INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Passadeiras Inteligentes nas Smart Cities - Gestão Computacional incluindo Comunicação na Luz Visível

Tiago Pereira Ribeiro - a39172

Dissertação apresentada à Escola Superior de Tecnologia e Gestão para obtenção do Grau de Mestre em Informática

Trabalho realizado sob a orientação de:

Professora Doutora Luísa Maria Garcia Jorge

Professor Doutor João Paulo Coelho

Bragança
Outubro de 2024

Agradecimentos

Este trabalho foi apoiado por fundos NORTE - FEDER e pelo Horizonte 2020, no âmbito do projeto ”VALLPASS”, NORTE-01-0247-FEDER-113439. O autor agradece igualmente à Fundação para a Ciência e a Tecnologia (FCT, Portugal) pelo apoio financeiro através de fundos nacionais FCT/MCTES (PIDDAC) ao CeDRI (UIDB/05757/2020 e UIDP/05757/2020), SusTEC (LA/P/0007/2021) e INESC Coimbra (UIDB/00308/2020).

Resumo

A segurança dos peões em passadeiras urbanas permanece um desafio significativo, mesmo com as melhorias alcançadas nas últimas décadas. Dado o abrandamento na redução dos atropelamentos, torna-se imperativo explorar novas abordagens que integrem tecnologias emergentes no contexto das cidades inteligentes. Esta dissertação teve como objetivo principal desenvolver soluções inovadoras que reforcem a eficiência e a conectividade das infraestruturas urbanas, com foco no projeto VALLPASS e na tecnologia de Comunicação por Luz Visível (VLC).

No âmbito do projeto VALLPASS, foi implementada uma plataforma de gestão remota ”Powered by FIWARE”, sustentada por um modelo de dados semântico conforme a especificação NGSI-LD. Este trabalho contribuiu para a definição do modelo de dados, a conceção da arquitetura da plataforma e o desenvolvimento de uma aplicação *web* de gestão. Estas ferramentas permitem a monitorização e o controlo eficaz das passadeiras inteligentes, promovendo a interoperabilidade e integração com outras infraestruturas de cidades inteligentes.

Além disso, inspirado pela arquitetura de alto nível do sistema VALLPASS, investigou-se a viabilidade e a eficiência da VLC para comunicação direta entre postes de iluminação pública. Este estudo envolveu a criação de um cenário experimental, baseado no padrão IEEE 802.15.7, onde se validou a capacidade da VLC como uma solução complementar às tecnologias de comunicação tradicionais.

Os resultados obtidos comprovam a eficácia da plataforma de gestão remota e a relevância do modelo de dados semântico na gestão de infraestruturas urbanas. Adicionalmente, a investigação sobre a VLC evidenciou o seu potencial para melhorar a conectividade em ambientes urbanos densamente povoados. Conclui-se que a integração destas tecnologias contribui significativamente para a evolução das cidades inteligentes e para a segurança rodoviária, abrindo caminho para futuros desenvolvimentos e otimizações.

Palavras-chave: FIWARE, NGSI-LD, VLC, Comunicação por Luz Visível, Cidades Inteligentes, Passadeiras Inteligentes

Abstract

The safety of pedestrians at urban crosswalks remains a significant challenge, even with improvements achieved in recent decades. Given the slowdown in the reduction of pedestrian accidents, it is imperative to explore new approaches that integrate emerging technologies within the context of smart cities. The primary aim of this dissertation was to develop innovative solutions that enhance the efficiency and connectivity of urban infrastructures, focusing on the VALLPASS project and Visible Light Communication (VLC) technology.

Within the scope of the VALLPASS project, a remote management platform "Powered by FIWARE"was implemented, supported by a semantic data model compliant with the NGSI-LD specification. This work contributed to defining the data model, designing the platform architecture, and developing a web management application. These tools enable effective monitoring and control of smart crosswalks, promoting interoperability and integration with other smart city infrastructures.

Furthermore, inspired by the high-level architecture of the VALLPASS system, the feasibility and efficiency of VLC for direct communication between street lighting poles were investigated. This study involved creating an experimental scenario based on the IEEE 802.15.7 standard, which validated VLC's capability as a complementary solution to traditional communication technologies.

The results obtained confirm the effectiveness of the remote management platform and the relevance of the semantic data model in urban infrastructure management. Additionally, the research on VLC demonstrated its potential to improve connectivity in densely populated urban environments. It is concluded that integrating these technologies significantly contributes to the evolution of smart cities and road safety, paving the way for future developments and optimizations.

Keywords: FIWARE, NGSI-LD, VLC, Visible Light Communication, Smart Cities, Smart Crosswalks

Índice Geral

Lista de Siglas

API Application Programming Interface, ou Interface de Programação de Aplicações.

CRC Cyclic Redundancy Check, ou Verificação de Redundância Cíclica.

CRUD Create (criar), Read (ler), Update (atualizar), Delete (excluir).

DAL Linked Open Data, ou Dados Abertos Ligados.

DOM Document Object Model.

EFD End Frame Delimiter, ou Delimitador de Fim de Quadro.

ER Entidade-Relacionamento.

FAB Floating Action Button, ou Botão de Ação Flutuante.

FCS Frame Check Sequence, ou Sequência de Verificação de Quadro.

FEC Forward Error Correction, ou Correção de Erro Direta.

GD Digital Twin, ou Gémeo Digital.

GE Generic Enabler.

IDE Integrated Development Environment, ou Ambiente de Desenvolvimento Integrado.

IoT Internet of Things, ou Internet das Coisas.

JSX JavaScript XML.

LD Linked Data, ou Dados Ligados.

LDR Light Dependent Resistor, ou Resistor Dependente de Luz.

LDs Laser Diodes, ou Díodos *Laser*.

LED Light Emitting Diode, ou Díodo Emissor de Luz.

LoRaWAN Long Range Wide Area Network, ou Rede de Área Alargada de Alcance Longo.

LOS Line of Sight, ou Linha de Vista.

MAC Media Access Control, ou Controlo de Acesso ao Meio.

MFR MAC Footer, ou Rodapé MAC.

MHR MAC Header, ou Cabeçalho MAC.

MSDU MAC Service Data Unit, ou Unidade de Dados de Serviço MAC.

NLOS Non-Line of Sight, ou Fora da Linha de Vista.

OOK ON-OFF Keying.

OTA Over-The-Air.

OWC Optical Wireless Communication, ou Comunicação Ótica Sem Fios.

OWL Web Ontology Language.

OWPAN Optical Wireless Personal Area Network, ou Rede Pessoal Óptica Sem Fios.

PDP Ponto de Aplicação de Política.

PEP Policy Enforcement Point, ou Ponto de Decisão de Políticas.

PHY Physical Layer, ou Camada Física.

PLC Power Line Communication, ou Comunicação por Linha de Energia.

PPDU Physical Protocol Data Unit, ou Unidade de Dados de Protocolo Físico.

QOS Quality of Service, ou Qualidade de Serviço.

RDF Resource Description Framework.

RDS Relational Database Service.

REST Representational State Transfer, ou Transferência de Estado Representacional.

RF Radio Frequencies, ou Frequências de Rádio.

SDK Software Development Kit, ou *Kit* de Desenvolvimento de *Software*.

SDM Smart Data Model.

SFD Start Frame Delimiter, ou Delimitador de Início de Quadro.

SWT Semantic Web Technologies, ou Tecnologias da *Web* Semântica.

TTL Transistor-Transistor Logic, ou Lógica Transistor-Transistor.

UI User Interface, ou Interface de Utilizador.

URI Uniform Resource Identifier, ou Identificador Uniforme de Recursos.

URN Uniform Resource Name, ou Nome Uniforme de Recurso.

VLC Visible Light Communication, ou Comunicação por Luz Visível.

VPPM Variable Pulse Position Modulation, ou Modulação por Posição de Pulso Variável.

XACML eXtensible Access Control Markup Language.

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Capítulo 1 Introdução

1.1 Enquadramento

1.1.1 Evolução dos atropelamentos de peões e desafios futuros

A mobilidade é um dos fatores essenciais para o desenvolvimento humano, sendo a criação de condições mais seguras, eficientes e sustentáveis uma preocupação constante. Em ambiente urbano, as passadeiras desempenham um papel relevante na gestão do tráfego; contudo, representam um desafio significativo em termos de segurança rodoviária. Entre 2010 e 2018, foram registadas mais de 51.000 mortes de peões na Europa [5].

No gráfico apresentado na Fig. 1.1, pode-se observar o número total de peões atropelados (incluindo vítimas mortais) e, separadamente, o número de peões que perderam a vida em Portugal Continental entre 2000 e 2023. A evolução geral demonstra uma tendência positiva, refletindo o trabalho significativo desenvolvido pelas instituições governamentais para tornar as estradas mais seguras para os peões. No entanto, com 4.873 atropelamentos registados em 2023, é evidente que há ainda um longo caminho a percorrer e que mais pode e deve ser feito.

Além disso, e ainda com base na análise da mesma figura, constata-se que, nos últimos anos, a curva de redução tem vindo a abrandar. Entre 2000 e 2010, o número de atropelamentos diminuiu de 8.176 para 5.964, representando uma redução de aproximadamente 27%. Em contraste, entre 2010 e 2023, a diminuição foi mais modesta, passando de 5.964 para 4.873, correspondendo a apenas 18% de decréscimo. Esta alteração no ritmo de redução pode ser atribuída a vários fatores.

Uma explicação plausível é que a implementação inicial de infraestruturas de segurança rodoviária, como passadeiras elevadas e zonas de redução de velocidade, teve um impacto substancial na primeira fase, mas os benefícios dessas medidas tendem a estabilizar à medida que a sua eficácia se consolida. Adicionalmente, o aumento do tráfego rodoviário em áreas urbanas e a maior exposição dos peões ao fluxo de veículos podem estar a contrabalançar os efeitos positivos das melhorias implementadas. Assim, é natural concluir que se impõe uma nova abordagem estratégica para alcançar reduções

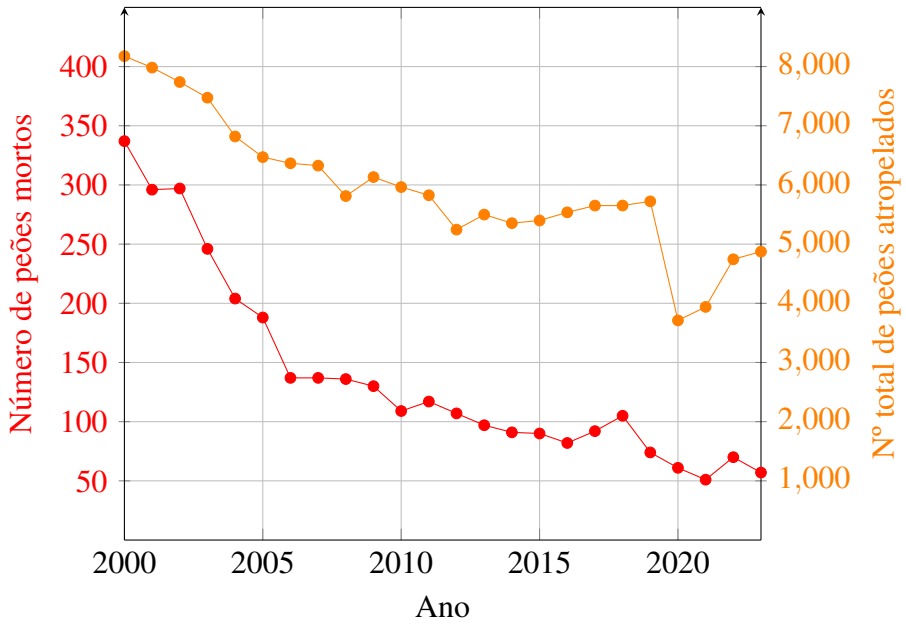

Figura 1.1: Peões atropelados: total e mortos - Portugal Continental (2000-2023)[1]

mais robustas, de modo a atingir o objetivo final: zero mortes.

1.1.2 O papel das Cidades Inteligentes na segurança rodoviária

O conceito de cidades inteligentes refere-se à utilização estratégica de tecnologias digitais e sistemas integrados para otimizar a gestão urbana, promovendo qualidade de vida, sustentabilidade e segurança [6, 7]. Estas cidades utilizam dados e sensores conectados para monitorizar e adaptar dinamicamente diversos aspetos do ambiente urbano, criando uma infraestrutura mais eficiente e responsiva [8]. Entre as várias áreas de atuação das cidades inteligentes, a mobilidade destaca-se como uma das mais relevantes, visando garantir deslocações mais seguras, rápidas e ambientalmente sustentáveis para todos os cidadãos [8].

No contexto da mobilidade, os sistemas de iluminação inteligente assumem uma função fundamental na segurança rodoviária, permitindo intervenções automatizadas e dinâmicas, como a adaptação da iluminação em função da presença de veículos e peões [9, 8]. García-Castellano et al., em [8], apontaram a iluminação rodoviária como uma das abordagens mais eficazes que um sistema de transporte inteligente pode implementar para reduzir a sinistralidade rodoviária. Estas soluções não apenas aumentam a visibilidade nas vias, mas também permitem alertar condutores sobre a presença de peões, especialmente em zonas de elevado risco, contribuindo assim para a redução de acidentes e a proteção dos utilizadores mais vulneráveis.

1.1.3 A VLC na infraestrutura de comunicação das cidades inteligentes

Um dos princípios basilares das cidades inteligentes é a existência de uma infraestrutura de comunicação robusta e eficiente [10], capaz de suportar a troca de informações em tempo real entre múltiplos dispositivos, mesmo em ambientes urbanos com elevada densidade populacional. Neste contexto, VLC surge como uma alternativa promissora às tecnologias tradicionais de comunicação sem fios, como as baseadas em Radio Frequencies, ou Frequências de Rádio (RF) [11, 12], especialmente em ambientes urbanos densamente povoados e com elevada procura por conectividade, onde o espectro de RF está a tornar-se insuficiente para lidar com o aumento massivo do tráfego [13].

A VLC permite a transmissão de dados através da modulação da intensidade de fontes de luz Light Emitting Diode, ou Díodo Emissor de Luz (LED), um recurso já amplamente presente em infraestruturas urbanas, como os sistemas de iluminação pública. Esta tecnologia pode aproveitar a infraestrutura de iluminação já existente [13, 14, 12], evitando a necessidade de novas instalações de cabos ou antenas dedicadas e promovendo um uso eficiente e sustentável dos recursos urbanos. Cumulativamente, ajuda a libertar o espectro de RF para outras aplicações melhorando a eficiência e a robustez da infraestrutura de comunicação das cidades inteligentes.

1.2 O projeto VALLPASS

O projeto VALLPASS - Vigilância Ativa e Inteligente com suporte LoRa para Passadeiras - foi um projeto de I&D que decorreu entre 2021 e 2023, financiado pelo programa Norte2020. Envolveu um consórcio liderado pela empresa Valled, com a colaboração do Instituto Politécnico de Bragança (IPB) e do laboratório colaborativo MORE. O objetivo principal foi desenvolver um sistema a ser implementado em passadeiras que garanta a travessia segura dos peões. Apresentam-se, de seguida, as principais funcionalidade e inovações tecnológicas que o caracterizam.

Autosuficiente em termos energéticos O sistema VALLPASS é independente da rede elétrica, gerando a sua própria energia através de painéis solares e utilizando técnicas inteligentes de gestão de energia, o que possibilita a sua instalação em praticamente qualquer local.

Autocomissionamento O sistema VALLPASS apenas necessita de ser colocado nas passadeiras, dispensando qualquer configuração por parte dos instaladores - instalação *plug-and-play*. Algoritmos baseados em técnicas de auto-organização coordenarão a sua ligação e operacionalização.

Utilização de técnicas de manutenção autónoma O sistema VALLPASS recorre a técnicas e algoritmos de manutenção autónoma para aumentar a sua resiliência e reduzir os custos operacionais.

Funcionalidades avançadas de segurança rodoviária As passadeiras de peões VALLPASS estão equipadas com um "túnel luminoso" para assegurar a travessia segura dos peões e um sistema de aviso eficiente direcionado aos condutores sempre que há peões a atravessar a passadeira.

Plataforma de gestão remota "Powered by FIWARE" Uma plataforma de gestão remota, baseada na especificação NGSI-LD, permite aos clientes gerir o sistema de forma eficiente a nível global, com enfoque na apresentação de telemetria relevante e estatísticas.

O sistema VALLPASS é composto por um conjunto de postes de iluminação autónomos em termos energéticos, distribuídos ao longo de áreas urbanas em passadeiras não controladas por semáforos. Em cada passadeira, são instalados dois postes, um em cada extremidade. Além de uma luminária LED de alta eficiência e de um sistema de gestão de energia, cada poste integra uma miríade de sensores incluindo um radar para medição da velocidade dos veículos. No que se refere à infraestrutura de transmissão de dados (telemetria), é utilizada uma arquitetura de rede Long Range Wide Area Network, ou Rede de Área Alargada de Alcance Longo (LoRaWAN).

1.3 Objetivos

Considerando a importância da gestão computacional e das tecnologias de VLC no contexto das cidades inteligentes, esta dissertação propõe-se a explorar e desenvolver soluções inovadoras que potenciem a eficiência e a conectividade das infraestruturas urbanas. Neste âmbito, foram estabelecidos os seguintes objetivos:

- Implementação de uma plataforma de gestão remota "*Powered* by FIWARE"para o projeto VALLPASS, suportada por um modelo de dados semântico orientado às cidades inteligentes.
- Definir e implementar um cenário de estudo que tem como objetivo investigar a viabilidade e a eficiência do uso de VLC para comunicação direta entre postes de iluminação pública, utilizando como referência o padrão IEEE 802.15.7 [4].

1.4 Contribuições científicas

Dentro âmbito do primeiro objetivo desta dissertação apresentado na Secção anterior, foi publicado um artigo intitulado "Modelling with NGSI-LD: the VALLPASS project case

study"[15], apresentado na 21st IEEE International Conference on Industrial Informatics (INDIN 2023), em Lemgo, Alemanha. Este artigo propôs uma metodologia para a criação de modelos de dados semânticos no contexto da Internet of Things, ou Internet das Coisas (IoT), com foco na representação de gémeos digitais. A metodologia foi apresentada de forma prática, ilustrada através do processo de construção de um modelo de dados semântico NGSI-LD para o projeto VALLPASS, inserido no domínio do tráfego, um dos mais relevantes no contexto das cidades inteligentes.

No contexto do segundo objetivo apresentado na Secção 1.3, encontra-se em fase de submissão um artigo adicional, descrevendo a implementação realizada e os seus resultados.

1.5 Estrutura do documento

Esta dissertação está organizada em 8 Capítulos - incluindo o presente. De seguida, apresenta-se uma descrição sucinta de cada um deles.

Capítulo 2 Estabelece os objetivos específicos desta dissertação e detalha a metodologia adotada para os alcançar.

Capítulo 3 Apresenta uma revisão bibliográfica abrangente dos principais conceitos e tecnologias relevantes para o projeto VALLPASS, incluindo a análise de trabalhos relacionados que contribuem para o avanço no domínio das passadeiras inteligentes e das soluções IoT em ambientes urbanos.

Capítulo 4 Introduz a VLC, discutindo os seus fundamentos teóricos, arquitetura básica e o padrão mais relevante - IEEE 802.15.7 [4]. São também revistos alguns trabalhos relacionados sobre a VLC com ênfase na sua integração com outras tecnologias de comunicação e na sua relevância na arquitetura de comunicação das cidades inteligentes.

Capítulo 5 Este Capítulo descreve a contribuição específica para o projeto VALLPASS, focando-se no desenvolvimento de uma plataforma de gestão remota apoiada por um modelo de dados semântico, direcionado para a aplicação em cidades inteligentes.

Capítulo 6 Dedicado à definição e implementação de um cenário de estudo que tem como objetivo investigar a viabilidade e a eficiência do uso da VLC para comunicação direta entre postes de iluminação pública afetos a sistemas de transportes inteligentes, tendo por base a arquitetura de alto nível do sistema VALLPASS.

Capítulo 7 Fornece uma visão abrangente das contribuições e inovações alcançadas, detalhando os métodos e resultados para cada objetivo específico da dissertação.

Capítulo 8 Resume os principais resultados da dissertação, destacando as contribuições para o projeto VALLPASS e o estudo da tecnologia VLC. São discutidas as limitações encontradas e propostas direções para trabalhos futuros que visam aprimorar e expandir as soluções desenvolvidas.

