A figura 5.7 mostra um módulo de *Step-Up*.



Figura 5.7: Módulo de *Step-Up* MT3608 (em <https://www.botnroll.com/pt/step-up/3652-mt3608-conversor-dc-dc-step-up-sa-da-max-28v-2a.html>).

Cada leitura dura no máximo 30 segundos, ou seja, é o tempo máximo em que os sensores podem estar alimentados. Este tempo deve-se à leitura da localização que é a que pode demorar mais. Porém, caso este detete a localização imediatamente, as outras leituras também são efetuadas e os sensores são desligados, com o objetivo de poupar energia.

5.5.2 Pannel Fotovoltaico

O pannel fotovoltaico escolhido é de 3W com tensão de saída de 5V, devido ao seu baixo custo, tamanho, rendimento e adaptação para carregar a bateria de 3.7V deste projeto, como se calculou.

A figura 5.8 mostra um pannel fotovoltaico de 5V e 3W.

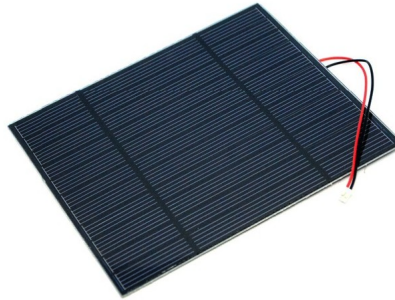


Figura 5.8: Pannel solar de 3W 5V (em <https://www.botnroll.com/pt/paineis-solares/316-pannel-solar-3w-5v.html>).

Um pannel de 3 W garante que há potência suficiente mesmo em condições menos ideais de luz solar e leva em consideração perdas de eficiência, o que pode ser confirmado de acordo com os seguintes cálculos.

$$\text{Energia (Wh)} = \text{Capacidade de bateria (Ah)} \times \text{Tensão de bateria (V)}$$

$$E = 2.5 \text{ Ah} \times 3.7 \text{ V} = 9.25 \text{ Wh}$$

$$\frac{3 \text{ W}}{5 \text{ V}} = 0.6 \text{ A}$$

A corrente máxima do pannel é de $I = 0.6 \text{ A}$, ou seja $I = 600\text{mA}$

De modo a determinar se os 3 W de potência gerada são suficientes, é calculado o tempo de carga necessário utilizando a energia do pannel.

$$t = \frac{\text{Energia necessária (Wh)}}{\text{Potência do pannel (W)}}$$

$$t = \frac{9.25 \text{ Wh}}{3 \text{ W}} = 3.08 \text{ h}$$

Uma vez que existe perda de eficiência do controlador de carga, com 90% de eficiência, a potência efetiva seria:

$$P_{\text{eff}} = 3.08 \times 0.9 = 2.7 \text{ W}$$

Em que o tempo de carga iria sofrer uma pequena alteração:

$$1h = 3600s$$

$$t = \frac{3600 \times 9.25 \text{ Wh}}{2.7 \text{ W}} \approx 12.333 \text{ Ks} \approx 3h \text{ 26m}$$

Em suma, o tempo necessário para carregar completamente a bateria de 2500 mAh utilizando um painel solar de 3 W, considerando eficiência de 90% é de aproximadamente 3 horas e 26 minutos.

Em condições mais extremas, estima-se que o sistema pode funcionar aproximadamente 24 dias sem luz solar, considerando um consumo médio de 4.23 mAh (ver mais informações no Anexo III).

$$\text{Consumo diário (mAh)} = \text{Consumo médio por hora (mA)} \times 24h$$

$$\text{Consumo diário (mAh)} = 4.23 \times 24 \approx 101.52 \text{ mAh}$$

$$\text{Dias sem luz solar} = \frac{\text{Capacidade de bateria (mAh)}}{\text{Consumo diário (mAh)}}$$

$$\text{Dias sem luz solar} = \frac{2500 \text{ mAh}}{101.76 \text{ mAh}} \approx 24.57 \text{ dias}$$

Posto isto, este painel oferece uma margem de segurança para garantir um carregamento eficiente.

5.5.3 Programação para baixo consumo energético: Modo *deep-sleep*

O modo *deep-sleep* do microcontrolador permite diminuir o consumo energético do dispositivo. Durante o modo *deep-sleep*, o coprocessador ULP e a memória RTC permanecem ligados. Assim, os periféricos como GPIO e I2C são capazes de operar, sendo possível armazenar um programa para o coprocessador ULP na memória do RTC para aceder aos periféricos, temporizadores internos e sensores internos durante o modo *deep-sleep*. Isto é útil para projetar aplicações onde o processador precisa ser acordado por um evento externo e/ou um temporizador, mantendo ao mesmo tempo um consumo mínimo de energia [Sys19].

Segundo a documentação, o consumo do coprocessador ULP ligado é de 150 μA e do temporizador RTC com a memória RTC é de 5 μA [Sys19].

Neste caso, o dispositivo acorda de hora a hora ou caso um botão de inclinação seja acionado ao abrir a tampa do caixote do lixo.

Existe um pino digital no TTGO que tem como função ativar o módulo de **Step-Up**, quando este acorda e que após a leitura e envio dos dados, desliga para que o TTGO possa entrar em modo de *deep-sleep* sem que os sensores estejam alimentados, isto resulta num menor consumo de energia e uma maior durabilidade da carga de bateria sem ser recarregada.

5.6 Tecnologia de *containers*

A descrição da tecnologia de *containers* é o primeiro assunto em termos da descrição da realização de *software*.

Os *containers* podem ser vistos como leves máquinas virtuais que permitem configurar um ambiente computacional, incluindo todas as dependências necessárias, como bibliotecas, configurações, códigos e outros dados necessários dentro de uma única unidade, a imagem. Essa imagem é documentada num ficheiro, um *script* que contém toda a configuração e comandos da infraestrutura. As imagens podem ser distribuídas publicamente e serem executadas perfeitamente no **Linux** [CFG16].

A principal diferença em relação às máquinas virtuais é que as imagens compartilham o *Kernel*, o que permite imagens muito reduzidas e com grande desempenho [CFG16].

Existem 2 tipos classificação para os *containers*, o *container* aplicacional e o *container* de sistema [Ris24].

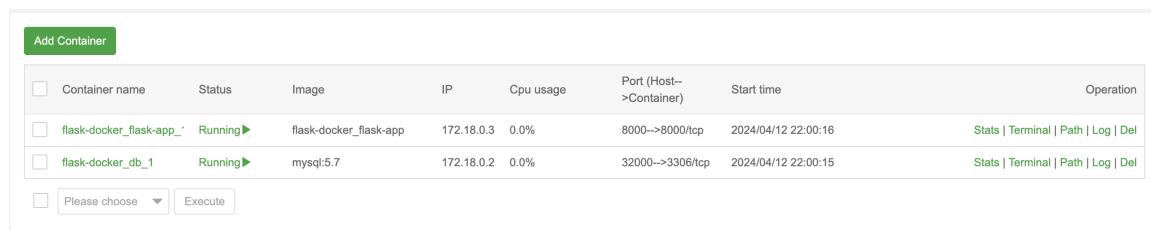
Um *container* aplicacional permite executar uma aplicação ou serviço isolado, é leve e permite encapsular a aplicação e todas as suas dependências de modo a funcionar em qualquer máquina. O *container* aplicacional mais utilizado é o **Docker**, uma vez que, tem sido muito adotado em todo o mundo. O **Docker** é uma tecnologia *open-source* que pode resolver os problemas de reproduzir a mesma aplicação noutras máquinas [CFG16].

Um *container* de sistema é semelhante a uma máquina virtual leve. Ao contrário do anterior, este isola um sistema operativo completo e os seus processos. Existem diversos *containers* de sistema, como o **LXC** e **Incus**, estes são utilizados para criar ambientes que replicam o comportamento de máquinas virtuais, porém mais leves, pois compartilham mesmo *kernel* [Ris24].

Toda a programação do lado do servidor deste projeto, assim como a base de dados, estão num *container* aplicacional **Docker**. Na figura 5.9 é possível observar os *containers* **Docker** no painel de controlo **AAPanel**.

O **Docker Compose** permite definir todos os serviços, redes e volumes de uma aplicação num único ficheiro `docker-compose.yml`. A listagem 5.1, apresenta o ficheiro `docker-compose.yml` deste projeto. Este, define dois serviços, um para a aplicação **Flask** e outro para a base de dados **MySQL**. Esta configuração permite construir a aplicação **Flask** a partir de um diretório específico, vincular os *containers*, selecionar portas para acesso externo e configurar o ambiente de desenvolvimento. Além disso, define a imagem do **MySQL**,

5. IMPLEMENTAÇÃO EXPERIMENTAL



The screenshot shows the AAPanel control panel. At the top left is a green 'Add Container' button. Below it is a table with columns: Container name, Status, Image, IP, Cpu usage, Port (Host->Container), Start time, and Operation. Two containers are listed: 'flask-docker_flask-app_1' and 'flask-docker_db_1', both in 'Running' status. Below the table is a 'Please choose' dropdown menu and an 'Execute' button.

☐	Container name	Status	Image	IP	Cpu usage	Port (Host->Container)	Start time	Operation
☐	flask-docker_flask-app_1	Running ▶	flask-docker_flask-app	172.18.0.3	0.0%	8000-->8000/tcp	2024/04/12 22:00:16	Stats Terminal Path Log Del
☐	flask-docker_db_1	Running ▶	mysql:5.7	172.18.0.2	0.0%	32000-->3306/tcp	2024/04/12 22:00:15	Stats Terminal Path Log Del

☐ Please choose ▼

Figura 5.9: Painel de controlo AAPanel.

que representa um serviço de SGBD e configura o mapeamento de portas e os volumes necessários para a iniciar a base de dados. Os volumes são locais de armazenamento de dados que o Docker utiliza para armazenar dados fora dos *containers*. Estes permitem que os dados permaneçam mesmo que o *container* seja interrompido, reiniciado ou apagado. O Docker Compose é descritivo, o utilizador define o local onde pretende que os dados sejam armazenados. É, assim, mais fácil efetuar os *backups*.

```
1  version: '3'
2  services:
3    flask-app:
4      build: flask-app/
5      links:
6        - db
7      ports:
8        - "8000:8000" # Porta App
9      volumes:
10       - ./flask-app:/usr/src/app/
11      environment:
12        - FLASK_DEBUG=1
13        - FLASK_APP=app.py
14        - FLASK_ENV=development
15    db:
16      image: mysql:5.7
17      ports:
18        - "32000:3306"
19      environment:
20        MYSQL_ROOT_PASSWORD: *****
21      volumes:
22        - ./db:/docker-entrypoint-initdb.d/:ro # Volume para inicialização
23        - db_data:/var/lib/mysql # Volume para dados permanentes no MySQL
24
```

Listing 5.1: Ficheiro docker-compose.yml.

5.7 Comunicação com o servidor

A comunicação entre os microcontroladores e o servidor é feita da seguinte forma, em primeiro lugar o microcontrolador comunica com o *gateway* através de tecnologia de comunicação LoRa, onde os dados estão codificados, de seguida o *gateway* processa esses dados e publica-os no tópico `application/867f8a4c-4678-4d3c-b92a-bb9b023d08e9/device/a80b8d8e76cca797/event/up` através do protocolo MQTT, utilizando bibliotecas para esse efeito. Posteriormente, o servidor conecta-se ao *Broker* e subscreve esse tópico, onde irá ser publicada a informação que este espera obter.

Caso não exista comunicação do dispositivo com o *gateway* mais próximo, este tentará ligar-se a outro que esteja ao seu alcance. A arquitetura do sistema *LoRaWAN* é apresentada mais detalhadamente na Secção 3.4.1.

Na figura 5.10 é possível visualizar os dados recebidos pelo *gateway* LoRaWAN na plataforma ChirpStack.

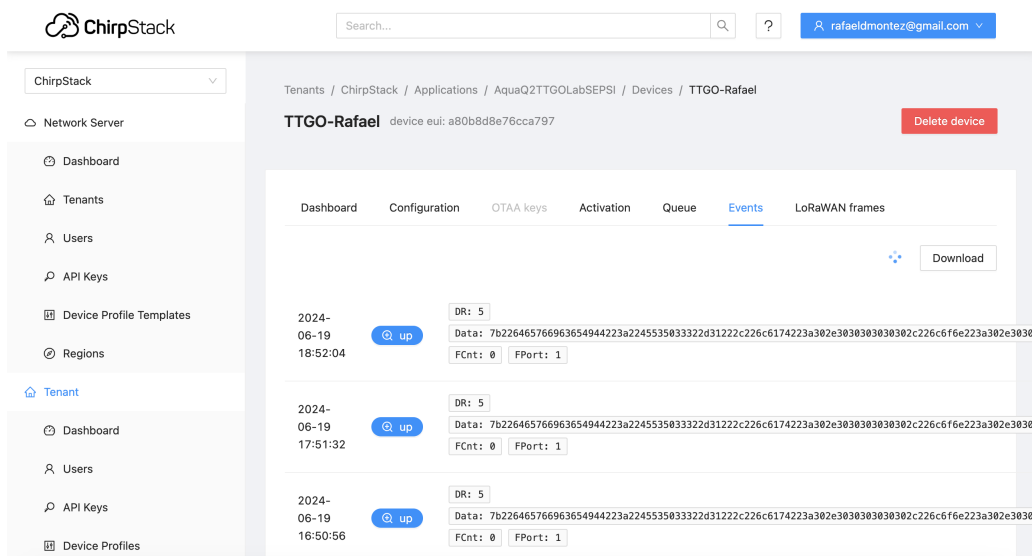


Figura 5.10: ChirpStack LoRaWAN Network-Server.

O ChirpStack é uma plataforma que permite a gestão de redes LoRaWAN. A plataforma mais utilizada é a TTN (The Things Network). Esta é mais apropriada para a partilha pública, sendo mais adequada a projetos que utilizam redes abertas. A TTN fornece um conjunto de ferramentas e uma rede aberta global para criar aplicações de IoT com baixo custo, segurança e escalável [Net24].

O ChirpStack é uma solução que permite a implementação de redes LoRaWAN em ambientes privados, pois a gestão não é exposta ao público. Neste caso existe controlo total sobre dados e infraestruturas, como *gateways* e servidores. Esta fornece uma interface *web* para a gestão de *gateways*, dispositivos, integração com *clouds*, bases de dados, entre outros [Chi24].

5.7.1 Bibliotecas do microcontrolador

Foram utilizadas 3 bibliotecas para efetuar a comunicação entre o microcontrolador e o *gateway*, a `lmic`, `hal` e a `SPI`.

De modo a obter a localização através do módulo NEO-6M V2, foi utilizada a biblioteca `TinyGPS++`, juntamente com a `SoftwareSerial`.

O nível de bateria é obtido através da biblioteca `Battery18650Stats`.

A resistência do sensor MQ-2 varia de acordo com a concentração do gás, esta é convertida num valor de tensão analógico, que é lido pelo microcontrolador sem a utilização de qualquer biblioteca e assim medir a quantidade de fumo/gás presente no ambiente.

5.7.2 Comunicações LoRaWAN

A LoRaWAN define o protocolo de comunicação e a arquitetura do sistema, enquanto a LoRa define a camada física [SYH17].

A rede LoRa é uma tecnologia que tem evoluído bastante, devido às suas vantagens técnicas e de utilização, nomeadamente, a transmissão sem fios de elevado alcance e a baixa potência que define a camada física da comunicação entre dispositivos, a segurança, a escalabilidade e a fácil utilização, o que a torna a mais utilizada [DK18].

Esta tecnologia tem uma taxa de dados mais baixa e maior latência do que o Bluetooth e é utilizada em cidades inteligentes, automações industriais e agrícolas. Possibilita ainda uma comunicação bidirecional segura, uma vez que utiliza o *Chirp Spread Spectrum*, utiliza também serviços de localização, o que permite ter boa interoperabilidade [AAA22].

A tecnologia LoRa opera na banda ISM não licenciada em 868 MHz na Europa e América do Norte e 433 MHz na Ásia. Esta tecnologia tem diversas vantagens, como o baixo custo e o longo alcance, também o sinal é muito resistente a interferências de sinais radioelétricos, devido a existir pouco ruído, porém não é adequado para aplicações que exijam baixa latência [AAA22].

Os dispositivos, comunicam com um ou vários *gateways* através de LoRa, enquanto todos os *gateways* são conectados ao servidor por IP (*Internet Protocol*) [SYH17].

A LoRaWAN utiliza a arquitetura em estrela de longo alcance em que os *gateways* são utilizados para retransmitir as mensagens entre os dispositivos e a unidade central de controlo [SYH17].

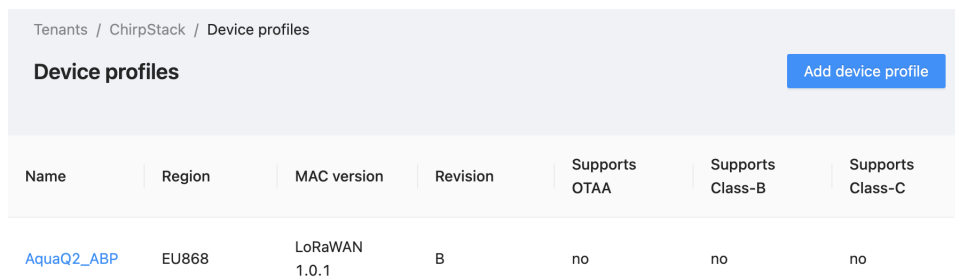
Numa rede LoRaWAN, os nós não estão associados a um *gateway* específico, pois os dados transmitidos por um nó normalmente são recebidos por vários *gateways*. Cada *gateway* encaminha o pacote recebido do nó final para o servidor [SYH17].

Existem varias vantagens e utilizar a LoraWAN, como o baixo custo, o baixo consumo de energia, permite também uma comunicação bidirecional e é uma tecnologia aberta,escalável e segura [Yaq+17]. Este é um protocolo sem fio desenvolvido pela *LoRa Alliance* para atender aos requisitos de comunicação de longo alcance das redes de sensores sem fios [Kha+19].

A LoRaWAN tem várias classes de dispositivos, para diferentes necessidades de aplicação, em que todos esses são bidirecionais [AAA22].

- Nos dispositivos de Classe A, os dispositivos fazem *Uplink* da transmissão seguida por duas janelas para receber *Downlink*;
- Nos dispositivos de Classe B, além das janelas de *Downlink* da Classe A, estes abrem janelas extras para receber em horários programados;
- Os dispositivos de Classe C, têm janelas de recepção abertas e fechadas durante a transmissão, porém consomem mais energia.

A classe LoRaWAN utilizada neste projeto é a Classe A, uma vez que, é a classe padrão e é suportada por qualquer dispositivo LoRaWAN, por isso o suporte às restantes classes permanecem desativadas como é possível verificar na figura 5.11.



Name	Region	MAC version	Revision	Supports OTAA	Supports Class-B	Supports Class-C
AquaQ2_ABP	EU868	LoRaWAN 1.0.1	B	no	no	no

Figura 5.11: Classes LoRaWAN na plataforma ChirpStack.

A tabela 5.1 apresenta a comparação de tecnologias de comunicação WAN para a IoT.

Tabela 5.1: Comparação de tecnologias de comunicação WAN (*em* [Mek+18]).

Tecnologia	Custo de utilização do espectro	Custo de desenvolvimento	Custo do dispositivo
Sigfox	Gratuito	Mais de 4000 EUR pela estação base	2 EUR
LoRaWAN	Gratuito	Mais de 1000 EUR pela estação base, Mais de 100 EUR por cada <i>gateway</i>	Entre 3 a 5 EUR
NB-IoT	Mais de 500000 EUR por MHz	Mais de 15000 EUR pela estação base	20 EUR

Ao optar pela utilização da tecnologia NB-IoT, o custo seria superior ao da LoRaWAN, visto que ao depender de uma operadora móvel, esta irá cobrar taxas mensais e será necessário a aquisição de cartões SIM para os dispositivos. Este consome mais energia do que a LoRaWAN, visto que o equipamento terá de estar sempre ativo para comunicar os dados [Mek+18].

A LoRaWAN é a solução mais barata em termos de custos energéticos, pois os dispositivos são criados para um consumo de energia muito inferior [Mek+18].

Ao utilizar LoRaWAN, descartam-se os custos financeiros para cobrir toda a cidade com *Access Points*, uma vez que é necessária uma ligação à Internet para utilizar MQTT sobre TCP/IP. Caso houvesse sinal Wi-Fi em toda a cidade esta seria a solução mais barata, porém as cidades que oferecem Wi-Fi não estão 100% cobertas e seria necessário instalar também infraestruturas [Mek+18].

A latência do NB-IoT é muito inferior em comparação com o LoRaWAN e o Wi-Fi e têm uma maior garantia das comunicações, visto não estar sujeito a interferências, pois depende de uma rede móvel licenciada [Mek+18].

A LoRaWAN não depende de serviços de cloud ao contrário do NB-IoT, pois a operadora móvel pode impor a utilização das suas *clouds* aumentando assim, o custo da manutenção da rede [Mek+18].

Em relação à segurança dos dados, o NB-IoT conta com os protocolos de segurança da operadora móvel, o que o torna bastante seguro, contudo a comunicação LoRaWAN é codificada, o que a torna também muito segura [Mek+18].

A figura 5.12 apresenta as diferenças entre as tecnologias WAN.

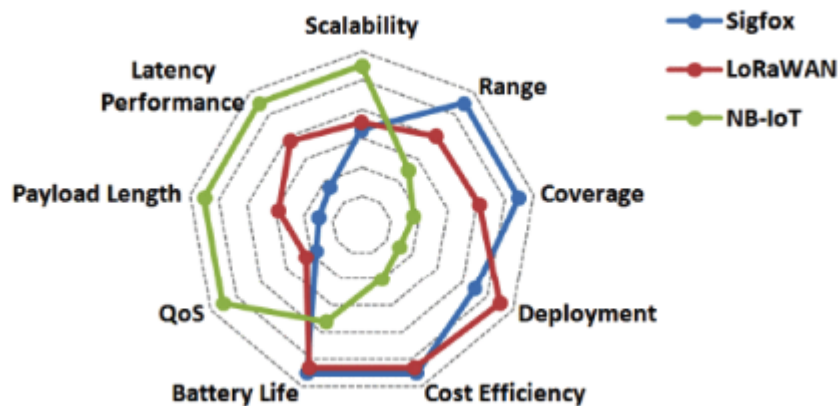


Figura 5.12: Diferenças entre tecnologias WAN [Mek+18].

O *gateway* LoRaWAN utilizado neste projeto, foi o MikroTik LtAP LR8 LTE6 kit, em que o seu sistema operativo é o RouterOS na versão 7.5 [Mik24a]. Este tem um alcance até 15 Km em ambiente rural e até 2 Km em ambiente urbano ao utilizar a antena MikroTik de 6,5 dBi [Mik24b]. A figura 5.13 mostra o *gateway* utilizado.



Figura 5.13: *Gateway* MikroTik LtAP LR8 LTE6 kit (em <https://www.mbit.pt/informatica/redes-comunicac-o/pontos-de-acesso-repetidores/mikrotik-router-iot-ltap-lr8-lte-kit-rbltap-2hnd-r11e-lte-lr>).

A figura 5.14 mostra a página LoRa do sistema operativo do *gateway* LoRaWAN, o RouterOS na versão 7.5.

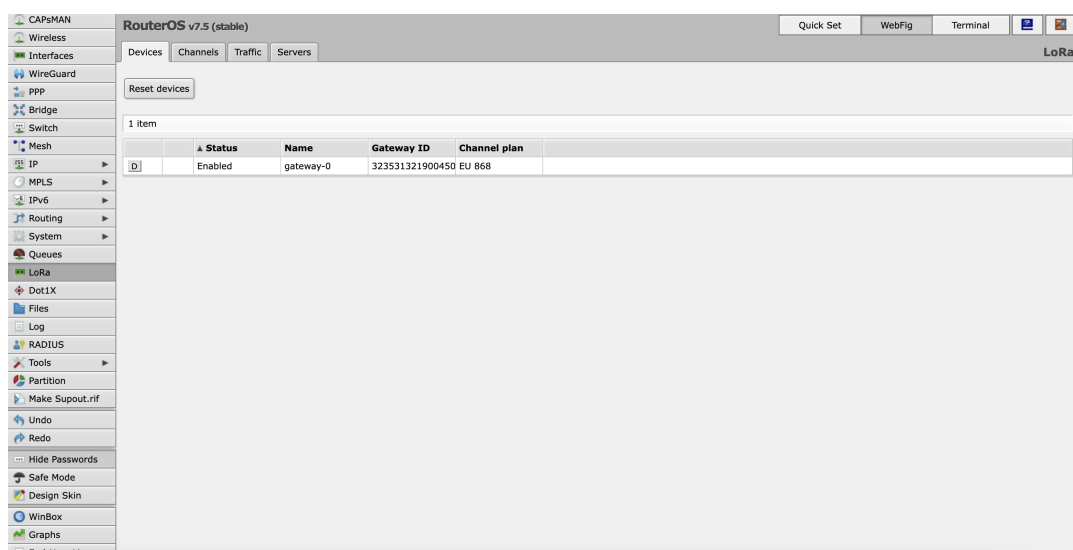


Figura 5.14: RouterOS 7.5.

5.7.3 Protocolo MQTT

O protocolo MQTT (*Message Queuing Telemetry Transport*) é um dos mais utilizados para a IoT, pois tem diversas vantagens, como a segurança, consome poucos recursos e é eficiente e confiável [MQT22].

O MQTT é leve e eficiente, pois os clientes MQTT, exigem recursos mínimos, por isso podem ser tão utilizados nos microcontroladores. O MQTT visa otimizar a largura de banda da rede [MQT22].

O MQTT ainda permite comunicações bidirecionais, o que facilita a transmissão de mensagens aos dispositivos. Este protocolo é escalável, por isso pode-se conectar a milhões de dispositivos IoT [MQT22].

O MQTT utiliza criptografia nas mensagens através do **OAuth**, o que o torna um protocolo seguro, se devidamente configurado [MQT22].

A comunicação entre o *gateway* LoRaWAN é feita através do protocolo MQTT, o *gateway* publica a mensagem num determinado tópico no MQTT **broker** e a API fica à escuta para adicionar os dados à base de dados e para enviar notificações e *emails*. Um *Broker* é um sistema intermediário e a sua principal função é gerir filas de mensagens.

No protocolo MQTT existe a qualidade de serviço (QoS), que é um acordo entre o remetente e o destinatário sobre a garantia de entrega de uma determinada mensagem [MQT22].

Existem três níveis de qualidade de serviço:

- QoS 0: No máximo uma vez;
- QoS 1: Pelo menos uma vez;
- QoS 2: Exatamente uma vez.

Este projeto utiliza QoS 1, neste caso existe a garantia que a mensagem chega ao *broker* pelo menos uma vez, isto é garantido, fazendo com que o destinatário envie um *PUBACK* (*Publication Acknowledgment*), confirmando que recebeu a mensagem. Se o remetente não receber um *PUBACK* dentro de um determinado período de tempo, ele enviará a mensagem novamente.

O cliente MQTT no lado do servidor escolhido foi o **Eclipse Paho MQTT** na linguagem de programação **Python** na versão 1.6.1. A unidade central de controlo irá subscrever o tópico onde o *gateway* LoRaWAN irá publicar os dados processados codificados em **Base64**, neste caso as leituras do caixote no formato **JSON**.

Quando uma leitura é publicada verifica-se se cumpre as condições para enviar alertas para que utilizadores responsáveis possam ser avisados. Os *emails* só são enviados de acordo com as preferências de cada utilizador, pois estes podem escolher se querem ou não receber *emails* na página de editar perfil.

5.7.4 Programação da unidade central de controlo

Na medida em que os dados processados pelo *gateway* LoRaWAN estão codificados em Base64.

A listagem 5.2 mostra o código utilizado para descodificar.

```

1  msg = json.loads(message.payload)
2
3  #----- Decode Data -----
4  convertbytes = msg['data']
5  convertedbytes = base64.b64decode(convertbytes) # Base64 Decode
6  decodedsample = convertedbytes.decode('utf-8')
7  print(decodedsample)
8
9  readings = json.loads(decodedsample)
10
11 print(readings['deviceID'])
12 print(readings['lat'])
13 print(readings['lon'])
14 print(readings['distance'])
15 print(readings['smoke'])
16 print(readings['battery'])

```

Listing 5.2: Descodificar os dados JSON em Base64.

Inicialmente, verifica-se se o ID do dispositivo está registado e está ativo e se esta condição for satisfeita é calculado o nível de enchimento, sabendo a altura do caixote, que foi introduzida no momento do registo do mesmo.

A figura 5.15 apresenta como o nível de enchimento de cada caixote é obtido, em que a distância sem resíduos é obtida pelo sensor e a distância total é a altura do caixote, assim é possível calcular a distância preenchida por resíduos.

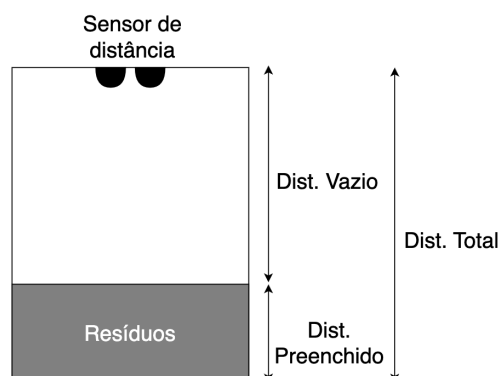


Figura 5.15: Ilustração de como é calculado o nível de enchimento.

Caso não seja possível ler as coordenadas nos 30 segundos de cada leitura é mantida a última localização lida.

O código da listagem 5.3 interpreta os valores no formato JSON e atualiza as leituras na base de dados.

```
1  deviceID = readings['deviceID']
2
3  device = Devices.query.filter_by(deviceID=deviceID,active=True).first()
4
5  # Check if the device exists
6  if device:
7
8      # Update
9      last_reading = Readings.query.filter_by(deviceID=deviceID).first()
10
11     # Calculate filling percentage
12     height = device.height
13     level = 100 - (readings['distance'] * 100 / height)
14     level = int(level)
15
16     smoke = readings['smoke']
17
18     if readings['lat'] != 0 or readings['lon'] != 0:
19         lat = readings['lat']
20         lon = readings['lon']
21     else:
22         lat = last_reading.lat
23         lon = last_reading.lon
24
25     battery = readings['battery']
26
27     # Update last reading
28     last_reading.level = level
29     last_reading.smoke = smoke
30     last_reading.lat = lat
31     last_reading.lon = lon
32     last_reading.battery = battery
33     last_reading.datetime = datetime.now(tz)
34
35     db.session.commit()
```

Listing 5.3: Lê e atualiza leituras na base de dados.

Neste projeto, também foram utilizados **WebSockets** para que na página das leituras fosse possível acompanhar a atualização dos dados sem ser necessário atualizar a página.

Na listagem 5.4 é mostrado como é emitido o evento **SocketIO** que permite aos utili-

zadores visualizar os dados das leituras em tempo real.

```

1  #-----SocketIO Emit-----
2
3  return_data = {
4      'deviceID': deviceID,
5      'lat': lat,
6      'lon': lon,
7      'level': level,
8      'smoke': smoke,
9      'battery': battery,
10     'datetime': datetime.now().strftime("%d/%m/%Y, %H:%M")
11 }
12
13 socketio.emit('readings', return_data)

```

Listing 5.4: Emitir evento SocketIO.