Capítulo 2 Objetivos e Metodologia de Investigação

2.1 Introdução

Neste Capítulo, apresentam-se os objetivos específicos desta dissertação no contexto do projeto VALLPASS, bem como a metodologia de investigação adotada para os alcançar. O principal objetivo é desenvolver uma plataforma de gestão remota orientada para cidades inteligentes, utilizando tecnologias emergentes como o *framework* FIWARE e explorando a integração de modelos de dados semânticos conforme a especificação NGSI-LD. Adicionalmente, propõe-se investigar a viabilidade da VLC como solução complementar à LoRaWAN para a comunicação entre postes de iluminação pública. A metodologia delineada segue uma abordagem iterativa e incremental, garantindo a contínua validação e ajustamento das soluções desenvolvidas ao longo do processo.

2.2 Contribuição para o projeto VALLPASS

2.2.1 Objetivos

O objetivo principal desta dissertação, no contexto do projeto VALLPASS, é desenvolver uma plataforma de gestão remota com uma abordagem orientada para cidades inteligentes, utilizando tecnologias emergentes. Os objetivos específicos são:

1. Definir o modelo de dados semântico VALLPASS orientado às cidades inteligentes
    - Desenvolver um modelo de dados semântico, conforme a especificação NGSI-LD, que permita a interoperabilidade e a integração do sistema de passadeiras em cidades inteligentes.
2. Definir a arquitetura e especificação da plataforma de gestão remota
    - Este objetivo consiste em delinear uma arquitetura segura, flexível e baseada em código aberto, utilizando o *framework* FIWARE (”*Powered by FIWARE*”).

Esta arquitetura facilitará a gestão remota de passadeiras e a integração com outras infraestruturas e sistemas de cidades inteligentes.

3. Prototipar a plataforma de gestão remota

    - Criar um protótipo funcional para validar tanto a arquitetura proposta como o modelo de dados, além de apoiar o desenvolvimento da Aplicação *Web*.

4. Desenvolver a Aplicação *Web* de gestão

    - Desenvolver uma Aplicação *Web* eficiente para a gestão das passadeiras, com uma interface intuitiva que permita monitorizar dados em tempo real, configurar o sistema e gerir telemetria.

2.2.2 Metodologia de Investigação

A metodologia de investigação adotada nesta dissertação, no contexto do projeto VALL-PASS, está organizada em etapas que correspondem diretamente aos objetivos específicos estabelecidos. Será utilizada uma abordagem iterativa e incremental ao longo do desenvolvimento, permitindo melhorias e ajustamentos contínuos à plataforma, de acordo com o feedback obtido durante os testes e validação. Cada etapa utiliza métodos e técnicas apropriados para alcançar os resultados pretendidos, conforme descrito abaixo.

Análise de requisitos

Na fase inicial, os requisitos do sistema são identificados e especificados, abrangendo tanto os requisitos funcionais quanto os não funcionais. As atividades envolvidas incluem:

- Caracterização dos utilizadores
    - Identificar os tipos de utilizadores do sistema e compreender as suas necessidades e expectativas.
- Especificação dos requisitos funcionais e não funcionais
    - Documentar detalhadamente as funcionalidades que o sistema deve proporcionar e estabelecer critérios de desempenho, segurança, usabilidade e compatibilidade.
- Elaboração do diagrama de casos de uso
    - Desenvolver um diagrama que ilustre as interações entre os utilizadores e o sistema, sintetizando os requisitos funcionais identificados.

Desenvolvimento do modelo conceptual de dados

Com base nos requisitos definidos, é elaborado o modelo conceptual de dados do sistema VALLPASS. Esta etapa envolve:

- Criação do diagrama ER
  - Identificar as entidades principais do sistema e as relações entre elas, representando-as num diagrama ER para visualizar a estrutura de dados.
- Descrição das entidades
  - Detalhar cada entidade identificada, especificando os seus atributos e relacionamentos, para compreender a estrutura necessária para a implementação do sistema.

Definição do modelo de dados semântico NGSI-LD

Nesta etapa, é desenvolvido o modelo de dados semântico NGSI-LD para o sistema VALLPASS, visando garantir a interoperabilidade com outras infraestruturas de cidades inteligentes. As atividades incluem:

- Estudo da especificação NGSI-LD
  - Compreender a estrutura e os princípios do NGSI-LD para orientar adequadamente o desenvolvimento do modelo de dados. Identificar boas práticas para a criação de modelos semânticos em sistemas de cidades inteligentes.
- Seleção de modelos de dados de referência
  - Pesquisar e identificar modelos de dados semânticos existentes que possam ser reutilizados ou adaptados para o sistema VALLPASS. Priorizar a reutilização de estruturas semânticas já aplicáveis.
- Ajuste dos modelos de dados de referência
  - Adaptar os modelos de dados selecionados às necessidades específicas do sistema, adicionando ou modificando entidades e atributos conforme necessário.

Definição da arquitetura da plataforma de gestão Remota

Com o modelo de dados definido, procede-se à conceção de uma arquitetura segura, flexível e suportada por código aberto, utilizando o *framework* FIWARE. As atividades envolvidas são:

- Análise do *framework* FIWARE
  - Compreender a arquitetura geral do FIWARE, estudando os seus GEs relevantes para o projeto, como o Orion Context Broker.
- Desenho da arquitetura
  - Definir os componentes da plataforma, as interações entre eles e os fluxos de dados, com especial ênfase na segurança e flexibilidade da solução.

Prototipagem da plataforma de gestão remota

Para validar a arquitetura proposta e o modelo de dados desenvolvido, é criada uma prototipagem da plataforma. O processo é altamente iterativo, permitindo ajustes e validações contínuas. As atividades incluem:

- Implementação e configuração dos serviços
  - Implementar os serviços definidos na arquitetura, configurando-os adequadamente para assegurar o correto funcionamento.
- Simulação do sistema VALLPASS
  - Criar um cenário de simulação que represente o comportamento do sistema VALLPASS, incluindo dispositivos relevantes, para validar a comunicação entre componentes e preparar o ambiente para testes.

Desenvolvimento da Aplicação *Web* de gestão

Nesta etapa, será desenvolvida a Aplicação *Web* que permitirá a gestão eficiente das passadeiras. As atividades incluem:

- Desenvolvimento do *back-end*
  - Analisar a necessidade de implementar um *back-end* para garantir a eficiência na comunicação entre o *front-end* e os serviços da plataforma FIWARE. Caso seja necessário, implementar a lógica de negócio e integrar a aplicação com a plataforma FIWARE.
- Desenvolvimento do *front-end*
  - Desenvolver a UI, com base nas representações de média fidelidade (*mockups*) previamente elaboradas fora do âmbito desta dissertação, garantindo uma experiência intuitiva para a monitorização de dados em tempo real, a configuração do sistema e a gestão de telemetria.

- Testes e Validação
  - Realizar testes funcionais e de usabilidade para garantir que a aplicação cumpre os requisitos e oferece uma experiência satisfatória.

Integração e Validação Final

Após o desenvolvimento dos componentes individuais, procede-se à integração completa do sistema, seguida de uma validação abrangente com base no cenário de simulação criado aquando da prototipagem da plataforma de gestão remota. As atividades incluem:

- Integração de Componentes
  - Unificar todos os módulos desenvolvidos, nomeadamente a Aplicação *Web* e os restantes serviços da plataforma de gestão remota, garantindo a comunicação eficiente entre todos os componentes da plataforma.
- Testes de Integração
  - Verificar o comportamento do sistema como um todo, corrigindo possíveis inconsistências.

2.3 VLC

2.3.1 Motivação e objetivo para o estudo da VLC

O projeto VALLPASS, no que concerne à infraestrutura de comunicação, e tal como apresentado na Secção 2.2, prevê a utilização da tecnologia LoRaWAN. No entanto, fora do âmbito imediato do projeto, mas tendo como base a sua arquitetura de alto nível, propõe-se explorar a VLC como uma solução complementar à LoRaWAN. O objetivo é investigar a viabilidade e a eficiência do uso de VLC para comunicação direta entre postes de iluminação pública, utilizando como referência o padrão IEEE 802.15.7 [4].

Para este fim, será simulado um cenário em que, numa passadeira equipada com dois postes de iluminação (colocados em lados opostos), apenas um dos postes possui conectividade LoRaWAN, funcionando como *gateway* ao nível local e com ligação direta à nuvem. O segundo poste, designado como poste secundário, comunicaria exclusivamente através da VLC com o *gateway*, de forma que todas as comunicações - tanto as provenientes do poste secundário para a nuvem, como as enviadas da nuvem para o poste secundário - teriam de ser encaminhadas pelo *gateway*. Esta alteração na arquitetura pode trazer várias vantagens em cenários reais, tais como:

- Simplificação da infraestrutura de comunicação;

- Redução dos custos globais do sistema VALLPASS;
- Aumento da eficiência energética e operacional do sistema;
- Libertação do espectro de RF para outras aplicações.

2.3.2 Metodologia de Investigação

Para investigar a viabilidade da VLC como uma solução complementar para a comunicação direta entre postes de iluminação pública, será adotada uma metodologia dividida em várias etapas, contemplando uma análise teórica e experimental. A metodologia é estruturada da seguinte forma:

1. Revisão bibliográfica:
    - Fundamentos teóricos da VLC;
    - Padrão IEEE 802.15.7 [4];
    - Estudos de caso de integração entre a VLC e outras tecnologias de comunicação (com ênfase nas cidades inteligentes).
2. Definição do cenário experimental, incluindo:
    - Arquitetura física
    - Arquitetura de comunicação suportada pelo padrão IEEE 802.15.7 [4]
3. Implementação do cenário experimental;
4. Análise do desempenho e resultados obtidos no cenário experimental;
5. Conclusões e perspetivas futuras.

2.4 Conclusão

Neste Capítulo, foram definidos os objetivos específicos da dissertação e detalhada a metodologia de investigação a ser seguida. Através da definição clara dos objetivos - desde a conceção da arquitetura da plataforma até à exploração da VLC - estabeleceu-se um plano estruturado para orientar o desenvolvimento do trabalho. A metodologia proposta, com etapas bem delineadas e uma abordagem iterativa, assegura que as soluções desenvolvidas serão continuamente validadas e aprimoradas. Este enquadramento metodológico constitui a base para os Capítulos seguintes, onde serão apresentados os resultados obtidos e as contribuições efetivas para o projeto VALLPASS e para o domínio das cidades inteligentes.

Capítulo 3 Revisão Bibliográfica - projeto VALLPASS

3.1 Introdução

Este Capítulo apresenta uma revisão bibliográfica no contexto do projeto VALLPASS. Inicialmente, discute-se a arquitetura da IoT e o papel do *middleware* na facilitação da interoperabilidade entre dispositivos e aplicações. Em seguida, são exploradas as plataformas IoT, detalhando os seus requisitos funcionais e não funcionais, assim como a sua classificação com base nesses critérios.

O conceito de Digital Twins, ou Gémeos Digitais (GDs) é introduzido, com destaque para a sua aplicação em cidades inteligentes, sublinhando a importância desses sistemas na gestão eficiente dos ambientes urbanos. A interoperabilidade no contexto da IoT é discutida, com ênfase no papel das Semantic Web Technologies, ou Tecnologias da *Web* Semântica (SWT) e Linked Data, ou Dados Ligados (LD) na integração de sistemas e dados de forma eficiente.

Apresenta-se o *framework* FIWARE como uma solução de código aberto que suporta o desenvolvimento de aplicações inteligentes. A seguir, é introduzido o Keyrock, componente responsável pela gestão de identidades e segurança na plataforma FIWARE. Adicionalmente, são abordadas tecnologias e ferramentas complementares, como Docker, Node-RED, Express.js, React e Material UI, fundamentais para o desenvolvimento de aplicações escaláveis e modulares.

Por fim, são analisados trabalhos relacionados que contribuíram para o avanço do estado da arte no domínio das passadeiras inteligentes e das soluções IoT em contextos urbanos.

3.2 Arquitetura da IoT e o papel do *middleware*

3.2.1 Introdução

A IoT tem revolucionado a forma como interagimos com dispositivos e sistemas, permitindo a comunicação e troca de dados entre uma variedade de ”coisas”ligadas entre si [16]. Com o crescimento assombroso de dispositivos IoT [17], é oportuno para os programadores de aplicações IoT compreender a arquitetura da IoT e o papel fundamental que o *middleware* desempenha nas interações entre os dispositivos IoT e as aplicações associadas a estes.

3.2.2 Arquitetura da IoT

Não obstante existirem vários modelos de arquitetura propostos para a IoT, nomeadamente no que ao nível de camadas diz respeito, optou-se por apresentar o modelo de três camadas neste trabalho devido à sua simplicidade e ampla aceitação na literatura [18]. Este modelo, simplificado, facilita a compreensão dos elementos essenciais da IoT, incluindo as suas funções e como eles se relacionam entre si.

O modelo de três camadas consiste nas seguintes camadas:

Camada de percepção É nesta camada onde habitam as ”coisas”, que ligam o mundo físico ao virtual, através da recolha de dados - sensores - e atuação no ambiente - atuadores (tomada inteligente, por exemplo);

Camada de rede Esta camada, intermédia, é responsável pela transmissão de dados bidirecional entre as camadas de perceção e aplicação;

Camada de aplicação Fornece serviços e soluções específicas aos utilizadores, em vários domínios (como as cidades inteligentes), através do processamento dos dados oriundos da camada de perceção.

3.2.3 O papel crucial do *middleware* na IoT

A existência de uma grande diversidade dispositivos, protocolos e tecnologias na IoT materializa uma das suas características mais desafiantes - heterogeneidade - que torna a tão desejada interoperabilidade (um dos pilares da IoT) difícil de alcançar. Neste contexto, torna-se extremamente útil a existência de uma camada de *software* que facilite a interoperabilidade e simplifique o desenvolvimento de aplicações [19, 20] - o *middleware*.

O *middleware*, de forma geral e no contexto da engenharia de *software*, pode ser definido como um *software* que atua como uma camada intermediária entre diferentes

sistemas, aplicações e componentes com o objetivo de os aglutinar, permitindo a sua comunicação e interoperabilidade. No que diz respeito à IoT e tendo por base a arquitetura apresentada na Subsecção 3.2.2, o *middleware* atua como uma ponte entre a camada de perceção e a camada de aplicação, ”mascarando”a heterogeneidade e complexidade dos dispositivos sitos na camada de perceção e disponibilizando um ambiente de execução e desenvolvimento para a camada de aplicação [21], abstraindo as aplicações e os programadores da complexidade das ”coisas”. Em outras palavras, o *middleware* fornece interfaces, serviços e ferramentas que possibilitam e/ou facilitam a comunicação, gestão e a integração de dispositivos heterogéneos.

3.3 Plataformas IoT

3.3.1 Introdução

As plataformas IoT podem ser vistas como uma forma de middleware, com o objetivo de integrar dispositivos, redes e aplicações [22]. De seguida, apresentam-se os principais requisitos funcionais e não funcionais associados a estas plataformas, bem como uma classificação das mesmas, baseada nesses requisitos, oferecendo assim uma introdução e uma visão geral das plataformas IoT de uma forma sistemática.

3.3.2 Requisitos funcionais

Descoberta e gestão de dispositivos Traduz-se na capacidade da plataforma IoT identificar e integrar dispositivos de forma automática, nomeadamente detetar novos dispositivos que se conectem à rede, reconhecer as suas capacidades (e limitações), e garantir uma gestão e monitorização eficiente e contínua desses dispositivos, como, por exemplo, ajustar a atribuição de tarefas aos dispositivos com base no seu nível de bateria.

Gestão de código Uma extensão do requisito acima, descreve a capacidade de uma plataforma IoT gerir o *firmware* e *software* dos dispositivos de forma remota - Over-The-Air (OTA).

Gestão de dados Os dados são um dos pilares da IoT e são gerados em quantidades massivas pelos seus dispositivos. Atentando a esta premissa, este requisito representa a gestão eficaz dos dados gerados pelos dispositivos, incluindo a capacidade de os obter, processar, armazenar e analisar, ao mesmo tempo que se garante a sua integridade e segurança.

Gestão de eventos Assim como a IoT gera uma enorme quantidade de dados, também produz um número significativo de eventos, aos quais a plataforma IoT deve identificar e reagir de acordo com regras ou condições predefinidas. Note-se que a origem dos eventos

não se limita apenas aos dispositivos IoT, podendo também provir de serviços externos integrados, por exemplo. O envio de notificações e alertas também está incluído. Este requisito pode ser visto como uma extensão da gestão de dados [22].

3.3.3 Requisitos não funcionais

Interoperabilidade Como já exposto anteriormente (Subsecção 3.2.3) é o *middleware* um grande garante da interoperabilidade na IoT. Além da interoperabilidade sintática - formato e estrutura dos dados - é cada vez mais importante uma plataforma IoT suportar SWT, garantindo uma interpretação e compreensão inequívocas dos dados processados. A compatibilidade com um amplo leque de protocolos de comunicação IoT e a exposição de funcionalidades através de Application Programming Interfaces, ou Interfaces de Programação de Aplicações (APIs) são dois exemplos que visam satisfazer este requisito não funcional.

Segurança e privacidade Requisitos válidos em praticamente todo o *software*, assumem especial importância na IoT, não só por representarem um enorme desafio (considerando as características da IoT), mas também pela gravidade das consequências que podem resultar no caso de não serem satisfeitos, devido ao teor de alguns dados processados na IoT (dados médicos, por exemplo) e o enorme poder de computação distribuído proporcionado por um número massivo de dispositivos *iot*.

Escalabilidade A IoT, pelas suas características, é a antítese total de estabilidade e está associada a topologias de rede bastante dinâmicas, devido às ligações instáveis e ao nomadismo das ”coisas”. Além disso, o número de dispositivos IoT tende a aumentar [17] e, por extensão, a quantidade de dados gerados e que necessitam de ser processados. Neste contexto, uma plataforma IoT precisa de ser, necessariamente, escalável, e manter níveis de Quality of Service, ou Qualidade de Serviço (QOS) estáveis à medida que o tempo passa e mais dispositivos são provisionados [22].

Disponibilidade e resiliência Uma plataforma *iot* deve apresentar elevados níveis de disponibilidade, especialmente, em aplicações críticas, mostrando-se resiliente a falhas.

Tempo real Uma parte significativa de aplicações e serviços é suportada por dados em tempo real, estando por vezes, associada a aplicações críticas (como no domínio da saúde, por exemplo), onde decisões tomadas com atraso podem ter consequências graves. Neste sentido, é imperativo assegurar a entrega e o processamento dos dados em tempo real.

Flexibilidade A flexibilidade (assim como a adaptabilidade) são requisitos de suma importância face ao ritmo acelerado de inovação verificado na IoT. Uma plataforma IoT necessita de possuir uma boa capacidade de adaptação para absorver alterações a longo

prazo, ao mesmo tempo que demonstra flexibilidade para ajustes imediatos e de curto prazo [22].

3.3.4 Classificação

Embora sejam apresentadas na literatura várias classificações para as plataformas IoT [23, 21, 22], irá ser apresentada a classificação apresentada por da Cruz *et al.* em [22], que divide as plataformas IoT em três categorias conforme os requisitos que satisfaçam: gestão de dispositivos, desenvolvimento de aplicações e habilitação de aplicações.

Gestão de dispositivos Focadas na gestão de dispositivos e na otimização de recursos de rede [22]. No âmbito da gestão de dispositivos, oferecem diversas funcionalidades, como o provisionamento (configuração inicial), atualizações OTA e a monitorização do estado dos dispositivos. Uma característica essencial é a capacidade de provisionamento *plug-and-play*, isto é, com nenhuma ou pouca configuração manual [22], o que é fundamental devido à dinamicidade da IoT. Além disso, no que toca à otimização da rede, estas plataformas podem analisar as capacidades da infraestrutura e otimizar o seu desempenho, fornecendo ferramentas que facilitam a entrega de dados, a deteção de dispositivos e o diagnóstico da rede [22]. O Azure IoT Hub [24] e a AWS IoT Device Management [25] são dois exemplos deste tipo de plataformas.

Desenvolvimento de aplicações Orientadas ao desenvolvimento de aplicações IoT, estas plataformas podem oferecer ferramentas de desenvolvimento, Software Development Kits, ou *Kits* de Desenvolvimento de *Software* (SDKs) e até um Integrated Development Environment, ou Ambiente de Desenvolvimento Integrado (IDE) baseado na *cloud* [26]. Exemplos incluem o CHOReVOLUTION [27] e o M2MLabs [28] [22].

Habilitação de aplicações As plataformas afetas a esta categoria são as mais populares (e numerosas) e oferecem a maior abrangência de funcionalidades, frequentemente combinando características das duas categorias anteriores. Da Cruz *et all.* [22] descreve este tipo de plataformas, que são o expoente máximo do conceito de *middleware* no mundo da IoT, como plataformas com foco na permissão (*enablement*) e integração de aplicações externas, proporcionando meios para gerir e visualizar dados - o que acelera o desenvolvimento de aplicações e facilita a integração com sistemas empresariais - ao mesmo tempo que garantem a segurança dos dados dos utilizadores e permitem a troca de informações entre vários dispositivos e aplicações. Alguns exemplos incluem o ThingsBoard [29] e o FIWARE [30].

É importante salientar que as três categorias mencionadas anteriormente baseiam-se no foco de uma plataforma, não sendo mutualmente exclusivas. De facto, é comum que uma plataforma IoT combine características das três categorias.

3.4 GDs

3.4.1 Conceito e aplicação

Os GDs são representações virtuais de ativos, processos ou sistemas, com expressão no mundo físico, que replicam fielmente as suas características e comportamento no mundo digital [31, 32]. Criados através da integração de dados em tempo real, sensores IoT e algoritmos avançados, estes modelos digitais permitem monitorizar, simular e prever o desempenho de seus homólogos físicos [33, 34]. Uma vantagem significativa dos GDs reside na capacidade de testar diferentes cenários e otimizar processos sem interferir diretamente no mundo real, proporcionando uma visão mais aprofundada para a tomada de decisões informadas [34].

3.4.2 Relevância no contexto das cidades inteligentes

Quando aplicados ao contexto das cidades inteligentes, os GDs tornam-se ferramentas essenciais e sofisticadas para a gestão eficiente e integrada dos sistemas urbanos [35]. No domínio dos transportes inteligentes, por exemplo, um GD pode representar a rede de tráfego de uma cidade, incluindo estradas, veículos, semáforos e comportamento dos pedestres. Este modelo digital permite simular e prever fluxos de tráfego, identificar gargalos, avaliar impactos de intervenções urbanas e testar políticas de mobilidade sustentável, como novas rotas de transporte público ou zonas de baixa emissão.

3.5 Interoperabilidade na IoT: o papel das SWT e dos LD

3.5.1 Introdução

O conceito de cidades inteligentes, inicialmente teórico, evoluiu para uma realidade concreta, com o mercado a alcançar um valor estimado de 457 mil milhões de dólares em 2021 [36]. Não obstante, a definição do termo "cidade inteligente" continua a ser ambígua e depende amplamente do contexto e da perspetiva adotada [37]. Segundo Hall [38], uma cidade inteligente pode ser descrita como um ambiente urbano que monitoriza e integra as condições das suas infraestruturas críticas - como redes de transporte, comunicações e fornecimento de energia - de modo a otimizar recursos e serviços para os cidadãos. A monitorização de tantas e complexas infraestruturas gera, naturalmente, uma quantidade colossal de dados, contudo, como salientam Abella *et al.* em [39], o simples aumento da quantidade de dados recolhidos não implica, por si só, a geração de valor ou a melhoria dos serviços disponibilizados. A chave está na capacidade de agregar dados de diferentes origens e extrair informação significativa que possa ser transformada em conhecimento

útil. Para alcançar esse objetivo, é necessário garantir a integração de dois níveis distintos de interoperabilidade: a sintática, que assegura a uniformização dos formatos de dados, e a semântica, que proporciona um entendimento contextual partilhado [15].

No contexto da IoT, a heterogeneidade não se limita aos dados; envolve também uma ampla diversidade de hardware, protocolos e plataformas, tornando a interoperabilidade sintática o primeiro desafio a enfrentar [40]. Após os dados serem convertidos para um formato comum, é imprescindível atribuir-lhes significado através de anotações semânticas, associando-os a conceitos ontológicos que lhes conferem um valor interpretativo para além do contexto específico de onde foram originados [15].

Assim, a utilização combinada de LD e a SWT torna-se fundamental para promover a interoperabilidade entre sistemas [15]. Os LD facilitam a integração dos dados num grafo conceptual unificado e navegável, preservando a sua distribuição e gestão em sistemas distintos (integração flexível). Simultaneamente, as SWT permitem estruturar esses dados de forma coerente e significativa, por meio de ontologias partilhadas e vocabulários da *Web*, criando modelos de dados que asseguram uma interpretação uniforme e consistente [15].

3.5.2 O modelo de informação NGSI-LD

Esta Subsecção é uma adaptação de [15], © 2023 IEEE. As modificações incluem a tradução do texto e a reescrita de algumas partes.

Em primeiro lugar, é essencial compreender o conceito de contexto. A IoT pode ser vista como um conjunto de entidades - físicas e não físicas. Exemplos dessas entidades incluem os diversos sensores e atuadores empregados em qualquer solução prática de IoT. O contexto refere-se a todas as características, estados e outras propriedades dinâmicas dessas entidades, juntamente com as relações relevantes que representam as conexões reais e virtuais entre elas [41]. Em suma, qualquer informação que caracterize a situação de uma entidade constitui o seu contexto.

O NGSI-LD é umo *framework* aberta para o processamento de informações de contexto, desenvolvida pelo ETSI Industry Specification Group for cross-cutting Context Information Management (ISG CIM) [42]. O termo NGSI está relacionado com trabalhos anteriores desenvolvidos pela Open Mobile Alliance e serviu de inspiração para o sufixo LD, o qual reflete a forte influência dos conceitos de LD [42]. Esto *framework* consiste num modelo de informação semântica e numa API RESTful.

O modelo de informação NGSI-LD facilita a modelação de entidades do mundo real, bem como das suas relações e propriedades, permitindo também a interligação e federação com outros modelos de informação através da utilização de JSON-LD [42]. Este modelo é ainda compatível com Resource Description Framework (RDF) [42]. No NGSI-LD,

os dados assumem a forma de um grafo de conexões entre unidades de informação que correspondem a entidades do mundo real - o modelo de dados de grafo [41]. A referência semântica utilizada pela NGSI-LD baseia-se na tipificação padrão RDF/Relational Database Service (RDS)/Web Ontology Language (OWL) e ontologias públicas [41]. Todos os nós e arestas do grafo são associados a diferentes classes dessas ontologias, que em conjunto descrevem as características partilhadas por todas as instâncias dessas classes [41].

As principais construções definidas no metamodelo NGSI-LD são:

Entidade Representante informacional de algo que supostamente existe no mundo real, fisicamente ou conceptualmente [41]. Qualquer instância de uma Entidade deve ser identificada de forma única por um Uniform Resource Identifier, ou Identificador Uniforme de Recursos (URI) e caracterizada por um ou mais Tipo(s) de Entidade, como classes na OWL [41].

Propriedade Uma instância descritiva que associa uma característica principal - o Valor - a uma Entidade, Relação ou outra Propriedade [41].

Valor Pode assumir uma de três formas: valor JSON, valor tipificado em JSON-LD, ou valor estruturado em JSON-LD [41].

Relação Representa uma ligação direta entre um sujeito, que pode ser uma Entidade, Propriedade ou outra Relação, e um objeto, que deve ser uma Entidade [41].

As mensagens associadas às operações da API são expressas em formato JSON-LD, sendo qualquer entidade representada por um objeto codificado em JSON-LD [42].

3.5.3 Aplicação das SWT em Cidades Inteligentes

Embora o conceito de SWT tenha sido apresentado ao mundo por Berners-Lee em 1999 [43], apenas nos últimos anos começou a ganhar relevância no contexto prático e atingiu um nível de maturidade [15].

No que se refere a cidades inteligentes, os investigadores têm explorado e implementado modelos [44, 45], *frameworks* [46, 47] e arquiteturas [48, 49, 50] sustentados pelas SWT.

3.6 o *framework* FIWARE

Esta Secção é uma adaptação de [15], © 2023 IEEE. As modificações incluem a tradução do texto, reescrita de algumas partes e adição de novo conteúdo.

O FIWARE é uma iniciativa de código aberto que visa definir e implementar um conjunto de padrões para o desenvolvimento de soluções inteligentes em diversos domínios, como Cidades, Agricultura e Indústrias Inteligentes, entre outros [51].

As aplicações inteligentes alimentam-se de dados provenientes de diferentes fontes sobre eventos relevantes para elas. Esta informação de contexto, após ser processada, visualizada e analisada, pode conduzir a comportamentos inteligentes [52].

É no âmbito da gestão de dados de contexto que o FIWARE intervém, promovendo um padrão que descreve como recolher, gerir e publicar esta informação, e, adicionalmente, adiciona certos elementos que permitem explorar esses dados recolhidos [52].

Em termos de arquitetura, o FIWARE pode ser entendido como umo *framework* de componentes de software que podem ser integrados entre si ou com componentes de terceiros, facilitando a construção de plataformas para o desenvolvimento de soluções inteligentes [53]. Esses componentes são denominados GEs [54].

O principal componente do *framework* FIWARE é o "FIWARE Context Broker GE", que é o único elemento obrigatório para que uma aplicação seja considerada uma solução "Powered by FIWARE"[53]. A sua função é gerir a informação de contexto, que é uma necessidade crucial e transversal a qualquer solução inteligente [53]. Este componente exporta uma API chamada "FIWARE NGSI", que permite a integração de componentes numa plataforma "Powered by FIWARE"e proporciona um meio para que as aplicações interajam com a informação de contexto [53]. As especificações desta API estão atualmente alinhadas com o padrão NGSI-LD [53].

Além do "FIWARE Context Broker GE", o FIWARE oferece uma gama de outros GEs que atuam em diferentes áreas, desde a interação com dispositivos IoT, robôs e sistemas de terceiros até ao processamento, análise e visualização de informação de contexto [53]. Na Fig. 3.1 encontra-se o diagrama com os principais domínios funcionais do *framework* FIWARE e na Tabela encontra-se a descrição e respetivo domínio funcional de alguns GE presentes no FIWARE.

O FIWARE prioriza a interoperabilidade através de um modelo de dados padronizado, permitindo uma utilização eficaz dos dados por todos os componentes [54], enquanto as suas capacidades de gestão de dados distribuídos facilitam a integração com outras plataformas [51]. Além disso, o FIWARE mantém retrocompatibilidade ao unificar modelos de dados e oferecer conversão de formatos de dados para compatibilidade com sistemas existentes [54].

O domínio das Cidades Inteligentes é de particular relevância para o FIWARE, com mais de 250 cidades já a utilizar a sua tecnologia em diversos casos de uso, como turismo inteligente e mobilidade [55]. Pode mesmo dizer-se que o FIWARE está a tornar-se o padrão *de facto* para Cidades Inteligentes. Na verdade, o uso de API e modelos de informação padrão tem várias vantagens associadas no desenvolvimento de soluções inte-

Figura 3.1: Diagrama de domínios funcionais do *framework* FIWARE [2]

Domínio funcional	GE	Descrição
Core	**Orion-LD**	Um corretor de contexto NGSI-LD compacto que requer menos recursos
	QuantumLeap	Suporta o armazenamento de dados de contexto numa base de dados de séries temporais (CrateDB e Timescale)
Agentes IoT	**Agente IoT para JSON**	Uma ponte entre e a comunicação HTTP/MQTT (com um *payload* em JSON) e NGSI
Segurança	**Keyrock**	Suporte à gestão de identidade para autenticação segura e privada baseada em OAuth2 de utilizadores e dispositivos, etc.
	Wilma	Funções de *proxy* em esquemas de autenticação baseados em OAuth2, além de funções Policy Enforcement Point, ou Ponto de Decisão de Políticas (PEP) num controlo de acesso eXtensible Access Control Markup Language (XACML)

Tabela 3.1: Alguns dos GEs presentes no *framework* FIWARE [2]

ligentes para cidades inteligentes. Por exemplo, aumenta a interoperabilidade, melhora a portabilidade e a reutilização, acelera o tempo de entrada no mercado, integra mecanismos de segurança e fomenta a colaboração e a co-criação entre cidades e países.

3.6.1 Smart Data Models Initiative

Esta Subsecção é uma adaptação de [15], © 2023 IEEE. As modificações incluem a tradução do texto e reescrita de algumas partes.

A Smart Data Models Initiative resulta de uma colaboração entre a FIWARE Foundation, o TM Forum, o IUDX e o OASC, com o objetivo de promover a adoção de uma arquitetura de referência comum e de modelos de dados compatíveis, permitindo o desenvolvimento de soluções inteligentes interoperáveis e replicáveis em diversos setores, com foco inicial nas Cidades Inteligentes [56].

Um Smart Data Model (SDM) consiste em três elementos: o esquema, a especificação e os exemplos [56]. O esquema é a representação técnica do modelo, onde a sua estrutura e os tipos de dados são definidos [56]. A especificação é um documento orientado para leitores humanos – a documentação [56]. Os exemplos são *payloads* que ilustram o uso dos modelos e estão disponíveis para as especificações NGSIv2 e NGSI-LD. Todos os modelos de dados são gratuitos para usar, de modificação livre e permitem a partilha livre de modificações [56].

Em termos de organização, os modelos de dados estão agrupados por temas e são atribuídos a repositórios GIT, sendo possível estar associados a um ou mais domínios, representando diferentes setores da indústria [56].

Partilhar informação estruturada entre diferentes partes interessadas e promover a troca aberta de dados é fundamental em qualquer SDM e é uma premissa essencial do FIWARE, onde o NGSI-LD aparece como um modelo de informação que apoia todos esses requisitos.

3.7 Introdução ao conceito de gestão de identidades e ao Keyrock

No âmbito da computação, a gestão de identidades refere-se à administração das identidades digitais de utilizadores e serviços dentro de um sistema, assegurando que apenas aqueles devidamente autorizados possam aceder aos recursos digitais necessários para executar as suas funções ou operar de forma adequada. Neste contexto, emergem dois conceitos fundamentais: autenticação e autorização. A autenticação corresponde ao processo de verificar que os utilizadores ou serviços são realmente quem dizem ser, constituindo a primeira etapa de qualquer sistema de segurança [57]. Após a verificação, procede-se à autorização, que consiste na atribuição de permissões ao utilizador ou serviço para aceder a recursos específicos.

O *framework* FIWARE, e, consequentemente, a plataforma de gestão remota do sistema VALLPASS, é composto por um conjunto de componentes distintos com os quais interagem tanto os utilizadores - administradores - como os dispositivos - postes das passadeiras VALLPASS -, que, neste contexto, podem ser considerados como ”utilizadores não humanos”. No que se refere à segurança, é necessário validar a identidade destes

componentes e utilizadores, processo designado por autenticação, e assegurar que apenas têm acesso aos recursos a que estão autorizados, i.e., autorização. Para esse propósito, o componente FIWARE Keyrock pode ser utilizado, dado que apresenta todas as funcionalidades típicas de um sistema de gestão de identidades, permitindo a implementação de mecanismos de autenticação padrão e um sistema de autorização baseado em protocolos reconhecidos na indústria [58].

Apresentam-se, a seguir, alguns conceitos fundamentais no âmbito da gestão de identidades, que se encontram representados na base de dados do Keyrock sob a forma de objetos [59]:

***User* (Utilizador)** Refere-se a qualquer utilizador registado que possa ser identificado por meio de um endereço de email e uma palavra-passe. A estes podem ser atribuídos direitos de forma individual ou em grupo.

***Application* (aplicação)** Denomina qualquer aplicação FIWARE passível de ser protegida, constituída por um conjunto de microsserviços.

***Organization* (organização)** Agrupamento de utilizadores ao qual se pode atribuir um conjunto de direitos. A modificação dos direitos de uma organização afeta todos os utilizadores que dela fazem parte.

***OrganizationRole* (função na organização)** Indica que os utilizadores podem ser membros ou administradores de uma organização. Os administradores têm a capacidade de adicionar ou remover membros da organização, enquanto os membros apenas adquirem as funções e permissões associadas à mesma. Este conceito permite a administração distribuída, garantindo que cada organização seja responsável pela gestão dos seus próprios membros, eliminando a necessidade de um "super-administrador" para controlar todos os direitos.

***Role* (função)** Representa um conjunto de permissões agregadas que podem ser atribuídas a um utilizador ou a uma organização. O utilizador autenticado obtém cumulativamente todas as permissões das funções a ele designadas, bem como das funções ligadas à organização a que pertence.

***Permission* (permissão)** Capacidade de executar uma determinada ação sobre um recurso específico do sistema.

Adicionalmente, existem dois objetos adicionais, que não se referem a utilizadores, mas que podem ser protegidos numa plataforma FIWARE [58]:

IoTAgent Componente que atua como uma ponte entre os dispositivos IoT - como os postes das passadeiras VALLPASS - e o corretor de contexto NGSI-LD.

PEPProxy *Middleware* utilizado para mediar a comunicação entre componentes FIWARE, assegurando a verificação dos direitos de acesso dos utilizadores.

Em termos de arquitetura, o Keyrock está integrado com uma base de dados MySQL, que é utilizada para armazenar de forma persistente as identidades dos utilizadores, aplicações, funções e permissões. Este componente disponibiliza as seguintes funcionalidades [58]:

Sistema de autenticação OAuth2 Implementa o protocolo OAuth2, um padrão amplamente adotado na indústria [60], tanto para aplicações como para utilizadores.

Aplicação *Web* Interface destinada à administração do sistema de gestão de identidades.

API Permite a gestão de identidades através de pedidos HTTP, utilizando o estilo arquitetural Representational State Transfer, ou Transferência de Estado Representacional (REST), possibilitando a integração com outras aplicações e serviços.

3.8 Docker

3.8.1 Introdução

O Docker é uma plataforma para desenvolver, distribuir e executar aplicações em contentores [61]. Estes contentores encapsulam uma aplicação juntamente com todas as suas dependências, garantindo um ambiente isolado que facilita a portabilidade e assegura a consistência em diferentes sistemas.

3.8.2 Conceito de contentor

A essência do Docker reside no conceito de contentores - unidades leves, independentes e executáveis [61] que contêm tudo o que uma aplicação necessita para operar, incluindo o código, bibliotecas, variáveis de ambiente e configurações de sistema. Estes contentores abstraem a aplicação da infraestrutura subjacente, permitindo que esta, além de ser portátil, seja executada de forma consistente em qualquer plataforma, seja num ambiente local de desenvolvimento, num servidor de produção ou na nuvem.

Ao contrário da virtualização tradicional, baseada em hypervisor, os contentores Docker partilham o sistema operativo do *host* (virtualização ao nível do sistema operativo), o que proporciona maior eficiência e portabilidade. Pese o facto de partilharem o sistema operativo, em particular o seu *kernel*, garantem ainda um nível adequado de isolamento em relação a outros contentores e à própria máquina *host*, graças ao uso da tecnologia Namespaces [61].

3.8.3 Arquitetura

A arquitetura do Docker segue um modelo de cliente-servidor, no qual o cliente envia comandos para o servidor (ou *daemon*), que executa as operações de gestão dos objetos Docker como contentores, imagens, volumes e redes [61]. De seguida, apresentam-se os principais objetos Docker.

Imagens As imagens Docker são ficheiros imutáveis (*read-only*) com um sequência de instruções (passos) para criar um contentor [61]. As imagens pode ser vistas como o modelo de um contentor e são definidas através de uma Dockerfile.

Contentores De forma concisa, um contentor é uma instância executável de uma imagem [61]. A este, é possível associar várias redes, anexar armazenamento persistente e até criar uma nova imagem com base no seu estado atual [61].

Redes O Docker permite a criação de redes virtuais que conectam contentores entre si ou com o exterior, isolando o tráfego de rede dos contentores e possibilitando a configuração de regras de comunicação de forma granular.

Volumes Unidades de armazenamento persistente usadas pelos contentores para preservar dados mesmo depois da paragem ou remoção de um contentor. São úteis para armazenar bases de dados ou ficheiros de configuração que necessitem de ser persistentes entre execuções.

3.8.4 Vantagens

O Docker proporciona importantes vantagens para o desenvolvimento e gestão de aplicações. De seguida, apresentam-se as principais.

Portabilidade Graças ao seu ambiente isolado, as aplicações em Docker podem ser executadas em qualquer sistema que suporte a plataforma de forma consistente, eliminando problemas de compatibilidade entre diferentes plataformas ou sistemas operativos.

Isolamento Cada contentor está, por defeito, relativamente bem isolado da máquina *host* e dos outros contentores, todavia, é possível ajustar e controlar o grau de isolamento conforme necessário [61]. Esta característica oferece várias vantagens em termos de segurança, gestão de recursos e manutenção.