Todas as leituras são armazenadas num ficheiro CSV para análise futura, como é demonstrado no código da listagem 5.5.

```

1  #-----CSV File-----
2
3  # List that we want to add as a new row
4  List = [deviceID,level,smoke,lat,lon,battery,datetime.now()]
5
6  # Open our existing CSV file in append mode
7  # Create a file object for this file
8  with open('readings.csv', 'a') as f_object:
9      # Pass this file object to csv.writer() and get a writer object
10     writer_object = writer(f_object)
11     writer_object.writerow(List) # Pass the list as an argument into the.writerow()
12     f_object.close() # Close the file object

```

Listing 5.5: Adicionar os dados ao ficheiro CSV.

Na parte da alarmística, caso algum valor esteja fora do esperado são enviados alertas para os utilizadores responsáveis, com cargos de administrador e gestão. Esses alertas, podem ser enviados pela central de notificações do *website*, pela aplicação móvel do smartphone ou por *email*. Os alertas por *email* e pela aplicação móvel podem ser desativados.

A listagem 5.5 mostra o código que verifica se a bateria está com pouca carga, se a localização do caixote foi alterada e se o caixote detetou a presença de fumo e mostra

5. IMPLEMENTAÇÃO EXPERIMENTAL

como são enviados os alertas por *email*, para a central de notificações da plataforma *web* e para os dispositivos móveis.

```
1  #-----Low Battery-----
2
3  # Notify active Admins in website and App
4  if battery <= 10:
5      #print('Alert - low battery')
6
7      # Website
8      users = Users.query.filter_by(role=1, active=True)
9      for user in users:
10         notifier_id = user.id
11         level = 'Alerta'
12         message = 'O dispositivo ' + deviceID + ' está com bateria fraca - '
13         + str(battery) + '%'
14         is_read = False
15
16         new_notification = Notifications(notifier_id=notifier_id,level=level,
17         message=message,is_read=is_read)
18
19         db.session.add(new_notification)
20         db.session.commit()
21         pass
22
23     # Send Smartphone Notifications to active admins
24     tokens_list = []
25
26     all_admins = Users.query.filter_by(role=1,active=True)
27     tokens = DeviceTokens.query.all()
28
29     # Add all admins tokens to list
30     user_ids = set(admin.id for admin in all_admins)
31     filtered_tokens = filter(lambda token: token.user_id in user_ids, tokens)
32     tokens_list = list(map(lambda token: token.token, filtered_tokens))
33
34     send_token_push('e-Bin', deviceID + ' está com bateria fraca',tokens_list)
35     pass
36
37 #-----Location-----
38
39 # Notify active Admins in Website, App and Email
40 # Check if the current location is different from the previous one
41 # Use 3 decimal places to check if a significant zone change
42
43 if round(float(lat), 3) != round(float(last_reading.lat), 3) or
44    round(float(lon), 3) != round(float(last_reading.lon), 3):
```

```

45     #print('Alert - location changed')
46
47     # Website
48     users = Users.query.filter_by(role=1, active=True)
49     for user in users:
50         notifier_id = user.id
51         level = 'Alerta'
52         message = 'A localização de ' + deviceID + ' foi alterada (' +
53             lat + ', ' + lon + ' )'
54         is_read = False
55
56         new_notification = Notifications(notifier_id=notifier_id,level=level,
57             message=message,is_read=is_read)
58
59         db.session.add(new_notification)
60         db.session.commit()
61         pass
62
63     # Send Smartphone Notifications to active admins
64     tokens_list = []
65
66     all_admins = Users.query.filter_by(role=1,active=True)
67     tokens = DeviceTokens.query.all()
68
69     # Add all admins tokens to list
70     user_ids = {admin.id for admin in all_admins}
71     tokens_list = [token.token for token in tokens if token.user_id in user_ids]
72
73     send_token_push('e-Bin', 'A localização de ' + deviceID + ' foi alterada',
74         tokens_list)
75
76     # Email
77     users = Users.query.filter_by(role=1, active=True, receive_emails=True)
78
79     with app.app_context():
80         with mail.connect() as conn:
81             for user in users:
82                 msg = Message('Alerta - e-Bin', sender = 'info@e-bin.pt',
83                     recipients = [user.email])
84                 msg.body = 'Caro/a {},\n\nA localização do dispositivo {} foi
85                     'alterada para ({} , {}) .\n\nAtenciosamente,\ne-Bin'.format(
86                     user.name,deviceID,str(lat),str(lon))
87                 mail.send(msg)
88             pass
89     pass
90
91     #-----Smoke detected-----

```

```
92
93 # Notify active Admins in Website, App and Email
94 if smoke == 1:
95     #print('Alert - smoke detected')
96
97     # Website
98     users = Users.query.filter_by(role=1, active=True)
99     for user in users:
100         notifier_id = user.id
101         level = 'Alerta'
102         message = 'Detetado fumo no dispositivo ' + deviceID
103         is_read = False
104
105         new_notification = Notifications(notifier_id=notifier_id,level=level,
106         message=message,is_read=is_read)
107
108         db.session.add(new_notification)
109         db.session.commit()
110     pass
111
112 # Send Smartphone Notifications to active admins
113 tokens_list = []
114
115 all_admins = Users.query.filter_by(role=1,active=True)
116 tokens = DeviceTokens.query.all()
117
118 # Add all admins tokens to list
119 user_ids = set(admin.id for admin in all_admins)
120 filtered_tokens = filter(lambda token: token.user_id in user_ids, tokens)
121 tokens_list = list(map(lambda token: token.token, filtered_tokens))
122
123 send_token_push('e-Bin','Detetado fumo em ' + deviceID ,tokens_list)
124
125 # Emails
126 users = Users.query.filter_by(role=1, active=True, receive_emails=True)
127
128 with app.app_context():
129     with mail.connect() as conn:
130         for user in users:
131             msg = Message('Alerta - e-Bin', sender = 'info@e-bin.pt',
132             recipients = [user.email])
133             msg.body = 'Caro/a {},\n\nFoi detetada a presença de fumo no
134             'dispositivo {} .\n\nAtenciosamente,\ne-Bin'.format(user.name,
135             deviceID)
136             mail.send(msg)
137         pass
138     pass
```

Listing 5.6: Verificar se os valores estão fora dos parâmetros normais e notifica os utilizadores.

A figura 5.16 mostra um *email* enviado automaticamente para um utilizador com o cargo de administrador ou gestão deste sistema, quando for detetado fumo no caixote com o dispositivo ESP32-1. Os *emails* só são enviados de acordo com as preferências do utilizador, este deve ter ativo a opção de receber *emails* na página de editar perfil caso pretenda receber os *emails*.

Alerta - e-Bin



De info@e-bin.pt em 2024-08-30 12:09



[Detalhes](#)



[Cabeçalhos](#)

Caro/a Rafael Montez,

Foi detetada a presença de fumo no dispositivo ESP32-1 .

Atenciosamente,
e-Bin

Figura 5.16: Email de alerta de deteção de fumo.

A figura 5.17 mostra o mesmo alerta mas exibido num dispositivo móvel.

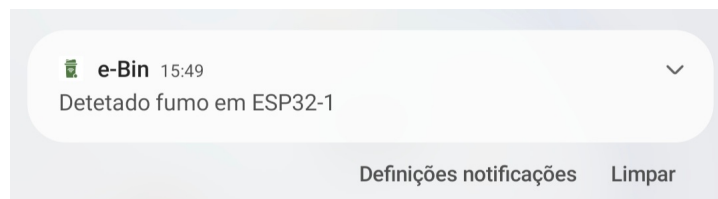


Figura 5.17: Notificação de alerta de deteção de fumo na aplicação móvel.

Caso a central de notificações não seja limpa a contagem das notificações não lidas no ícone da aplicação irá aumentar, como é possível observar na figura 5.18.

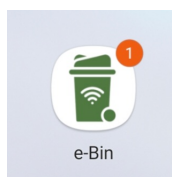


Figura 5.18: Contagem de Notificações da aplicação móvel.

5. IMPLEMENTAÇÃO EXPERIMENTAL

As notificações são enviadas para os dispositivos móveis, através da API deste projeto que comunica com o **Firestore Cloud Messaging**, que é responsável por enviar o alerta, na figura 5.19 é apresentada a interface do **Firestore Cloud Messaging**.

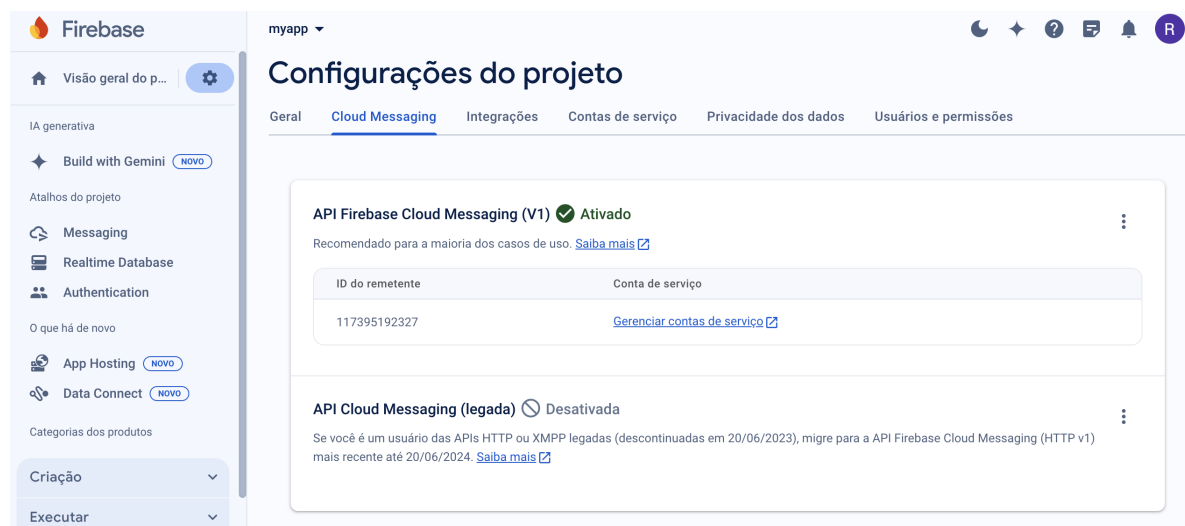


Figura 5.19: Firestore Cloud Messaging.

5.8 Sistema de gestão de base de dados

O sistema de gestão de base de dados relacional escolhido para este projeto foi o **MySQL**, pois é rápido, é leve, é mantido pela **Oracle**, utiliza a linguagem padronizada **SQL**, é *open-source* e estas características tornam-no o mais utilizado em todo o mundo [Ora24].

A base de dados deste projeto conta com 8 tabelas baseadas no diagrama ER na Secção 3.5: **users**, **devices**, **readings**, **vehicles**, **routes**, **drivingData**, **notifications** e **deviceTokens**.

A tabela **users** contém os dados dos utilizadores. A tabela **devices** contém os dados dos caixotes, como a sua capacidade, entre outros. A tabela **readings** contém as leituras dos caixotes. A tabela **vehicles** contém os dados dos camiões, como a capacidade e a matrícula. A tabela **routes** contém todas as rotas geradas. A tabela **drivingData** contém dados da rota realizada pelo motorista como as coordenadas e a velocidade em Km/h. A tabela **notifications** contém as notificações do sistema de cada utilizador e a tabela **deviceTokens** contém os *tokens* de cada dispositivo móvel de cada utilizador para que sejam enviadas as notificações automaticamente.

Estima-se que a quantidade de dados da tabela responsável por armazenar as rotas seja aproximadamente 2 KB por registo e as restantes tabelas 1 KB por registo. Por exemplo, 2 milhões de rotas são aproximadamente 4 GB ocupados e 1 milhão de leituras são aproximadamente 1 GB ocupado.

5.9 Segurança Informática

Na elaboração de um projeto existem vários aspetos a ter em conta, nomeadamente, a segurança informática.

Neste projeto, as *passwords* dos utilizadores são encriptadas com **bcrypt**, uma vez que é mais seguro que outros algoritmos de encriptação como o MD5 ou da família SHA. Uma máquina moderna com muito poder de processamento de CPU e GPU pode calcular milhões ou bilhões de *hashes* SHA-256 por segundo, como por exemplo, *passwords* de uma base de dados comprometida [Ari21].

Ao existir máquinas que consigam efetuar cálculos mais rápidos, significa que os ataques de *brute force* também seriam mais rápidos e mais eficientes. De modo a prevenir isto, é necessária uma função de *hash* mais lenta para tentar combater os ataques dos invasores [Ari21].

O **bcrypt** é baseado na cifra Blowfish e requer o uso de uma *salt* que é um valor adicional aleatório utilizado no *hashing* [Ari21].

O maior benefício do **bcrypt** é que com o passar do tempo a contagem de iterações pode ser aumentada para tornar mais lenta a decifração, o que permite o **bcrypt** escalar com o poder de computação, diminuindo o benefício da utilização de *hardware* mais rápido, aumentando o número de iterações para tornar o **bcrypt** mais lento [Ari21].

Foi utilizado o **flask-login** e **flask-admin**, para aumentar a segurança da autenticação, protegendo os *endpoints* dos utilizadores não autorizados, através de sessões.

Os utilizadores veem a sua sessão terminada após 2 horas do início da sessão, para que haja mais segurança, obrigando o utilizador a efetuar o *login* novamente.

Dentro da plataforma *web*, caso um funcionário tente aceder a uma página que não tenha permissão, através do URL, este é redirecionado para a página de *login*.

Na aplicação móvel os *endpoints*, estão protegidos através de *tokens* JWT, como é possível observar no código da listagem 5.7.

```
1  try:
2      token_passed = request.headers['TOKEN']
3      if request.headers['TOKEN'] != '' and request.headers['TOKEN'] != None:
4          try:
5              data = jwt.decode(token_passed, app.config['SECRET_KEY'],
6                               algorithms=['HS256'])
7              return something()
8          except jwt.exceptions.ExpiredSignatureError:
9              return_data = {
10                 "error": "1",
11                 "message": "Token has expired"
12             }
13             return app.response_class(response=json.dumps(return_data),
14                                     mimetype='application/json'), 401
```

```
15         except:
16             return_data = {
17                 "error": "1",
18                 "message": "Invalid Token"
19             }
20             return app.response_class(response=json.dumps(return_data),
21                                     mimetype='application/json'),401
22         else:
23             return_data = {
24                 "error" : "2",
25                 "message" : "Token required",
26             }
27             return app.response_class(response=json.dumps(return_data),
28                                     mimetype='application/json'),401
29     except Exception as e:
30         return_data = {
31             "error" : "3",
32             "message" : "An error ocurred"
33         }
34         return app.response_class(response=json.dumps(return_data),
35                                 mimetype='application/json'),500
```

Listing 5.7: Código para proteger os *endpoints* da aplicação móvel com *tokens* JWT em python.

No código da listagem 5.8, é possível observar como se protege um *endpoint* utilizando o método definido acima, isto faz com que o cliente ao fazer a solicitação tenha de enviar um *token* JWT válido.

```
1  # List Users
2  @app.route('/app/users', methods=['GET'])
3  @token_required # <-----
4  def users_app():
5      all_users = Users.query.all()
6      results = users_schema.dump(all_users)
7
8      return jsonify(results)
```

Listing 5.8: Código para proteger os *endpoints* da aplicação móvel com *tokens* JWT em python.

No código da listagem 5.9, é mostrado como é feita uma solicitação protegida pelo *token* JWT, na aplicação móvel. Este, é gerado no momento do *login* e fica armazenado na memória do dispositivo móvel até ao *logout*.

```

1  var url = "https://e-bin.pt/app/users";
2  var response = await http.get(Uri.parse(url),
3    headers: {
4      // Send authorization headers to the backend
5      "TOKEN": "$jwt",
6    },
7  );
8
9  return json.decode(response.body);

```

Listing 5.9: Código para proteger os *endpoints* da aplicação móvel com *tokens* JWT em flutter.

5.10 Validações de formulários

De modo a garantir que os dados são recebidos conforme esperado existe uma validação de formulários do lado do cliente. A validação é feita com JavaScript utilizando expressões regulares e verificação do comprimento após cada clique numa tecla através do evento `onKeyUp`. Durante a validação dos formulários as classes CSS também são alteradas.

A figura 5.20 mostra um exemplo de validação de formulários na plataforma *web*.

The screenshot shows the 'e-Bin' web application interface. On the left is a green sidebar with navigation links: Dashboard, Leituras, Rotas (selected), Veículos, Dispositivos, and Utilizadores. The top bar includes a search bar, a notification bell, and the user profile 'Rafael Montez | Administrador'. The main content area is titled 'Editar Rota'. It contains a form with the following fields: 'ID Funcionário*' (text input with value 'funcX' and a red error message 'O ID do funcionário não é válido'), 'Matrícula*' (text input with value 'AA-00-AA'), 'Data*' (date picker with value '12/04/2024'), 'Hora*' (time picker with value '19:00'), and 'Estado*' (dropdown menu with value 'Concluída'). At the bottom of the form are two buttons: 'Atualizar' (green) and 'Voltar' (grey).

Figura 5.20: Exemplo de validação de formulários na plataforma *web*.

Os campos de introdução dos dados em que a *label* contém um asterisco (*) são obrigatórios e essa validação de formulários não permite submeter o formulário sem que estes estejam preenchidos.

5. IMPLEMENTAÇÃO EXPERIMENTAL

A figura 5.21 mostra a validação de formulários na aplicação móvel.

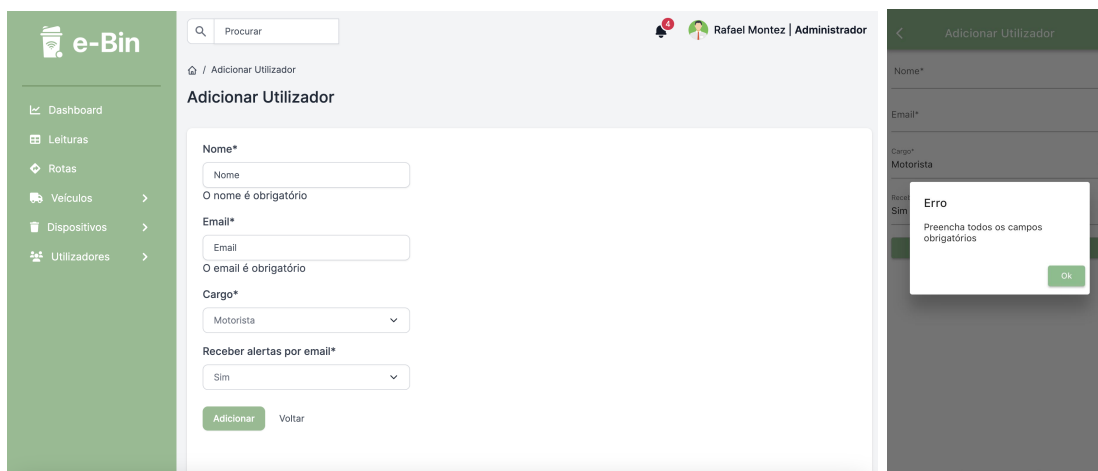


Figura 5.21: Exemplo de validação de formulários.

Neste projeto para que exista uma segurança mínima, todas as *passwords* têm que ter pelo menos uma letra minúscula, uma letra maiúscula, um número e não pode ter menos que 8 caracteres. Não é obrigatório introduzir caracteres especiais, uma vez que é previsto que este sistema funcione internamente, ligado a uma *firewall* e dentro de uma *vlan*.

Existe também uma validação de formulários do lado do servidor, para garantir que o mesmo utilizador não se regista 2 vezes, por exemplo, tanto na plataforma *web* como na aplicação móvel.

Na figura 5.22 é possível observar um exemplo de uma validação de formulários na plataforma *web*.

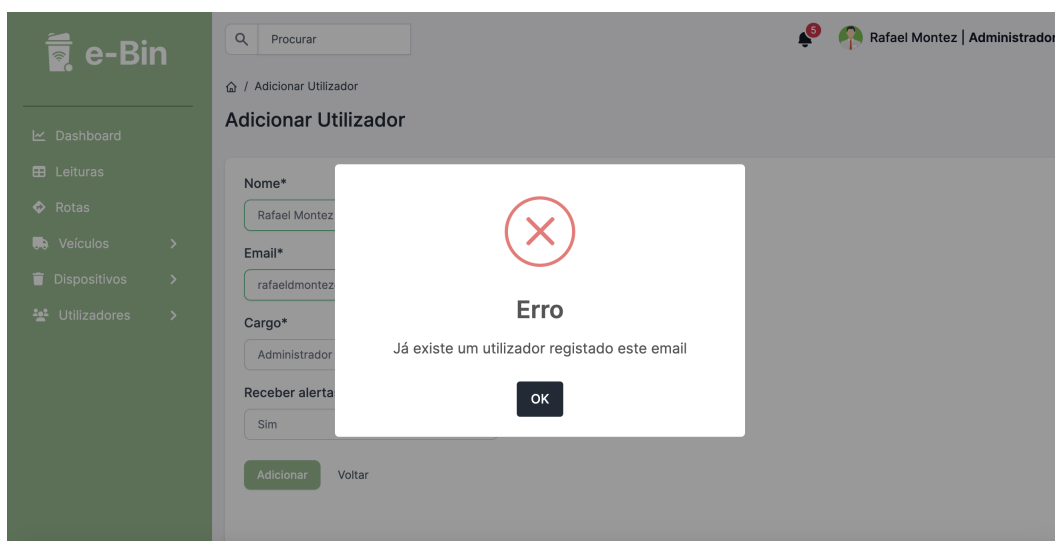


Figura 5.22: Exemplo de validação de formulários na plataforma *web*.

5.11 Envio de *emails*

Atualmente, o envio de *emails* é fundamental para qualquer plataforma *web*, por isso foi decidido implementar este sistema neste projeto.

Os *emails* são enviados através do **Flask-Mail**, este permite a utilização do servidor **SMTP** configurado para o envio de *emails*.

Os utilizadores poderão escolher se querem receber os alertas por *email*, através da página de editar perfil. Esses *emails* são relacionados aos caixotes ou às rotas.

Os *emails* também são recebidos por exemplo quando um caixote deteta a presença de fumo.

No momento do registo de cada utilizador é enviado um *email* de boas-vindas com a uma *password*, que o utilizador deve trocar o mais rapidamente possível.

A figura 5.23 o *email* recebido após o registo de um novo utilizador.

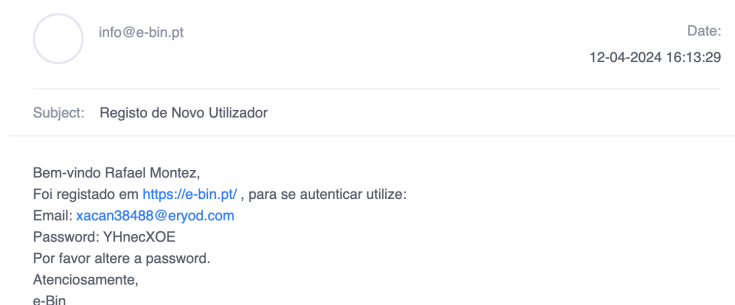


Figura 5.23: Email de registo de novo utilizador.

5.12 Análise dos resultados

5.12.1 Dispositivo e-Bin

O dispositivo **e-Bin** incorpora uma placa de circuito impresso, que conta com algumas alterações em relação ao circuito na *breadboard*. Os sensores e o microcontrolador estão conectados à placa de circuito impresso através de *headers*, que permitem retirar os componentes sem a necessidade de remover a solda e soldar novamente. A figura 5.24 mostra um *header* semelhante ao utilizado neste protótipo.



Figura 5.24: *Headers* para placa de circuito impresso (em <https://www.rapidonline.com/oupiin-2141-1-04g00sb-4w-single-row-2mm-pcb-header-socket-22-3232>).

5. IMPLEMENTAÇÃO EXPERIMENTAL

A figura 5.25 mostra o circuito na placa de circuito impresso colocada numa caixa de plástico, onde fornece uma melhor proteção contra a entrada de pó e resíduos para este protótipo.

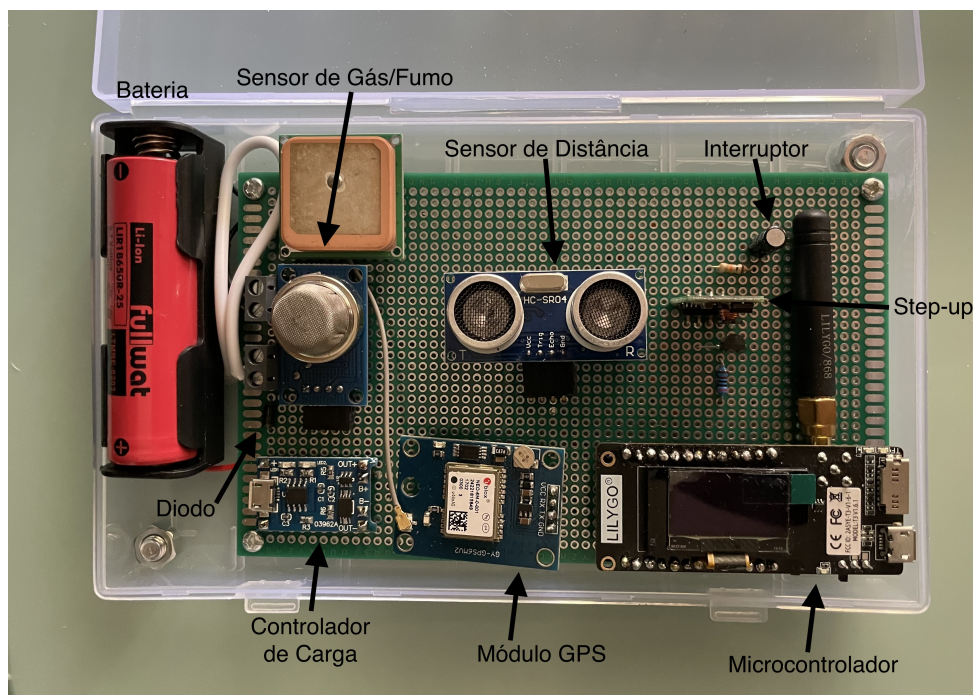


Figura 5.25: Circuito do projeto numa caixa.

Na placa de circuito impresso o botão de pressão foi substituído por um interruptor de inclinação e um módulo TP4056 com proteção, para carregar a bateria por *micro usb* e através do painel solar de 3W e 5V. Também foi introduzido um diodo retificador 1N4007 para o terminal positivo do painel solar. Isso garante que a corrente do painel solar só possa mover-se para fora e não retornar para o painel.

As leituras são enviadas de hora a hora ou cada vez que a tampa é aberta. Na figura 5.26 é possível observar o dispositivo e-Bin colocado num caixote do lixo.



Figura 5.26: Caixote do lixo com dispositivo e-Bin.

5.12.2 Aplicação móvel

Para que os motoristas possam visualizar e seguir a rota gerada pelo sistema, foi desenvolvida uma aplicação móvel em **Flutter** que conta com várias funcionalidades para os diferentes cargos. O logótipo utilizado na plataforma *web* foi simplificado para que pudesse ser visualizado num ecrã inicial de um dispositivo móvel **iOS** ou **Android**. A figura 5.27 mostra o ícone da aplicação móvel **e-Bin**.

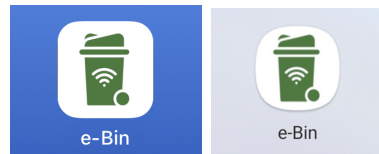


Figura 5.27: Ícone da aplicação no ecrã inicial no **iOS** e **Android** respetivamente.

A aplicação móvel **e-Bin** no **Android** ocupa mais espaço do que no **iOS**, apesar de ser a mesma. A figura 5.28 mostra o ícone da aplicação móvel **e-Bin**.

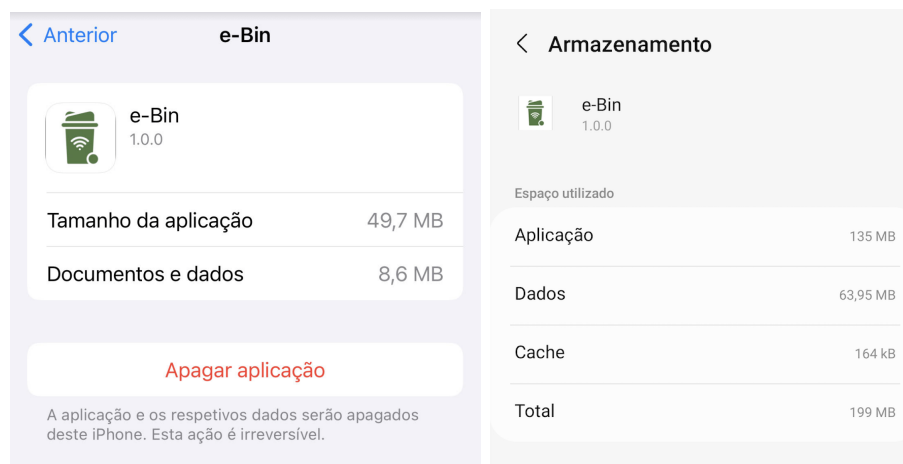


Figura 5.28: Espaço ocupado pela aplicação **e-Bin** no **iOS** e no **Android** respetivamente.

5. IMPLEMENTAÇÃO EXPERIMENTAL

As figura 5.29 mostra o *splash screen* utilizado na aplicação móvel, ou seja, na abertura da aplicação mostra o logótipo com o fundo de acordo com o tema escolhido, claro ou escuro. A última figura mostra o alerta que é recebido ao iniciar a aplicação pela primeira vez, esta pede para confirmar se pretende receber notificações, após a confirmação é pedido o acesso à localização do utilizador, o que é necessário para que a aplicação funcione corretamente.

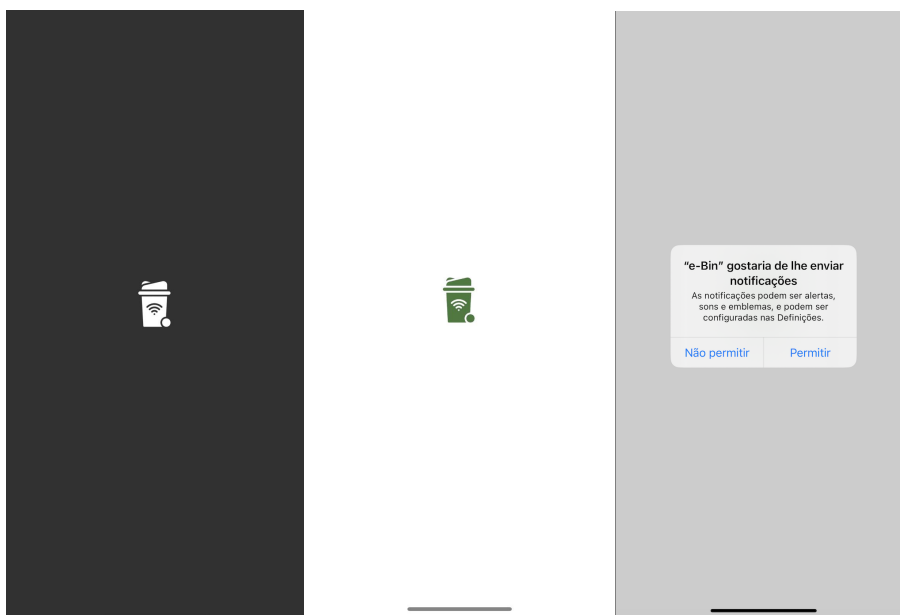


Figura 5.29: Capturas de ecrã da aplicação móvel no iOS.

Nas figuras 5.30 é possível observar a página de *login* e reposição de *password*.