Escalabilidade O Docker facilita a escalabilidade horizontal (distribuição de contentores) das aplicações, permitindo a criação rápida e eficiente de múltiplas instâncias de um serviço. Esta abordagem melhora a resposta a picos de carga (através de balanceamento de carga) e aumenta a resiliência do sistema, assegurando maior disponibilidade e desempenho.

Eficiência Ao partilharem o *kernel* do sistema operativo do *host*, os contentores são intrinsecamente mais leves e consomem menos recursos do que as máquinas virtuais, reduzindo a sobrecarga e melhorando o desempenho. Isto permite a execução simultânea de múltiplos contentores, otimizando a utilização da capacidade de processamento e dos recursos disponíveis no servidor. Consequentemente, aumenta-se a densidade de aplicações e a eficiência na utilização de recursos.

3.8.5 Ecossistema e ferramentas

O ecossistema Docker inclui várias ferramentas e serviços que complementam e expandem as suas funcionalidades. De seguida, apresentam-se duas bastante populares.

Docker Hub O Docker Hub é o registo oficial de imagens Docker, onde os utilizadores podem partilhar e descarregar imagens públicas, sendo também possível armazenar imagens privadas, tornando-se uma ferramenta essencial para a distribuição e gestão de imagens em diferentes ambientes [62, 63].

Docker Compose O Docker Compose é uma ferramenta que facilita a definição e execução de aplicações Docker que utilizam múltiplos contentores. Na arquitetura Docker, o Docker Compose assume o papel de um cliente Docker [61]. Usando um ficheiro `docker-compose.yml`, é possível definir a configuração de uma aplicação composta por vários serviços, simplificando tanto o processo de orquestração quanto a gestão de dependências.

3.9 Node-RED

O Node-RED é uma ferramenta de programação *low-code*, baseada em fluxos, que facilita a interligação de dispositivos de *hardware*, APIs e serviços online de forma intuitiva e inovadora [64]. Esta plataforma é suportada por um ambiente de execução Node.js, proporcionando uma desenvolvimento orientado a eventos [64]. Desenvolvida originalmente pela IBM e agora parte da OpenJS Foundation, o Node-RED oferece um ambiente de desenvolvimento visual que simplifica a criação de aplicações através da interconexão de nós, cada um representando uma funcionalidades específica [65]. Esta abordagem reduz significativamente a complexidade da integração de sistemas e acelera a prototipagem, sendo particularmente atrativa para programadores que desejam integrar diferentes componentes sem recorrer a grandes quantidades de código.

A interface do Node-RED é acessível através de um navegador *web*, onde os utilizadores podem arrastar e soltar nós a partir de uma paleta, conectando-os para formar fluxos que representam o comportamento desejado. Além disso, possui um editor de texto rico,

que permite a escrita de funções em JavaScript. Os fluxos criados no Node-RED são armazenados em formato JSON [64].

O Node-RED é altamente extensível, oferecendo mais de 225000 módulos disponíveis no repositório de pacotes do Node.js [64], e beneficia de uma comunidade que contribui com uma vasta gama de fluxos adicionais, facilitando a integração com uma ampla diversidade de serviços e dispositivos [66].

3.10 Express.js

O Express.js é um *framework web* minimalista e flexível para o ambiente de execução Node.js, oferecendo um conjunto robusto de funcionalidades para o desenvolvimento de aplicações *web* e móveis [67]. Desenvolvido em JavaScript, este *framework* simplifica o processo de criação de *software* ao fornecer um conjunto abrangente de ferramentas para lidar com pedidos HTTP, roteamento, *middleware*, entre outros [68, 69]. Esta abordagem facilita aos programadores a construção de APIs e aplicações *web* de forma rápida e eficiente.

Uma das principais vantagens do Express.js é a sua simplicidade e extensibilidade. Ele permite aos programadores integrar apenas as funcionalidades necessárias através de *middleware* e módulos, promovendo uma arquitetura modular e escalável [68]. Além disso, suporta diversos motores de *template*, facilitando a renderização dinâmica de páginas *web* [70].

Devido à sua versatilidade, eficiência e ao suporte de um vasto ecossistema de ferramentas e bibliotecas, incluindo extensões da comunidade e soluções de grandes empresas, o React tornou-se uma escolha dominante tanto para o desenvolvimento de aplicações *web* quanto móveis, evidenciado pela sua posição de destaque como uma das tecnologias *web* mais populares na pesquisa de 2024 do Stack Overflow [71].

3.11 React

O React é uma biblioteca JavaScript de código aberto, desenvolvida pela Facebook em 2013, cujo objetivo principal é facilitar a criação de User Interfaces, ou Interfaces de Utilizador interativas e dinâmicas, particularmente em aplicações *web*. Ao contrário de *frameworks* completos, como Angular ou Vue.js, o React foca-se exclusivamente na camada de visualização, permitindo que os programadores criem componentes reutilizáveis e reativos que atualizam eficientemente a interface à medida que os dados mudam.

Uma das principais inovações do React é o uso do Virtual Document Object Model, uma representação em memória do Document Object Model (DOM) real [72]. Em vez de modificar diretamente o DOM, o React gere atualizações de forma eficiente, compa-

rando as mudanças entre o Virtual DOM e o DOM real, aplicando apenas as alterações mínimas necessárias. Este mecanismo resulta numa melhoria significativa no desempenho de aplicações de grande escala.

Além disso, o React adota uma arquitetura baseada em componentes, em que cada componente encapsula a lógica e a estrutura da UI associada a uma parte específica da aplicação [73]. Estes componentes podem ser combinados e reutilizados em diferentes partes da aplicação, o que promove a modularidade, a manutenibilidade e a escalabilidade do código.

O React também introduziu o conceito de JavaScript XML, uma sintaxe que permite escrever código semelhante a HTML diretamente dentro de JavaScript, facilitando a construção de UIs complexas de forma mais intuitiva e declarativa [74].

Atualmente, o React é amplamente utilizado tanto no desenvolvimento de aplicações *web* quanto de aplicações móveis (através do React Native), devido à sua versatilidade, eficiência e ao vasto ecossistema de ferramentas e bibliotecas que o suportam [75].

3.12 Material UI

A Material UI é uma biblioteca de código aberto amplamente utilizada, composta por componentes React, que segue as diretrizes de *design* do Material Design, desenvolvidas pela Google [76]. Esta biblioteca oferece uma vasta gama de componentes reutilizáveis, estilizados e de elevada qualidade, facilitando o desenvolvimento de UIs consistentes e modernas para aplicações *web*.

3.13 Trabalhos relacionados

Em [77], Bhardwaj et al. propõem uma arquitetura inovadora para passagens de peões em cidades inteligentes utilizando uma abordagem baseada em IoT. Os autores realçam que as passagens de peões são pontos críticos de segurança nas infraestruturas urbanas, locais onde o aumento da densidade populacional e do tráfego frequentemente resulta em mais acidentes e lesões graves. A arquitetura proposta utiliza um sistema de deteção de gestos e câmaras de alta definição para identificar quando um peão deseja atravessar a via. O sistema, montado em postes de semáforo, comunica-se com os veículos próximos através de sinais de luz, indicando se a travessia está ativa, com o objetivo de garantir maior segurança aos peões. O artigo aborda ainda a integração deste sistema com uma base de dados na nuvem para manter registos de eventos, o que pode ser útil para diagnósticos posteriores em caso de acidentes. Os autores comparam a sua proposta com outras soluções existentes, indicando que a arquitetura é mais fiável, económica e eficiente na redução do consumo de energia.

Em [78], Tomás de J. Mateo Sanguino *et al.* apresentam um sistema inovador de passadeira inteligente que utiliza técnicas de deteção baseadas em inteligência artificial para melhorar a segurança rodoviária em passadeiras. O artigo aborda os desafios das soluções tradicionais de sinalização, como elevados custos e limitações de cobertura, e propõe um sistema que integra deteção inteligente de peões e sinalização luminosa autónoma alimentada por energia solar. O sistema utiliza técnicas de fusão sensorial e algoritmos de inteligência artificial para identificar a presença de peões e veículos, atuando de forma adaptativa para alertar condutores e garantir uma travessia mais segura. Testes em condições reais demonstraram uma taxa de precisão de 99,11% na deteção de peões, assim como reduções significativas na velocidade nos veículos - 32,83% durante o dia e 70,6% à noite, promovendo um ambiente mais seguro para os peões. O estudo também incluiu um inquérito a utilizadores, que indicou uma melhoria na perceção de segurança e no respeito pelas normas de trânsito, demonstrando o potencial de sistemas de sinalização inteligentes para melhorar a segurança rodoviária e otimizar a interação entre peões e veículos através de IoT e técnicas de *machine learning*.

Em [79], os autores apresentam a implementação do City Data Hub, uma plataforma de dados para cidades inteligentes desenvolvida com foco na interoperabilidade, especialmente ao nível dos dados. A plataforma foi criada no contexto do Programa Nacional Estratégico de Cidades Inteligentes da Coreia do Sul. Para atingir a interoperabilidade, foram adotados princípios de interoperabilidade baseados em APIs padronizadas e modelos de dados unificados, nomeadamente através da utilização de interfaces NGSI-LD e da definição de modelos de dados compatíveis com a especificação NGSI-LD. Além disso, a abordagem modular da plataforma possibilita a redefinição e expansão com novos módulos NGSI-LD compliant. Os autores ilustram a aplicabilidade da solução através de provas de conceito como a previsão de disponibilidade de estacionamento e o suporte a investigações epidemiológicas no contexto da pandemia de COVID-19. Assim, o City Data Hub destaca-se pela sua arquitetura aberta e extensível, que visa superar silos de dados e criar um ecossistema sustentável para soluções de cidades inteligentes.

Em [80] os autores relatam as suas experiências no âmbito do projeto CityIoT, cujo objetivo foi definir uma plataforma IoT independente de fornecedores para aplicações em cidades inteligentes. Após uma análise prévia no início do projeto, optaram pela adoção do FIWARE como *framework* técnico devido à sua abordagem aberta (mesma premissa do projeto CityIoT) e à sua popularidade nas cidades inteligentes. No artigo, resumem as experiências na recolha, pré-processamento, unificação e armazenamento de dados provenientes de 12 casos-piloto que cobrem diversos domínios, como iluminação pública, transportes públicos e monitorização de condições de edifícios. Estas experiências foram categorizadas em quatro áreas principais no contexto dos dados: modelos, conversão, envio e processamento e armazenamento. Relativamente aos modelos de dados, os autores

descobriram que, embora os modelos de dados existentes do FIWARE fossem bastante úteis e aplicáveis na maioria dos casos, frequentemente era necessário fazer adaptações para acomodar necessidades específicas. Concluem que o uso prático do FIWARE em casos reais pode revelar desafios não evidentes na documentação ou nos tutoriais. Apesar das dificuldades mencionadas, os autores não pretendem desencorajar o uso do FIWARE, mas sim fornecer informações adicionais que possam ajudar outros a planear e implementar soluções de forma mais eficaz, aproveitando as capacidades do FIWARE para lidar com dados complexos em ambientes de cidades inteligentes.

O artigo [81] de Javier Conde et al. explora o uso do *framework* FIWARE para a construção de GDs, apresentando uma arquitetura de referência que facilita a implementação de soluções inteligentes em diversos domínios. Para resolver os desafios comuns na construção de GDs - como a integração de grandes volumes de dados, a interoperabilidade e a segurança - os autores demonstram como os componentes de FIWARE, incluindo o Context Broker, Agentes IoT, Draco e Keyrock, podem ser utilizados para criar uma solução modular e escalável. A proposta é validada através de um caso de estudo que utiliza um GD aplicado a um parque de estacionamento, com o objetivo de melhorar o tráfego neste, auxiliando os utilizadores na localização dos seus veículos através de uma aplicação 3D e prevendo a ocupação do parque. Neste caso, os dados são adquiridos a partir de sensores IoT e fontes externas, como previsões meteorológicas, e são processados utilizando o *framework* FIWARE. O Context Broker é utilizado como ponto central de gestão de contexto, enquanto os restantes componentes tratam da aquisição, transformação, análise e persistência dos dados. A segurança é garantida através do uso de mecanismos de autenticação e autorização para controlar o acesso aos dados e aos dispositivos. O artigo conclui que o FIWARE, com os seus GEs e os SDMs, constitui uma solução sólida e flexível para a criação de GDs em diferentes setores. Os autores sugerem como trabalho futuro a validação da arquitetura com novos casos de uso e a transição para o padrão NGSI-LD, explorando as vantagens semânticas oferecidas por esta especificação.

Como extensão dessa arquitetura, o trabalho de Conde *et al.* em [82] propõe a integração de Linked Open Data, ou Dados Abertos Ligados (DAL) para facilitar a colaboração entre GDs. A integração de DALs com o *framework* FIWARE é proposta como uma opção de referência para implementar GDs. O artigo valida a proposta através de um caso de uso em que um GD urbano colabora com um GD de estacionamento. No exemplo, os dados dos estacionamentos *off-street* são publicados como DAL, enquanto um GD urbano consome essa informação para gerir a informação de estacionamentos *off-street*.

3.14 Conclusão

Este Capítulo apresentou uma revisão bibliográfica abrangente sobre tópicos essenciais para o desenvolvimento do projeto VALLPASS. Iniciou-se pela análise da arquitetura da IoT e do papel do *middleware*, prosseguindo com a exploração das plataformas IoT e a sua classificação com base em requisitos funcionais e não funcionais.

Foram introduzidos os GDs e discutida a sua aplicação em cidades inteligentes. Foi ainda abordada a importância da interoperabilidade na IoT, destacando-se as SWT e os LD como elementos chave para a integração eficaz de sistemas e dados.

o *framework* FIWARE foi apresentada, assim como o componente Keyrock, destacando-se o seu papel na gestão de identidades e segurança em plataformas IoT. Além disso, tecnologias como Docker, Node-RED, Express.js, React e Material UI foram discutidas como ferramentas relevantes para o desenvolvimento de aplicações modernas e escaláveis.

Por fim, foram analisados trabalhos relacionados que forneceram contributos significativos no desenvolvimento de passadeiras inteligentes e soluções IoT em ambientes urbanos, oferecendo uma base sólida para a continuação do trabalho desta dissertação no projeto VALLPASS.

Capítulo 4 Revisão bibliográfica - VLC

4.1 Introdução

Este Capítulo apresenta uma revisão bibliográfica sobre a VLC. Inicialmente, é fornecida uma visão geral da VLC, explorando o seu funcionamento, características e vantagens, especialmente no contexto das cidades inteligentes. Em seguida, é analisado o padrão mais relevante - IEEE 802.15.7 [4] - que define a Physical Layer, ou Camada Física (PHY) e a MAC. Posteriormente, é descrita a arquitetura básica de um sistema VLC, detalhando os seus componentes principais: transmissor, canal de comunicação e recetor. Por fim, são apresentados trabalhos relacionados que ilustram aplicações práticas da VLC em diferentes cenários e discutidas as suas contribuições.

4.2 Visão geral da VLC

A Optical Wireless Communication, ou Comunicação Ótica Sem Fios (OWC) caracteriza-se pela transmissão de dados através da modulação da intensidade de fontes óticas como luminárias LED e Laser Diodes, ou Díodos *Laser* (LDs), a uma velocidade superior à que o olho humano consegue perceber [4]. A VLC é um subtipo de OWC que restringe as fontes óticas à luz visível, localizada no espectro entre 380 nm e 750 nm de comprimento de onda de luz [13].

Um dos grandes facilitadores do desenvolvimento da VLC foi a adoção massiva da iluminação LED, que, além das suas inúmeras vantagens, especialmente no contexto da sustentabilidade, permite, ao contrário das fontes de iluminação convencionais (como a iluminação de halogéneo), ajustar a intensidade luminosa a uma velocidade extremamente elevada - impercetível ao olho humano [13].

A VLC possui inúmeras vantagens, especialmente no contexto das cidades inteligentes. Destacam-se as seguintes:

Versatilidade e eficiência A infraestrutura de iluminação existente pode ser utilizada simultaneamente para transmitir dados [13, 14, 12];

Simplificação da infraestrutura Elimina a necessidade de implementar novas infraestruturas de comunicação, reduzindo a complexidade e os custos, o que promove a implementação mais célere das cidades inteligentes;

Libertação do espectro de RF O atual espectro de RF, mesmo com as otimizações realizadas, está a tornar-se insuficiente para lidar com o aumento massivo do tráfego [13], especialmente em áreas densamente povoadas. A VLC, com vários centenas de terahertz de espectro não licenciado [4], pode ajudar a colmatar este problema. Conforme mencionado por Elgala et al. em [14], é amplamente aceite que a OWC será essencial para comunicações de curto alcance.

4.3 Padrão IEEE 802.15.7

O padrão mais relevante neste domínio é o IEEE 802.15.7, que define a PHY e a subcamada de MAC para OWC de curta distância, utilizando comprimentos de onda de luz entre 190 nm e 10 000 nm (abrangendo desde a luz visível até parte do espectro infravermelho e ultravioleta) [4]. Este padrão aborda aspetos como topologias de rede, modulação de luz, mitigação de interferências, segurança, suporte à mobilidade e integração com dispositivos como LEDs e câmaras, garantindo ainda suporte a serviços multimédia e controlo de iluminação [4].

4.4 Arquitetura básica

Um sistema VLC é composto, principalmente, por três blocos fundamentais: transmissor, canal de comunicação e recetor. Cada um desses blocos desempenha um papel fundamental na transmissão e receção de dados através da modulação de luz visível. A seguir, descreve-se de forma sucinta cada um dos seus elementos constituintes.

Transmissor O transmissor de um sistema VLC é geralmente constituído por uma luminária LED, devido às características já apresentadas na Secção 4.2, nomeadamente pela sua capacidade de modular a intensidade luminosa a altas frequências. Note-se, contudo, que esta não é a única opção viável. Em determinados contextos, pode ser interessante utilizar outras fontes luminosas, como LDs, especialmente em situações onde não se pretende obter iluminação através do transmissor ou onde se desejam alcançar distâncias de comunicação mais longas. O processo de transmissão envolve a conversão de dados digitais em sinais luminosos, utilizando técnicas de modulação de luz apropriadas, como ON-OFF Keying (OOK) ou Variable Pulse Position Modulation, ou Modulação por Posição de Pulso Variável (VPPM) [4], para codificar a informação na variação de intensidade da

luz. Sempre que o transmissor desempenhe também a função de iluminação, pode ser necessário implementar mecanismos de mitigação de cintilação (*flicker*), que o padrão IEEE 802.15.7 define como "a modulação temporal da iluminação a frequências superiores à frequência crítica de fusão, que pode afetar a eficiência humana de várias formas" [4].

Canal de comunicação O canal de comunicação na VLC é o meio pelo qual a luz visível se propaga. Este canal pode ser afetado por diversos fatores ambientais, como interferência de outras fontes de luz (luz solar, lâmpadas fluorescentes, etc.), reflexões em superfícies, obstáculos físicos e ruído ótico. Dependendo do ambiente, o canal pode ser considerado de Line of Sight, ou Linha de Vista (LOS) ou Non-Line of Sight, ou Fora da Linha de Vista (NLOS) [83], influenciando diretamente a qualidade da transmissão e a taxa de erro dos dados recebidos.

Recetor Os recetores são responsáveis por converter a luz modulada em sinais elétricos que representam a informação transmitida. Dividem-se em dois tipos: fotodetetores (também designados por fotodíodos ou recetores *non-imaging*) e sensores de imagem (também conhecidos como sensores de câmara) [13]. Um fotodetetor é um dispositivo semicondutor que converte a luz recebida em corrente elétrica. Já os sensores de imagem, que correspondem a um conjunto de fotodetetores organizados numa matriz, estão presentes de forma ubíqua no nosso dia a dia - em smartphones, tablets, computadores portáteis, entre outros -, o que garante, potencialmente, uma elevada disponibilidade de recetores para a VLC.

Na Fig. 4.1 é apresentado um diagrama que representa a arquitetura básica de um sistema VLC.

4.5 Trabalhos relacionados

Em [84], Delgado-Rajo et al. propõem uma arquitetura de rede híbrida para a IoT, combinando comunicação via rádio (LoRa e ZigBee) com VLC. O objetivo é fornecer uma solução de baixo consumo energético para áreas remotas com infraestrutura de comunicação limitada. A rede adota um modelo celular, no qual o LoRa atua como *backbone* para comunicações de longo alcance, enquanto o ZigBee e a VLC interligam sensores e atuadores. A interoperabilidade entre essas tecnologias é assegurada por uma camada de abstração de protocolo, que facilita o roteamento e a identificação dos nós. Os testes experimentais realizados comprovaram a viabilidade da arquitetura proposta.