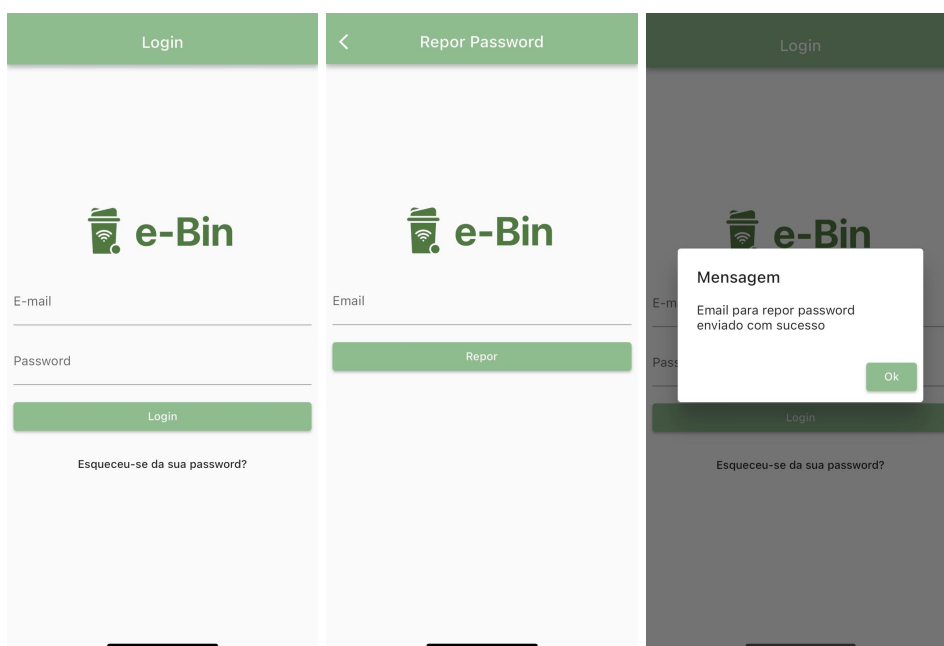


Figura 5.30: Capturas de ecrã da aplicação móvel no iOS.

Na reposição de *password* será enviado um *email* com o link para nova *password*, esse link contém um token gerado para que só o próprio utilizador consiga repô-la, como é possível observar na figura 5.31.

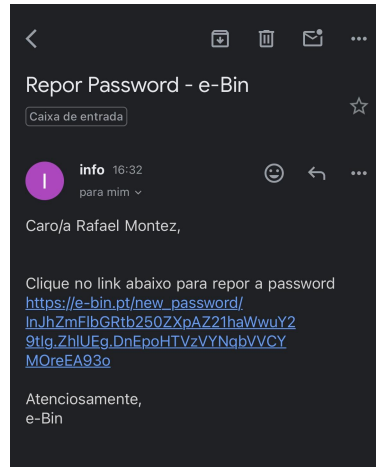


Figura 5.31: Email para reposição de *password*.

Se o link recebido no *email* for aberto num dispositivo móvel com a aplicação instalada, é perguntado se pretende abrir pela aplicação ou através da página *web*. Esta funcionalidade foi implementada graças à biblioteca `Uni Links` do `Flutter`. A figura 5.32 mostra o funcionamento dessa característica.

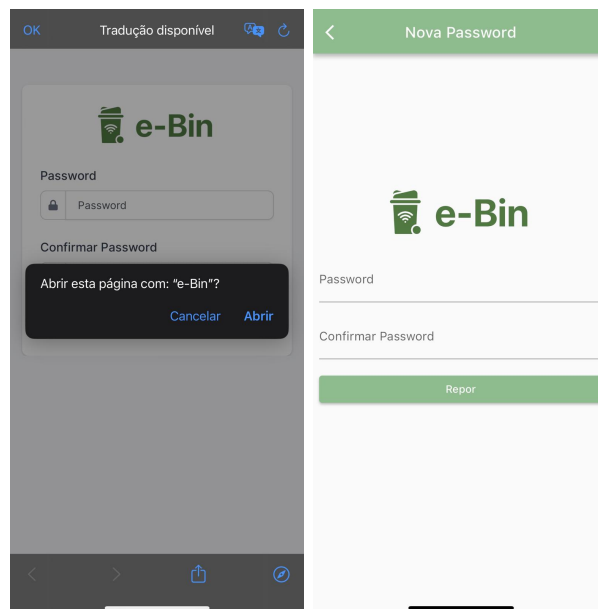


Figura 5.32: Capturas de ecrã da aplicação móvel no iOS.

A aplicação móvel guarda dados localmente, como a aplicação GPS escolhida e se o utilizador já efetuou o *login*, para que até ao momento do *logout* não seja necessário repetir o *login*. Os dados são guardados através da biblioteca `Shared Preferences` do `Flutter`.

Cargo de Motorista

Após um *login* bem-sucedido, os motoristas podem verificar a localização e o nível de enchimento de todos os caixotes. Esta página é alterada quando uma rota está selecionada, mostrando assim o percurso no mapa para a rota selecionada, como é possível verificar na figura 5.33.

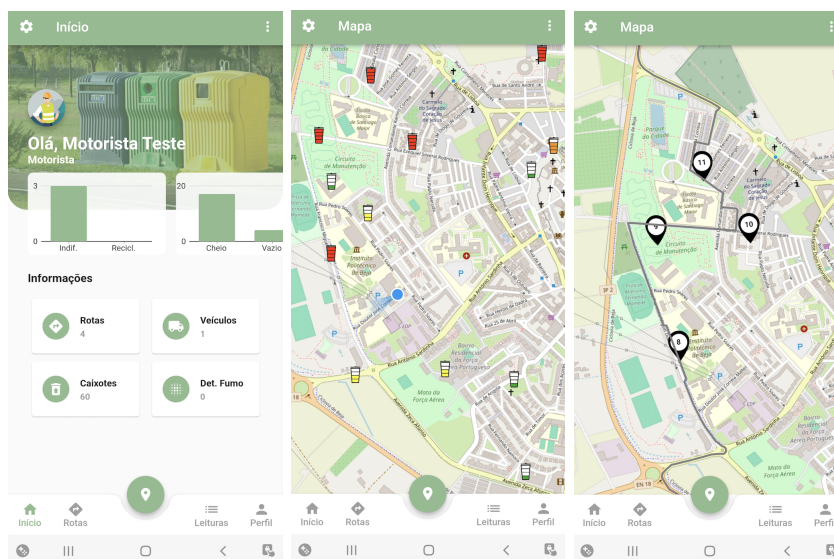


Figura 5.33: Capturas de ecrã da aplicação móvel no Android.

A figura 5.34, mostra a página de visualizar rotas, onde é possível iniciar as rotas e ver as respetivas paragens através da aplicação móvel.

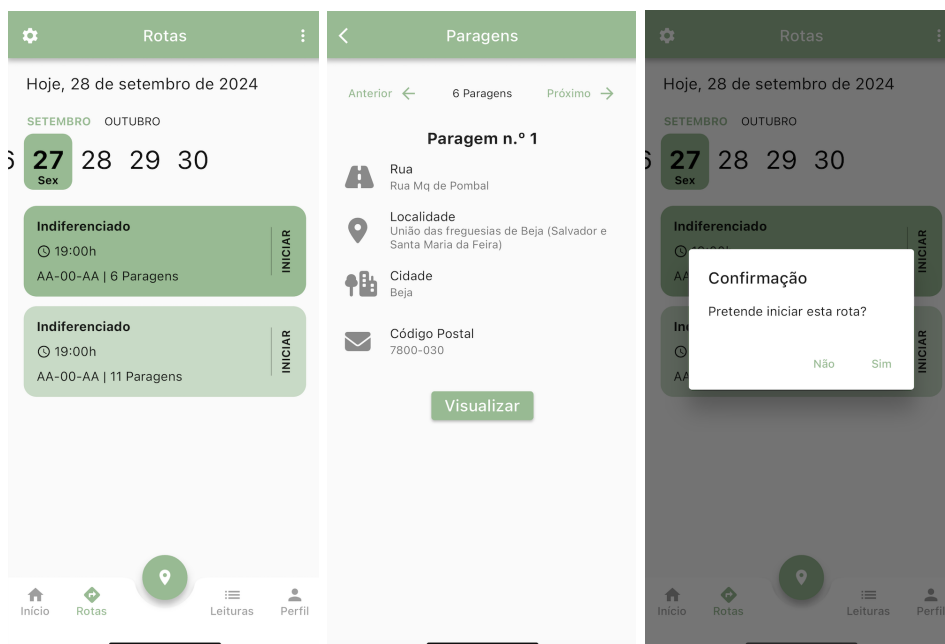


Figura 5.34: Capturas de ecrã da aplicação móvel no iOS.

Ao iniciar uma rota, o dispositivo móvel irá abrir a aplicação de GPS selecionada pelo utilizador. Em segundo plano enviará através de **WebSockets**, dados do utilizador, dados da rota, a localização, a velocidade em Km/h e o ângulo para onde se move. Essa informação é recebida pelos utilizadores com cargo de administrador ou gestão que estejam a utilizar a aplicação móvel ou a plataforma *web*, obtendo assim um maior controlo sobre a frota.

O tema da aplicação pode ser alterado para claro, escuro ou automático, consoante as preferências do utilizador, como mostra a figura 5.35.

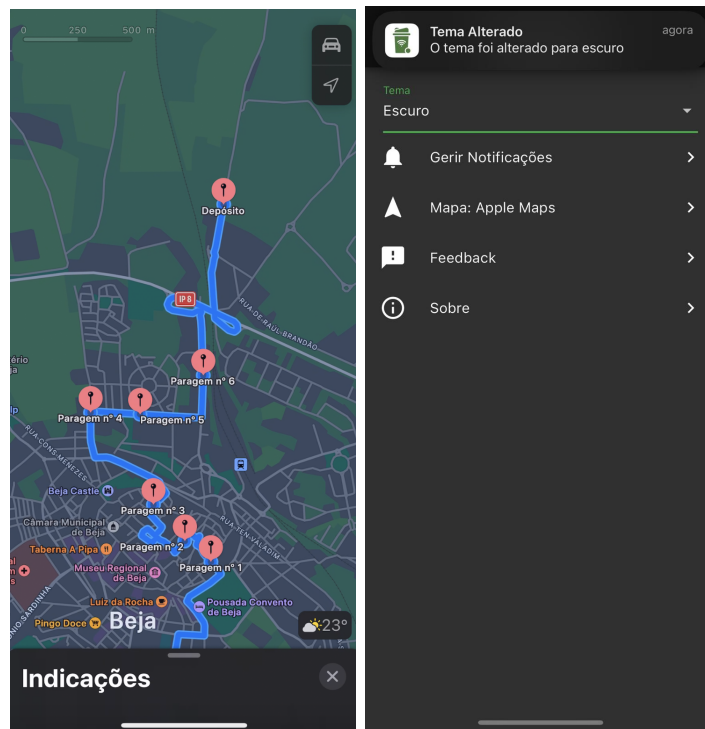


Figura 5.35: Capturas de ecrã da aplicação móvel no iOS.

É possível alterar a aplicação de navegação pré-definida, uma vez que a biblioteca **map launcher** do Flutter tem compatibilidade com diversas aplicações, como: **Google Maps**, **Waze**, **Tomtomgo**, **Sygyic Truck**, entre outras. Ao trocar de aplicação irá ser mostrada uma notificação e começará a utilizar a aplicação escolhida.

Caso alguma aplicação de navegação externa seja desinstalada, o sistema vai verificar que esta já não existe e pede para que o utilizador selecione novamente uma aplicação de navegação. Na página de configurações da aplicação é possível enviar *feedback*, ver mais sobre a aplicação, aceder ao menu de ajuda e configurar as notificações.

5. IMPLEMENTAÇÃO EXPERIMENTAL

Na figura 5.36 é possível verificar a página de configurações da aplicação móvel.

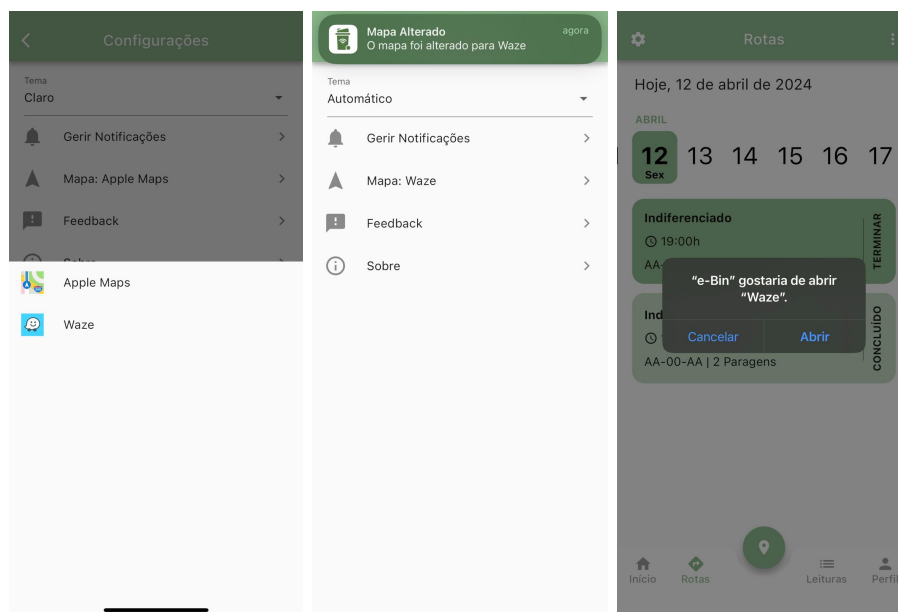


Figura 5.36: Capturas de ecrã da aplicação móvel no iOS.

Cargo de administrador ou gestão

Com uma autenticação feita por um utilizador com cargo de administrador ou gestão, é possível aceder a mais funcionalidades, que com o cargo de motorista não era possível. Na figura 5.37 é possível observar as páginas da gestão de utilizadores.

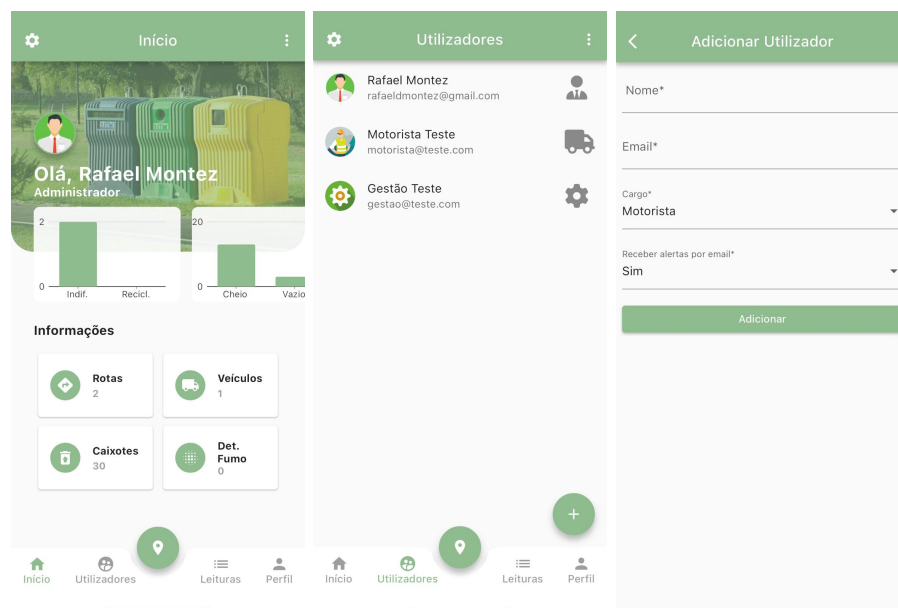


Figura 5.37: Capturas de ecrã da aplicação móvel no iOS.

A figura 5.38 mostra as páginas de apagar e editar utilizadores. Nesta, também é possível desativar os utilizadores, para que não possam autenticar-se novamente no sistema.

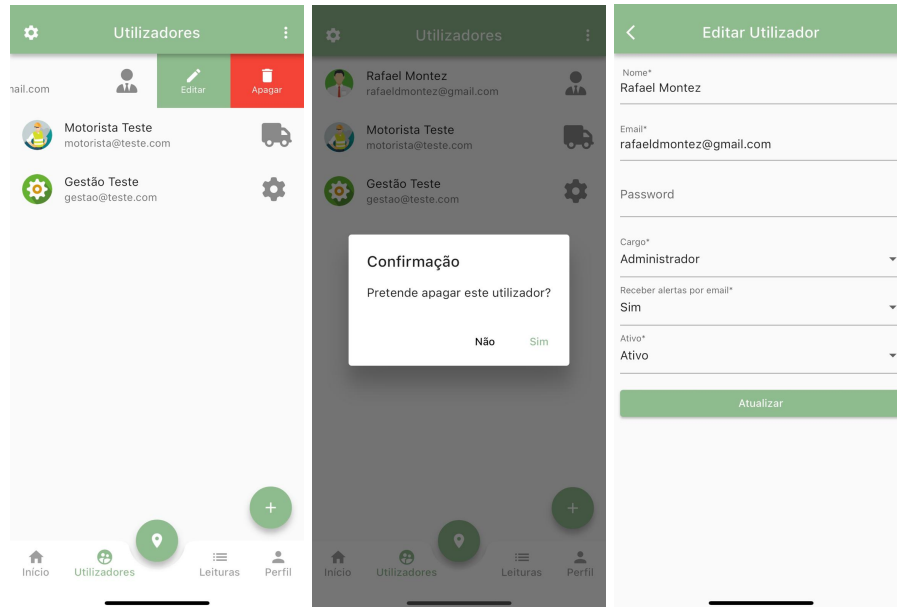


Figura 5.38: Capturas de ecrã da aplicação móvel no iOS.

A figura 5.39 mostra o mapa, onde é possível observar todos os caixotes, o seu nível de enchimento e verificar a localização de todos os motoristas que estejam a percorrer uma rota. A aplicação também conta com uma página de ajuda.

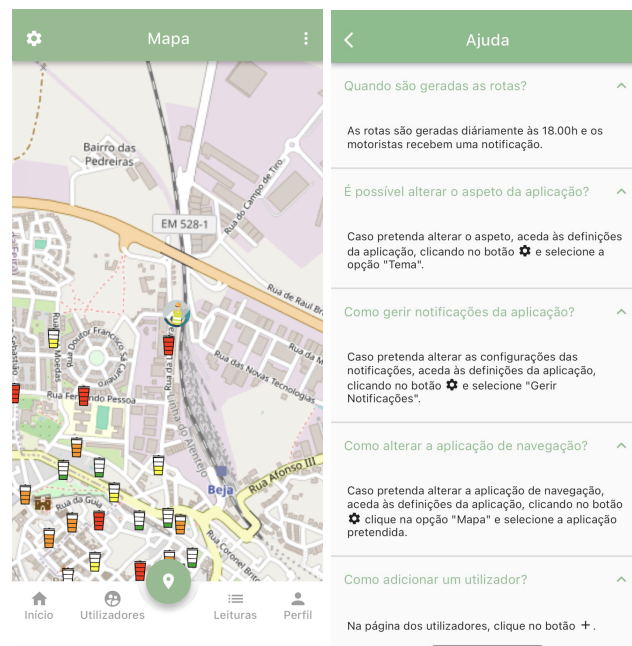


Figura 5.39: Capturas de ecrã da aplicação móvel no iOS.

5. IMPLEMENTAÇÃO EXPERIMENTAL

A figura 5.40 mostra a página Sobre, que permite obter mais informações relacionadas à aplicação e a página de leituras, que mostra dados relativos ao dispositivo instalado no caixote. Ambas são comuns a todos os cargos.

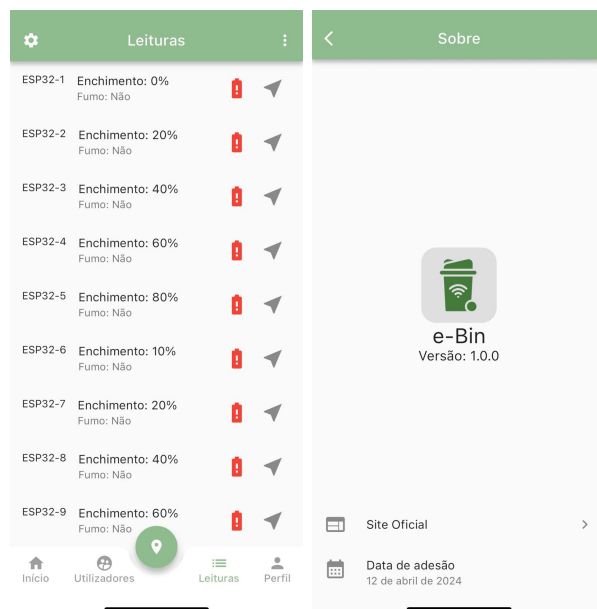


Figura 5.40: Capturas de ecrã da aplicação móvel no iOS.

A figura 5.41 mostra a página de editar perfil, esta permite alterar dados pessoais do utilizador e escolher se pretende ou não receber *emails* informativos.

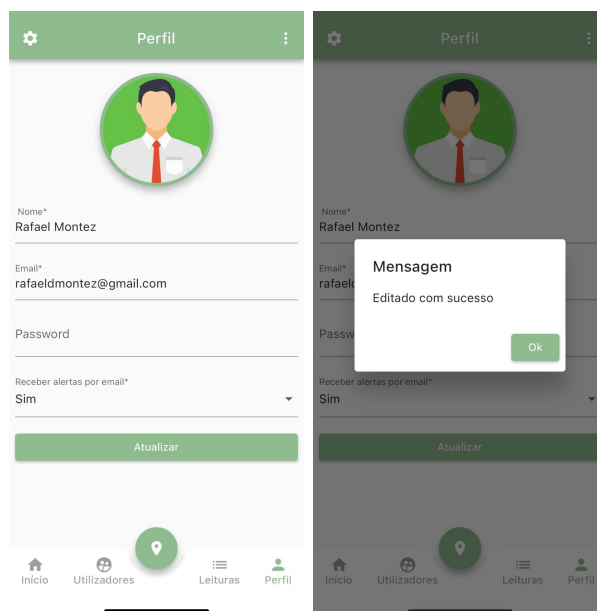


Figura 5.41: Capturas de ecrã da aplicação móvel no iOS.

Ao clicar em cima da foto de perfil atual, é aberta a galeria onde a aplicação só tem acesso à foto escolhida, mantendo a privacidade dos utilizadores. A figura 5.42 mostra a página onde é possível escolher uma nova foto de perfil.

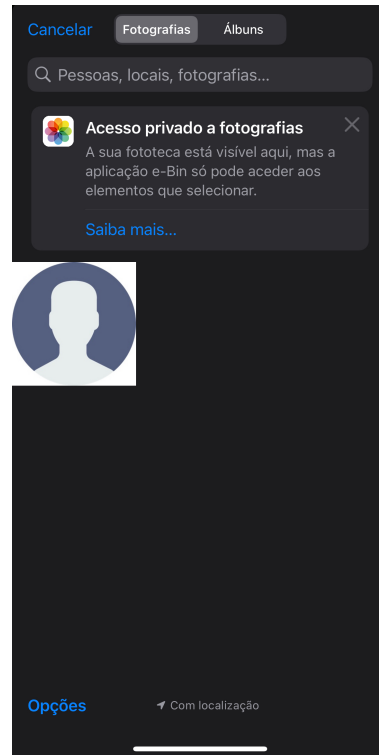


Figura 5.42: Escolher nova foto de perfil no iOS.

5.12.3 Plataforma *web*

A plataforma *web* tem vários elementos comuns à aplicação móvel, como a autenticação, que é feita com o mesmo *email* e *password* que é utilizado na aplicação móvel.

A figura 5.43 mostra a página de *login*.

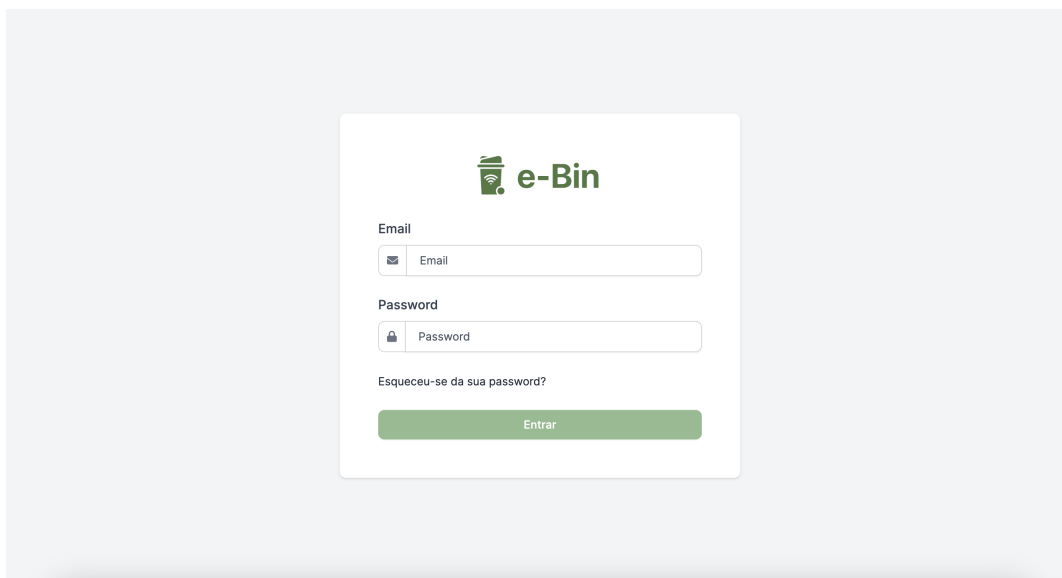


Figura 5.43: Captura de ecrã da página de *login* da plataforma *web*.

A figura 5.44 mostra a página de reposição de *password*.

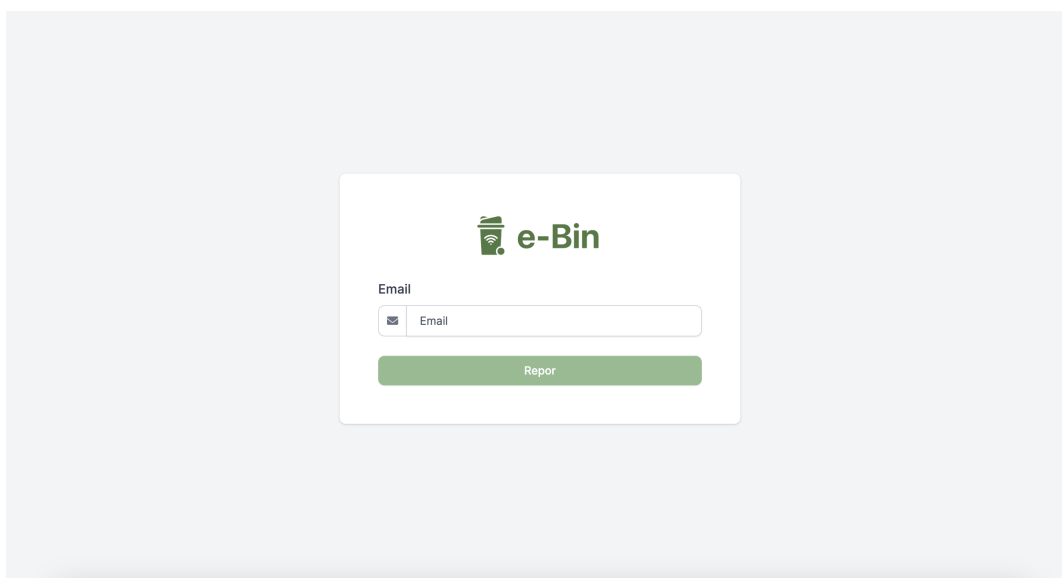
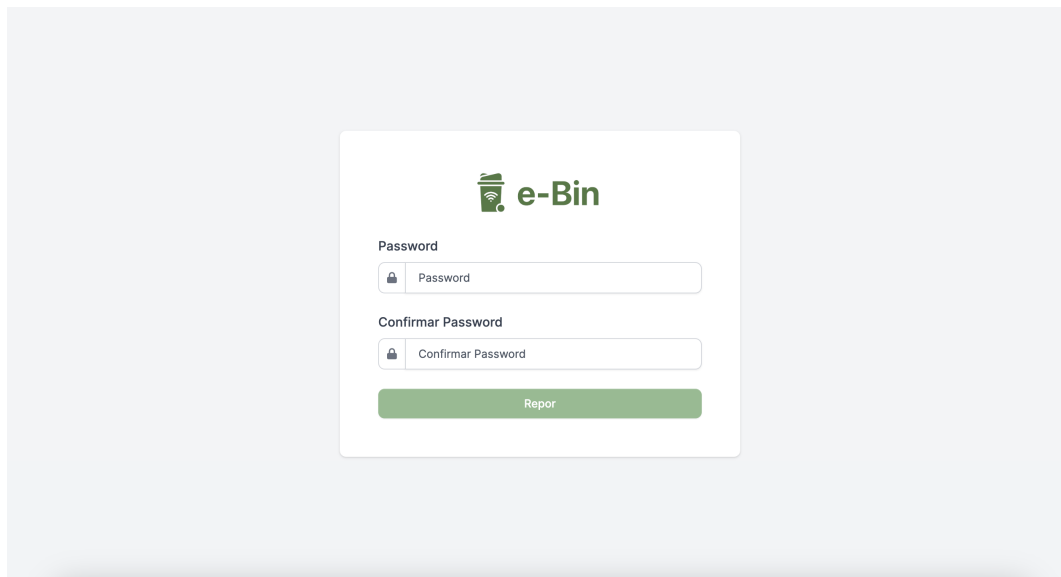


Figura 5.44: Captura de ecrã da página de reposição de *password* da plataforma *web*.

A figura 5.43 mostra a página de inserir uma nova *password*.



A screenshot shows a web form for creating a new password. At the top is the 'e-Bin' logo, which consists of a green trash can icon with a Wi-Fi symbol and the text 'e-Bin'. Below the logo, there are two input fields. The first is labeled 'Password' and has a small lock icon to its left. The second is labeled 'Confirmar Password' and also has a small lock icon to its left. Below these fields is a green button labeled 'Repor'.

Figura 5.45: Captura de ecrã da página de nova *password* da plataforma *web*.

Cargo de administrador ou gestão

Todos os utilizadores podem visualizar a localização e o nível de enchimento de cada caixote, porém, apenas os utilizadores com cargo de Administrador ou Gestão podem localizar os motoristas e saber a que velocidade estão e para onde se dirigem.

Na figura 5.46 e 5.47 é possível observar a *dashboard* para o cargo de Administrador.

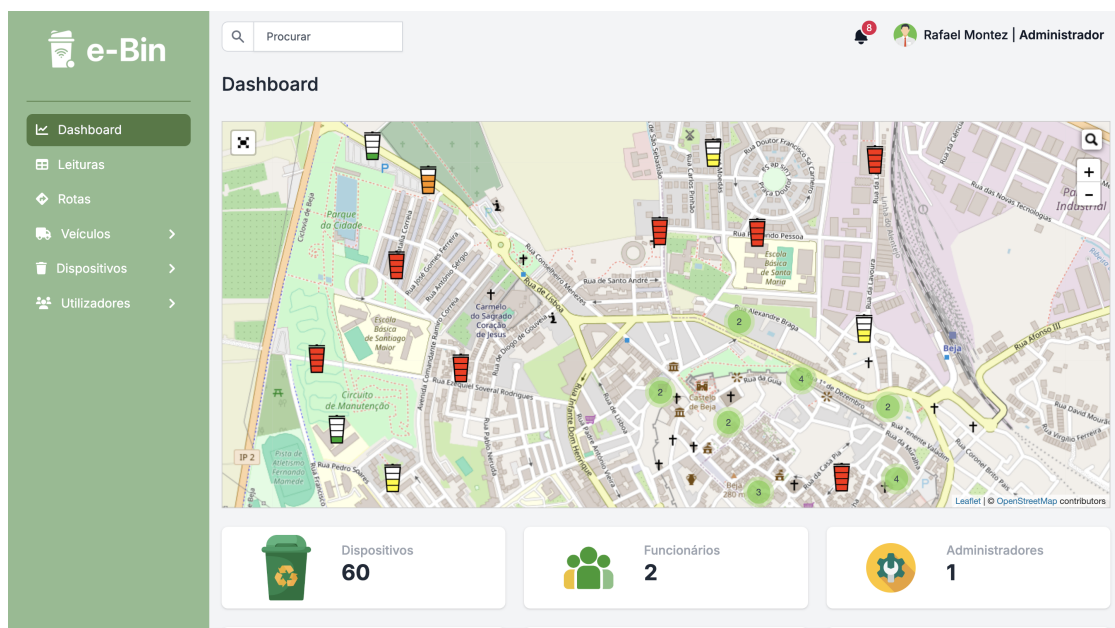


Figura 5.46: Captura de ecrã da página *dashboard* da plataforma *web*.

5. IMPLEMENTAÇÃO EXPERIMENTAL

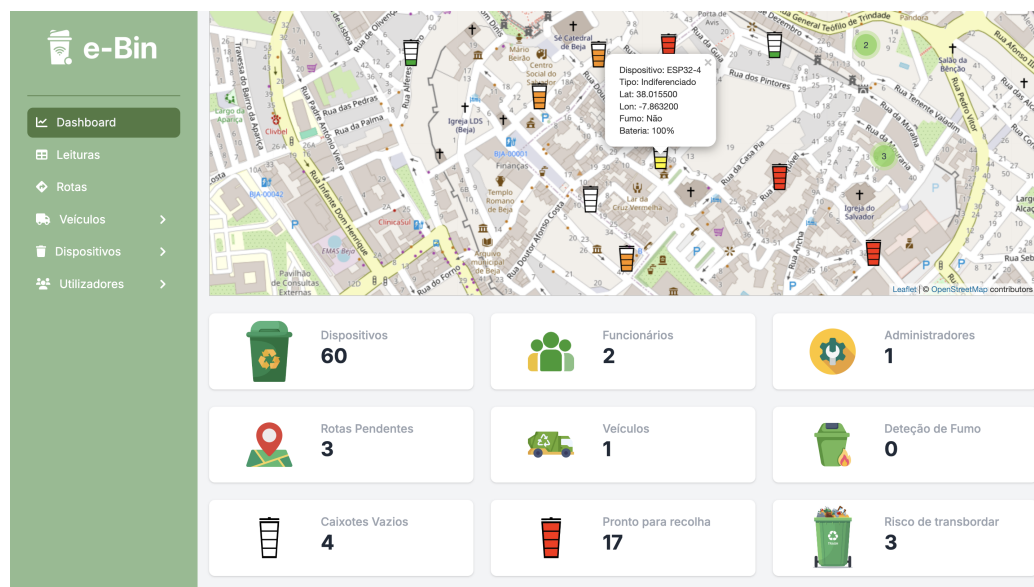


Figura 5.47: Captura de ecrã da página *dashboard* da plataforma *web*.

Caso os motoristas estejam a efetuar alguma rota, este pode ser localizado pelos utilizadores com cargo de administrador ou gestão.

A informação relativa à localização do motorista, velocidade e o ângulo para onde se move é obtida pela aplicação móvel através da biblioteca `background location` do Flutter que é enviada através de `WebSockets`, assim, os utilizadores com cargos de administrador ou gestão, conseguem ter o controlo de toda a frota tanto na aplicação móvel como na plataforma *web*. Os dados da rota real efetuada pelos motoristas são guardados na base de dados para que possam ser analisados e comparados com a rota gerada pelo sistema na página de detalhes das rotas concluídas. Essa informação pode ser visualizada no mapa da *dashboard* como mostra a figura 5.48.

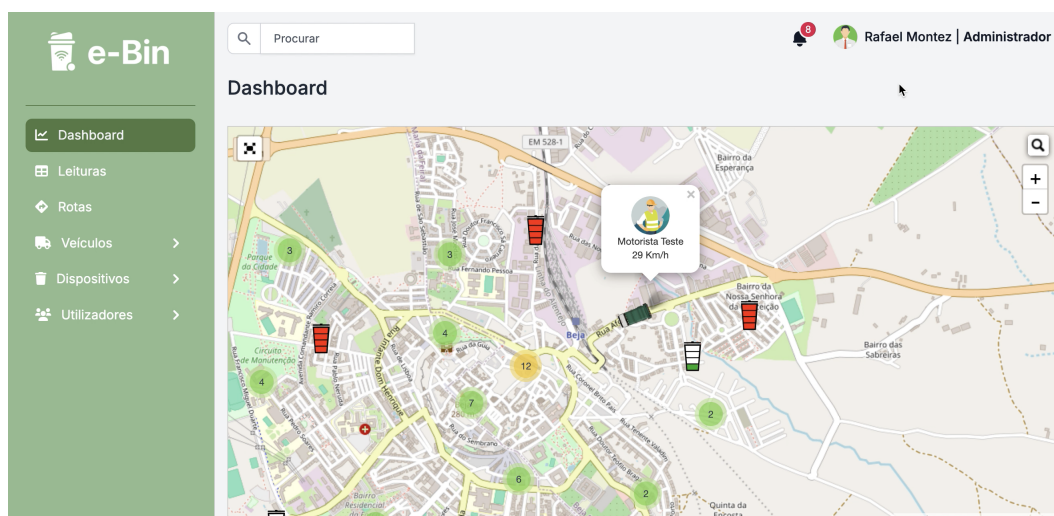


Figura 5.48: Captura de ecrã da página *dashboard* da plataforma *web*.

A figura 5.49 mostra a central de notificações onde é possível ver e ocultar notificações.



Figura 5.49: Central de notificações da plataforma web.

A figura 5.50 mostra a página de leituras, nesta página é possível visualizar dados relativos ao dispositivo instalado no caixote, como o nível de enchimento, deteção de fumo, localização e o nível de bateria. O nível de bateria é importante para que a equipa responsável pela manutenção dos dispositivos nunca deixe a bateria descarregar mais do que 20% e carregar mais de 80%, de modo a prolongar a vida útil da bateria. Esta página também utiliza WebSockets e a figura circular no canto superior direito, indica que a ligação deste foi bem sucedida, assim, a cada leitura recebida essa linha é atualizada sem ser necessário recarregar a página.

ID DISPOSITIVO	ESTADO	FUMO	LOCALIZAÇÃO (LAT,LON)	BATERIA	DATA/HORA
ESP32-1	27%	Não	(38.016294, -7.875425)	85%	03/04/2024, 15:01h
ESP32-2	10%	Não	(38.014987, -7.864234)	100%	03/04/2024, 15:02h
ESP32-3	90%	Não	(38.017843, -7.875968)	100%	03/04/2024, 15:03h
ESP32-4	40%	Não	(38.015500, -7.863200)	100%	03/04/2024, 15:04h
ESP32-5	30%	Não	(38.022481, -7.874438)	100%	03/04/2024, 15:05h
ESP32-6	70%	Não	(38.017522, -7.865966)	100%	03/04/2024, 15:06h
ESP32-7	85%	Não	(38.007738, -7.862309)	100%	03/04/2024, 15:07h
ESP32-8	60%	Não	(38.011810, -7.861869)	100%	03/04/2024, 15:08h
ESP32-9	50%	Não	(38.022334, -7.864988)	100%	03/04/2024, 15:09h
ESP32-10	95%	Não	(38.020585, -7.863765)	100%	03/04/2024, 15:10h

Figura 5.50: Captura de ecrã da página de leituras da plataforma web.

5. IMPLEMENTAÇÃO EXPERIMENTAL

Ao clicar sobre uma leitura é possível visualizar as últimas 10 leituras desse dispositivo através de um gráfico criado utilizando o **ChartJS**. A figura 5.51 mostra as últimas 10 leituras de um caixote.

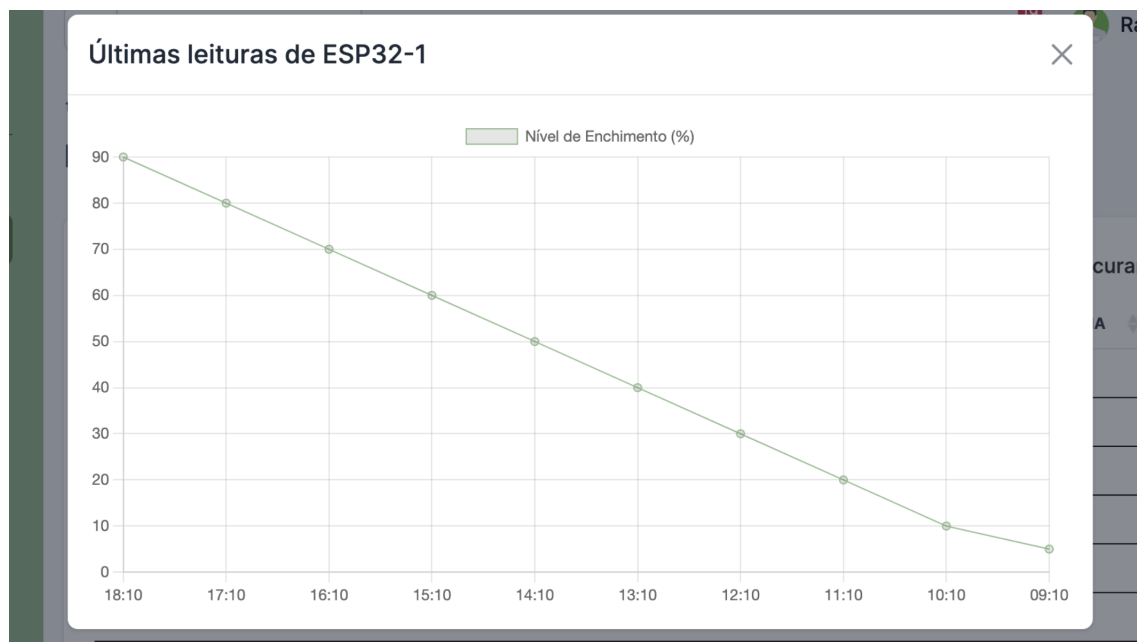
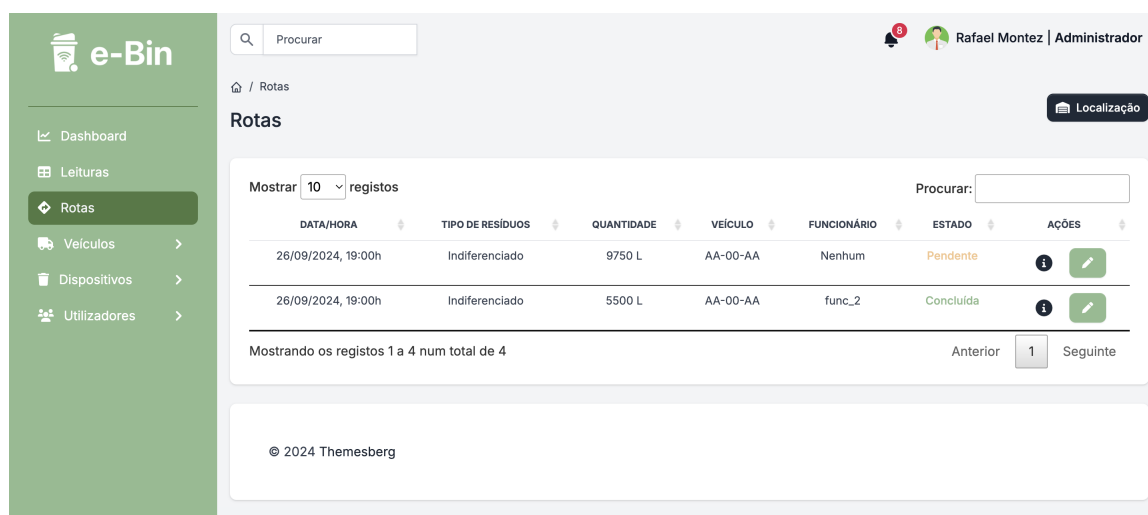


Figura 5.51: Gráfico das últimas 10 leituras.

A figura 5.52, mostra a página das rotas, onde é possível verificar as rotas geradas e ver detalhes, assim como editá-las.



Interface de usuário da plataforma web, mostrando a página das rotas. O cabeçalho inclui o logo "e-Bin", uma barra de busca, o nome de usuário "Rafael Montez | Administrador" e um botão "Localização". O menu lateral contém links para Dashboard, Leituras, Rotas (destacado), Veículos, Dispositivos e Utilizadores. A seção principal, intitulada "Rotas", apresenta uma tabela com 7 colunas: DATA/HORA, TIPO DE RESÍDUOS, QUANTIDADE, VEÍCULO, FUNCIONÁRIO, ESTADO e AÇÕES. A tabela exibe 2 registros. Abaixo da tabela, há uma barra de paginação indicando "Mostrando os registros 1 a 4 num total de 4" e botões "Anterior", "1" e "Seguinte".

DATA/HORA	TIPO DE RESÍDUOS	QUANTIDADE	VEÍCULO	FUNCIONÁRIO	ESTADO	AÇÕES
26/09/2024, 19:00h	Indiferenciado	9750 L	AA-00-AA	Nenhum	Pendente	[Info] [Editar]
26/09/2024, 19:00h	Indiferenciado	5500 L	AA-00-AA	func_2	Concluída	[Info] [Editar]

Figura 5.52: Captura de ecrã da página das rotas da plataforma web.

A figura 5.53 mostra a página de editar rota.

A screenshot displays the 'e-Bin' web application interface for editing a route. On the left is a green sidebar with navigation links: Dashboard, Leituras, Rotas (highlighted), Veículos, Dispositivos, and Utilizadores. The top header includes a search bar, a notification bell, and the user profile 'Rafael Montez | Administrador'. The main content area is titled 'Editar Rota' and contains a form with the following fields: 'ID Funcionário*' (func_2), 'Matricula*' (AA-00-AA), 'Data*' (26/09/2024), 'Hora*' (19:00), and 'Estado*' (Concluida). At the bottom of the form are 'Atualizar' and 'Voltar' buttons.

Figura 5.53: Captura de ecrã da página de editar rota da plataforma *web*.

Caso o motorista esteja a executar a rota, é possível verificar qual é o motorista, onde está e a que velocidade se dirige, ao clicar sobre o camião, como também acontece no mapa da *dashboard*.

Na figura 5.54 é possível verificar o camião a executar uma rota. Nesta página existem outras informações da rota, como o tipo de resíduos, o total de paragens, a distância, o tempo estimado e o custo com o combustível.

The screenshot shows the 'Detalhes da Rota' page in the 'e-Bin' application. The sidebar and header are identical to Figure 5.53. The main area features a map with a route and several location markers. To the right of the map is a summary panel titled 'Rota' with tabs for 'Detalhes' (selected) and 'Paragens'. The 'Detalhes' tab lists the following information: 'Tipo de Resíduo: Indiferenciado', 'Data/Hora: 26/09/2024, 19:00h', 'Distância: 7.18 Km', 'Total de paragens: 6', 'Tempo Estimado: 19.10 min', and 'Custo com combustível: 2.76 €'. The footer of the page indicates '© 2024 Themesberg'.

Figura 5.54: Captura de ecrã da página de detalhes da rota da plataforma *web*.

5. IMPLEMENTAÇÃO EXPERIMENTAL

A figura 5.55 mostra o percurso real do caminhão, através de uma linha mais escura, que surge após uma rota ser concluída.

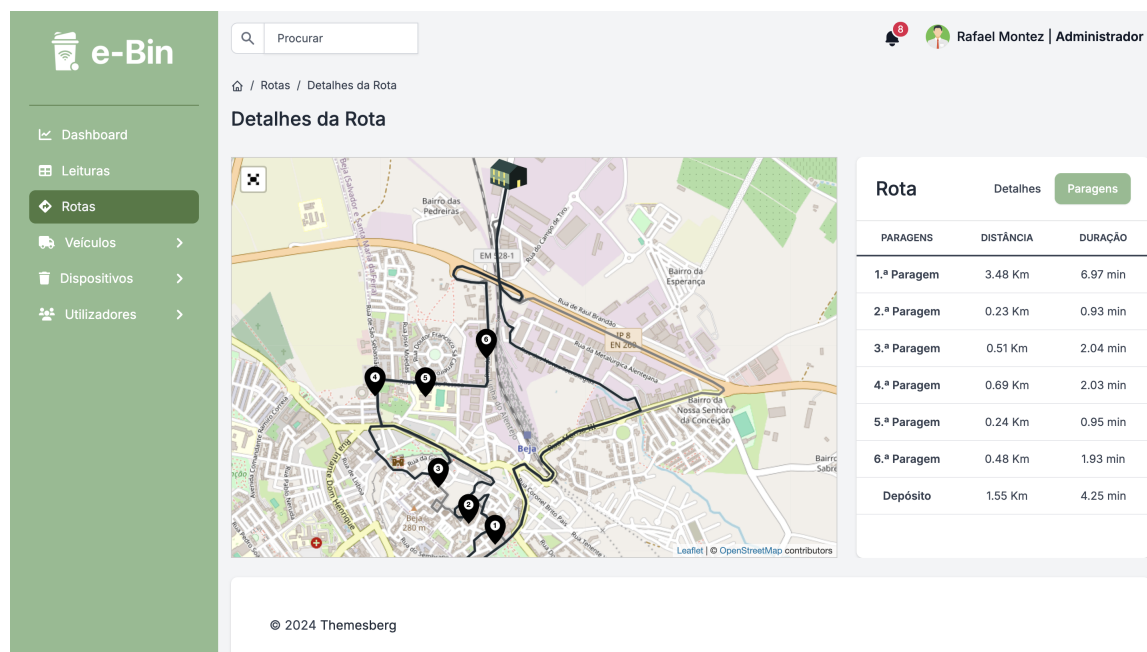


Figura 5.55: Captura de ecrã da página de detalhes da rota da plataforma *web*.

A figura 5.56 mostra a página de adicionar veículos, que é exclusiva dos utilizadores com cargo de administrador ou gestão.

Adicionar Veículo

Marca*

Modelo*

Capacidade (L)*

Consumo (l/100 Km)*

Tipo de Combustível*

Matrícula*

Mês/Ano*

Figura 5.56: Captura de ecrã da página de adicionar veículo da plataforma *web*.

A figura 5.57, mostra a página de gestão de veículos.

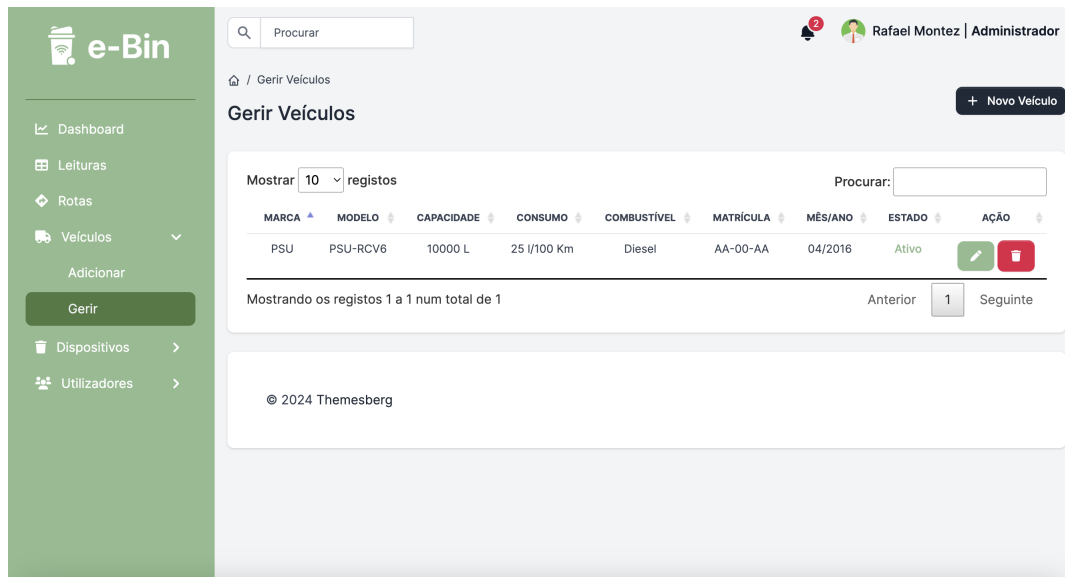


Figura 5.57: Captura de ecrã da página para gerir veículos da plataforma *web*.

A figura 5.57, mostra a confirmação de apagar um veículo.

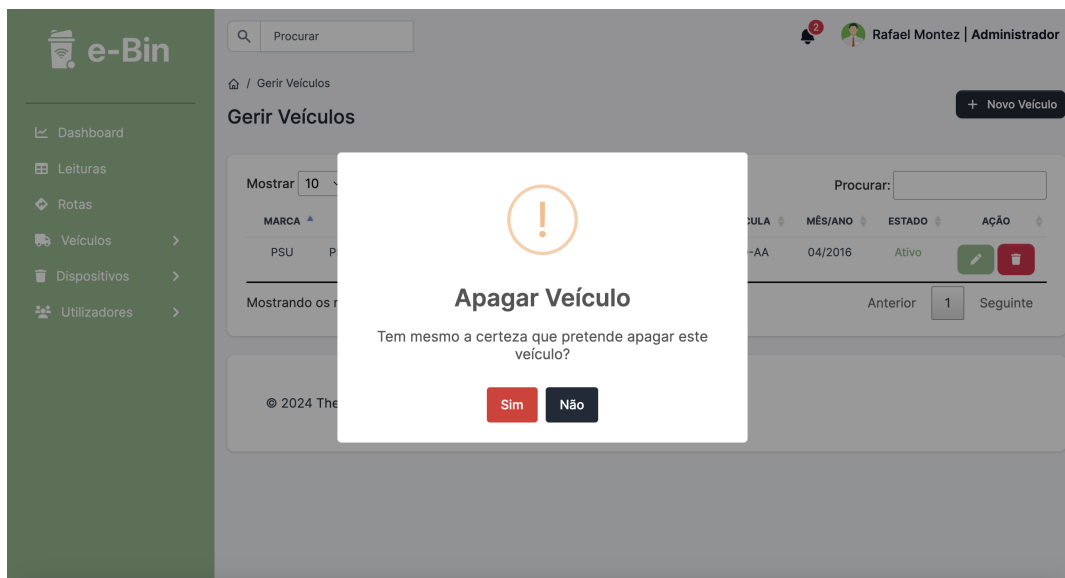
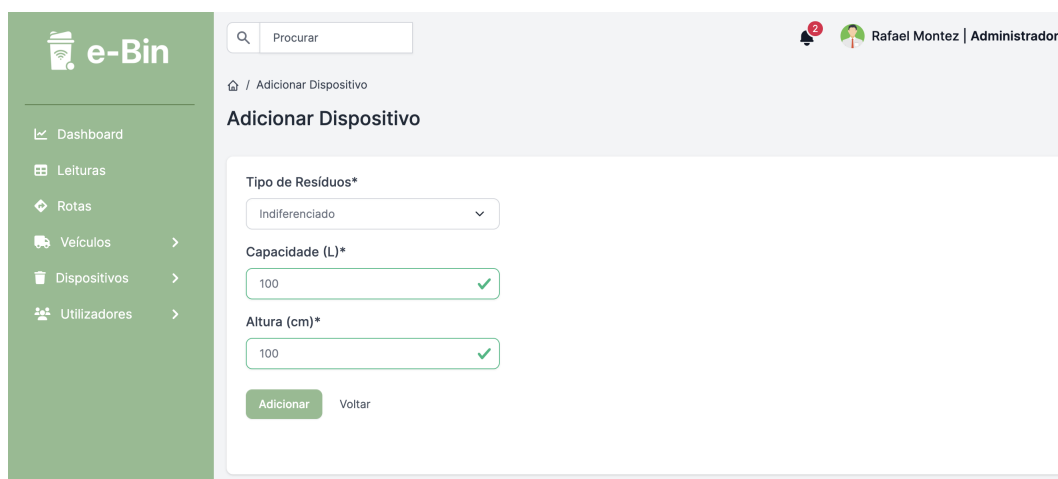


Figura 5.58: Captura de ecrã de apagar veículo na plataforma *web*.

5. IMPLEMENTAÇÃO EXPERIMENTAL

Os utilizadores com cargos de administrador ou gestão podem gerir os dispositivos. Na página de adicionar dispositivo é necessário adicionar o tipo de resíduo, a sua capacidade e a altura. No momento da criação da rota é verificada a capacidade de cada caixote para certificar que os resíduos que estão no caixote somados com outros não excedem a capacidade máxima do camião. A altura é necessária para calcular o nível de enchimento, este é baseado na altura do caixote vazio e a distância medida pelo sensor. A figura 5.59 mostra a página de adicionar dispositivo.



A captura de ecrã mostra a interface de administração da plataforma 'e-Bin'. No topo, há um menu lateral com opções como Dashboard, Leituras, Rotas, Veículos, Dispositivos e Utilizadores. O cabeçalho principal contém uma barra de pesquisa, um ícone de notificação e o nome de usuário 'Rafael Montez | Administrador'. O formulário 'Adicionar Dispositivo' está no centro, com campos para 'Tipo de Resíduos*' (Indiferenciado), 'Capacidade (L)*' (100) e 'Altura (cm)*' (100). Cada campo tem um ícone de marcação verde. Abaixo dos campos, há dois botões: 'Adicionar' (em verde) e 'Voltar'.

Figura 5.59: Captura de ecrã da página de adicionar dispositivo da plataforma *web*.

O ID do dispositivo é incrementado automaticamente, para que não existiam dispositivos repetidos e para que todos sigam o mesmo padrão, ou seja **ESP32-x**, como mostra a figura 5.60



A captura de ecrã mostra a mesma interface de administração, mas com uma mensagem de sucesso sobreposta no centro. A mensagem é: 'Dispositivo ESP32-31 adicionado com sucesso', acompanhada de um ícone de marcação verde. O formulário de fundo está desativado e cinzento.

Figura 5.60: Captura de ecrã da página de adicionar dispositivo da plataforma *web*.

A figura 5.61 mostra a página para gerir dispositivos.

A screenshot of the 'Gerir Dispositivos' (Manage Devices) page in the e-Bin web application. The page features a sidebar with navigation links: Dashboard, Leituras, Rotas, Veículos, Dispositivos, and Utilizadores. The main content area displays a table of devices with columns for ID, Type, Capacity, Height, Status, Date/Time, and Actions. A modal dialog is open, showing a confirmation message for deleting a device.

ID DISPOSITIVO	TIPO DE RESÍDUOS	CAPACIDADE	ALTURA	ESTADO	DATA/HORA	AÇÃO
ESP32-1	Indiferenciado	1000 L	100 cm	Ativo	03/04/2024, 18:01h	 
ESP32-2	Indiferenciado	1000 L	100 cm	Ativo	03/04/2024, 18:02h	 
ESP32-3	Indiferenciado	1000 L	100 cm	Ativo	03/04/2024, 18:03h	 
ESP32-4	Indiferenciado	1000 L	100 cm	Ativo	03/04/2024, 18:04h	 
ESP32-5	Indiferenciado	1000 L	100 cm	Ativo	03/04/2024, 18:05h	 
ESP32-6	Indiferenciado	1000 L	100 cm	Ativo	03/04/2024, 18:06h	 
ESP32-7	Indiferenciado	1000 L	100 cm	Ativo	03/04/2024, 18:07h	 
ESP32-8	Indiferenciado	1000 L	100 cm	Ativo	03/04/2024, 18:08h	 

Figura 5.61: Captura de ecrã da página para gerir dispositivos da plataforma web.

A figura 5.62 mostra a confirmação de apagar um dispositivo.

A screenshot of the 'Gerir Dispositivos' page showing a confirmation modal for deleting a device. The modal has a title 'Apagar Dispositivo' and a message 'Tem mesmo a certeza que pretende apagar este dispositivo?'. It includes two buttons: 'Sim' (Yes) and 'Não' (No). The background shows the same table of devices as in Figure 5.61.

Figura 5.62: Captura de ecrã da página de apagar dispositivo da plataforma web.

5. IMPLEMENTAÇÃO EXPERIMENTAL

A figura 5.63 mostra a página de editar dispositivo.

A screenshot displays the 'e-Bin' web application interface. On the left is a green sidebar with a menu containing 'Dashboard', 'Leituras', 'Rotas', 'Veículos', 'Dispositivos', and 'Utilizadores'. The main content area has a top header with a search bar labeled 'Procurar' and a user profile for 'Rafael Montez | Administrador'. Below the header, the breadcrumb trail shows 'Gerir Dispositivos / Editar Dispositivo'. The title 'Editar Dispositivo' is centered above the form. The form contains several fields: 'ID Dispositivo*' with the value 'ESP32-1', 'Tipo de Resíduos*' with a dropdown set to 'Indiferenciado', 'Capacidade (L)*' with the value '1000', 'Altura (cm)*' with the value '100', and 'Ativo*' with a dropdown set to 'Ativo'. At the bottom of the form are two buttons: 'Atualizar' (highlighted in green) and 'Voltar'.

Figura 5.63: Captura de ecrã da página de editar dispositivo da plataforma *web*.

A figura 5.64 mostra a página de adicionar utilizadores que é exclusiva dos utilizadores com cargo de administradore ou gestão.

The screenshot shows the 'e-Bin' web application interface for adding a user. The sidebar and top header are identical to the previous figure. The breadcrumb trail indicates 'Adicionar Utilizador'. The title 'Adicionar Utilizador' is centered above the form. The form includes the following fields: 'Nome*' with a text input containing 'Nome' and a validation message 'O nome é obrigatório'; 'Email*' with a text input containing 'Email' and a validation message 'O email é obrigatório'; 'Cargo*' with a dropdown menu set to 'Motorista'; and 'Receber alertas por email*' with a dropdown menu set to 'Sim'. At the bottom of the form are two buttons: 'Adicionar' (highlighted in green) and 'Voltar'.

Figura 5.64: Captura de ecrã da página de adicionar utilizador da plataforma *web*.

A figura 5.64 mostra a página para gerir utilizadores.

Mostrar 10 registros

Procurar:

ID	NOME	EMAIL	IMAGEM	CARGO	RECEBER EMAILS	ESTADO	DATA/HORA	AÇÃO
func_1	Rafael Montez	rafaeldmontez@gmail.com		Administrador	Sim	Ativo	03/04/2024, 14:01h	
func_2	Motorista Teste	motorista@teste.com		Motorista	Sim	Ativo	03/04/2024, 14:02h	
func_3	Gestão Teste	gestao@teste.com		Gestão	Sim	Ativo	03/04/2024, 14:03h	

Mostrando os registros 1 a 3 num total de 3

Anterior 1 Seguinte

Figura 5.65: Captura de ecrã da página para gerir utilizadores da plataforma *web*.

A figura 5.66 mostra a página de editar um utilizador.