Em [85], Shao et al. apresentam um sistema híbrido de acesso à Internet para ambientes internos que combina WiFi e VLC. A proposta visa mitigar problemas de congestionamento no espectro de RF ao utilizar a VLC, com LEDs como transmissores, para a

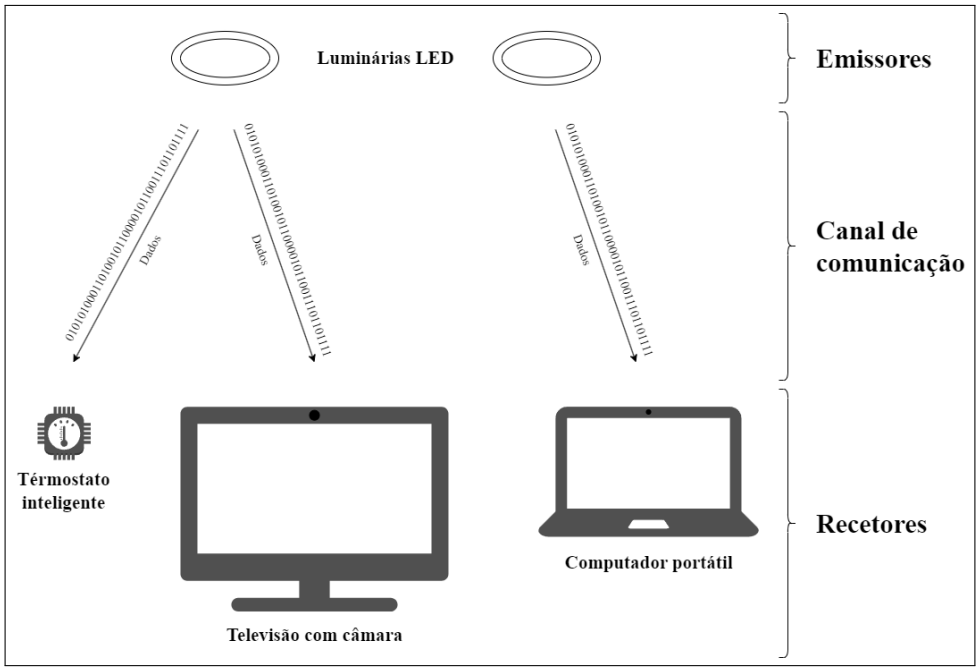

Figura 4.1: Diagrama da arquitetura básica de um sistema VLC

comunicação no canal de descida (*downlink*), enquanto a comunicação no canal de subida (*uplink*) é realizada através de WiFi. A implementação prática do sistema demonstra que, em ambientes congestionados, o desempenho do sistema híbrido supera o WiFi tradicional em termos de taxa de transferência e tempo de carregamento de páginas *web*.

Em [11], Kamruzzaman analisa as tecnologias-chave para a comunicação sem fios 6G em cidades inteligentes, destacando o papel da IoT, através de uma revisão sistemática. Entre as tecnologias abordadas, a VLC é identificada como uma solução promissora, sendo considerada uma tendência relevante para a gestão de tráfego nas cidades inteligentes. A capacidade de suportar comunicações híbridas é também destacada como uma característica importante. Contudo, o artigo também discute os desafios associados à VLC, tais como o fenómeno de cintilação e a necessidade de ajuste da intensidade luminosa (*dimming*) em ambientes interiores.

Em [12], Ayub et al. apresentam uma abordagem prática para integrar a VLC na arquitetura de cidades inteligentes. O artigo destaca a VLC como uma solução promissora para aliviar a sobrecarga do espectro de RF e melhorar a conectividade em ambientes urbanos. A utilização da VLC no desenho de sistemas de transporte inteligentes - onde as luzes de tráfego LED podem ser utilizadas como um sistema de *broadcast* para os veículos - e a integração com a Power Line Communication, ou Comunicação por Linha de Energia (PLC) são consideradas aplicações para exploração comercial. O estudo também apresenta resultados de testes experimentais que mostram o desempenho da VLC em termos de taxa de transferência e alcance.

4.6 Conclusão

Neste Capítulo, foi realizada uma revisão abrangente sobre a VLC. Iniciou-se com uma visão geral da VLC, destacando as suas vantagens, como a eficiência energética, a reutilização da infraestrutura de iluminação existente e a libertação do espectro de RF saturado. Explorou-se o padrão IEEE 802.15.7 [4], que fornece as bases para o desenvolvimento e implementação de sistemas VLC, definindo aspetos técnicos essenciais para garantir a interoperabilidade e a eficiência das comunicações. Descreveu-se a arquitetura básica de um sistema VLC, enfatizando o papel de cada componente - transmissor, canal de comunicação e recetor - e os desafios associados, como a mitigação de interferências e a adaptação às condições ambientais. Por fim, foram apresentados trabalhos relacionados que demonstram a aplicação prática da VLC em diferentes contextos, evidenciando o seu potencial para melhorar a conectividade e a eficiência em ambientes urbanos e interiores. Esta revisão permite compreender o estado atual da tecnologia e as oportunidades que a VLC oferece para futuras pesquisas e desenvolvimento no campo das comunicações sem fios.

Capítulo 5 Contribuição para o projeto VALLPASS

5.1 Introdução

No presente Capítulo, apresenta-se a contribuição para o projeto VALLPASS sob a forma de uma plataforma de gestão remota *Powered by FIWARE*, apoiada por um modelo de dados semântico direcionado para cidades inteligentes. O desenvolvimento inicia-se com uma análise aprofundada dos requisitos do sistema, englobando a caracterização dos utilizadores e a especificação dos requisitos funcionais e não funcionais, culminando com a apresentação do diagrama de casos de uso.

Em seguida, desenvolve-se o modelo conceptual de dados, onde se apresenta o diagrama ER e a descrição das entidades principais do sistema. Este modelo serve de base para a definição do modelo de dados semântico NGSI-LD, que é abordado posteriormente. Nesta Secção, são selecionados modelos de dados de referência e ajustados às necessidades específicas do VALLPASS, com o objetivo de gerar um ficheiro NGSI-LD `@context`, essencial para a interoperabilidade semântica.

A definição da arquitetura da plataforma de gestão remota, baseada em microsserviços, é então apresentada, detalhando-se os componentes principais, as suas interações e os *endpoints* do *proxy* reversa (NGINX). Esta arquitetura modular e escalável é fundamental para garantir a eficiência e a robustez do sistema.

Para validar a arquitetura proposta e o modelo de dados semântico desenvolvido, procede-se à prototipagem da plataforma de gestão remota. Esta etapa inclui a implementação e configuração dos serviços definidos, bem como a simulação do sistema VALLPASS, permitindo testar e refinar as funcionalidades desenvolvidas.

Finalmente, são descritos os processos de desenvolvimento do *back-end* e do *front-end* da aplicação *web*. No *back-end*, aborda-se o roteamento, a comunicação com o *front-end* e a atualização do estado operativo dos atuadores do sistema. No *front-end*, detalham-se as funcionalidades de autenticação, o *layout* da aplicação, a integração da especificação OpenAPI do modelo de dados semântico VALLPASS e as ferramentas de gestão de passadeiras e clientes.

5.2 Disponibilização do Código-Fonte

Todo o código relacionado com a plataforma de gestão remota desenvolvida no âmbito do projeto VALLPASS, está disponível num repositório online no GitHub - `https://github.com/a39172/PlataformaGestaoRemotaVALLPASS`.

5.3 Análise de requisitos

Os requisitos do sistema foram elaborados sob a perspetiva da Aplicação *Web*, que constitui a parte tangível da plataforma de gestão remota do projeto VALLPASS. Embora a aplicação *Web* seja apenas um dos componentes da plataforma, as funcionalidades e necessidades identificadas para esta aplicação influenciam diretamente a definição dos restantes elementos do sistema. Por exemplo, determinadas funcionalidades requeridas pela aplicação *web* podem implicar a integração de serviços adicionais ou o desenvolvimento de componentes específicos na infraestrutura subjacente.

Esta abordagem, centrada na Aplicação *Web*, permite focar nas necessidades do utilizador final e nas interações diretas com o sistema, garantindo uma experiência de utilização eficiente e intuitiva. Além disso, facilita a identificação de dependências e requisitos técnicos que afetam todo o sistema, assegurando uma coerência global no desenvolvimento da plataforma.

De seguida, apresenta-se a caracterização dos utilizadores e a especificação detalhada dos requisitos funcionais e não funcionais do sistema.

5.3.1 Caracterização dos utilizadores

No âmbito desta dissertação, considera-se apenas um tipo de utilizador: o administrador do sistema. Este utilizador tem acesso total a todas as funcionalidades da Aplicação *Web*, incluindo a gestão das passadeiras inteligentes, monitorização dos dispositivos e configuração de parâmetros do sistema.

A opção por centrar o desenvolvimento na perspetiva do administrador deve-se à necessidade de simplificar a gestão e operacionalização do sistema durante a fase de prototipagem e validação. Embora, numa implementação comercial do sistema VALLPASS, possam existir diferentes perfis de utilizadores com níveis de acesso distintos - por exemplo, clientes que gerem apenas as suas próprias passadeiras -, esta diferenciação não é abordada neste trabalho. O foco recai nas funcionalidades nucleares da plataforma, evitando a complexidade adicional associada à gestão de múltiplos perfis e modelos de negócio.

Assim, todas as funcionalidades e interfaces da Aplicação *Web* foram desenhadas con-

siderando as necessidades e responsabilidades do administrador, garantindo uma visão abrangente e controlo completo sobre o sistema VALLPASS.

5.3.2 Requisitos Funcionais

1. O sistema deve permitir que o utilizador não autenticado se autentique com as suas credenciais.
2. O sistema deve permitir que o administrador altere a sua palavra-passe.
3. O sistema deve permitir que o administrador altere o seu nome de apresentação.
4. O sistema deve permitir que o administrador altere a sua foto de perfil.
5. O sistema deve apresentar ao administrador uma *dashboard* que sintetize o estado do sistema VALLPASS, contendo as seguintes informações:
   (a) Número de postes inoperativos;
   (b) Número de postes com bateria fraca;
   (c) Número de postes com a luminária ligada;
   (d) Número de acidentes na última semana.
6. O sistema deve apresentar ao administrador todas as passadeiras inteligentes provisionadas.
7. O sistema deve permitir que o administrador altere a ordem de visualização de todas as passadeiras inteligentes provisionadas.
8. O sistema deve apresentar ao administrador todos os clientes registados numa tabela.
9. O sistema deve permitir que o administrador altere a ordem de visualização de todos os clientes.
10. O sistema deve permitir que o administrador pesquise clientes pelo seu nome.
11. O sistema deve permitir que o administrador registe novos clientes.
12. O sistema deve apresentar ao administrador uma *dashboard* para cada passadeira inteligente com as seguintes informações:
    (a) Nome atribuído à passadeira inteligente;
    (b) Temperatura exterior;

(c) Humidade exterior;

(d) Luminosidade;

(e) Mapa com a localização da passadeira inteligente;

(f) Número de postes com a luminária ligada;

(g) Número de postes inoperativos;

(h) Número de postes com bateria fraca;

(i) Número de acidentes na última semana.

13. O sistema deve apresentar ao administrador os detalhes de cada passadeira inteligente.

14. O sistema deve permitir que o administrador altere o nome de apresentação da passadeira inteligente.

15. O sistema deve permitir que o administrador associe as passadeiras inteligentes aos respetivos clientes.

16. O sistema deve apresentar ao administrador uma *dashboard* com as seguintes informações de cada poste:

    (a) Estado operativo;

    (b) Temperatura exterior;

    (c) Humidade exterior;

    (d) Luminosidade;

    (e) Mapa com a localização do poste;

    (f) Estado da luminária;

    (g) Nível de bateria.

17. O sistema deve apresentar ao administrador os detalhes de cada poste.

18. O sistema deve permitir que o administrador visualize o registo de anomalias de cada poste.

19. O sistema deve permitir que o administrador altere todas as configurações passíveis de serem realizadas OTA de cada poste.

20. O sistema deve permitir que o administrador registe as manutenções realizadas em cada poste.

21. O sistema deve permitir que o administrador altere o estado operativo do poste.

22. O sistema deve permitir que o administrador force o estado da luminária.

23. O sistema deve permitir que o administrador pesquise passadeiras inteligentes.

5.3.3 Requisitos Não Funcionais

1. O sistema deve seguir o paradigma *web-based* (aplicação Web).

2. O sistema deve ser compatível com diferentes ecossistemas e dispositivos - desktop e *mobile* - garantindo que o *front-end* seja responsivo.

3. O sistema deve permitir que o *front-end* comunique com o *back-end* seguindo o paradigma REST.

4. O sistema deve utilizar *dashboards* compostas por cartões no *front-end*.

5. O sistema deve atualizar as *dashboards* no *front-end* automaticamente sempre que existam mudanças de estado no *back-end*.

6. O sistema deve ter o *front-end* desenvolvido utilizando *React*.

7. O sistema deve utilizar a biblioteca de componentes *Material-UI* no *front-end*.

5.3.4 Diagrama de Casos de Uso

O diagrama de casos de uso apresentado na Fig. 5.1 ilustra as principais interações entre os atores e o sistema VALLPASS. Este diagrama foi elaborado com base nos requisitos funcionais descritos na Subsecção 5.3.2.

5.4 Modelo conceptual de dados

Nesta Secção, apresenta-se o modelo conceptual de dados do sistema VALLPASS, que serve de base para compreender as entidades envolvidas e os relacionamentos entre elas. Este modelo facilita a visualização da estrutura do sistema e estabelece as fundações para a definição detalhada do modelo de dados semântico NGSI-LD.

5.4.1 Diagrama ER

A Fig. 5.2 apresenta o diagrama ER que ilustra o modelo conceptual de dados do sistema VALLPASS. Este diagrama detalha as entidades principais, atributos, e as interligações essenciais entre elas, fornecendo uma visão holística dos elementos e da sua integração no contexto do sistema.

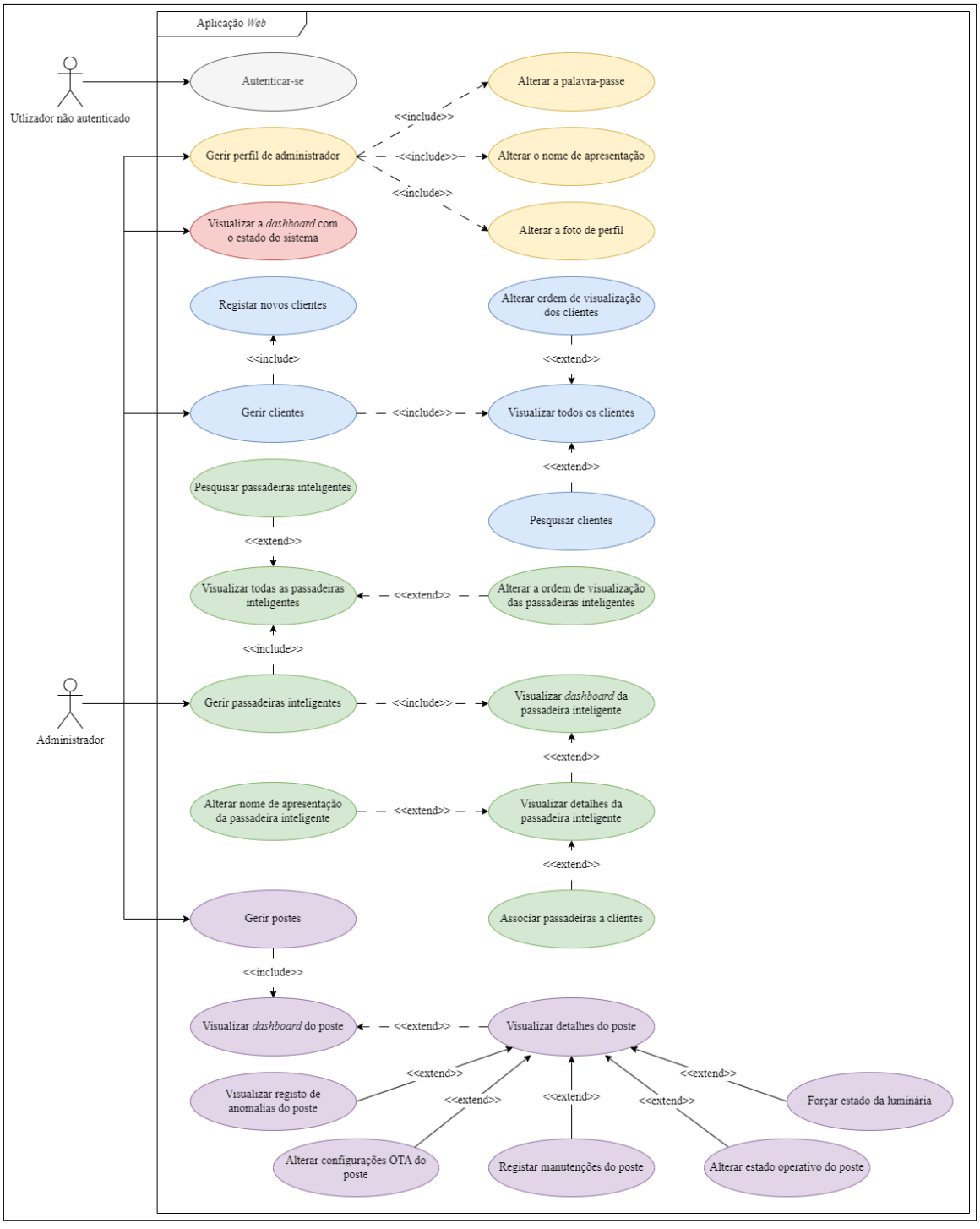

Figura 5.1: Diagrama de casos de uso para a Aplicação *Web*

5.4.2 Descrição das entidades

- **Empresa**

  - **Descrição**: Representa a organização que desenvolve, comercializa e mantém o sistema VALLPASS.

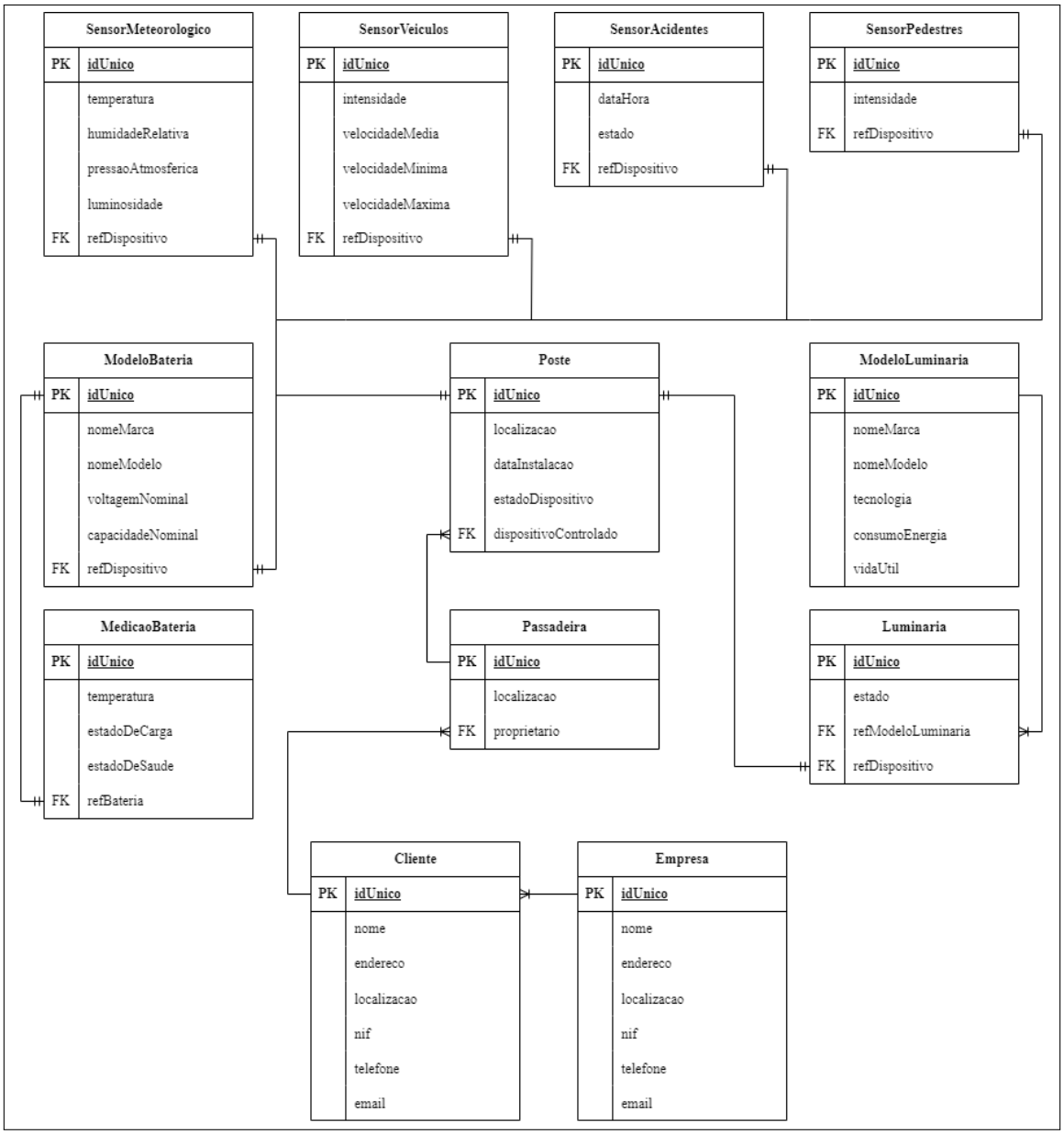

Figura 5.2: Diagrama ER do modelo concetual de dados do sistema VALLPASS

  - **Atributos Principais**: Nome, Endereço, Informação de Contacto.
  - **Relacionamentos**:
    - Possui múltiplos Clientes.

- **Cliente**
  - **Descrição**: Organizações que adquirem e utilizam o sistema VALLPASS.
  - **Atributos Principais**: Nome, Endereço, Informação de Contacto.
  - **Relacionamentos**:
    - Cliente de uma Empresa.

    * Possui múltiplas Passadeiras.

- **Passadeira**
  - **Descrição**: Representa uma passadeira inteligente equipada com tecnologia VALLPASS.
  - **Atributos Principais**: Identificador e Localização
  - **Relacionamentos**:
    * Está associada a um Cliente.
    * É composta por múltiplos (dois) Postes.

- **Poste**
  - **Descrição**: Poste de iluminação, equipado com vários sensores e uma luminária.
  - **Atributos Principais**: Identificador e Estado Operacional.
  - **Relacionamentos**:
    * Faz parte de uma Passadeira.
    * Contém múltiplos Sensores e uma Luminária.

- **SensorMeteorológico**
  - **Descrição**: Sensor responsável pela medição da temperatura, humidade relativa, pressão atmosférica e brilho ambiente.
  - **Atributos Principais**: Leitura de Temperatura, Leitura de Humidade, Leitura de Pressão, Leitura de luminosidade.
  - **Relacionamentos**:
    * Está associado a um Poste.

- **SensorVeículos**
  - **Descrição**: Sensor que contabiliza o fluxo de veículos.
  - **Atributos Principais**: Contagem de Veículos, Velocidade Mínima, Velocidade Média e Velocidade Máxima.
  - **Relacionamentos**:
    * Está associado a um Poste.

- **SensorPeões**
  - **Descrição**: Sensor que contabiliza o fluxo de peões.

  - **Atributos Principais**: Contagem de Peões.
  - **Relacionamentos**:
    * Está associado a um Poste.

- **SensorAcidentes**
  - **Descrição**: Sensor, do tipo, acelerómetro, que monitoriza a integridade estrutural do Poste.
  - **Atributos Principais**: Data e Hora, Estado de Resolução.
  - **Relacionamentos**:
    * Está associado a um Poste.

- **ModeloLuminária**
  - **Descrição**: Especificações técnicas da luminária instalada no poste (atributos estáticos).
  - **Atributos Principais**: Tipo de Lâmpada, Potência, Fluxo Luminoso.
  - **Relacionamentos**:
    * Define as características de uma Luminária.

- **Luminária**
  - **Descrição**: Atuador responsável pela iluminação na passadeira.
  - **Atributos Principais**: Estado (Ligada/Desligada).
  - **Relacionamentos**:
    * É parte de um Poste.

- **ModeloBateria**
  - **Descrição**: Especificações técnicas da bateria utilizada no poste (atributos estáticos).
  - **Atributos Principais**: Marca, Modelo, Voltagem Nominal e Capacidade Nominal.
  - **Relacionamentos**:
    * Define as características de uma Bateria.
    * É parte de um Poste

- **MediçãoBateria**

- **Descrição**: Registo das medições atuais da bateria.
- **Atributos Principais**: Temperatura, Nível de Carga e Estado de Saúde.
- **Relacionamentos**:
  * Refere-se a uma Bateria.

Note-se que, neste contexto, a palavra "sensor" refere-se a dispositivos virtuais, e não a dispositivos físicos. Por exemplo, o SensorMeteorologico representa, na verdade, dois sensores físicos distintos: um que mede a temperatura, a humidade relativa e a pressão atmosférica, e outro que mede o brilho ambiente.

5.5 Definição do modelo de dados semântico NGSI-LD

Esta Secção é uma adaptação de [15], © 2023 IEEE. As modificações incluem a tradução do texto e reescrita de algumas partes.

Nesta Secção, será descrito o processo sistemático utilizado para desenvolver o modelo de dados NGSI-LD para o sistema VALLPASS, bem como a metodologia proposta para a criação de modelos de dados no âmbito da IoT. Embora seja possível criar o modelo de dados com diversas ferramentas, este será delineado de acordo com a especificação OpenAPI Versão 3.1, especificamente através dos seus objetos do tipo Schema [86]. Nesta versão do OpenAPI, o objeto Paths não é necessário. Por conseguinte, não é preciso definir caminhos e operações para a API, visto que estes já estão especificados na API NGSI-LD. Contudo, para garantir a validade da especificação em versões anteriores à 3.1, onde o objeto Paths é obrigatório, pode-se definir um caminho fictício como solução alternativa.

5.5.1 Modelos de dados de referência

Ao desenvolver um modelo de dados semânticos, como um modelo de informações NGSI-LD, não é necessário, e até mesmo desaconselhável, começar do zero, uma vez que o objetivo principal é adotar ontologias comuns e partilhadas. Assim, sempre que possível, serão utilizados os modelos de dados da Smart Data Models Initiative como ponto de partida para representar as entidades relacionadas com o nosso cenário.

Conforme já mencionado, os sensores no contexto atual do sistema VALLPASS representam dispositivos virtuais, em vez de dispositivos físicos. Em situações como esta, onde um sistema de sensores virtuais realiza diversos tipos de medições, [87] recomenda modelá-los como um sistema de sensores onde cada sensor está ligado a um concentrador, que, no caso do VALLPASS, será o poste de iluminação. Seguindo esta abordagem, o SensorMeteorologico será subdividido em dois sensores distintos:

- **SensorTemperaturaHumidadePressao** - responsável pela monitorização da temperatura, da humidade relativa e da pressão atmosférica.
- **SensorLuminosidade** - encarregado de monitorizar o brilho ambiente.

Para cada uma das entidades identificadas no sistema VALLPASS, será realizada uma pesquisa para encontrar o SDM mais adequado, se disponível. Caso o SDM encontrado não represente fielmente a entidade em análise, isso não constitui um problema, uma vez que pode ser ajustado conforme necessário.

Apresentam-se de seguida os SDMs de referência encontrados para as entidades do sistema VALLPASS.

Empresa O SDM "Organisation" [88], atribuído ao domínio Cross-Sector, será utilizado. Trata-se de um modelo de dados genérico, concebido para representar diversos tipos de organizações, sendo, por isso, bastante adequado para esta entidade.

Cliente Os clientes-alvo do sistema VALLPASS serão organizações, pelo que será utilizado o mesmo SDM da entidade Empresa.

Passadeira Para esta entidade, não foi encontrado um SDM específico. Será utilizado o SDM "RoadSegment" [89] do domínio Smart Cities. Embora seja destinado a representar segmentos de estrada, tem uma certa compatibilidade [89].

Poste Para esta entidade, não foi encontrado um SDM específico; no entanto, a entidade Poste pode ser vista como um dispositivo IoT que, além de ser um atuador por si só, conterá outros sensores e outro atuador - a luminária. Por esta razão, será utilizado o SDM "Device" [90], associado ao domínio Smart Sensoring, que se destina a representar um "aparelho (*hardware* + *software* + *firmware*) com o objetivo de cumprir uma determinada tarefa (monitorizar o ambiente, atuar, etc.)"[90].

SensorTemperaturaHumidadePressao Tal como a entidade Poste, esta entidade também pode ser vista como um dispositivo IoT (sensor, neste caso), mas considera-se mais apropriado representá-la como uma entidade que descreve determinadas medições provenientes do Poste. Sob esta perspetiva, o SDM "WeatherObserved" [91], associado ao domínio Smart Environment, ajusta-se perfeitamente, uma vez que é destinado a descrever observações meteorológicas e define todas as propriedades necessárias para esta entidade [91].

SensorLuminosidade Assim como a entidade SensorTemperaturaHumidadePressao, esta entidade não é considerada um dispositivo IoT por si só, mas sim uma entidade que descreve determinadas medições provenientes do Poste. Assim, será utilizado o mesmo SDM da entidade SensorTemperaturaHumidadePressao, que também define todas as propriedades necessárias para esta entidade.

SensorVeiculos Para esta entidade, não foi encontrado um SDM específico. Será utilizado o SDM "ItemFlowObserved" [92], que pertence ao domínio Smart Cities e destina-se a descrever "uma observação relacionada com o movimento de um item num determinado local e durante um determinado período"[92]. Embora seja um SDM genérico, é compatível, visto que o sensor associado a esta entidade monitoriza o fluxo de um item - veículos - durante um certo período.

SensorPeoes Tal como o sensor relacionado com a entidade SensorVeiculos, o sensor associado a esta entidade também monitoriza o fluxo de um item - peões - durante um determinado período, pelo que será utilizado o mesmo SDM da entidade SensorVeiculos.

SensorAcidentes Para esta entidade, será utilizado o SDM "RoadAccident" [93]. Este modelo pertence ao domínio Smart Cities e é destinado a descrever um acidente rodoviário, incluindo as suas causas e consequências [93].

ModeloLuminaria O SDM "StreetlightModel", associado ao domínio Smart Cities, foi escolhido porque se destina a descrever as características técnicas de um poste de iluminação, incluindo a luminária (que é o nosso foco de interesse) [94].

Luminaria Será utilizado o SDM "Streetlight", também associado ao domínio Smart Cities, que permite representar o estado operacional de uma luminária [95].

ModeloBateria Para esta entidade, foi escolhido o SDM "StorageBatteryDevice", pertencente ao domínio Cross-Sector, que se destina especificamente a descrever as características técnicas de uma bateria [96].

MedicaoBateria O SDM "StorageBatteryMeasurement" será utilizado para esta entidade [97]. Atribuído ao domínio Cross-Sector, permite descrever o estado atual de uma bateria (nomeadamente a sua capacidade energética restante) [97].

Os esquemas dos SDMs selecionados são integrados no modelo de dados do sistema VALLPASS por referência, utilizando a palavra-chave `$ref`.

Antes de avançar, é pertinente clarificar como as relações são definidas no modelo de dados semânticos NGSI-LD. Ao contrário dos sistemas de bases de dados relacionais, onde as relações são estabelecidas utilizando chaves estrangeiras, no NGSI-LD as relações não são formalmente definidas da mesma forma. As relações são criadas ou modificadas no momento em que as entidades são manipuladas, especificamente através das operações Create (criar), Read (ler), Update (atualizar), Delete (excluir) (CRUD) fornecidas pela API NGSI-LD. O que se pode fazer é definir, nas entidades que formarão as relações, as propriedades que incentivarão (mas não "forçarão") essas relações. Por exemplo, o atributo obrigatório `controlledAsset` da entidade Poste será preenchido com

o Uniform Resource Name, ou Nome Uniforme de Recurso (URN) da Passadeira que irá controlar (ou da qual fará parte), estabelecendo assim uma relação de um para muitos. De certa forma, este atributo comporta-se como uma chave estrangeira, contudo, não impõe restrições sobre o tipo de entidade cujo URN será atribuído, podendo receber o URN de qualquer entidade.

5.5.2 Ajuste dos modelos de dados de referência

Os SDMs escolhidos anteriormente como base constituem um ponto de partida sólido. No entanto, pode ser necessário ajustar a definição das entidades para criar um GD que represente de forma fiel o objeto físico correspondente. As alterações feitas a cada entidade do sistema VALLPASS (quando aplicável) podem envolver a adição de novos esquemas e propriedades.

Tal como os esquemas dos SDMs previamente selecionados, novos esquemas também são adicionados por referência e combinados com os existentes, utilizando a propriedade `allOf` do objeto Schema para realizar a composição.

Sempre que for necessário adicionar novas propriedades, existem duas abordagens possíveis: uma consiste em adicioná-las diretamente à entidade, utilizando a palavra-chave `properties` do objeto Schema. Nesse caso, cada uma dessas propriedades seria, então, um objeto Schema, e como se está a desenvolver um modelo de dados semântico, é essencial que, sempre que possível, se faça referência a definições já existentes, como aquelas de propriedades semelhantes definidas num SDM. A outra abordagem seria adicionar um novo esquema que defina as propriedades a serem incluídas. A escolha entre uma opção e outra dependerá de dois fatores: o número de propriedades a adicionar e a relação entre elas. Por exemplo, se for necessário adicionar várias propriedades a diferentes entidades que estão todas definidas num mesmo esquema, pode ser mais eficiente adicionar esse esquema à entidade, em vez de adicionar as propriedades uma a uma, tornando assim o modelo de dados mais compacto.

Note-se que o modelo de dados do sistema VALLPASS especifica quais propriedades são obrigatórias, ou seja, as que devem ser definidas no momento da criação de novas entidades. É importante destacar que, embora não seja um requisito obrigatório, esta prática é recomendada para garantir a consistência dos dados. A seguir, são descritas as alterações feitas ao modelo de dados base (ao nível das entidades).

Empresa As propriedades telefone e e-mail não estão presentes no esquema base selecionado anteriormente para esta entidade. Para adicionar essas propriedades, optou-se por incluir um novo esquema à entidade - uma definição OpenAPI do tipo Schema.org ”ContactPoint- que também define outras propriedades relacionadas com um ponto de contacto, as quais podem ser úteis no futuro [98, 99].

Cliente Tal como a entidade Empresa, é necessário adicionar as propriedades telefone e e-mail a esta entidade, pelo que foi aplicada a mesma solução. Esta entidade representa o lado ”muitos”de uma relação de um para muitos com a entidade Empresa, pelo que deverá conter uma propriedade que receba o URN da Empresa, indicando assim a relação. Esta propriedade não está definida no esquema base, sendo desenvolvida conforme apresentado na Listagem 5.1 sob o nome `refProvider`. Na definição atual, utiliza-se o esquema “EntityIdentifierType” [100], que faz parte de um conjunto de esquemas subjacente a todos os SDMs e que se destina a identificar inequivocamente qualquer entidade NGSI. O campo `x-ngsi` é uma extensão da especificação [86] e, como o nome sugere, serve para expandir a especificação OpenAPI em desenvolvimento. Todos os campos associados a essas propriedades devem começar com o prefixo `-x`. Embora não seja obrigatório definir essas propriedades [101], é uma boa prática, especialmente se se pretender utilizar o modelo de dados fora de aplicações NGSI-LD, como em aplicações JSON-LD ”genéricas”.

SensorAcidentes O esquema base escolhido para esta entidade não define uma propriedade que descreva a relação entre esta entidade e a entidade Poste (uma relação um-para-um), ou seja, a propriedade que indica qual o Poste associado a um acidente. Por este motivo, tal propriedade foi adicionada. Relativamente à definição desta propriedade, ela baseia-se na propriedade `refDevice`, presente no modelo de dados ”ItemsFlowObserved”(utilizado nas entidades ”SensorPeoes”e ”SensorVeiculos”) e destinada a identificar de forma única ”o dispositivo ou dispositivos utilizados para obter os dados expressos por este registo”[92]. Embora genérica, esta definição adequa-se perfeitamente à entidade em questão.

5.5.3 Ficheiro NGSI-LD `@context`

Com o modelo de dados semânticos criado, o próximo passo para a sua utilização na plataforma de gestão remota VALLPASS consiste em gerar o ficheiro NGSI-LD `@context`. Este ficheiro será responsável por expandir os termos, convertendo cadeias de caracteres abreviadas em conceitos, especificados pelos respetivos URIs, e também por compactar os URIs em termos abreviados. Deste modo, as aplicações que interagem com o corretor NGSI-LD do VALLPASS poderão ”compreender”programaticamente os dados armazenados. Mais especificamente, o ficheiro NGSI-LD `@context` especificará os URIs correspondentes aos tipos de entidades, propriedades e metadados (propriedades de propriedades).

Listagem 5.1: Definição da propriedade `refProvider` para a entidade Cliente do modelo de dados semântico VALLPASS

```
{
    refProvider:
        description: "Organization from which the system was
        acquired."
        anyOf:
            - description: Property. Identifier format of any
            NGSI entity
            type: string
            minLength: 1
            maxLength: 256
            pattern: ^[\w\-\.\{\}\$\+\*\[\]‘|~^@!,:\\]+$
            - description: Property. Identifier format of any
            NGSI entity
            type: string
            format: uri
        x-ngsi:
            type: Relationship
}
```

5.6 Definição da arquitetura da plataforma de gestão remota

5.6.1 Introdução

A plataforma de gestão remota descrita adota uma arquitetura de microsserviços que promove a independência operacional de cada componente, encapsulado em contentores Docker e interligado por interfaces bem definidas. Esta abordagem confere modularidade, escalabilidade e facilita a manutenção, possibilitando que cada serviço seja desenvolvido, implantado e escalado de forma autónoma.

A arquitetura inclui uma variedade de serviços colaborativos que oferecem uma solução completa para monitorização e controlo dos dispositivos VALLPASS, facilitando a comunicação entre sensores, atuadores e as aplicações de gestão. A conceção arquitetural assenta em princípios de modularidade e escalabilidade, utilizando tecnologias e padrões abertos, incluindo o *framework* FIWARE, para garantir interoperabilidade e simplificar a integração com sistemas externos.

Nesta Secção, apresenta-se uma visão geral dos serviços que integram a plataforma, das interações entre eles e da implementação da arquitetura de microsserviços através de contentores Docker.

5.6.2 Visão geral

A Fig. 5.3 ilustra a arquitetura geral da plataforma e as principais interações entre os seus serviços. Todos os GEs FIWARE foram identificados com o prefixo FIWARE, para os distinguir dos restantes. Abaixo, descrevem-se os principais serviços que integram esta arquitetura.