Nome*

Gestão Teste

Email*

gestao@teste.com

Password

Password

Confirmar Password

Confirmar Password

Cargo*

Gestão

Receber alertas por email*

Sim

Ativo*

Ativo

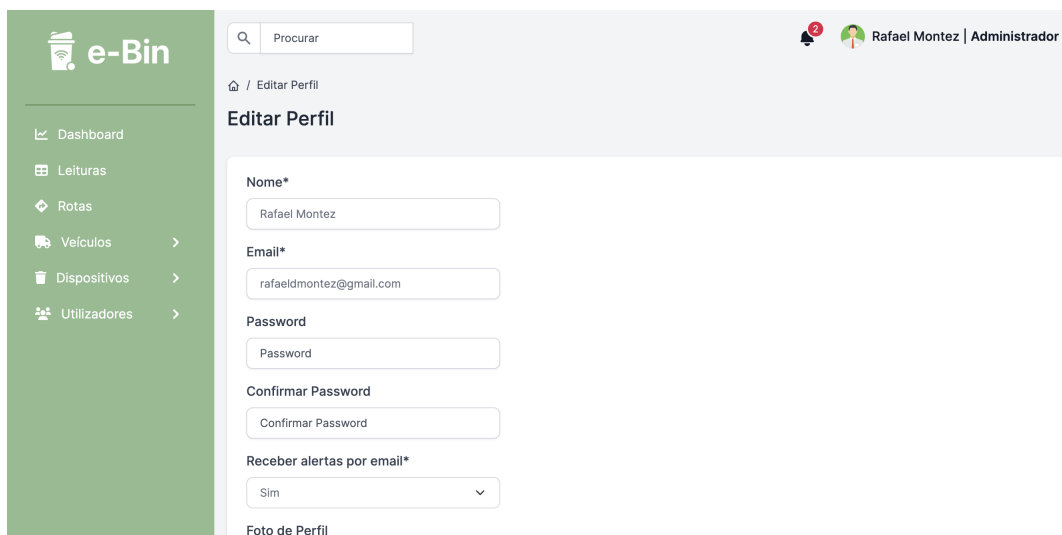
Atualizar Voltar

Figura 5.66: Captura de ecrã da página de editar utilizador da plataforma *web*.

5. IMPLEMENTAÇÃO EXPERIMENTAL

Caso algum utilizador necessite de alterar alguma informação pessoal ou foto de perfil, assim como faz na aplicação móvel, pode efetuá-lo na plataforma *web*.

A figura 5.66 mostra a página de editar perfil.



A screenshot displays the 'e-Bin' web application's 'Editar Perfil' (Edit Profile) page. On the left is a green sidebar with a menu containing 'Dashboard', 'Leituras', 'Rotas', 'Veículos', 'Dispositivos', and 'Utilizadores'. The main content area has a light gray header with a search bar and a user profile for 'Rafael Montez | Administrador'. Below the header, the page title 'Editar Perfil' is shown. The form includes fields for 'Nome*' (filled with 'Rafael Montez'), 'Email*' (filled with 'rafaeldmontez@gmail.com'), 'Password', and 'Confirmar Password'. There is also a dropdown for 'Receber alertas por email*' set to 'Sim' and a section for 'Foto de Perfil'.

Figura 5.67: Captura de ecrã da página de editar perfil da plataforma *web*.

A plataforma *web* também dispõe de um menu de ajuda para os utilizadores que necessitem, como mostra a figura 5.68.



The screenshot shows the 'Ajuda' (Help) page of the 'e-Bin' web application. The layout is consistent with the previous figure, but the main content area is titled 'Ajuda'. It contains a list of help topics, each with a checkbox: 'Quando são geradas as rotas?', 'Como deixar de receber emails informativos?', 'Como visualizar as rotas no mapa?', 'É possível comparar a rota gerada com a rota realmente efetuada?', 'Como verificar dados dos caixotes?', and 'Como localizar os motoristas?'. The last item is expanded, revealing a detailed explanation: 'Pode localizar em tempo real todos os motoristas no mapa da página "Dashboard" ou apenas o motorista correspondente a cada rota, dentro da página "Rotas" no botão [ícone], com isto é possível saber dados relevantes como a velocidade do camião.'

Figura 5.68: Captura de ecrã da página de ajuda da plataforma *web*.

Cargo de Motorista

Existem vários recursos que são ocultos ou mostrados de acordo com o cargo de cada utilizador. O utilizador com cargo de motorista apenas tem acesso à *dashboard* e as páginas de leituras e das rotas. Por exemplo, o motorista não pode controlar os outros motoristas. A figura 5.69 mostra a *dashboard* para os utilizadores com cargo de motorista.

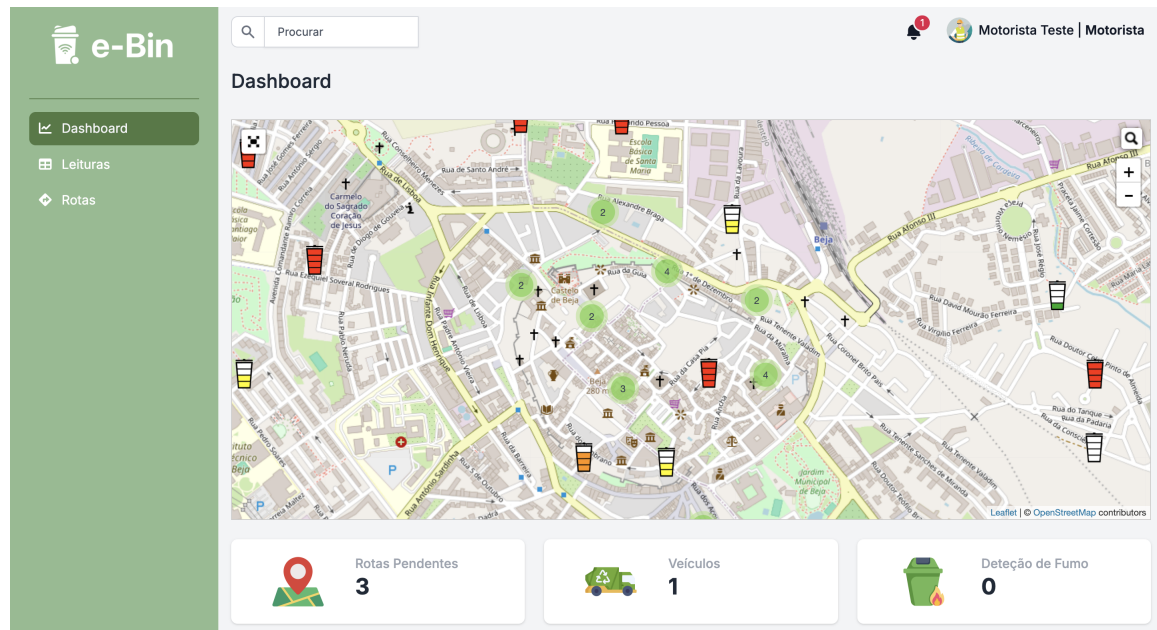


Figura 5.69: Captura de ecrã da página *dashboard* da plataforma *web*.

A figura 5.70 mostra a página de leituras dos motoristas.

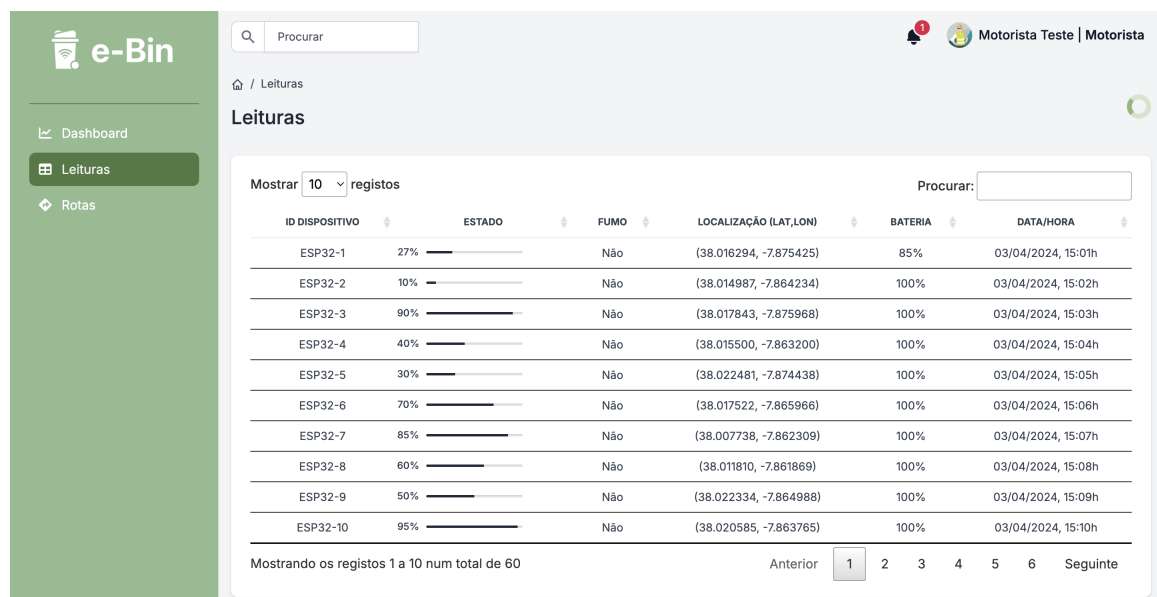


Figura 5.70: Captura de ecrã da página de leituras da plataforma *web*.

5. IMPLEMENTAÇÃO EXPERIMENTAL

A figura 5.71 mostra a página das rotas na vista dos utilizadores com cargo de motorista.

A screenshot displays the 'Rotas' (Routes) page for a user with the role of 'Motorista' (Driver). The interface includes a sidebar on the left with the 'e-Bin' logo and navigation links for 'Dashboard', 'Leituras' (Readings), and 'Rotas' (Routes). The main content area features a search bar at the top right, a breadcrumb trail ' / Rotas', and a title 'Rotas'. Below this, there's a section for displaying records, with a dropdown set to '10 registros' and a search input. A table lists two records, both dated '26/09/2024, 19:00h' and categorized as 'Indiferenciado'. The first record shows a quantity of '9750 L' and a vehicle 'AA-00-AA' with a status of 'Pendente' (Pending). The second record shows a quantity of '5500 L' and the same vehicle with a status of 'Concluída' (Completed). Each record has an information icon. At the bottom of the table, it indicates 'Mostrando os registros 1 a 4 num total de 4' and provides navigation links 'Anterior', '1', and 'Seguinte'. A footer at the very bottom reads '© 2024 Themesberg'.

DATA/HORA	TIPO DE RESÍDUOS	QUANTIDADE	VEÍCULO	ESTADO	AÇÕES
26/09/2024, 19:00h	Indiferenciado	9750 L	AA-00-AA	Pendente	
26/09/2024, 19:00h	Indiferenciado	5500 L	AA-00-AA	Concluída	

Figura 5.71: Captura de ecrã da página de rotas da plataforma *web*.

A figura 5.72 mostra a página de ajuda dos utilizadores com cargo de motorista.

The screenshot shows the 'Ajuda' (Help) page for a 'Motorista' user. The sidebar is identical to the previous figure. The main area has a breadcrumb trail ' / Ajuda' and a title 'Ajuda'. It contains a list of help topics, each with a question and a corresponding icon: 'Quando são geradas as rotas?' (checkbox), 'Como deixar de receber emails informativos?' (checkbox), 'Como visualizar as rotas no mapa?' (diamond), 'Entre na página "Rotas" e clique no botão .' (no icon), 'É possível comparar a rota gerada com a rota realmente efetuada?' (checkbox), and 'Como verificar dados dos caixotes?' (checkbox).

Figura 5.72: Captura de ecrã da página de ajuda da plataforma *web*.

5.13 Testes realizados

De modo a garantir o bom funcionamento e desempenho deste projeto, este foi submetido a diversos testes.

5.13.1 Consumo médio de corrente

Foram efetuadas medições com um multímetro digital de gama profissional, o KEYSIGHT 34465A na gama de 1A de medição. A figura 5.73 mostra um multímetro, como o que foi utilizado para efetuar as medições.



Figura 5.73: Multímetro digital KEYSIGHT 34465A. (em <https://www.newark.com/keysight-technologies/34465a/digital-multimeter-bench-6-1-2/dp/30Y4302>).

O microcontrolador TTGO juntamente com LoRa consome 61 mA, enquanto em *deep-sleep* consome apenas 2 mA.

O sensor HC-SR04, consome 15mA, o sensor MQ-2 cerca de 150mA e o NEO-6M V2 consome cerca de 45 mA. Foi possível observar que os consumos reais são muito próximos dos anunciados pelos fabricantes.

O tempo de carregamento de bateria (t) pode ser calculado da seguinte forma:

$$t = \text{Capacidade de bateria (mAh)} \times \text{Corrente de carga (mA)}$$

Neste caso é utilizado o módulo de carregamento *micro usb* TP4056 de 1A para carregar a bateria de 2500 mAh.

$$t = 2500 \text{ mAh} \times 1000 \text{ mA} = 2.5 \text{ horas}$$

É possível verificar que irá demorar 2 horas e meia com 100% de eficiência, o que em condições reais demora cerca de 3 horas, o que se verificou na prática.

5.13.2 Endpoints

De forma a garantir o bom funcionamento do projeto é necessário executar testes. Os *endpoints* desta API foram testados utilizando o *software* Postman.

No Postman foi criado uma *collection* que continha as rotas da API. Em toda a *collection* foi executado um teste para verificar se o código de *status* era 200, uma vez que este indica que o *request* foi bem-sucedido. Foi ainda verificado se o tempo de resposta era menor que 2 segundos, ou seja 2000 ms. Na figura 5.74 é possível verificar que passou nos testes do Postman.

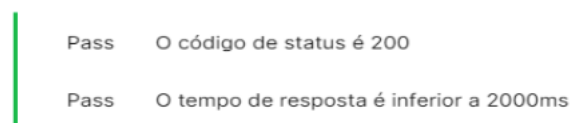


Figura 5.74: Resultado dos testes com Postman.

De modo a testar o funcionamento das rotas que recebem os dados em formato JSON, foi utilizado o *pytest*.

O *pytest* pode ser utilizado para produzir vários tipos de testes de *software* em Python, como testes unitários.

O código da listagem 5.10, mostra um exemplo de um teste para testar a rota de adicionar novos utilizadores.

```
1  from app import app
2  import json
3
4  app.config['TESTING'] = True
5  app.config['WTF_CSRF_ENABLED'] = False # Disable all CSRF protection
6
7  def test_add_user():
8      response = app.test_client().post(
9          '/add_user',
10         data=json.dumps({'name': 'test', 'email': 'test@test.com', 'role': 1,
11             'receive_emails': 1}),
12         content_type='application/json',
13     )
14
15     data = json.loads(response.get_data(as_text=True))
16
17     assert response.status_code == 200
18     assert data
```

Listing 5.10: Teste com *pytest*.

A figura 5.75, mostra o *output* dos testes executados utilizando o `pytest`, onde foi obtido sucesso.

```
collected 6 items

tests.py . . . . .

===== 6 passed in 0.67s
```

Figura 5.75: Resultado dos testes com `pytest`.

5.13.3 Conexão SocketIO

De forma a testar a conexão `SocketIO`, foram emitidos eventos para testar se a informação para preencher os mapas em tempo real com a localização dos motoristas estava a ser corretamente enviada e recebida pelo sistema. Este teste foi executado através do código da listagem 5.11.

```
1  import socketio
2  import time
3
4  sio = socketio.Client()
5
6  @sio.event
7  def connect():
8      print("Connected to the server.")
9
10     sio.emit('userCoord', {
11         'user_id': 3,
12         'route_id': 2,
13         'lat': 38.016430,
14         'lon': -7.875406,
15         'bearing': 300,
16         'speed': 50,
17         'username': 'Teste',
18         'img': 'static/img/driver_default.png'
19     })
20
21     time.sleep(10)
22     sio.emit('client_disconnecting', {'user_id': 3})
23
24  @sio.event
25  def response(data):
26      print("Response from the server:", data)
27
28  # Connect to the Socket.IO server
29  sio.connect('https://e-bin.pt')
```

```
30
31 # Keep the client running
32 try:
33     sio.wait()
34 except KeyboardInterrupt:
35     print("Disconnecting...")
36     sio.disconnect()
```

Listing 5.11: Teste SocketIO.

5.13.4 Testes adicionais

De modo a garantir que o sistema funcione conforme previsto foram efetuados testes complementares.

A fim de melhorar a usabilidade, foram colocados três indivíduos a realizar as tarefas, pois o *feedback* permitiu melhorar a usabilidade.

Também foram introduzidos dados de uma forma exaustiva, para garantir que caso o projeto esteja sobre alguma carga, este mantém-se estável.

5.14 Discussão Geral dos Resultados

Os objetivos propostos inicialmente foram cumpridos. A principal limitação deste projeto prende-se pelo facto de o sistema não gerar rotas para tipos diversos de resíduos. Os resíduos recicláveis necessitam de ser recolhidos em camiões específicos para esse efeito. É sabido que os camiões que recolhem os resíduos recicláveis, contêm no seu interior diferentes depósitos para armazenar vidro, papel e embalagens, o que dificulta a sua integração neste projeto. O sistema encontra-se preparado para gerir dispositivos com diversos tipos de resíduos, porém o algoritmo VRP não o permite.

Outra limitação deste protótipo, relaciona-se com o facto de não contar com a taxa de prensagem efetuada pelo camião no momento da recolha, uma vez que, esta diminui o volume dos resíduos. Esta característica pode variar de veículo para veículo e de acordo com o tipo de resíduo que está a prensar.

5.15 Conclusão

Neste capítulo, foi implementado o caixote inteligente *e-Bin*. Com base nesta implementação experimental, foi possível desenvolver um sistema funcional que integra *hardware* e *software*. Inicialmente, o circuito projetado foi testado na *breadboard*, demonstrando a viabilidade do *layout* do circuito.

A alimentação foi configurada para garantir o funcionamento do dispositivo, mesmo sem luz solar.

A comunicação do dispositivo IoT com o servidor foi estabelecida com sucesso, permitindo a transmissão de dados para a base de dados, onde são armazenados para análise. Foram implementadas medidas de segurança informática para proteger o sistema.

Os testes realizados validaram o desempenho do sistema, comprovando que o caixote do lixo inteligente é capaz de operar em condições reais, apresentando os resultados esperados.

Capítulo 6

Conclusão

6.1 Conclusões gerais

A gestão de resíduos urbanos é um tema de crescente importância global, dada a necessidade de implementar estratégias que promovam a sustentabilidade em todo o mundo. Neste contexto, incluir a gestão de resíduos como uma componente essencial dessas estratégias é fundamental para o sucesso de iniciativas voltadas à melhoria do meio ambiente.

De forma a melhorar a qualidade da recolha, reduzir o custo da recolha de resíduos e otimizar o percurso dos camiões do lixo, é importante a utilização de caixotes inteligentes [Zan+14].

Diversos projetos com caixotes do lixo inteligentes têm sido desenvolvidos com o principal objetivo de melhorar a gestão de resíduos. Neste relatório é apresentado o desenvolvimento do caixote do lixo inteligente **e-Bin**, uma solução que aplica a utilização da arquitetura IoT para resolver várias questões relacionadas à gestão de resíduos. A arquitetura proposta neste projeto, foi resultado de um estudo detalhado sobre as necessidades e desafios da gestão de resíduos, este mostra-se como uma solução eficiente e adaptável para o contexto urbano.

A análise dos sensores demonstrou uma grande importância para selecionar componentes que garantam não só a robustez, como também a eficiência na comunicação do sistema, tendo em conta as características do ambiente onde os caixotes serão instalados.

De modo a gerar rotas otimizadas foi implementada a heurística de *Clarke and Wright*, uma vez que é eficaz e flexível. A otimização das rotas dos veículos de recolha, não só contribui para uma operação mais eficiente, como também reduz significativamente os custos operacionais e as emissões de gases poluentes, contribuindo para uma gestão de resíduos mais sustentável.

A utilização do VRP neste projeto mostrou-se vantajosa, uma vez que possibilitou uma redução de tempo e da distância percorrida pelos veículos de recolha, o que, por sua vez, tem impacto na diminuição dos custos e no aumento da eficiência das recolhas.

A implementação prática do sistema **e-Bin** envolveu a integração de uma parte de

hardware e outra parte de *software* para criar um sistema funcional e eficaz. Inicialmente, os testes foram implementados em *breadboards*, onde também o circuito foi projetado. Os testes iniciais foram fundamentais para assegurar que o sistema funcionasse conforme esperado.

Durante os testes, o sistema **e-Bin** apresentou resultados satisfatórios, demonstrando a sua capacidade de operar autonomamente e de acordo com os requisitos estabelecidos.

De modo geral, o projeto do caixote do lixo inteligente **e-Bin** demonstrou ser uma solução viável e eficaz para a gestão de resíduos sólidos urbanos. A utilização da arquitetura IoT permite que o sistema funcione de maneira autónoma, recolhendo e processando dados sem a necessidade de intervenção humana direta. Isso resulta numa gestão de resíduos mais eficiente, reduzindo os custos operacionais e as emissões de dióxido de carbono (CO₂).

O protótipo do caixote do lixo inteligente **e-Bin** demonstrou ser capaz de operar eficientemente em condições reais. A sua implementação em grande escala poderá contribuir para a melhoria do meio ambiente e da qualidade de vida nas cidades. O dispositivo tem um custo de cerca de 65 EUR e um consumo total de aproximadamente 4.23 mAh.

Para complementar as funcionalidades do dispositivo **e-Bin**, foi desenvolvida uma aplicação web e uma aplicação móvel, para facilitar o acesso dos utilizadores à informação em tempo real.

Em última instância, o **e-Bin** é um projeto que oferece uma solução prática e sustentável para os desafios da gestão de resíduos nas cidades inteligentes. Este combina a tecnologia IoT com uma abordagem voltada à sustentabilidade, o sistema permite mudar o modo de como os resíduos são geridos, contribuindo para um ambiente mais limpo e sustentável.

A incorporação de tecnologias IoT neste setor, permitirá uma gestão de resíduos cada vez mais eficiente, sustentável e economicamente viável, beneficiando as cidades, o meio ambiente e a sociedade.

6.2 Trabalho futuro

O **e-Bin** é um protótipo que visa melhorar a gestão de resíduos urbanos, contudo existem melhorias desde a robustez física e proteção contra vandalismo, até a segurança informática e integração de inteligência artificial, aumentando assim a eficiência e a confiabilidade do sistema.

Uma vez que se trata de um protótipo, atualmente, o **e-Bin** está implementado numa caixa simples. De modo a prolongar a sua vida útil, em condições reais é fundamental utilizar uma caixa com certificação IP-65 ou superior, garantindo proteção contra pó e jatos de água. Além disso, é importante adotar materiais mais robustos e resistentes ao vandalismo com revestimentos reforçados dificultando tentativas de destruição.

Uma vez que o dispositivo, integra um módulo GPS e os utilizadores responsáveis são alertados cada vez que a sua localização é alterada, assim é necessário a utilização de

mecanismos que dificultem sua remoção.

Uma das principais melhorias sugeridas para a arquitetura IoT do **e-Bin** é a introdução de redundância nos pontos críticos. Isso pode ser feito através da duplicação de sensores e módulos de comunicação, garantindo que o sistema continue a funcionar mesmo em caso de falhas. Por exemplo, pode ser incluída uma segunda rede de comunicação, como **LTE** ou **Sigfox**, que pode servir de *backup* para o **LoRaWAN**, aumentando assim a resiliência do sistema.

Outra melhoria passa também pela utilização de sensores mais precisos, como a utilização de um sensor de distância a laser (**LIDAR**).

Outras futuras melhorias será relacionado com a parte da segurança informática, como encriptar todos os dados transmitidos e implementar mecanismos para a deteção de intrusões.

Futuramente, a introdução de inteligência artificial (**IA**) no projeto **e-Bin** permitirá prever os padrões de enchimento dos caixotes de lixo com base em dados históricos. O que permitirá ajustar automaticamente as rotas dos veículos de recolha, reduzindo o consumo de combustível e melhorando a eficiência operacional. Esta também pode ser adotada para analisar imagens no interior dos caixotes, devido à sua capacidade de identificar e classificar resíduos de forma eficiente, melhorando o processo de reciclagem.

Atualmente, o sistema gera rotas apenas para os caixotes de resíduos comuns quase cheios. No entanto, essa abordagem pode ser melhorada com algoritmos de otimização de rotas mais avançados que levam em consideração a heterogeneidade dos camiões de recolha. Isto garantirá que as rotas geradas sejam mais eficientes e economicamente viáveis. Estes deveriam ser implementados em linguagens de programação de mais baixo nível como **C** ou linguagens matemáticas como **Julia**, de modo a serem executados mais rapidamente e assim lidar com uma maior quantidade de dados, conforme apresentado na Secção 4.5.

A heurística de *Clarke and Wright* é uma primeira aproximação, para obter uma melhor qualidade de rotas, deve ser adotado um algoritmo mais avançado.

Alguns aspetos iniciais do desempenho computacional devem ser melhorados futuramente, uma vez que, foram obtidos com dados gerados aleatoriamente. De modo a garantir uma noção real do crescimento do procedimento computacional deve-se utilizar dados reais, através de *datasets* de **VRP**.

As interfaces dos utilizadores podem ser melhoradas de modo a facilitar a utilização da plataforma *web* e da aplicação móvel. Também poderá ser incluída uma área respetiva à manutenção preventiva dos dispositivos que irá notificar os responsáveis de manutenção quando necessário, garantindo assim que os dispositivos estejam sempre operacionais.

Uma futura melhoria do **e-Bin** seria suportar diferentes tipos de resíduos como o papel, vidro, plástico, entre outros. O sistema poderá informar em tempo real a quantidade de cada resíduo, o que permitirá utilizar camiões de recolha equipados com compartimentos subdivididos internamente.

6. CONCLUSÃO

Por último, outra melhoria importante é a consideração da taxa de prensagem dos veículos de recolha. Como os resíduos são compactados dentro dos camiões, o sistema deverá ajustar automaticamente a quantidade de caixotes a serem recolhidos com base no volume reduzido após a prensagem. Isso otimizará ainda mais as rotas e a utilização da capacidade dos camiões.

Bibliografia

- [1Bi24] 1BitSquared. *KiCad EDA*. <https://kicad.org/>. Consultado em 2024/10/11. 2024 (ver p. 72).
- [AAA22] Omer Al-Dulaimi, Mohammed Al-Dulaimi e A. Al-Dulaimi. «Analysis of Low Power Wireless Technologies used in the Internet of Things (IoT)». Em: abr. de 2022, pp. 1–6. DOI: 10.1109/ICMI55296.2022.9873714. URL: https://www.researchgate.net/publication/363312661_Analysis_of_Low_Power_Wireless_Technologies_used_in_the_Internet_of_Things_IoT (ver pp. 80, 81).
- [ABD15] Vito Albino, Umberto Berardi e Rosa Dangelico. «Smart Cities: Definitions, Dimensions, Performance, and Initiatives». Em: *Journal of Urban Technology* 22 (fev. de 2015), p. 2015. DOI: 10.1080/10630732.2014.942092. URL: <https://www.tandfonline.com/doi/abs/10.1080/10630732.2014.942092> (ver pp. 5, 6).
- [Add+19] Tommaso Addabbo et al. «A LoRa-based IoT Sensor Node for Waste Management Based on a Customized Ultrasonic Transceiver». Em: mar. de 2019, pp. 1–6. DOI: 10.1109/SAS.2019.8705980. URL: <https://ieeexplore.ieee.org/document/8705980> (ver p. 14).
- [Águ20] Câmara Municipal de Águeda. «Relatório de balanço e revisão do sistema de gestão». Em: (jun. de 2020). URL: https://www.cm-agueda.pt/cmageda/uploads/document/file/6188/relat_revisao_2020.pdf (ver p. 8).
- [All24] LoRa Alliance. *LoRaWAN - Most Common Applications and Use Cases*. <https://loro-alliance.org/lorawan-news/lorawan-most-common-applications-and-use-cases/>. Consultado em 2024/06/01. 2024 (ver p. 69).
- [Amb22] Porto Ambiente. *Porto e a Gestão Sustentável dos Resíduos Urbanos*. <https://www.porto.pt/>. Consultado em 2024/08/01. 2022 (ver p. 9).
- [Ari21] Dan Arias. *Hashing in Action: Understanding bcrypt*. <https://auth0.com/blog/hashing-in-action-understanding-bcrypt/>. Consultado em 2024/07/31. 2021 (ver p. 93).

- [Bai+17] Zubair A. Baig et al. «Future challenges for smart cities: Cyber-security and digital forensics». Em: *Digital Investigation* 22 (2017), pp. 3–13. ISSN: 1742-2876. DOI: <https://doi.org/10.1016/j.diin.2017.06.015>. URL: <https://www.sciencedirect.com/science/article/pii/S1742287617300579> (ver p. 10).
- [Big24] Bigbelly. *Bigbelly Waste Management*. <https://bigbelly.com/solutions/municipal>. Consultado em 2024/06/01. 2024 (ver pp. 18, 19).
- [Bra22] Braval. *Contentores Inteligentes em Braga para uma Cidade Sustentável*. <https://www.braval.pt/>. Consultado em 2024/08/01. 2022 (ver p. 9).
- [Cac+14] Jose Caceres Cruz et al. «A Savings-based Randomized Heuristic for the Heterogeneous Fixed Fleet Vehicle Routing Problem with Multi-trips». Em: *Journal of Applied Operational Research* 6 (jan. de 2014), pp. 69–81. URL: https://www.researchgate.net/publication/259276416_A_Savings-based_Randomized_Heuristic_for_the_Heterogeneous_Fixed_Fleet_Vehicle_Routing_Problem_with_Multi-trips (ver pp. 49–51).
- [Car21] Filipa Cardoso. *Esposende Smart City*. <https://smart-cities.pt/noticias/esposende-smart-city3003/>. Consultado em 2024/05/20. Mar. de 2021 (ver p. 8).
- [Cas21] Câmara Municipal de Cascais. *Cascais e a Gestão de Resíduos: A Implementação de Contentores Inteligentes*. <https://www.cascais.pt/>. Consultado em 2024/08/01. 2021 (ver p. 9).
- [CFG16] Jürgen Cito, Vincenzo Ferme e Harald Gall. «Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research». Em: jun. de 2016, pp. 609–612. ISBN: 978-3-319-38790-1. DOI: 10.1007/978-3-319-38791-8_58. URL: https://www.researchgate.net/publication/303515069_Using_Docker_Containers_to_Improve_Reproducibility_in_Software_and_Web_Engineering_Research (ver p. 77).
- [Chi24] ChirpStack. *The ChirpStack project*. <https://www.chirpstack.io/docs/>. Consultado em 2024/10/11. 2024 (ver pp. 34, 79).
- [Cit22] Aveiro STEAM City. *Inovação e Sustentabilidade com Caixotes Inteligentes em Aveiro*. <https://www.cm-aveiro.pt/>. Consultado em 2024/08/01. 2022 (ver p. 8).
- [Cor15] Semtech Corporation. *SX1276 Datasheet*. <https://pdf1.alldatasheet.com/datasheet-pdf/view/800239/SEMTECH/SX1276.html>. Consultado em 2024/05/15. 2015 (ver p. 71).

- [CSC21] Nuno Cruz, António Serrador e Nuno Cota. *Rede de Comunicação LoRa*. <https://lisboainteligente.cm-lisboa.pt/lxi-iniciativas/rede-de-comunicacao-lora/>. Consultado em 2024/05/20. Out. de 2021 (ver p. 9).
- [CTM21] Diogo Correia, Leonor Teixeira e João Lourenço Marques. «Reviewing the State-of-the-Art of Smart Cities in Portugal: Evidence Based on Content Analysis of a Portuguese Magazine». Em: *Publications* 9.4 (2021). ISSN: 2304-6775. DOI: 10.3390/publications9040049. URL: <https://www.mdpi.com/2304-6775/9/4/49> (ver p. 8).
- [Dee23] Matthew Deery. *Hashing in Action: Understanding bcrypt*. <https://careerfoundry.com/en/blog/web-development/what-is-flask/>. Consultado em 2024/07/31. 2023 (ver p. 44).
- [Día06] Bernabé Dorronsoro Díaz. *The VRP Web*. <https://bernabe.dorronsoro.es/vrp/>. Consultado em 2024/10/25. 2006 (ver pp. 65, 67).
- [DK18] Shilpa Devalal e A. Karthikeyan. «LoRa Technology - An Overview». Em: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 2018, pp. 284–290. DOI: 10.1109/ICECA.2018.8474715. URL: <https://ieeexplore.ieee.org/document/8474715> (ver p. 80).
- [DZ16] Dan Ding e Xin Zou. «The Optimization of Logistics Distribution Route Based on Dijkstra’s Algorithm and C-W Savings Algorithm». Em: jun. de 2016. DOI: 10.2991/mmabc-16.2016.200. URL: https://www.researchgate.net/publication/308113903_The_Optimization_of_Logistics_Distribution_Route_Based_on_Dijkstra’s_Algorithm_and_C-W_Savings_Algorithm (ver p. 15).
- [Eco22] EcoGestus. *Soluções Smart Waste: Caixotes de Lixo Inteligentes*. <https://www.ecogestus.com/>. Consultado em 2024/08/01. 2022 (ver p. 9).
- [Ein21] Einstronic. *TTGO T-Display ESP32*. <https://einstronic.com/product/ttgo-t-display-esp32/>. Consultado em 2024/07/01. 2021 (ver p. 70).
- [Fri20] Fritzing. *Fritzing*. <https://fritzing.org/>. Consultado em 2024/10/11. 2020 (ver p. 73).
- [Ful22] James Fuller. *NEO-6M V2 Datasheet*. <https://www.datasheethub.com/wp-content/uploads/2022/08/NEO6MV2-GPS-Module-Datasheet.pdf>. Consultado em 2024/05/15. 2022 (ver pp. 45, 46).
- [Fun20] Câmara Municipal do Funchal. *Funchal Eco: Contentores Inteligentes nas Zonas Turísticas*. <https://www.funchal.pt/>. Consultado em 2024/08/01. 2020 (ver p. 9).

- [Gui21] Câmara Municipal de Guimarães. *Guimarães Mais Verde: Um Compromisso Sustentável*. <https://www.guimaraes.pt/>. Consultado em 2024/08/01. 2021 (ver p. 9).
- [Hal+00] R. Hall et al. «The vision of a smart city». Em: *2nd Int. Life*. (jan. de 2000). URL: <https://www.osti.gov/servlets/purl/773961> (ver p. 6).
- [Haq+20] Khandaker Haque et al. *An IoT Based Efficient Waste Collection System with Smart Bins*. Jun. de 2020. DOI: 10.13140/RG.2.2.13406.25924. URL: https://www.researchgate.net/publication/341900743_An_IoT_Based_Efficient_Waste_Collection_System_with_Smart_Bins (ver p. 14).
- [Jul24] JuliaLang. *Julia Micro-Benchmarks*. <https://julialang.org/benchmarks/>. Consultado em 2024/10/11. 2024 (ver p. 60).
- [Kar+19] Ahmadh Kariapper et al. «Smart Garbage Collection Using GPS & Shortest Path Algorithm». Em: dez. de 2019. DOI: 10.1109/PuneCon46936.2019.9105674. URL: <https://ieeexplore.ieee.org/document/9105674> (ver p. 1).
- [Kha+19] Ala Khalifeh et al. «A Survey of 5G Emerging Wireless Technologies Featuring LoRaWAN, Sigfox, NB-IoT and LTE-M». Em: mar. de 2019, pp. 561–566. DOI: 10.1109/WiSPNET45539.2019.9032817. URL: <https://ieeexplore.ieee.org/document/9032817> (ver p. 80).
- [KL09] Rosabeth Kanter e Stanley Litow. «Informed and Interconnected: A Manifesto for Smarter Cities». Em: *SSRN Electronic Journal* (jun. de 2009). DOI: 10.2139/ssrn.1420236. URL: https://www.researchgate.net/publication/256001928_Informed_and_Interconnected_A_Manifesto_for_Smarter_Cities (ver p. 5).
- [KZ16] Fabio Kon e Eduardo Felipe Zambom Santana. «Cidades Inteligentes: Tecnologias, Aplicações, Iniciativas e Desafios». Em: jul. de 2016, pp. 13–60. ISBN: 9788576693260. DOI: 10.5753/sbc.6.1. URL: <https://repositorio.usp.br/item/003184594> (ver pp. 10, 11).
- [LIL24] LILYGO. *LoRa32 - LILYGO*. <https://www.lilygo.cc/products/lora3>. Consultado em 2024/05/15. 2024 (ver pp. 70, 71).
- [Meh+17] Yasir Mehmood et al. «Internet-of-Things-Based Smart Cities: Recent Advances and Challenges». Em: *IEEE Communications Magazine* 55.9 (2017), pp. 16–24. DOI: 10.1109/MCOM.2017.1600514. URL: <https://ieeexplore.ieee.org/document/8030479> (ver pp. 11, 12).

- [Mek+18] Kais Mekki et al. «Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT». Em: mar. de 2018. DOI: 10.1109/PERCOMW.2018.8480255. URL: <https://ieeexplore.ieee.org/document/8480255> (ver pp. 81, 82).
- [Mik24a] MikroTik. *Kit LtAP LR8 LTE*. https://mikrotik.com/product/ltap_lr8_lte_kit. Consultado em 2024/07/31. 2024 (ver p. 83).
- [Mik24b] MikroTik. *LtAP LR8 LTE kit*. https://cdn.mikrotik.com/web-assets/product_files/LtAPLR8LTEkit_230204.pdf. Consultado em 2024/07/01. 2024 (ver p. 83).
- [Moh16] Saraju Mohanty. «Everything You Wanted to Know About Smart Cities». Em: *IEEE Consumer Electronics Magazine* 5 (jul. de 2016), pp. 60–70. DOI: 10.1109/MCE.2016.2556879. URL: <https://ieeexplore.ieee.org/document/7539244> (ver p. 7).
- [Mor14] Elijah J. Morgan. *HC-SR04 Ultrasonic Sensor Datasheet*. <https://pdf1.alldatasheet.com/datasheet-pdf/view/1132204/ETC2/HCSR04.html>. Consultado em 2024/05/15. 2014 (ver pp. 46, 47).
- [Mou21] Radouan Mouha. «Internet of Things (IoT)». Em: *Journal of Data Analysis and Information Processing* 09 (jan. de 2021), pp. 77–101. DOI: 10.4236/jdaip.2021.92006. URL: <https://www.scirp.org/journal/paperinformation?paperid=108574> (ver p. 10).
- [MP17] B.S. Malapur e Vani Pattanshetti. «IoT based waste management: An application to smart city». Em: ago. de 2017, pp. 2476–2486. DOI: 10.1109/ICECDS.2017.8389897. URL: <https://ieeexplore.ieee.org/document/8389897> (ver p. 1).
- [MQT22] MQTT.org. *MQTT: The Standard for IoT Messaging*. <https://mqtt.org/>. Consultado em 2024/06/01. 2022 (ver p. 84).
- [MR20] Ayaskanta Mishra e Arun Ray. «IoT cloud-based cyber-physical system for efficient solid waste management in smart cities: a novel cost function based route optimisation technique for waste collection vehicles using dustbin sensors and real-time road traffic informatics». Em: *IET Cyber-Physical Systems Theory & Applications* 5 (out. de 2020). DOI: 10.1049/iet-cps.2019.0110. URL: <https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-cps.2019.0110> (ver pp. 2, 15).
- [MRT15] Somayya Madakam, R Ramaswamy e Siddharth Tripathi. «Internet of Things (IoT): A Literature Review». Em: *Journal of Computer and Communications* 3 (abr. de 2015), pp. 164–173. DOI: 10.4236/jcc.2015.35021. URL: <https://www.scirp.org/journal/paperinformation?paperid=56616> (ver p. 10).

- [Net24] The Things Network. *The Things Network*. <https://www.thethingsnetwork.org/>. Consultado em 2024/10/25. 2024 (ver p. 79).
- [OOY17] Z. Oralhan, Burcu Oralhan e Yonca Yigit. «Smart city application: Internet of Things (IoT) technologies based smart waste collection using data mining approach and ant colony optimization». Em: *International Arab Journal of Information Technology* 14 (jan. de 2017), pp. 423–427. URL: <https://www.iajit.org/upload/files/Smart-City-Application-Internet-of-Things-IoT-Technologies-Based-Smart-Waste-Collection-Using-Data-Mining-Approach-and-Ant-Colony-Optimization.pdf> (ver p. 15).
- [OPB21] Pedro Oliveira, Isabel Pedrosa e Jorge Bernardino. «IoT nas Smart Cities - Revisão da literatura». Em: *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao* (fev. de 2021), p. 330. URL: <https://www.proquest.com/openview/447edae31d1463bb886cffd18e0f5560/1.pdf?pq-origsite=gscholar&cbl=1006393> (ver p. 6).
- [Ope24] OpenStreetMap. *OpenStreetMap*. <https://www.openstreetmap.org/about>. Consultado em 2024/06/01. 2024 (ver p. 44).
- [Ora24] Oracle. *What is MySQL?* <https://www.oracle.com/mysql/what-is-mysql/>. Consultado em 2024/07/31. 2024 (ver p. 92).
- [Pan+13] Gang Pan et al. «Trace analysis and mining for smart cities: issues, methods, and applications». Em: *IEEE Communications Magazine* 51.6 (2013), pp. 120–126. DOI: 10.1109/MCOM.2013.6525604. URL: <https://ieeexplore.ieee.org/document/6525604> (ver p. 7).
- [PiB24] PiBorg. *Ultrasonic Distance Sensor (HC-SR04)*. <https://www.piborg.org/sensors-1136/hc-sr04>. Consultado em 2024/07/01. 2024 (ver p. 46).
- [Pin14] Pedro Pinto. *iEcoSys: Sistema inteligente para gestão de resíduos*. <https://empresashoje.pt/high-tech/iecosys-sistema-inteligente-para-gestao-de-residuos/>. Consultado em 2024/05/20. 2014 (ver p. 17).
- [Pol24] Polulu. *Flammable Gas & Smoke Sensor MQ-2*. <https://www.pololu.com/product/1480>. Consultado em 2024/07/01. 2024 (ver p. 47).
- [Ram+21] Jino Ramson et al. «A LoRaWAN IoT enabled Trash Bin Level Monitoring System». Em: *IEEE Transactions on Industrial Informatics* PP (mai. de 2021), pp. 1–1. DOI: 10.1109/TII.2021.3078556. URL: <https://ieeexplore.ieee.org/document/9427265> (ver pp. 1, 12, 13).
- [Rei+15] Pedro Reis et al. «iEcoSys - An Intelligent Waste Management System». Em: *Advances in Intelligent Systems and Computing* 353 (jan. de 2015), pp. 843–853. DOI: 10.1007/978-3-319-16486-1_84. URL: https://link.springer.com/chapter/10.1007/978-3-319-16486-1_84 (ver p. 17).

- [Ris24] RisingStack. *Operating System Containers vs. Application Containers*. <https://blog.risingstack.com/operating-system-containers-vs-application-containers/>. Consultado em 2024/10/25. 2024 (ver p. 77).
- [Rod+20] Hélder Rodrigues et al. «Protótipo para recolha de informação de apoio à gestão de resíduos». Em: nov. de 2020. URL: https://www.researchgate.net/publication/346096553_Prototipo_para_recolha_de_informacao_de_apoio_a_gestao_de_residuos (ver p. 21).
- [Sch+11] Hans Schaffers et al. «Smart Cities and the Future Internet: Towards Cooperation Frameworks for Open Innovation». Em: vol. 6656. Mai. de 2011, pp. 431–446. ISBN: 978-3-642-20897-3. DOI: 10.1007/978-3-642-20898-0_31. URL: https://link.springer.com/chapter/10.1007/978-3-642-20898-0_31 (ver p. 7).
- [Sen24] Sensoneo. *Sensoneo Municipal Waste Collection*. <https://sensoneo.com/solutions/municipal-waste-collection/>. Consultado em 2024/06/01. 2024 (ver pp. 20, 21).
- [Sin23] SEMAS Sintra. *Sintra Sustentável: Gestão Inteligente de Resíduos*. <https://www.smas-sintra.pt/>. Consultado em 2024/08/01. 2023 (ver p. 9).
- [Sma24] SmartBin.io. *SmartBinIO*. <https://www.smartbin.io/>. Consultado em 2024/06/01. 2024 (ver pp. 19, 20).
- [SOM24] SOMA. *SOMA*. <https://www.soma.pt/index.php>. Consultado em 2024/06/01. 2024 (ver p. 22).
- [SYH17] Rashmi Sinha, Wei Yiqiao e Seung-Hoon Hwang. «A survey on LPWA technology: LoRa and NB-IoT». Em: *ICT Express* 3 (mar. de 2017). DOI: 10.1016/j.icte.2017.03.004. URL: https://www.researchgate.net/publication/315505158_A_survey_on_LPWA_technology_LoRa_and_NB-IoT (ver p. 80).
- [Sys19] Espressif Systems. *ESP32 Datasheet*. <https://pdf1.alldatasheet.com/datasheet-pdf/view/1148023/ESPRESSIF/ESP32.html>. Consultado em 2024/05/15. 2019 (ver pp. 70, 76).
- [Uni22] European Union. *Tempo de condução e períodos de repouso no setor dos transportes rodoviários*. <https://eur-lex.europa.eu/PT/legal-content/summary/driving-time-and-rest-periods-in-the-road-transport-sector.html>. Consultado em 2024/07/01. 2022 (ver p. 49).
- [UTA08] UTAD. *Smart Containers for domestic used oils collection*. https://smartcontainers.utad.pt/project_en.html. Consultado em 2024/05/20. 2008 (ver p. 16).
- [Vis23] Câmara Municipal de Viseu. *Viseu EcoCidade: Gestão Inteligente de Resíduos Urbanos*. <https://www.cm-viseu.pt/>. Consultado em 2024/08/01. 2023 (ver p. 9).

- [WH22] Aria Bisma Wahyutama e Mintae Hwang. «YOLO-Based Object Detection for Separate Collection of Recyclables and Capacity Monitoring of Trash Bins». Em: *Electronics* 11.9 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11091323. URL: <https://www.mdpi.com/2079-9292/11/9/1323> (ver p. 16).
- [Yaq+17] Ibrar Yaqoob et al. «Enabling Communication Technologies for Smart Cities». Em: *IEEE Communications Magazine* 55 (jan. de 2017). DOI: 10.1109/MCOM.2017.1600232CM. URL: <https://ieeexplore.ieee.org/document/7823347> (ver pp. 7, 8, 12, 80).
- [Zai+20] Muhammad Zar Zaid Harith et al. «Prototype Development of IoT Based Smart Waste Management System for Smart City». Em: *IOP Conference Series: Materials Science and Engineering* 884 (jul. de 2020), p. 012051. DOI: 10.1088/1757-899X/884/1/012051. URL: https://www.researchgate.net/publication/343109149_Prototype_Development_of_IoT_Based_Smart_Waste_Management_System_for_Smart_City (ver pp. 1, 13).
- [Zan+14] Andrea Zanella et al. «Internet of Things for Smart Cities». Em: *IEEE Internet of Things Journal* 1.1 (2014), pp. 22–32. DOI: 10.1109/JIOT.2014.2306328. URL: <https://ieeexplore.ieee.org/document/6740844> (ver p. 129).
- [Zhe18] Ltd Zhengzhou Winsen ElectronicsTechnology Co. *Flammable Gas Sensor Datasheet*. <https://pdf1.alldatasheet.com/datasheet-pdf/view/1304539/WINSEN/MQ-2.html>. Consultado em 2024/05/15. 2018 (ver p. 47).
- [Zyg12] Sotiris Zygiaris. «Smart City Reference Model: Assisting Planners to Conceptualize the Building of Smart City Innovation Ecosystems». Em: *Journal of the Knowledge Economy* 4 (jun. de 2012). DOI: 10.1007/s13132-012-0089-4. URL: <https://link.springer.com/article/10.1007/s13132-012-0089-4> (ver p. 6).

Anexos

Anexo I

Mockups

I.1 *Mockups* da plataforma *web*

Na figura I.1 é possível observar o *mockup* da página de efetuar *login*, onde os utilizadores iniciam sessão com o seu *email* e a sua *password*.

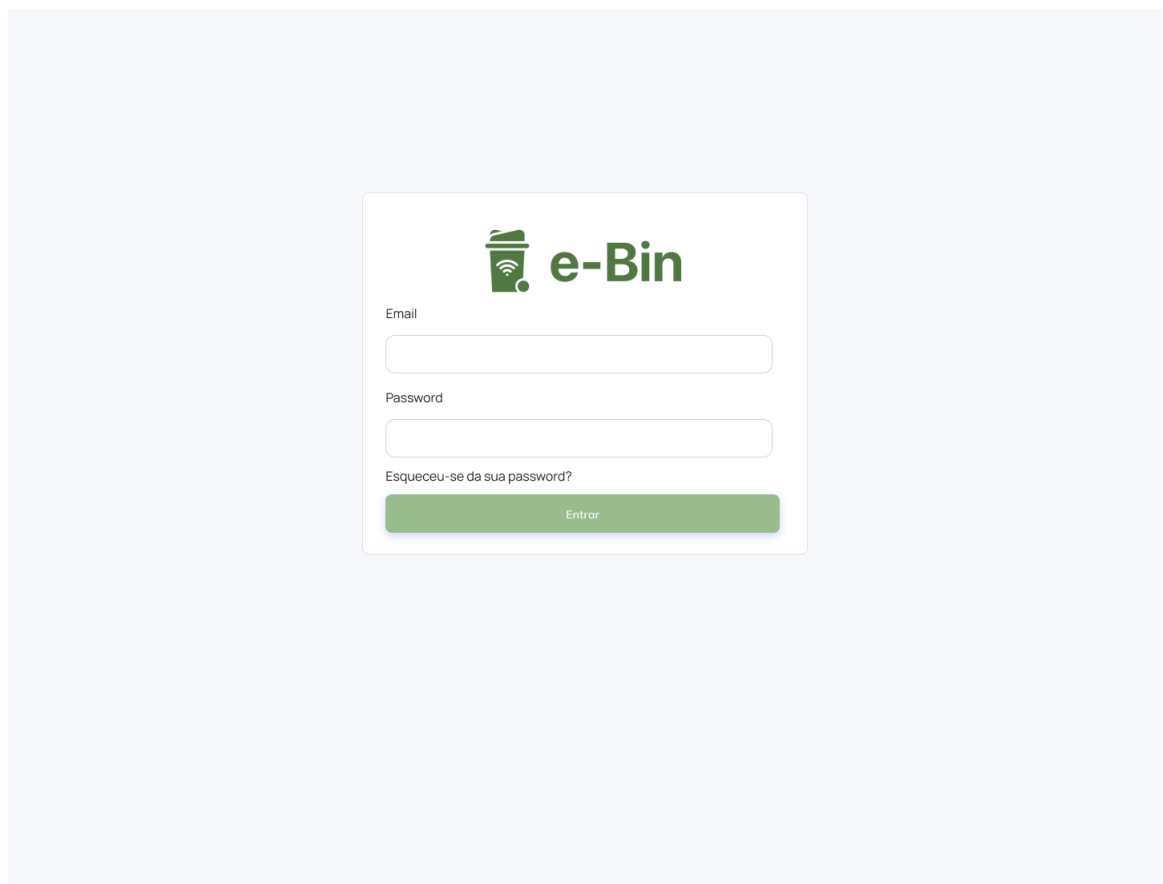


Figura I.1: *Mockup* de alta fidelidade da página de *password*.

A figura I.2 apresenta o *mockup* da página de reposição de *password*, ou seja, caso os utilizadores se esqueçam da *password*, poderão recuperá-la. Caso o *email* inserido seja válido, o utilizador recebe um *email* para repô-la.

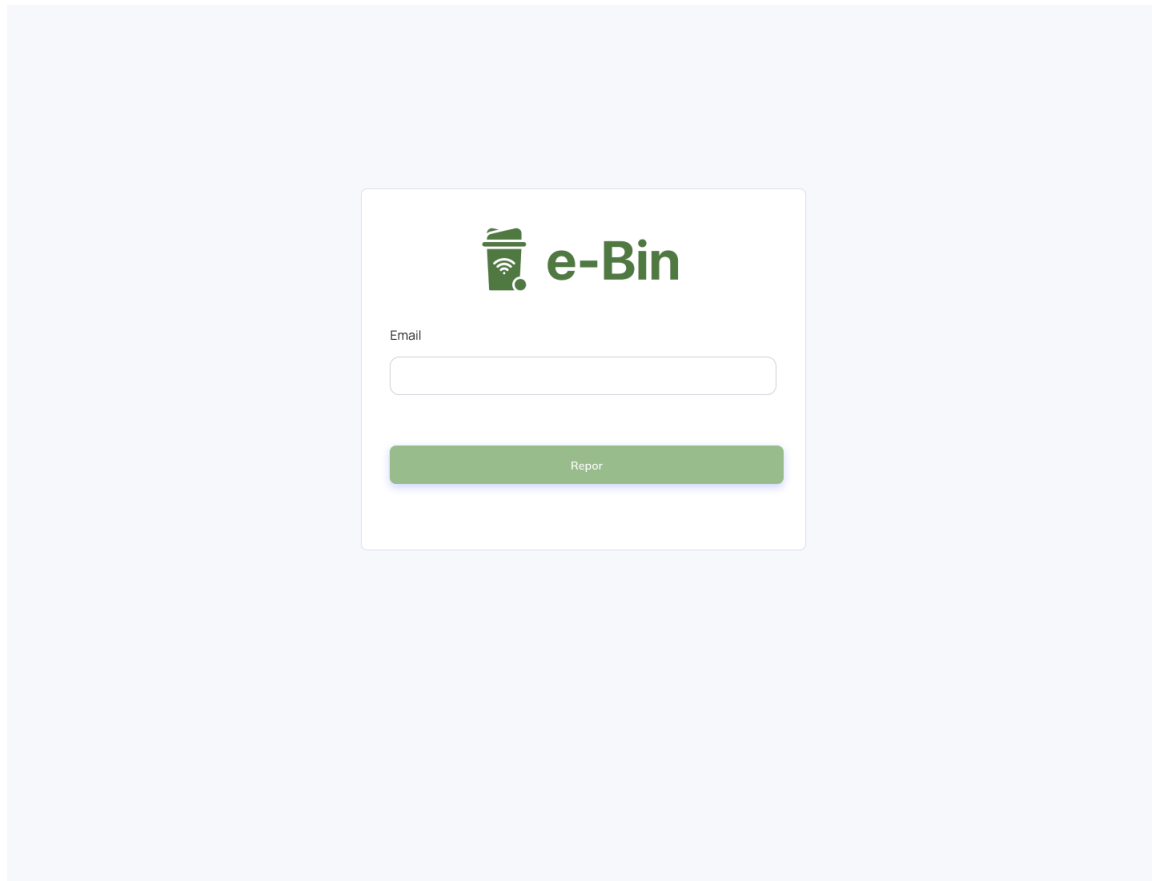
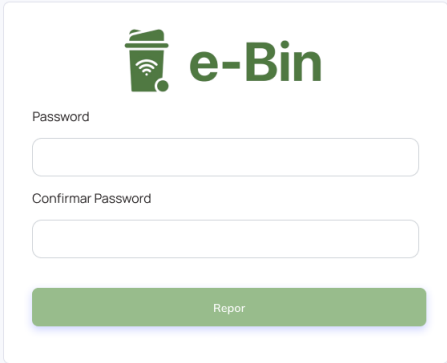


Figura I.2: *Mockup* de alta fidelidade da página de reposição de *password*.

Na figura I.3 é possível observar o *mockup* da página onde os utilizadores inserem a sua nova *password*.



The mockup shows a central white card on a light blue background. At the top of the card is the e-Bin logo, which consists of a green trash bin icon with a Wi-Fi symbol and the text 'e-Bin' in green. Below the logo, there are two input fields: the first is labeled 'Password' and the second is labeled 'Confirmar Password'. Both fields are empty and have a light gray border. At the bottom of the card is a green button with the text 'Repor' in white.

Figura I.3: *Mockup* de alta fidelidade da página de nova *password*.

I. MOCKUPS

A figura I.4 mostra o *mockup* da *dashboard* que é a primeira página apresentada após um *login* bem-sucedido. Nesta página é possível observar um mapa com a localização dos camiões, caixotes e outras informações sobre o sistema.

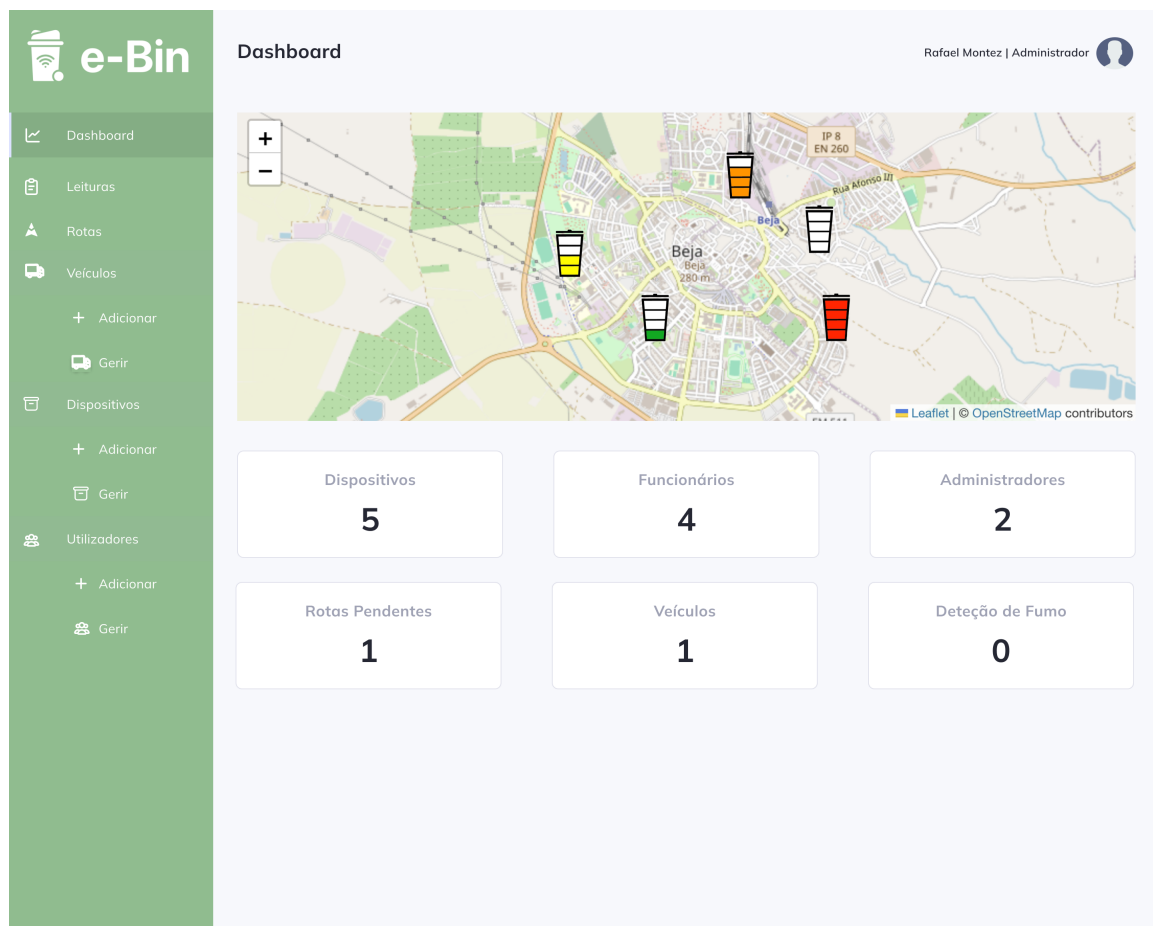


Figura I.4: *Mockup* de alta fidelidade da *dashboard*.

A figura I.5 corresponde ao *mockup* da página das leituras dos sensores de cada dispositivo.

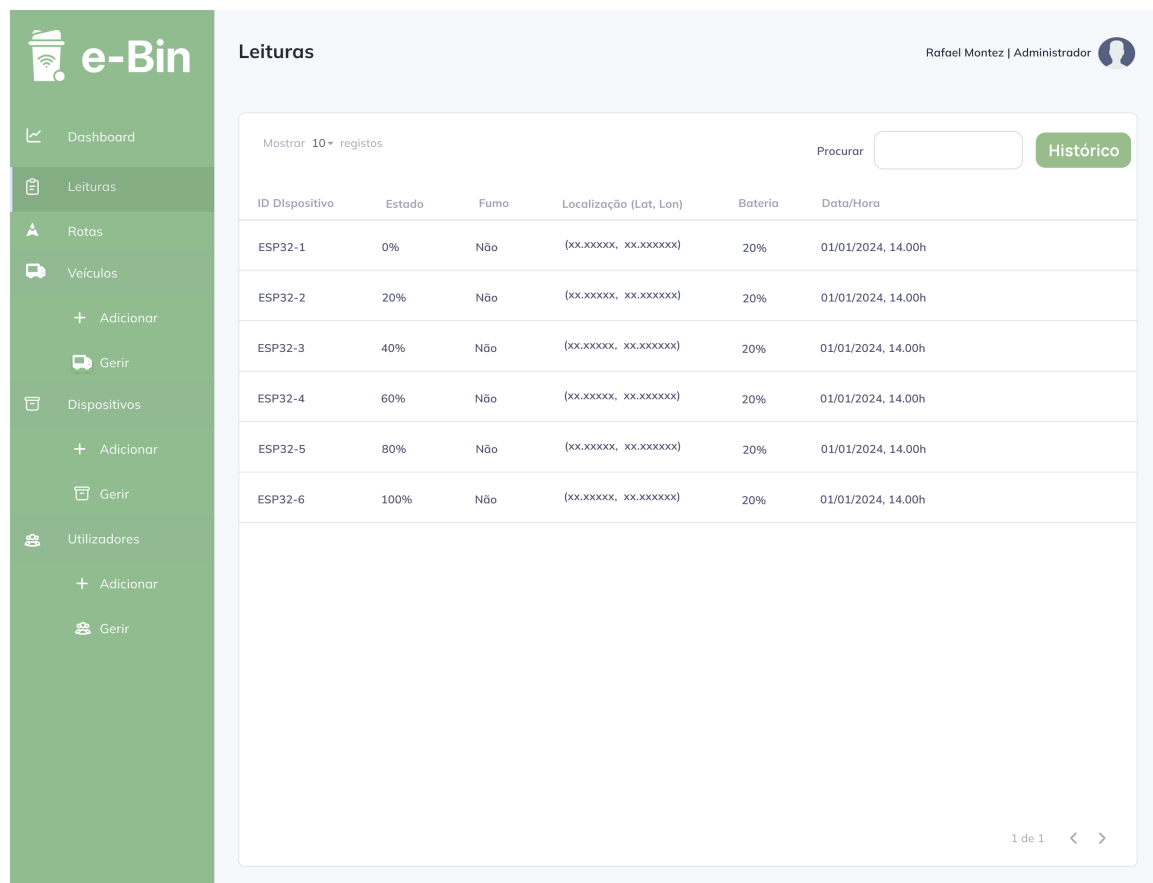


Figura I.5: *Mockup* de alta fidelidade da página de leituras.

I. MOCKUPS

Na figura I.6 é possível observar o *mockup* da página das rotas onde se pode visualizar quais já foram realizadas e quais ainda estão pendentes.