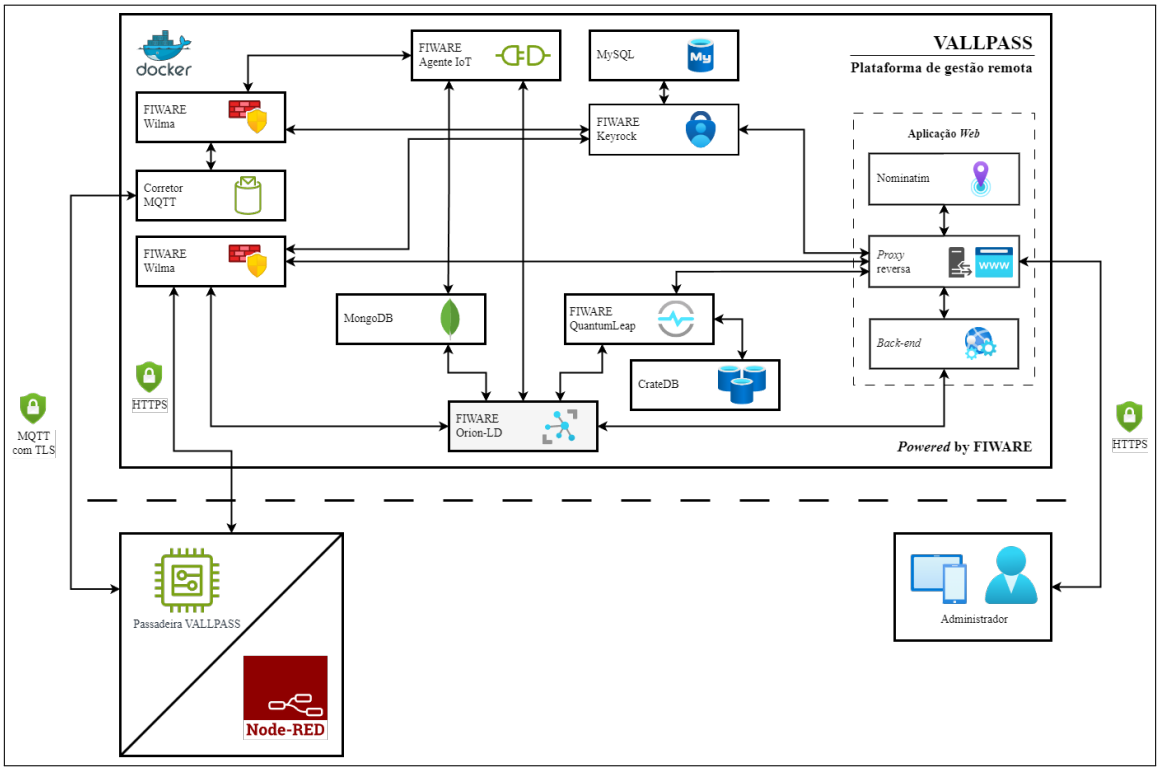

Figura 5.3: Diagrama da arquitetura da plataforma de gestão remota

Proxy Reversa - NGINX Servidor HTTP responsável pelo fornecimento de ficheiros estáticos que constituem a *build* de produção do *front-end* da Aplicação *Web*, desenvolvida em React, bem como dos ficheiros associados ao modelo de dados semântico VALLPASS, destacando-se os ficheiros NGSI-LD @context utilizados pelo Orion-LD. Além disso, o NGINX atua como *proxy* reversa para alguns serviços Docker acessíveis a partir do *front-end* da Aplicação *Web* - *back-end* da Aplicação *Web*, Nominatim e QuantumLeap - implementando autorização de cliente para os serviços Nominatim e QuantumLeap (a autorização dos pedidos ao *back-end* da Aplicação *Web* é processada pelo próprio serviço).

Back-end* da Aplicação *Web Desenvolvida em Express.js, esta componente suporta a atualização dinâmica das *dashboards* do *front-end* e também é utilizada na gestão de contexto.

FIWARE Keyrock Proporciona autenticação OAuth2 para o *front-end* da Aplicação *Web* e atua como Ponto de Aplicação de Política (PDP) para as *proxies* PEP FIWARE Wilma, assegurando a gestão de acessos e o controlo de permissões de forma centralizada. Armazena as identidades dos utilizadores, aplicações, funções e permissões na base de dados MySQL.

FIWARE Wilma São utilizadas duas destas *proxies* PEP: uma protege o acesso ao Orion-LD (com origem nas passadeiras VALLPASS e no *front-end* da Aplicação *Web*) e a outra é responsável por proteger o acesso à porta sul do Agente IoT para JSON, ao qual os postes das passadeiras VALLPASS se conectam através do corretor MQTT. Sempre que um utilizador (ou dispositivo) tenta aceder a um recurso protegido (atrás de um *proxy* PEP), este *proxy* descreve os atributos do utilizador ao PDP, solicita uma decisão de segurança - permitir ou rejeitar o acesso - e aplica essa decisão conforme determinado [102]. O papel de PDP é desempenhado pelo FIWARE Keyrock.

FIWARE Orion-LD Responsável pela gestão da informação de contexto. Utiliza uma base de dados MongoDB para armazenar o estado atual da informação de contexto e outros tipos de informação, como os dados associados às subscrições [103].

FIWARE Agente IoT para JSON Facilita a comunicação entre dispositivos IoT e o Orion-LD, funcionando como um ”tradutor”entre as comunicações MQTT (com *payload* em JSON) e a interface NGSI-LD disponibilizada pelo Orion-LD, permitindo comunicação bidirecional. Armazena informação relativa aos dispositivos (como chaves) na base de dados MongoDB [3].

FIWARE QuantumLeap Permite a persistência de informação de contexto histórica na base de dados de séries temporais CrateDB e a realização de consultas (através de uma API que envolve o *back-end* da CrateDB) [2, 104].

Nominatim Serviço de geocodificação [105] utilizado no suporte à pesquisa por localização das passadeiras inteligentes no *front-end* da Aplicação *Web*

Corretor MQTT - Eclipse Mosquitto Utilizado nas comunicações MQTT entre os postes das passadeiras VALLPASS e a plataforma de gestão remota, estabelecendo a conexão com o FIWARE Agente IoT para JSON (vio *proxy* PEP FIWARE Wilma).

Node-RED Utilizado para simular os postes das passadeiras VALLPASS no desenvolvimento do protótipo da plataforma de gestão remota.

5.6.3 *Endpoints* do *proxy* reversa (NGINX)

Foram definidos os seguintes *endpoints* para o *proxy* reversa:

- Servidor HTTP para os seguintes ficheiros estáticos:
  - `/management` - *Build* de produção do *front-end* da Aplicação *Web*
  - `/data-model` - Modelo de dados semântico VALLPASS (*autoindexing*)
    - Ficheiro `.yaml` com a especificação OpenAPI
    - Ficheiros NGSI-LD @context
    - Documentação
- *Proxy* reversa para os seguintes serviços, acedidos pelo *front-end* da Aplicação *Web*:
  - `/backend` - *Back-end* da Aplicação *Web*
  - `/nominatim` - Nominatim
  - `/time-series-data` - QuantumLeap

5.7 Prototipagem da plataforma de gestão remota

5.7.1 Visão geral

De modo a validar tanto o modelo de dados semântico desenvolvido quanto a arquitetura proposta para a plataforma de gestão remota, além de apoiar o desenvolvimento da Aplicação Web, foi efetuada a prototipagem da plataforma de gestão remota, incidindo em duas frentes:

1. Implementação e configuração dos serviços definidos na arquitetura apresentada na Secção 5.6
2. Simulação do sistema VALLPASS

5.7.2 Implementação e configuração dos serviços da plataforma de gestão remota

A orquestração dos contentores Docker é realizada através do Docker Compose, tendo-se definido os serviços, as suas configurações, redes, volumes e dependências num ficheiro de configuração `docker-compose.yml`. Este processo permite implementar e configurar de forma simplificada todos os serviços necessários, promovendo a modularidade e a estabilidade do ambiente. Com esta abordagem declarativa, toda a plataforma pode ser gerida de forma reproduzível, facilitando a implementação e a gestão dos componentes em contentores Docker.

Para permitir uma comunicação consistente entre serviços e organizar o tráfego interno de forma isolada do restante da infraestrutura, foi criada uma rede específica, denominada `vallpass`, com um intervalo de endereços IP fixos atribuídos a cada serviço.

Embora voltada para a prototipagem, a implementação aplica práticas que aproximam o ambiente de um cenário de produção. Foram configurados volumes persistentes para assegurar a integridade e a durabilidade dos dados, especialmente nas bases de dados: MongoDB, MySQL e CrateDB. Esses volumes garantem que os dados permaneçam disponíveis e inalterados, mesmo em casos de reinicialização ou atualização dos contentores. Adicionalmente também foram definidos volumes adicionais para *logs* e configurações de serviços como o Mosquitto e o Node-RED (fluxos). Estas configurações específicas, implementadas via `docker-compose.yml`, reforçam a integridade dos dados e a continuidade operacional da plataforma.

Quanto à segurança, foram utilizados `secrets` para armazenar credenciais sensíveis, como palavras-passe do MySQL e do Keyrock, e credenciais de autenticação para *proxies* e agentes IoT. Esses `secrets`, configurados no `docker-compose.yml` e armazenados em ficheiros externos, reduzem a exposição a dados sensíveis, proporcionando uma camada extra de segurança.

Apesar de configurada para prototipagem, a implementação adota práticas robustas que facilitam uma transição futura para um ambiente de produção seguro e resiliente, com uma infraestrutura sólida em termos de gestão de dados e segurança.

5.7.3 Simulação do sistema VALLPASS

Para simular o sistema VALLPASS, foi definido um cenário de simulação com uma passadeira inteligente (2 postes).

Na Fig. 5.4 apresenta-se o diagrama de atividades que especifica o processo de provisionamento automático das passadeiras inteligentes, incluindo, em forma de notas, os fluxos associados do Node-RED, apresentados de seguida.

Fluxo 1 - Entidades Na Fig. 5.5 encontra-se o fluxo Node-RED associado à criação das entidades Empresa, Cliente, Passadeira 0, Passadeira 1, Modelo luminária, Modelo bateria Poste 0 e Modelo bateria Poste 1. Para este fim são efetuados 6 pedidos HTTP POST com os atributos das entidades - um para cada entidade - ao corretor de contexto Orion-LD; na Listagem 5.2 encontra-se, a título de exemplo, a *payload* utilizada na criação da entidade Passadeira 0 (associada ao respetivo nó).

Fluxo 2 - Grupos de serviço A Fig. 5.6 representa o fluxo Node-RED utilizado para o provisionamento dos serviços, o primeiro passo no provisionamento de dispositivos [3].

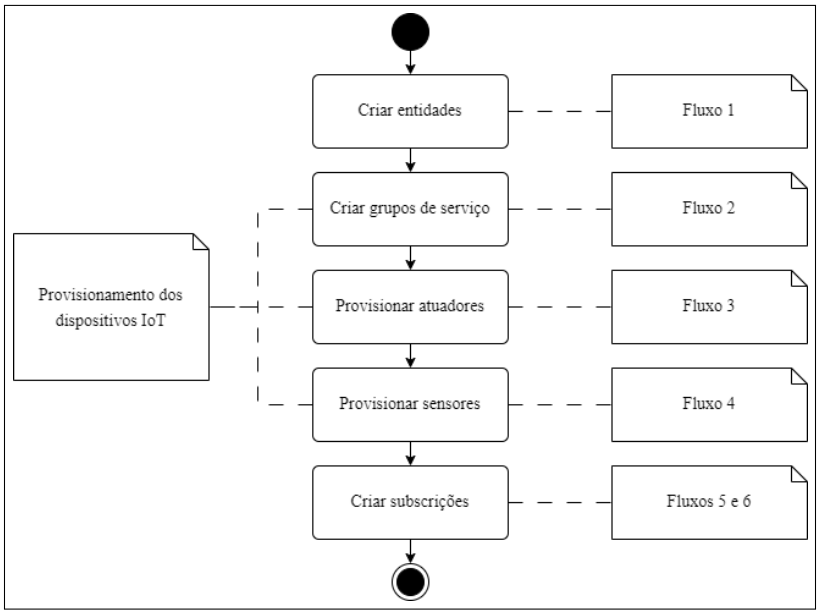

Figura 5.4: Diagrama de atividades associado ao processo de provisionamento automático das duas passadeiras inteligentes

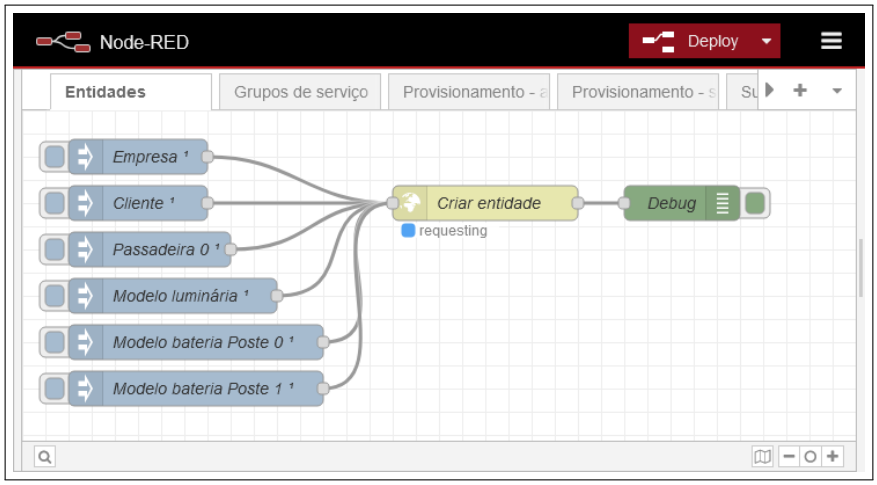

Figura 5.5: Fluxo 1 Node-RED - criação de entidades

Estes serviços configuram um conjunto comum de parâmetros para um grupo de dispositivos [106], evitando a repetição de configurações no subsequente provisionamento individual de cada dispositivo. Outra vantagem reside no facto de, ao definirmos uma API Key para cada serviço, protegermos o acesso ao grupo de dispositivos correspondente, isolando também as comunicações associadas a esse grupo e permitindo assim um controlo mais granular dos dispostivos [106]. No caso presente, esta API Key será utilizada nas comunicações com o corretor MQTT. Um caso de uso relevante para esta funcionalidade seria, por exemplo, o isolamento dos dispositivos de cada cliente. Na Listagem 5.3 encontra-se a *payload* JSON utilizada na criação do grupo de serviço para os sensores de luminosidade.

Listagem 5.2: *Payload* JSON utilizada na criação da entidade Passadeira 0

```
{
    "id": "urn:ngsi-ld:Crosswalk:IPB_BGC_0000000",
    "type": "Crosswalk",
    "name": {
        "type": "Property",
        "value": "IPB_BGC_0000000"
    },
    "location": {
        "type": "Point",
        "coordinates": [
            41.8184594659375,
            -6.749247147510722
        ]
    },
    "owner": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:Client:ESTiG"
    },
    "@context": "http://reverse-proxy/data-model/context-files
       /context-ngsi.jsonld"
}
```

Listagem 5.3: *Payload* JSON utilizada na criação do grupo de serviço para os sensores de luminosidade

```
{
    "services": [
        {
            "apikey": "a8obddvi6hg6yxnyqozr",
            "entity_type": "BrightnessSensor",
            "resource": "/iot/json"
        }
    ]
}
```

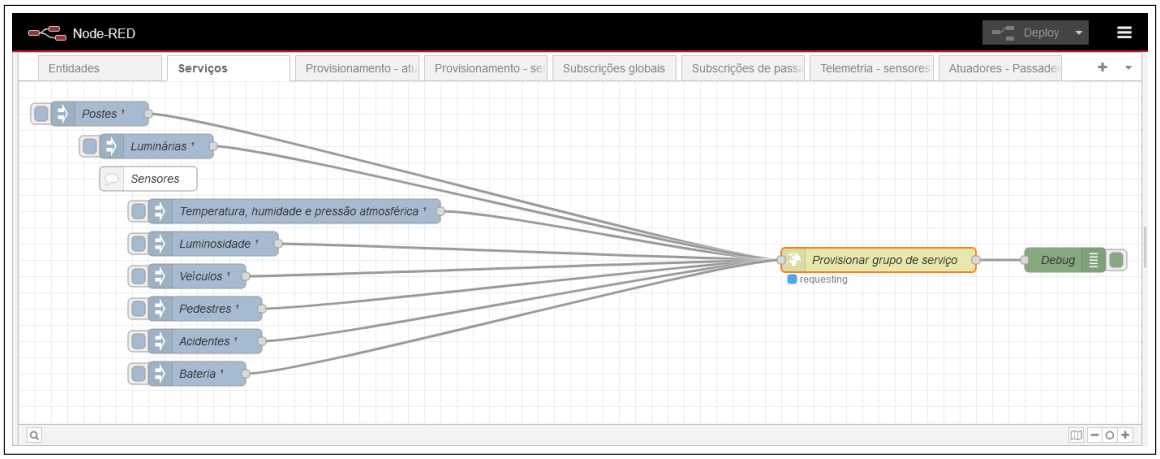

Figura 5.6: Fluxo 2 Node-RED - criação dos serviços

Fluxo 3 - Provisionamento dos atuadores O fluxo Node-RED utilizado para simular o provisionamento dos atuadores é mostrado na Fig. 5.7. Neste processo, são declarados os comandos suportados pelo atuador, bem como os seus atributos estáticos; na Listagem 5.4 encontra-se, como exemplo, a *payload* HTTP POST utilizada no provisionamento da Luminária integrada no Poste 0 da Passadeira 0. Na declaração dos comandos, que no contexto presente, se resumem às ações de ligar e desligar os atuadores, não se podia simplesmente declarar apenas um comando com o mesmo nome do atributo correspondente no modelo de dados semântico - `deviceState`. Em primeiro lugar, essa abordagem não é recomendada (os comandos devem ser atómicos); em segundo lugar, ele seria automaticamente expandido em dois atributos sem significado direto no modelo de dados: `deviceState_info` - o resultado real do comando - e `deviceState_status` - o estado do comando [15]. Neste sentido, foram definidos dois comandos atómicos para cada atuador com a finalidade de alterar o seu estado operativo - `on` e `off` - e, para atualizar o atributo correto no modelo de dados semântico (`deviceState`), será aproveitado o serviço de subscrição fornecido pelo Orion-LD e estendida a funcionalidade do *back-end* da Aplicação *Web*, que vai processar os resultados de execução associados aos atributos `on_info` e `off_info` e atualizar o atributo correto no Orion-LD.

Fluxo 4 - Provisionamento dos sensores O fluxo Node-RED utilizado para simular o provisionamento dos sensores pode ser visualizado na Fig. 5.8. No provisionamento dos sensores, caracterizam-se as leituras que ele irá fornecer e, à semelhança dos atuadores, os seus atributos estáticos; a título de exemplo, na Listagem 5.5 encontra-se a *payload* HTTP POST utilizada no provisionamento do sensor de temperatura, humidade e pressão atmosférica presente no Poste 0 da Passadeira 0.