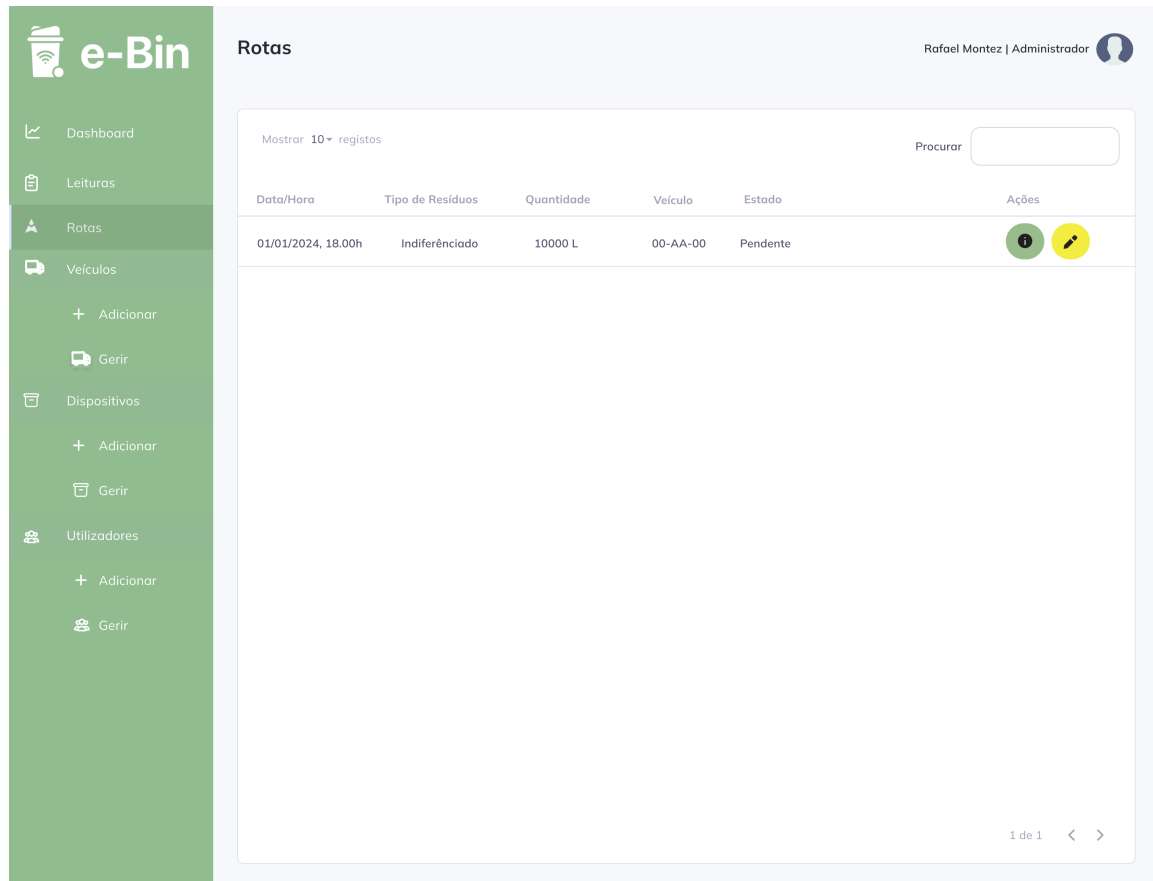


Figura I.6: *Mockup* de alta fidelidade da página de rotas.

Na página dos detalhes da rota é possível observar algumas informações correspondentes à rota selecionada como o tipo de resíduo, a distância, o total de paragens e o custo com o combustível, como se pode verificar no *mockup* da figura I.7.

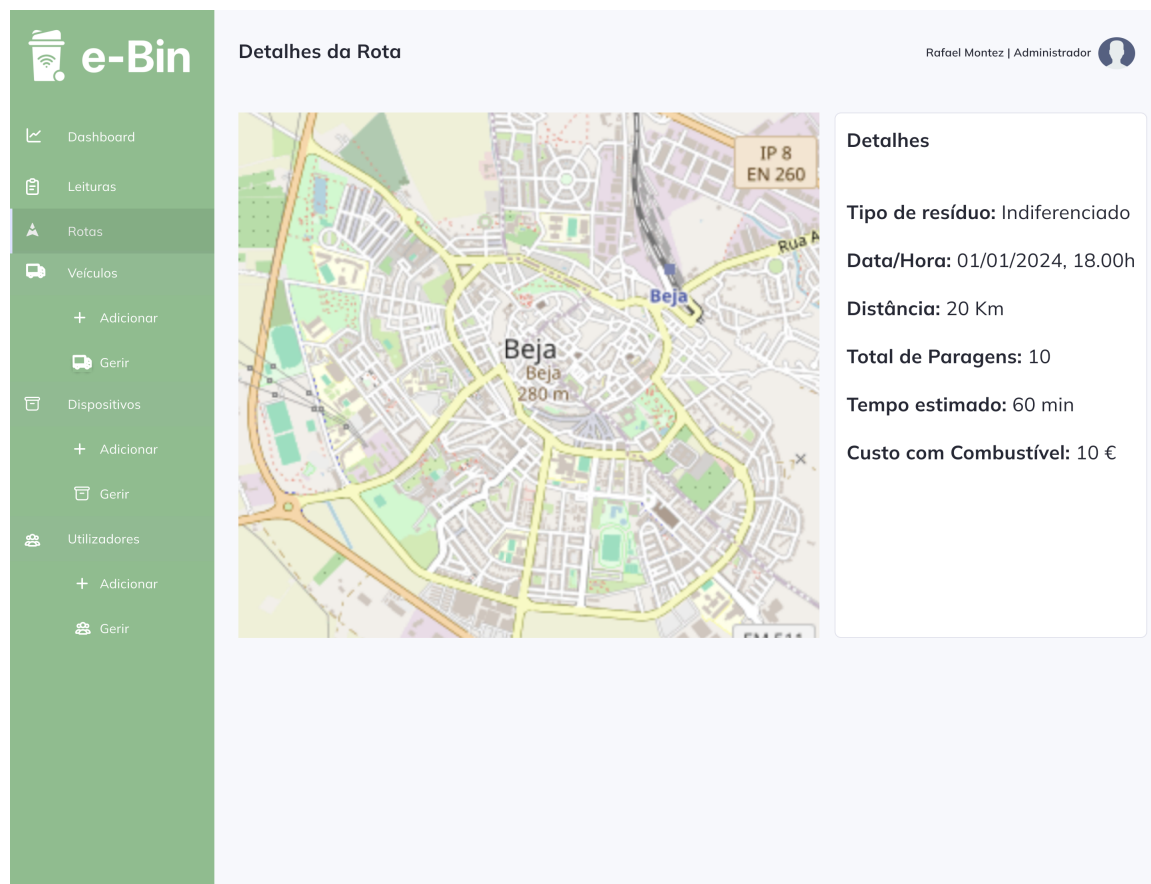


Figura I.7: Mockup de alta fidelidade da página de detalhes da rota.

I. MOCKUPS

Na figura I.8 é possível observar o *mockup* da página de editar rota, onde é possível alterar o funcionário, o veículo, a data e hora e o estado da mesma.

O mockup da página de edição de rota do sistema e-Bin apresenta uma interface com uma barra lateral verde à esquerda e uma área principal branca. A barra lateral contém o logotipo 'e-Bin' e um menu com as seguintes opções: Dashboard, Leituras, Rotas (destacada), Veículos, Dispositivos e Utilizadores. Cada opção principal possui sub-opções de '+ Adicionar' e 'Gerir'. No topo da área principal, o título 'Editar Rota' e o nome de usuário 'Rafael Montez | Administrador' com um ícone de perfil são exibidos. O formulário principal contém cinco campos de entrada rotulados como 'ID Funcionário*', 'Matrícula*', 'Data*', 'Hora*' e 'Estado*', todos obrigatórios. Abaixo dos campos, há dois botões: 'Atualizar' em verde e 'Voltar' em cinza.

Figura I.8: *Mockup* de alta fidelidade da página de editar rota.

A figura I.9 corresponde ao *mockup* da página de adicionar novos veículos.

The mockup shows a web application interface for adding a new vehicle. On the left is a green sidebar with the 'e-Bin' logo and a menu with options: Dashboard, Leituras, Rotas, Veículos (highlighted), and sub-options for each (e.g., '+ Adicionar', 'Gerir'). The main content area is titled 'Adicionar Veículo' and includes a user profile 'Rafael Montez | Administrador'. The form contains six required fields: 'Marca*', 'Modelo*', 'Capacidade (L)*', 'Consumo (l/100 Km)*', 'Tipo de Combustível*', 'Matricula*', and 'Mês/Ano*'. At the bottom are 'Adicionar' and 'Voltar' buttons.

 **e-Bin**

Dashboard

Leituras

Rotas

Veículos

+ Adicionar

Gerir

Dispositivos

+ Adicionar

Gerir

Utilizadores

+ Adicionar

Gerir

Adicionar Veículo

Rafael Montez | Administrador 

Marca*

Modelo*

Capacidade (L)*

Consumo (l/100 Km)*

Tipo de Combustível*

Matricula*

Mês/Ano*

Adicionar

Voltar

Figura I.9: *Mockup* de alta fidelidade da página de adicionar veículos.

I. MOCKUPS

Na figura I.10 é possível observar o *mockup* da página para gerir todos os veículos.

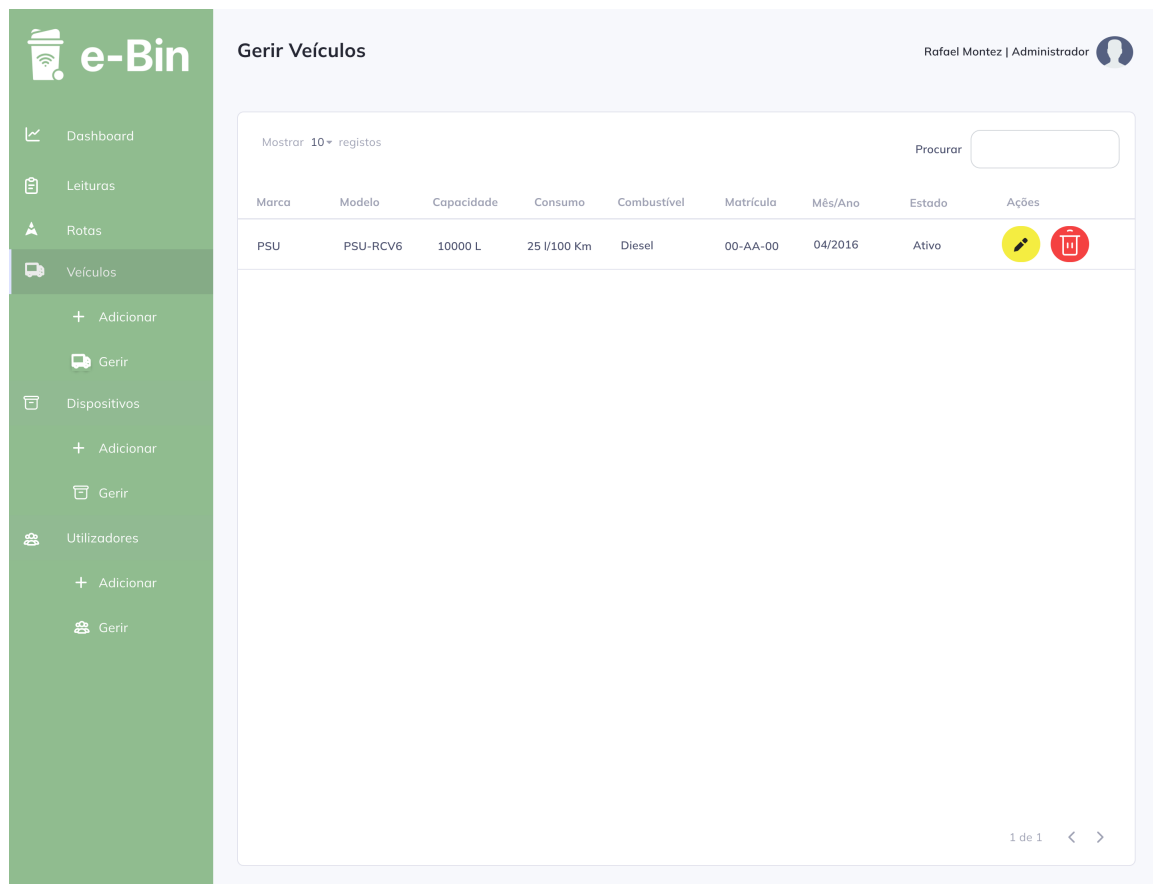


Figura I.10: *Mockup* de alta fidelidade da página para gerir veículos.

Na figura I.11 é possível observar o *mockup* da página de editar um veículo.

The image shows a web application interface for editing a vehicle. On the left is a green sidebar with the 'e-Bin' logo and a menu with options like Dashboard, Leituras, Rotas, Veículos, Dispositivos, and Utilizadores. The 'Veículos' option is selected. The main area is titled 'Editar Veículo' and shows a form with fields for Marca*, Modelo*, Capacidade (L)*, Consumo (l/100 Km)*, Tipo de Combustível*, Matricula*, and Mês/Ano*. At the bottom are 'Atualizar' and 'Voltar' buttons. The top right shows the user 'Rafael Montez | Administrador' with a profile icon.

e-Bin

Editar Veículo Rafael Montez | Administrador

Dashboard
Leituras
Rotas
Veículos
+ Adicionar
Gerir
Dispositivos
+ Adicionar
Gerir
Utilizadores
+ Adicionar
Gerir

Marca*

Modelo*

Capacidade (L)*

Consumo (l/100 Km)*

Tipo de Combustível*

Matricula*

Mês/Ano*

Atualizar Voltar

Figura I.11: *Mockup* de alta fidelidade da página de editar veículo.

I. MOCKUPS

Na figura I.12 é possível visualizar o *mockup* da página de adicionar novos dispositivos.

The mockup shows a web interface for adding a new device. On the left is a green sidebar with the 'e-Bin' logo and a menu containing: Dashboard, Leituras, Rotas, Veículos, + Adicionar, Gerir, Dispositivos (highlighted), + Adicionar, Gerir, Utilizadores, + Adicionar, and Gerir. The main content area is titled 'Adicionar Dispositivo' and shows the user 'Rafael Montez | Administrador' with a profile icon. The form includes three input fields: 'Tipo de Resíduos*', 'Capacidade (L)*', and 'Altura (cm)*'. At the bottom of the form are two buttons: a green 'Adicionar' button and a 'Voltar' link.

Figura I.12: *Mockup* de alta fidelidade da página de adicionar dispositivo.

A figura I.13 corresponde ao *mockup* da página para gerir todos os dispositivos.

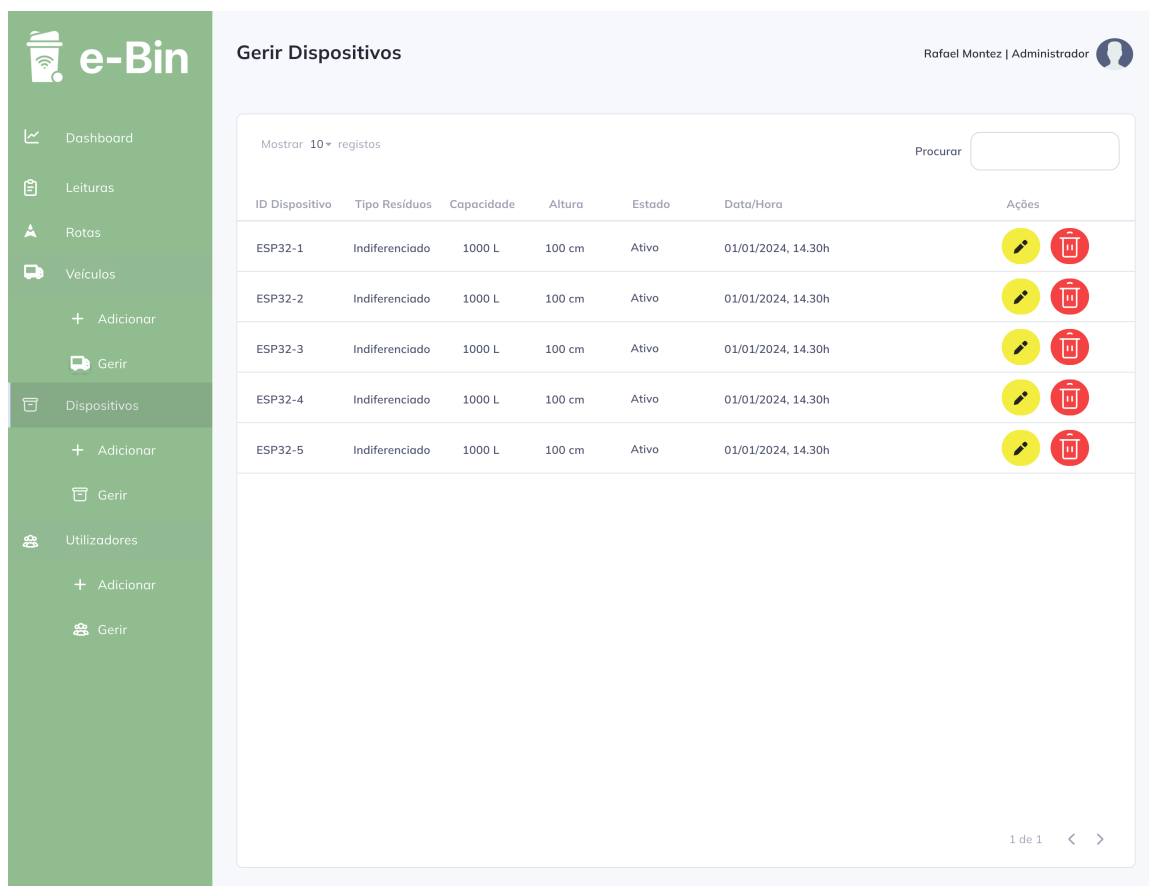


Figura I.13: *Mockup* de alta fidelidade da página para gerir dispositivos.

I. MOCKUPS

Na figura I.14 é possível observar o *mockup* da página de editar um dispositivo.

e-Bin

Editar Dispositivo

Rafael Montez | Administrador

ID Dispositivo*

Tipo de Resíduos*

Capacidade (L)*

Altura (cm)*

Ativo*

Atualizar Voltar

Figura I.14: *Mockup* de alta fidelidade da página de editar dispositivo.

Na figura I.15 é possível observar o *mockup* da página de adicionar um novo utilizador.

The mockup shows a web interface for adding a new user. On the left is a green sidebar with the 'e-Bin' logo and a menu with options: Dashboard, Leituras, Rotas, Veículos, + Adicionar, Gerir, Dispositivos, + Adicionar, Gerir, Utilizadores (highlighted), + Adicionar, and Gerir. The main content area is titled 'Adicionar Utilizador' and shows the user's role as 'Rafael Montez | Administrador'. The form includes fields for 'Nome*', 'Email*', 'Cargo*', and 'Receber alertas por email*'. At the bottom are 'Adicionar' and 'Voltar' buttons.

**e-Bin**

Dashboard

Leituras

Rotas

Veículos

+ Adicionar

Gerir

Dispositivos

+ Adicionar

Gerir

Utilizadores

+ Adicionar

Gerir

Adicionar Utilizador

Rafael Montez | Administrador 

Nome*

Email*

Cargo*

Receber alertas por email*

Adicionar

Voltar

Figura I.15: *Mockup* de alta fidelidade da página de adicionar utilizador.

157

I. MOCKUPS

Na figura I.16 é possível observar o *mockup* da página para gerir todos os utilizadores.

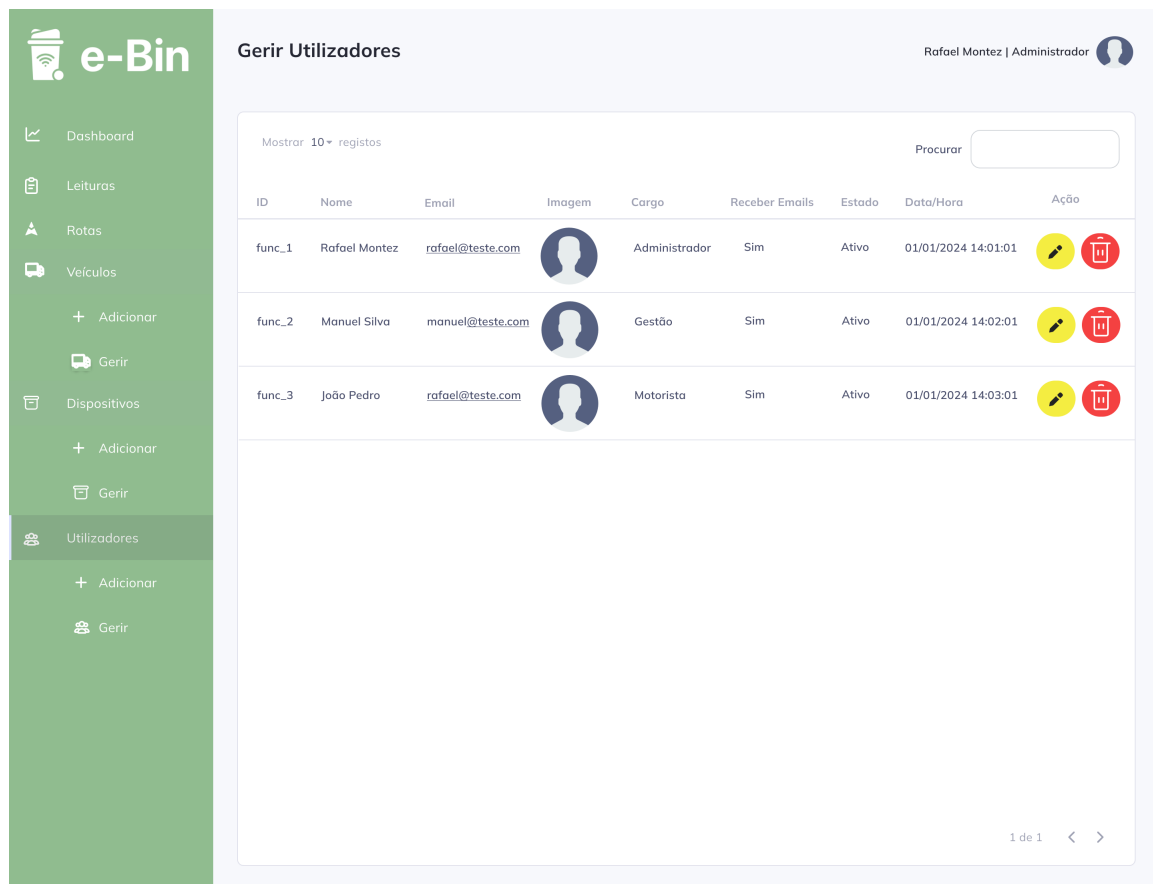


Figura I.16: *Mockup* de alta fidelidade da página para gerir utilizadores.

A figura I.17 corresponde ao *mockup* da página de editar um utilizador.

The mockup shows a web application interface for editing a user. On the left is a green sidebar with the 'e-Bin' logo and a menu containing: Dashboard, Leituras, Rotas, Veículos, + Adicionar, Gerir, Dispositivos, + Adicionar, Gerir, Utilizadores (highlighted), + Adicionar, and Gerir. The main content area is titled 'Editar Utilizador' and shows the user 'Rafael Montez | Administrador'. It contains two identical sets of form fields: 'Nome*', 'Email*', 'Password*', 'Confirmar Password*', 'Nome*', 'Password*', and 'Cargo*'. Each field is represented by a white input box with a light gray border. At the bottom of the form area are two buttons: 'Atualizar' (green) and 'Voltar' (gray).

Figura I.17: *Mockup* de alta fidelidade da página de editar utilizador.

I. MOCKUPS

A figura I.18 mostra o *mockup* da página de visualizar rotas. Nesta página, utilizadores com cargo de motorista não conseguem editar as rotas, como é possível verificar na figura I.18.

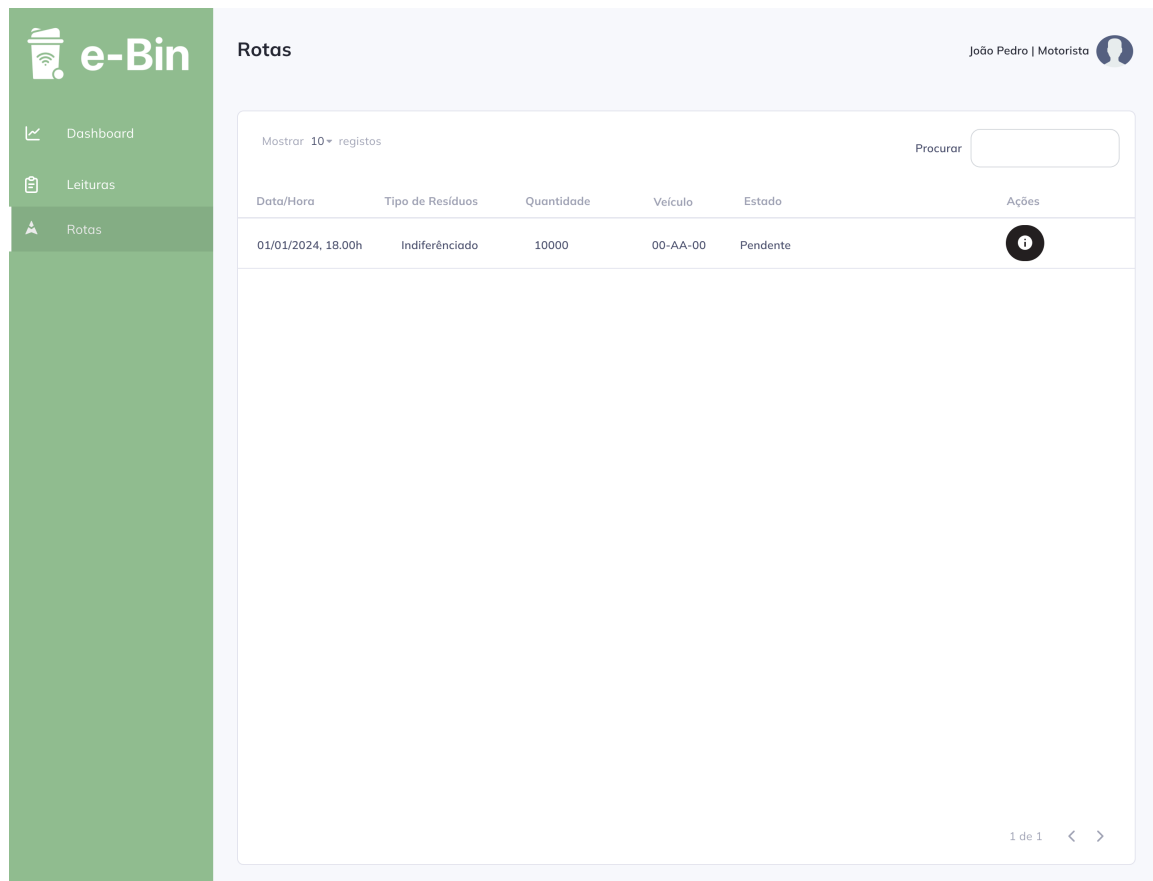


Figura I.18: *Mockup* de alta fidelidade da página de visualizar rotas.

I.2 *Mockups* da aplicação móvel

Nas figuras I.19, é possível observar, os *mockups* de média fidelidade da página de *password*, reposição de *password* para atender à parte de autenticação necessária na aplicação móvel.

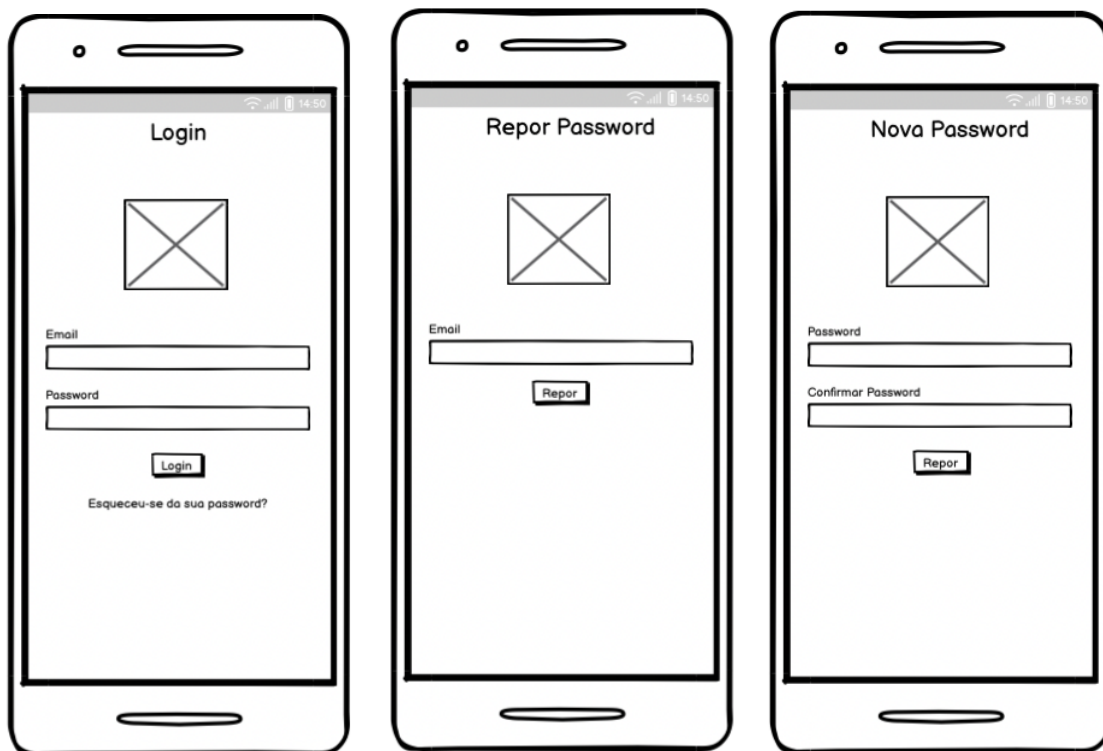


Figura I.19: *Mockups* de média fidelidade da aplicação móvel.

I. MOCKUPS

Nas figuras I.20, é possível observar, os *mockups* de alta fidelidade da página de *password*, reposição de *password* e nova *password*.

The figure displays three mobile application screens for the 'e-Bin' app, each with a green header bar and a status bar at the top showing the time as 9:41 and signal/battery icons.

- Login Screen:** Features the 'e-Bin' logo (a green trash can with a Wi-Fi symbol) at the top. Below it are two input fields labeled 'Email' and 'Password'. A green 'Login' button is positioned below the 'Password' field. At the bottom, there is a link that reads 'Esqueceu-se da sua password?'.
- Repor Password Screen:** Features the 'e-Bin' logo at the top. Below it is an input field labeled 'Email'. A green 'Repor' button is positioned below the 'Email' field.
- Nova Password Screen:** Features the 'e-Bin' logo at the top. Below it are two input fields labeled 'Password' and 'Confirmar Password'. A green 'Repor' button is positioned below the 'Confirmar Password' field.

Figura I.20: *Mockups* de alta fidelidade da aplicação móvel.

Nas figuras I.21, é possível observar os *mockups* de média fidelidade da página da *dashboard*, a página de visualizar e iniciar rotas e a página de visualizar detalhes das paragens.

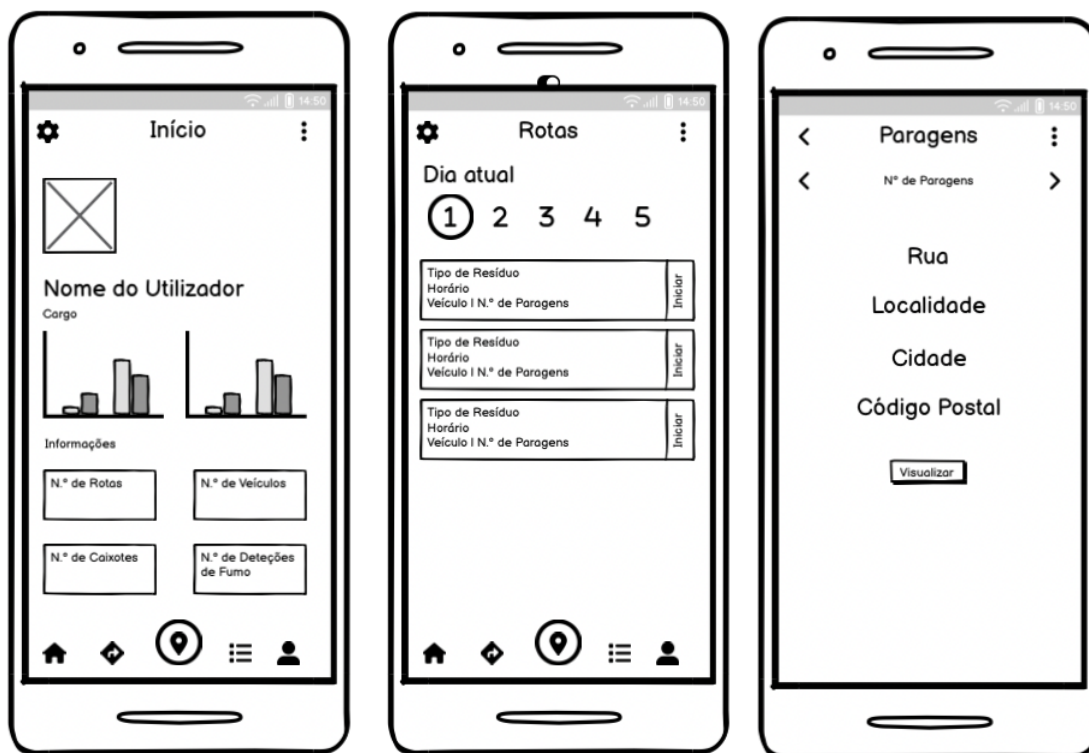


Figura I.21: *Mockups* de média fidelidade da aplicação móvel.

I. MOCKUPS

Nas figuras I.22, é possível observar os *mockups* de alta fidelidade da página da *dashboard*, a página de visualizar e iniciar rotas e a página de visualizar detalhes das paragens.

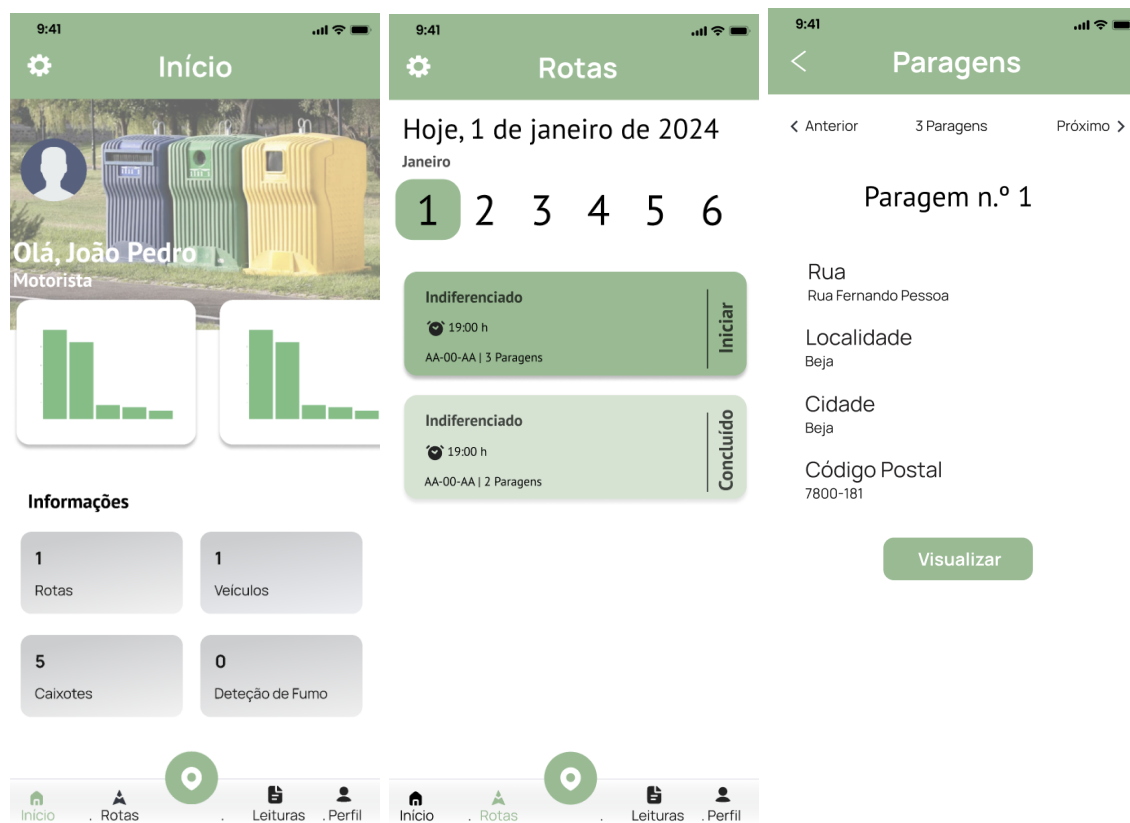


Figura I.22: *Mockups* de alta fidelidade da aplicação móvel.

Nas figuras I.23, é possível observar os *mockups* de média fidelidade da página com o mapa dos caixotes, a página de leituras e de editar perfil.

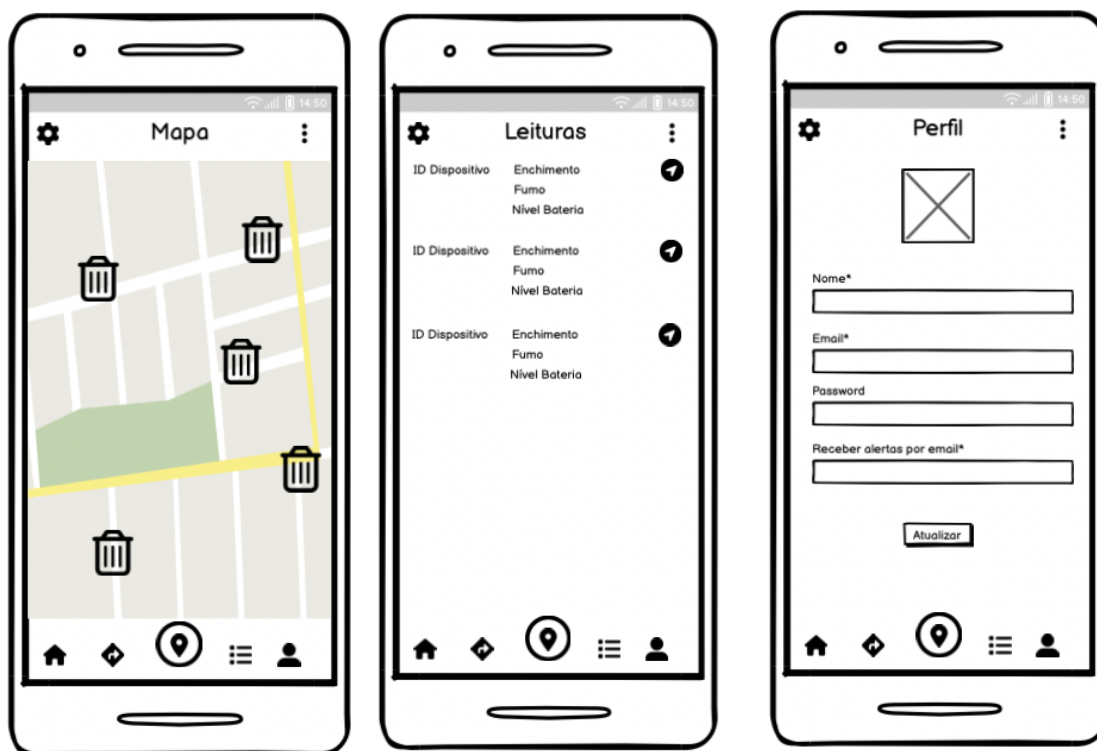


Figura I.23: *Mockups* de média fidelidade da aplicação móvel.

I. MOCKUPS

Nas figuras I.24, é possível observar os *mockups* de alta fidelidade da página com o mapa dos caixotes, a página de leituras e de editar perfil.

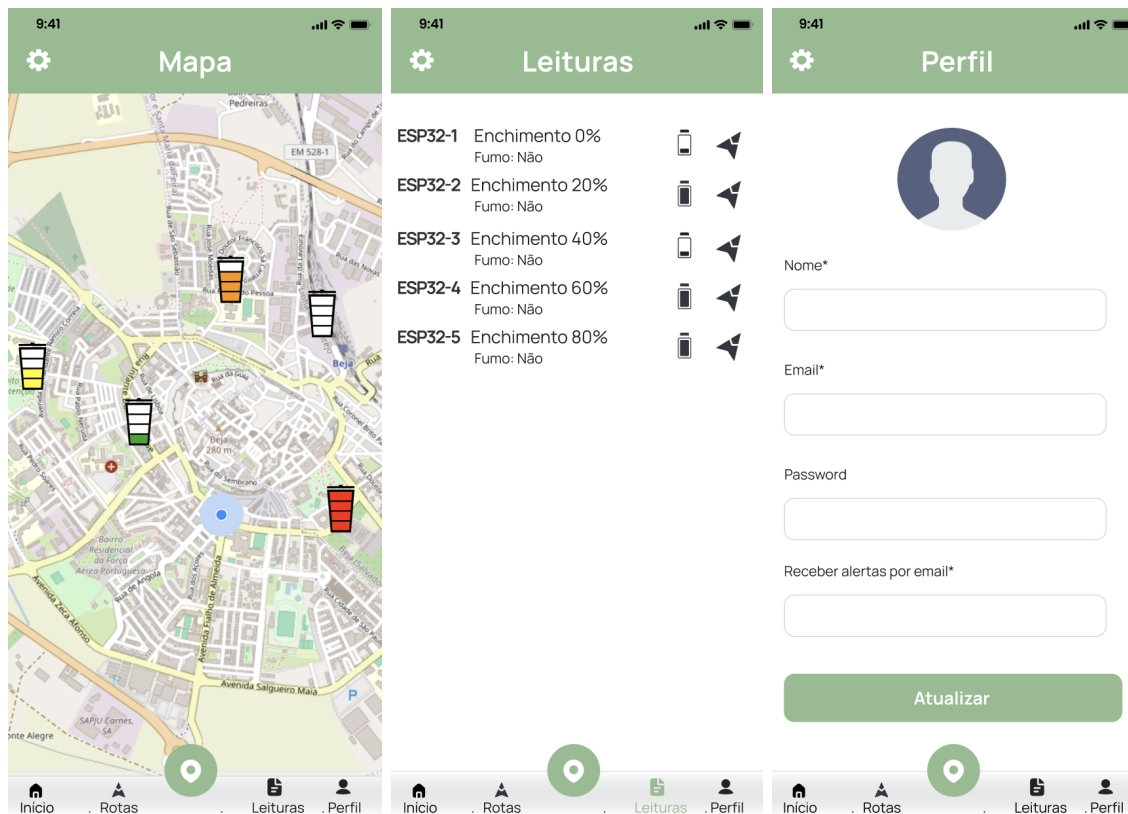


Figura I.24: *Mockups* de alta fidelidade da aplicação móvel.

Nas figuras I.25, é possível observar os *mockups* de média fidelidade das páginas exclusivas aos utilizadores com cargo de administrador ou gestão, ou seja, as páginas para gerir, adicionar e editar utilizadores.

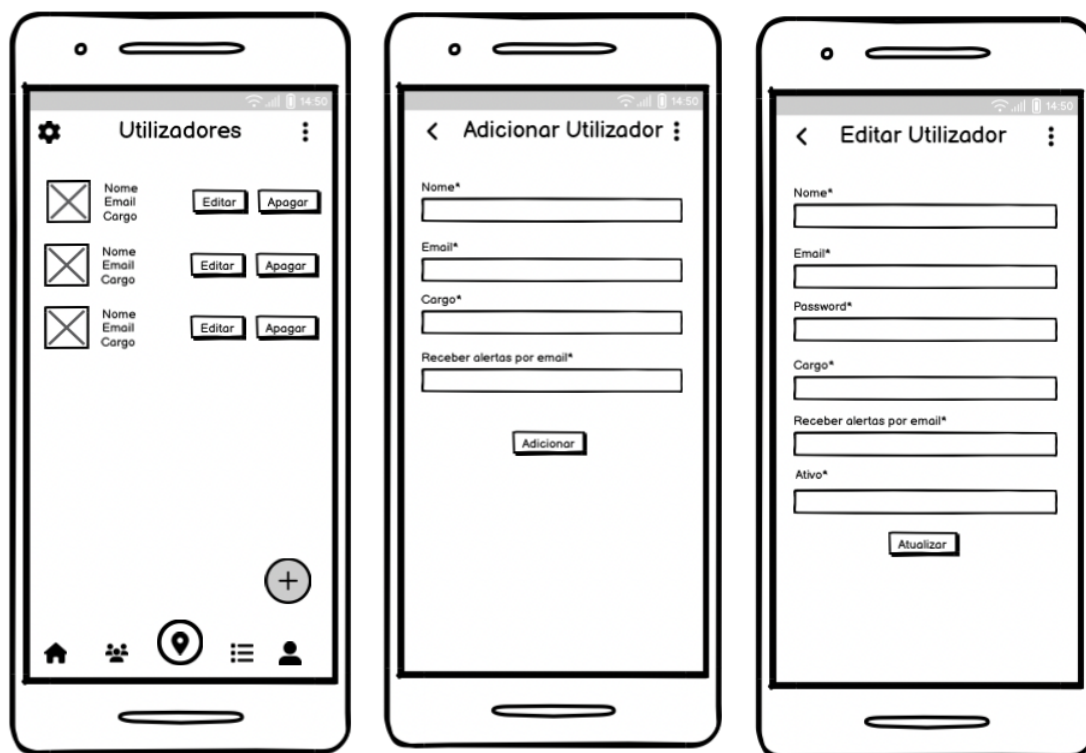


Figura I.25: *Mockups* de média fidelidade da aplicação móvel.

I. MOCKUPS

Nas figuras I.26, é possível observar os *mockups* de alta fidelidade das páginas de páginas para gerir utilizadores, adicionar utilizadores e editar utilizadores.

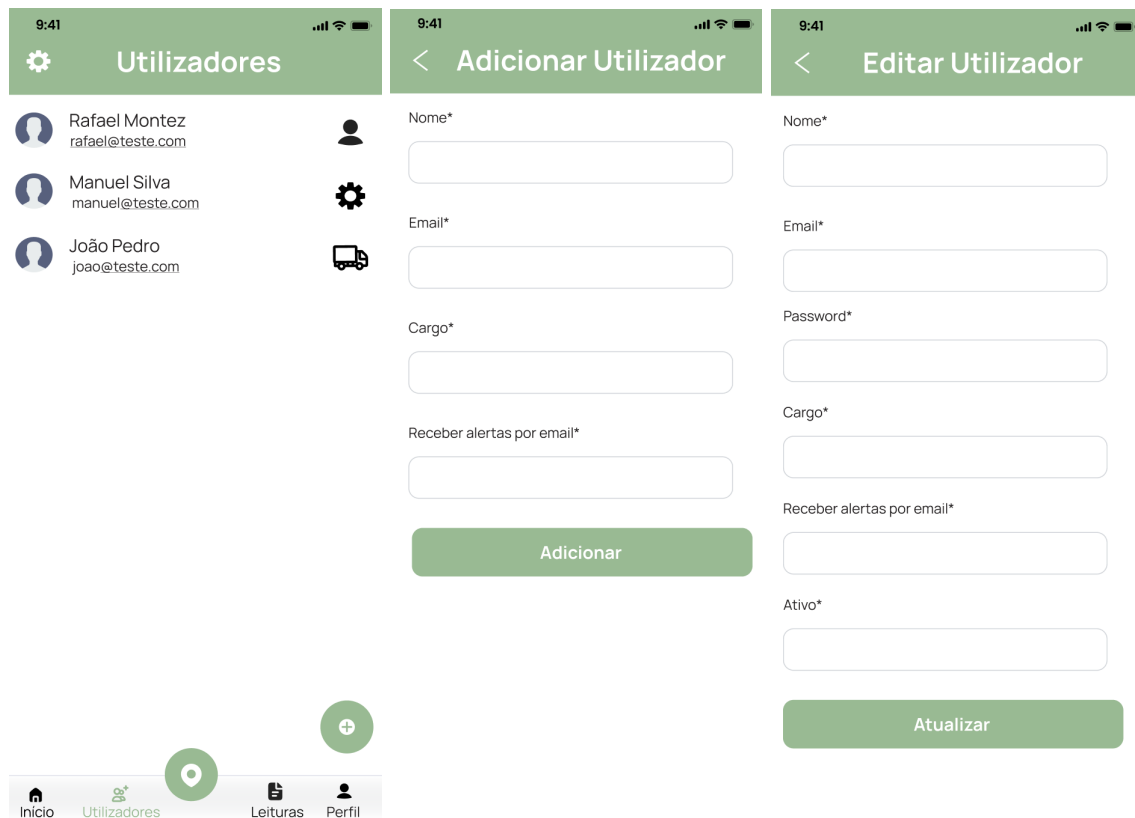


Figura I.26: *Mockups* de alta fidelidade da aplicação móvel.

Nas figuras I.27, é possível observar os *mockups* de média fidelidade das restantes páginas comuns a todos os cargos, ou seja, a página de configurações e a página sobre a aplicação.



Figura I.27: *Mockups* de média fidelidade da aplicação móvel.

I. MOCKUPS

Nas figuras I.28, é possível observar os *mockups* de alta fidelidade da página de configurações e a página sobre a aplicação.

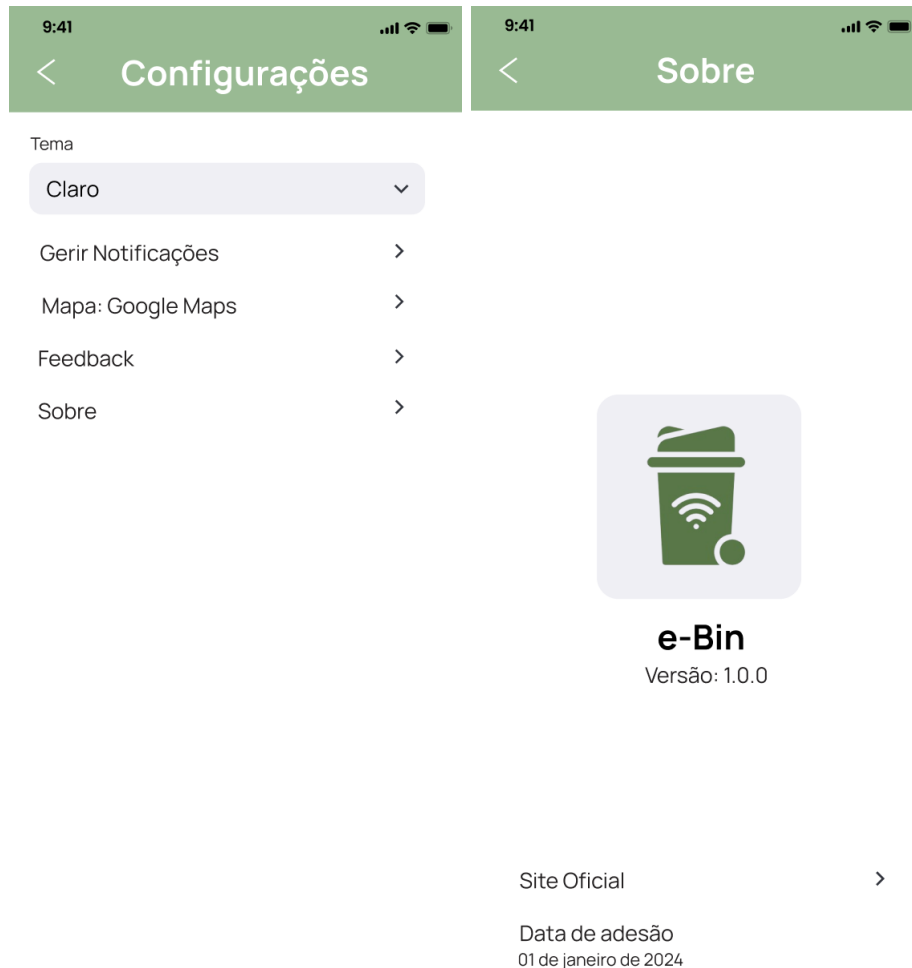


Figura I.28: *Mockups* de alta fidelidade da aplicação móvel.

Anexo II

Conferir *outputs* do procedimento computacional de VRP

Os *outputs* do procedimento computacional de VRP foram conferidos com dados conhecidos anteriormente, de modo a garantir que este opere conforme esperado.

A tabela II.1 corresponde à matriz de custos.

Tabela II.1: Matriz de custos.

	0	1	2	3	4	5
0	0	49	23	20	18	18
1	21	0	39	14	6	11
2	42	40	0	29	41	31
3	41	43	38	0	4	4
4	16	7	18	49	0	33
5	50	45	39	17	12	0

A tabela II.2 corresponde à procura dos clientes. A procura dos clientes neste contexto, é a capacidade dos caixotes.

Tabela II.2: Procura de clientes.

	Procura dos clientes
1	12
2	14
3	18
4	21
5	25

A capacidade máxima do veículo é 50.

Passo 1: Inicialização

Na tabela II.3 é possível verificar os custos de ir apenas a um caixote e voltar ao depósito (0-n-0).

Tabela II.3: Inicialização.

Rota	Custo
0-1-0	$c_{01}+c_{10}=49+21=70$
0-2-0	$c_{02}+c_{20}=23+42=65$
0-3-0	$c_{03}+c_{30}=20+41=61$
0-4-0	$c_{04}+c_{40}=18+16=34$
0-5-0	$c_{05}+c_{50}=18+50=68$

Custo total da solução inicial: $70+65+61+34+68=298$

Passo 2: Calcular as Poupanças

$$P_{12} = c_{12} - (c_{10} + c_{02}) = 39 - (21 + 23) = -5$$

$$P_{13} = c_{13} - (c_{10} + c_{03}) = 14 - (21 + 20) = -27$$

$$P_{14} = c_{14} - (c_{10} + c_{04}) = 6 - (21 + 18) = -33$$

$$P_{15} = c_{15} - (c_{10} + c_{05}) = 11 - (21 + 18) = -28$$

$$P_{23} = c_{23} - (c_{20} + c_{03}) = 29 - (42 + 20) = -33$$

$$P_{24} = c_{24} - (c_{20} + c_{04}) = 41 - (42 + 18) = -19$$

$$P_{25} = c_{25} - (c_{20} + c_{05}) = 31 - (42 + 18) = -29$$

$$P_{34} = c_{34} - (c_{30} + c_{04}) = 4 - (41 + 18) = -55$$

$$P_{35} = c_{35} - (c_{30} + c_{05}) = 4 - (41 + 18) = -55$$

$$P_{45} = c_{45} - (c_{40} + c_{05}) = 33 - (16 + 18) = -1$$

$$P_{21} = c_{21} - (c_{20} + c_{01}) = 40 - (42 + 49) = -51$$

$$P_{31} = c_{31} - (c_{30} + c_{01}) = 43 - (41 + 49) = -47$$

$$P_{41} = c_{41} - (c_{40} + c_{01}) = 7 - (16 + 49) = -58$$

$$P_{51} = c_{51} - (c_{50} + c_{01}) = 45 - (50 + 49) = -54$$

$$P_{32} = c_{32} - (c_{30} + c_{02}) = 38 - (41 + 23) = -26$$

$$P_{42} = c_{42} - (c_{40} + c_{02}) = 18 - (16 + 23) = -21$$

$$P_{52} = c_{52} - (c_{50} + c_{02}) = 39 - (50 + 23) = -34$$

$$P_{43} = c_{43} - (c_{40} + c_{03}) = 49 - (16 + 20) = 13$$

$$P_{53} = c_{53} - (c_{50} + c_{03}) = 17 - (50 + 20) = -53$$

$$P_{54} = c_{54} - (c_{50} + c_{04}) = 12 - (50 + 18) = -56$$

Passo 3: Ordenar as poupanças e fundir as rotas

$$P_{41} - 58 \quad 0 - 4 - 1 - 0 \quad \text{Capacidade} = 21 + 12 = 33$$

É possível fundir, porque $33 \leq 50$

$$P_{54} - 56$$

$$P_{34} - 55$$

$$P_{35} - 55 \quad 0 - 3 - 5 - 0$$

$$\text{Capacidade} = 18 + 25 = 43$$

É possível fundir, porque $43 \leq 50$

$$P_{51} - 54$$

$$P_{53} - 53$$

$$P_{21} - 51$$

$$P_{31} - 47$$

$$P_{52} - 34 \quad 0 - 5 - 2 - 0 \quad \text{fusão} \quad 0 - 3 - 5 - 2 - 0 \quad \text{Capacidade} = 18 + 25 + 14 = 57$$

Não é possível fundir, o sentido da rota permite, mas a capacidade não $57 \geq 50$

$$P_{14} - 33$$

$$P_{23} - 33$$

$$P_{25} - 29$$

$$P_{15} - 28$$

$$P_{13} - 27$$

$$P_{32} - 26$$

$$P_{42} - 21$$

$$P_{24} - 19 \quad 0 - 2 - 4 - 0 \quad \text{fusão} \quad 0 - 2 - 4 - 1 - 0 \quad \text{Capacidade} = 14 + 21 + 12 = 47$$

É possível fundir, o sentido da rota permite e a capacidade também

$$P_{12} - 5$$

$$P_{45} - 1$$

$$P_{43} - -13$$

Na figura II.1 é possível observar que tanto as poupanças como as rotas geradas pelo procedimento computacional VRP em *Python* coincidem com os dados conhecidos anteriormente.

```

4 -- 1 58
5 -- 4 56
3 -- 4 55
3 -- 5 55
5 -- 1 54
5 -- 3 53
2 -- 1 51
3 -- 1 47
5 -- 2 34
1 -- 4 33
2 -- 3 33
2 -- 5 29
1 -- 5 28
1 -- 3 27
3 -- 2 26
4 -- 2 21
2 -- 4 19
1 -- 2 5
4 -- 5 1
4 -- 3 -13

== == == Multi-start process == == ==
current sol 1: [[1], [3], [5], [2, 4]]
a better sol found
current sol 2: [[1], [2, 4], [3, 5]]
a better sol found
current sol 3: [[3, 5], [2, 4], [1]]
a better sol found

```

Figura II.1: Execução do procedimento computacional VRP.

A tabela II.4 mostra a rota otimizada utilizando a heurística de *Clarke and Wright*, com o respetivo custo e capacidade.

Tabela II.4: Melhores rotas.

Solução	Custo	Capacidade
0-3-5-0	$c_{02}+c_{24}+c_{41}+c_{10}=23+41+7+21=92$	47
0-2-4-1-0	$c_{03}+c_{35}+c_{50}=20+4+50=74$	43

Custo total: $92 + 74 = 166$

A figura II.2 mostra a solução final, que inclui as idas ao depósito, ou seja, os zeros nas extremidades.

```
== == == Result == == ==  
route: [[0, 3, 5, 0], [0, 2, 4, 1, 0]] total cost: 166
```

Figura II.2: *Output* do procedimento computacional VRP.

Anexo III

Consumo total do dispositivo

De forma a diminuir o consumo do dispositivo **e-Bin**, este fica ativo no máximo 30 segundos a cada hora (0.0083h), no restante tempo permanece no modo *deep-sleep* (ver mais informações na secção 5.5.3).

$$\frac{30 \text{ s}}{3600 \text{ s}} \approx 0.0083 \text{ h}$$

O consumo de cada componente durante 30 segundos ativo, é calculado da seguinte forma $mAh = I \times t$, onde I é a corrente em mA e t é o tempo em horas.

$$\text{Consumo TTGO} = 61 \text{ mA} \times 0.0083 \text{ h} = 0.5063 \text{ mAh}$$

$$\text{Consumo HC-SR04} = 15 \text{ mA} \times 0.0083 \text{ h} = 0.1245 \text{ mAh}$$

$$\text{Consumo NEO-6M} = 45 \text{ mA} \times 0.0083 \text{ h} = 0.3735 \text{ mAh}$$

$$\text{Consumo MQ-2} = 150 \text{ mA} \times 0.0083 \text{ h} = 1.245 \text{ mAh}$$

Durante 30 segundos o dispositivo **e-Bin** consome aproximadamente 2.25 mAh.

$$\text{Consumo ativo} = 0.5063 + 0.1245 + 0.3735 + 1.245 \approx 2.25 \text{ mAh}$$

O dispositivo vai estar 59 minutos e 30 segundos no modo *deep-sleep*, que corresponde a aproximadamente 0.9917h.

$$59\text{m} \times 60\text{s} = 3540 \text{ s}$$

III. CONSUMO TOTAL DO DISPOSITIVO

$$\frac{3540 \text{ s} + 30 \text{ s}}{3600 \text{ s}} = \frac{3570 \text{ s}}{3600 \text{ s}} \approx 0.9917 \text{ h}$$

Os sensores estão desligados durante o modo *deep-sleep*. O microcontrolador TTGO utilizado consome apenas 2mA neste modo.

Durante 59 minutos e 30 segundos o dispositivo **e-Bin** consome aproximadamente 1.98 mAh.

$$\text{Consumo deep-sleep} = 2 \text{ mA} \times 0.9917 \text{ h} \approx 1.9834 \text{ mAh}$$

Para calcular o consumo total por hora, é somado o consumo em 30 segundos ativo, com o consumo em 59 minutos e 30 segundos no modo *deep-sleep*. O dispositivo **e-Bin** consome no total aproximadamente 4.23 mAh.

$$\text{Consumo total} = 2.25 \text{ mAh} + 1.9834 \text{ mAh} \approx 4.2334 \text{ mAh}$$

Anexo IV

Custos relacionados

No presente momento da elaboração deste relatório o custo médio para produzir um dispositivo destes seria de aproximadamente 65 EUR com IVA de 23% incluído, sem contar os custos de mão-de-obra.

Ao consultar a loja *online* de componentes eletrónicos **Botnroll**, foi possível verificar que o microcontrolador **TTGO** utilizado neste projeto custa 24,90 EUR, o módulo GPS custa 11,50 EUR, o sensor de deteção de fumo/gás **MQ-2** custa 3,50 EUR, o carregador *micro usb* para baterias de lítio com proteção **TP4056** custa 1,90 EUR, o painel solar custa 4,75 EUR, a bateria de **LI-ION 18650 3.7V** de 2500 mAh tem um custo de 4,40 EUR. O restante valor corresponde aos restantes componentes utilizados como a Placa de Circuito Impresso, os *headers*, resistências, cabos, módulo de *Step-up*, suporte de bateria, interruptor de inclinação, transistor, bloco de terminais, caixa e parafusos.

A implementação destes dispositivos requer a utilização de *gateways* **LoRaWAN**, o número de *gateways* a utilizar depende do tamanho da área a cobrir. O *gateway* utilizado neste projeto, o **MikroTik LtAP LR8 LTE6** kit têm um custo de 244.99 EUR na loja **mBit**, porém existem diversas opções.