Fluxo 5 - Criação das subscrições globais Na Fig.5.9 é apresentado o fluxo do Node-RED responsável por configurar subscrições globais no Orion-LD, ou seja, subscrições

Listagem 5.4: *Payload* JSON utilizada na provisionamento da Luminária integrada no Poste 0 da Passadeira 0

```
{
    "devices": [
        {
            "device_id": "IPB_BGC_0000000_0_Luminaire",
            "entity_name": "urn:ngsi-ld:Luminaire:
               IPB_BGC_0000000_0_Luminaire",
            "entity_type": "Luminaire",
            "protocol": "PDI-IoTA-JSON",
            "transport": "MQTT",
            "commands": [
                {
                    "name": "on",
                    "type": "Property"
                },
                {
                    "name": "off",
                    "type": "Property"
                }
            ],
            "static_attributes": [
                {
                    "name": "powerState",
                    "type": "Property",
                    "value": ""
                },
                {
                    "name": "refStreetlightModel",
                    "type": "Relationship",
                    "value": "urn:ngsi-ld:LuminaireModel:
                       MicroplusGermany_Microled_80_4000"
                },
                {
                    "name": "refDevice",
                    "type": "Relationship",
                    "value": "urn:ngsi-ld:Pole:
                       IPB_BGC_0000000_0"
                }
            ]
        }
    ]
}
```

Listagem 5.5: *Payload* JSON utilizada na provisionamento do sensor de temperatura, humidade e pressão atmosférica integrada no Poste 0 da Passadeira 0

```
{
    "devices": [
        {
            "device_id": "IPB_BGC_0000000_0_THP",
            "entity_name": "urn:ngsi-ld:
                TemperatureHumidityPressureSensor:
                IPB_BGC_0000000_0_THP",
            "entity_type": "TemperatureHumidityPressureSensor"
                ,
            "protocol": "PDI-IoTA-JSON",
            "transport": "MQTT",
            "attributes": [
                {
                    "object_id": "t",
                    "name": "temperature",
                    "type": "Property"
                },
                {
                    "object_id": "h",
                    "name": "relativeHumidity",
                    "type": "Property"
                },
                {
                    "object_id": "p",
                    "name": "atmosphericPressure",
                    "type": "Property"
                }
            ],
            "static_attributes": [
                {
                    "name": "refDevice",
                    "type": "Relationship",
                    "value": "urn:ngsi-ld:Pole:
                        IPB_BGC_0000000_0"
                }
            ]
        }
    ]
}
```

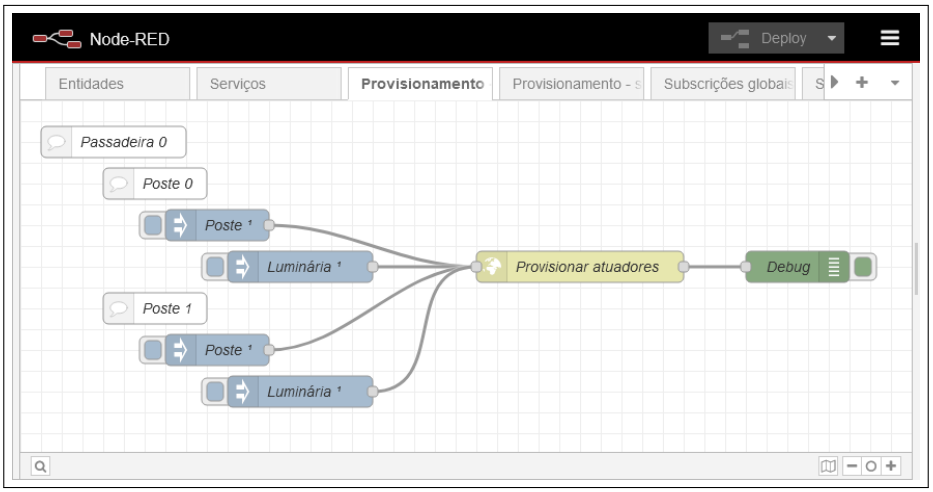

Figura 5.7: Fluxo 3 Node-RED - provisionamento dos atuadores

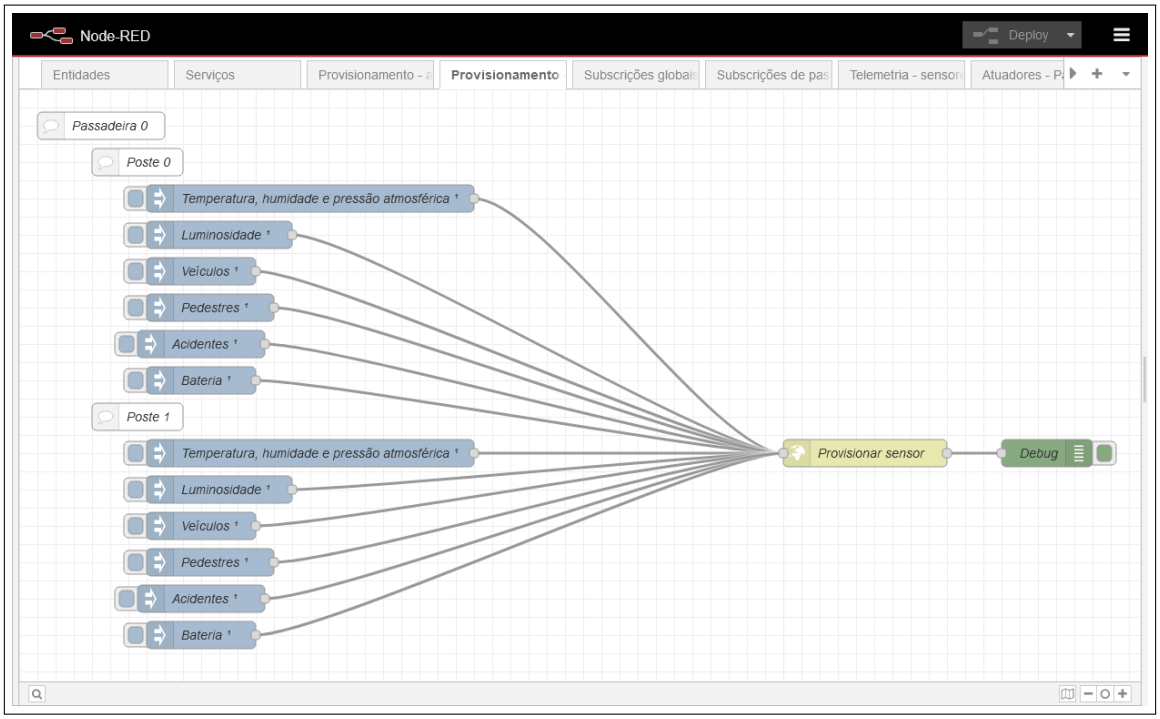

Figura 5.8: Fluxo 4 Node-RED - provisionamento dos sensores

que não são específicas a uma passadeira ou dispositivo. Antes de prosseguir, é importante compreender o conceito de subscrição no contexto do Orion-LD. As subscrições permitem que uma aplicação seja informada sempre que houver alterações na informação de contexto. Para isso, durante a criação das subscrições no Orion-LD, especifica-se, entre outros parâmetros, o *endpoint* da aplicação que receberá essas atualizações de contexto, bem como os atributos monitorizados, cuja alteração desencadeia o envio de um pedido HTTP POST para a aplicação subscrita. As subscrições globais configuradas neste fluxo Node-RED dividem-se em dois grupos principais: aquelas destinadas ao *back-end* da aplicação *Web* e aquelas destinadas ao QuantumLeap. As subscrições voltadas para o

back-end da aplicação *Web* permitem, conforme detalhado anteriormente no fluxo 3, que este serviço atualize o atributo que representa o estado operativo dos atuadores no modelo de dados semântico, com base nos resultados da execução dos comandos `on` e `off`. Estes resultados são armazenados nos atributos `on_info` e `off_info`, respetivamente. Assim, sempre que um atuador confirma o resultado de um comando, o Orion-LD notifica o *back-end* da aplicação *Web*, que, em resposta, realiza um pedido HTTP PATCH ao Orion-LD para atualizar o atributo correspondente com significância no modelo de dados semântico. As subscrições destinadas ao QuantumLeap, por outro lado, têm como objetivo assegurar que esta aplicação seja notificada sobre todas as alterações de contexto, permitindo que persista os dados temporais na base de dados CrateDB. Por exemplo, a Listagem 5.6 apresenta a *payload* HTTP POST utilizada para criar uma subscrição que informa o *back-end* da aplicação *Web* sobre o desligamento de uma luminária. A Listagem 5.7 ilustra a *payload* HTTP POST necessária para criar uma subscrição que notifica o QuantumLeap de uma alteração no estado de carga de uma bateria.

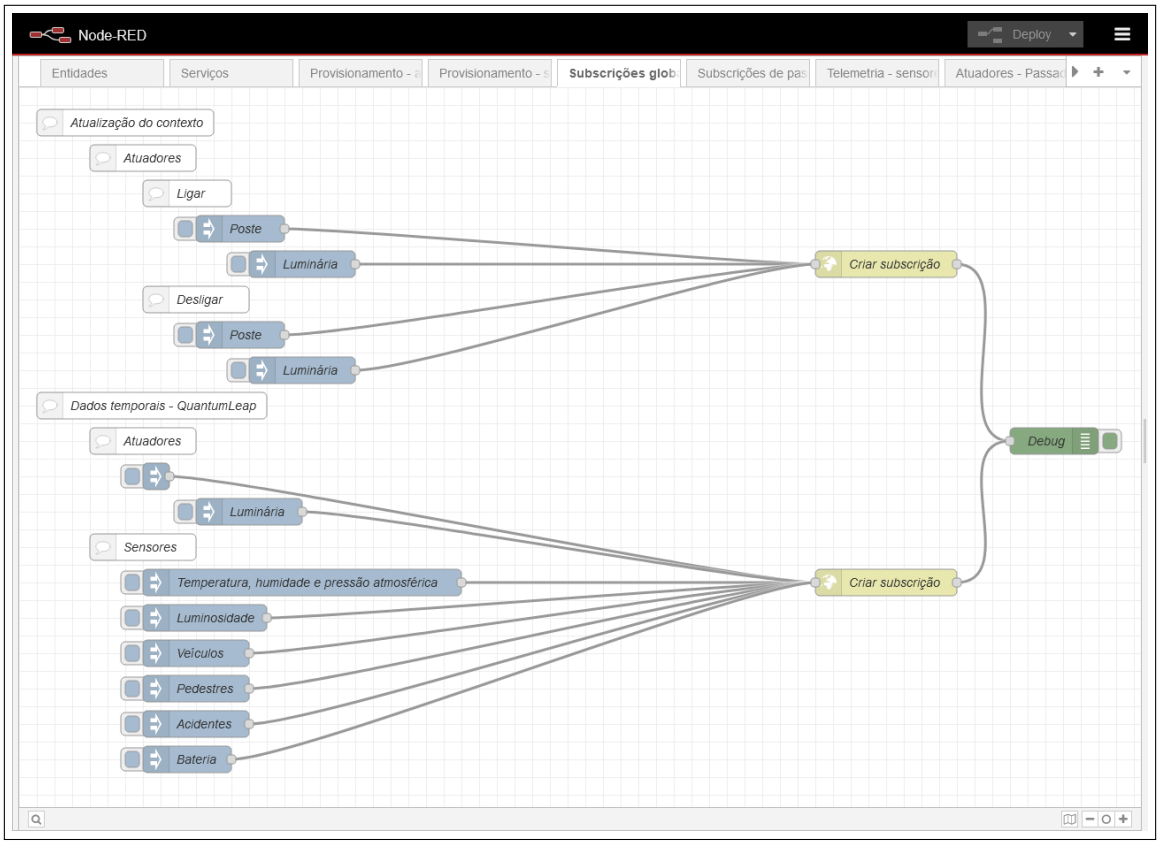

Figura 5.9: Fluxo 5 Node-RED - criação das subscrições globais

Fluxo 6 - Criação das subscrições de passadeira Na Fig.5.10, é ilustrado o fluxo do Node-RED encarregado de configurar subscrições específicas para cada passadeira ou, mais precisamente, para cada poste associado. As notificações geradas por estas

Listagem 5.6: *Payload* JSON utilizada na criação de uma subscrição global para notificar o *back-end* da Aplicação *Web* de que uma luminária foi desligada

```
{
    "description": "Atualizacao do contexto - luminaria",
    "type": "Subscription",
    "entities": [
        {
            "type": "Luminaire"
        }
    ],
    "watchedAttributes": [
        "off_info"
    ],
    "notification": {
        "attributes": [
            "off_info"
        ],
        "format": "keyValues",
        "endpoint": {
            "uri": "http://aplicacao-web-backend:49153/context
               -update/luminaire/off",
            "accept": "application/json"
        }
    },
    "@context": "http://reverse-proxy/data-model/context-files
       /context-ngsi.jsonld"
}
```

Listagem 5.7: *Payload* JSON utilizada na criação de uma subscrição global para notificar o QuantumLeap de alterações no estado de carga de uma bateria

```
{
    "description": "Dados temporais | QuantumLeap - sensor da
       bateria",
    "type": "Subscription",
    "entities": [
        {
            "type": "BatteryMeasurement"
        }
    ],
    "watchedAttributes": [
        "stateOfCharge"
    ],
    "notification": {
        "attributes": [
            "temperature",
            "stateOfCharge",
            "stateOfHealth"
        ],
        "format": "normalized",
        "endpoint": {
            "uri": "http://quantumleap:8668/v2/notify",
            "accept": "application/json"
        }
    },
    "@context": "http://reverse-proxy/data-model/context-files
       /context-ngsi.jsonld"
}
```

subscrições são enviadas para *endpoints* dedicados a cada poste, identificados de forma única, no *back-end* da aplicação *Web*. O objetivo destas notificações é permitir a atualização dinâmica das *dashboards* do *front-end* da aplicação *Web*, evitando que o utilizador precise de recarregar a página sempre que houver alterações no contexto do sistema VALLPASS. Como exemplo, encontra-se na Listagem 5.8 a *payload* HTTP POST utilizada na criação de uma subscrição deste tipo para o sensor de pedestres do Poste 1 da Passadeira 0.

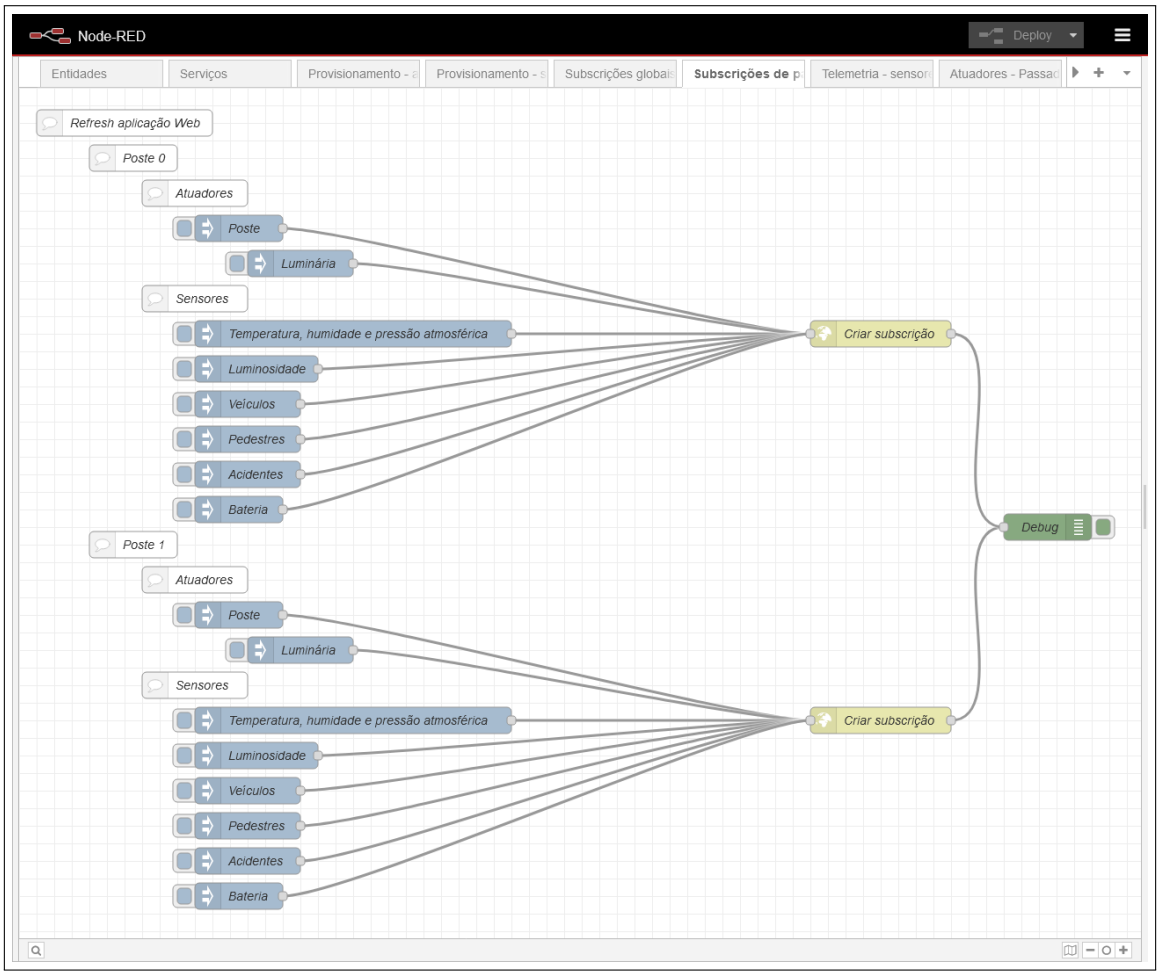

Figura 5.10: Fluxo 6 Node-RED - criação das subscrições de passadeira

Fluxo 7 - Simulação do envio de telemetria Este fluxo, representado na Fig. 5.11, possibilita a simulação do envio periódico de dados de telemetria provenientes dos sensores das passadeiras VALLPASS. Para cada sensor, gera-se periodicamente uma ou mais medições aleatórias, que são enviadas para o corretor MQTT. Os tópicos MQTT seguem a estrutura `/json/apiKey/idDispositivo/attrs`, onde `apiKey` corresponde à API Key definida nos serviços do fluxo 2 do Node-RED, e `idDispositivo` representa o identificador único atribuído aos dispositivos durante o seu provisionamento. Como exemplo, a Listagem 1

Listagem 5.8: *Payload* JSON utilizada na criação de uma subscrição de passadeira para notificar o *back-end* da Aplicação *Web* de novas medições oriundas do sensor de pedestres

```
{
    "description": "Refresh aplicacao Web | Poste
       IPB_BGC_0000000_1 - sensor de pedestres",
    "type": "Subscription",
    "entities": [
        {
            "type": "PedestrianSensor"
        }
    ],
    "watchedAttributes": [
        "intensity"
    ],
    "q": "refDevice == %22urn:ngsi-ld:Pole:IPB_BGC_0000000_1
       %22",
    "notification": {
        "attributes": [],
        "format": "keyValues",
        "endpoint": {
            "uri": "http://aplicacao-web-backend:49153/
               subscriptions/Pole/IPB_BGC_0000000_1",
            "accept": "application/json"
        }
    },
    "@context": "http://reverse-proxy/data-model/context-files
       /context-ngsi.jsonld"
}
```

exibe o processo de geração de uma *payload* MQTT contendo uma medição aleatória dos valores de temperatura, estado de carga e estado de saúde associados ao sensor de bateria instalado no Poste 0 da Passadeira 0.

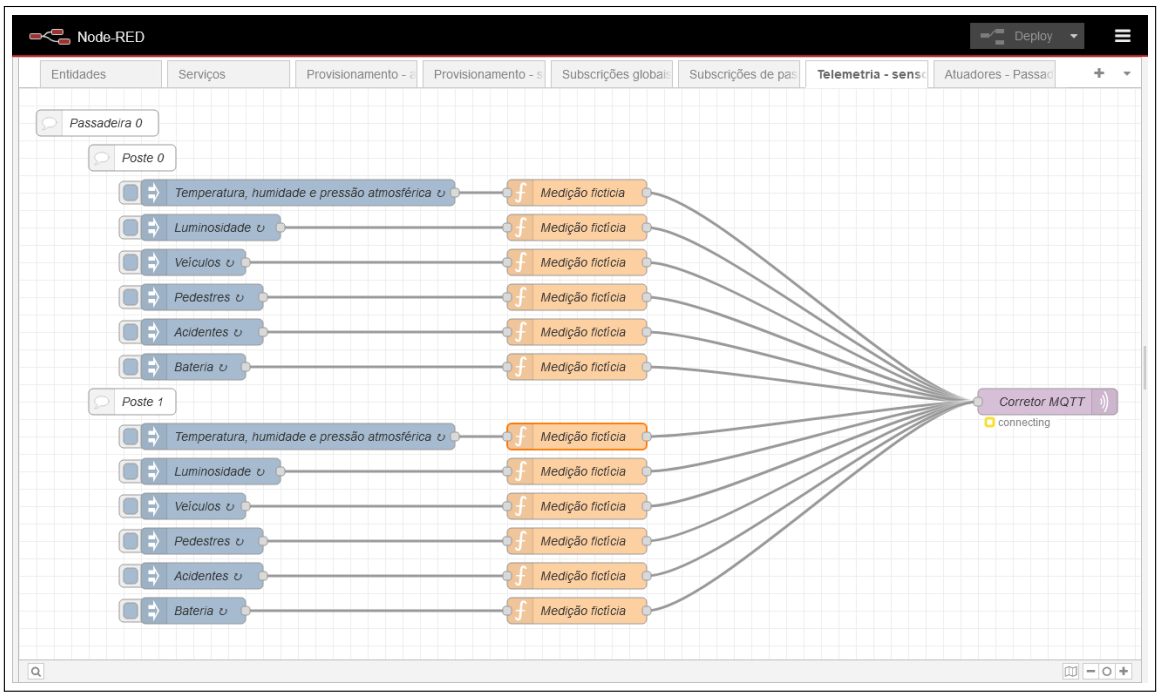

Figura 5.11: Fluxo 7 Node-RED - simulação do envio de telemetria com origem nos sensores

Fluxo 8 - Simulação da alteração do estado dos atuadores Este fluxo, apresentado na Fig. 5.12, simula a gestão remota dos atuadores - especificamente, do poste e da luminária - permitindo replicar as ações de ligar e desligar, que serão posteriormente implementadas no *front-end* da aplicação *Web*. Este fluxo inclui tanto o envio dos comandos (via pedidos HTTP PATCH) para o Orion-LD como a confirmação, por parte dos atuadores, da execução bem-sucedida dos mesmos, pois o contexto só é atualizado no Orion-LD após a receção dessa confirmação. Na Listagem 5.10, é exibida a *payload* MQTT que confirma a execução bem-sucedida do comando para ligar um atuador.

5.8 Desenvolvimento do *back-end* da Aplicação *Web*

5.8.1 Visão geral

O backend consiste numa aplicação desenvolvida com o *framework Express.js* e possui as finalidades descritas abaixo.

Listagem 5.9: Processo de geração de uma *payload* MQTT contendo uma medição aleatória dos valores de temperatura, estado de carga e estado de saúde associados ao sensor de bateria instalado no Poste 0 da Passadeira 0

```
1   {
2       function getRandomFloat(min, max, decimals) {
3           return (Math.random() * (max - min) + min).toFixed(
                decimals);
4       }
5
6       msg.payload = `{"t": ${getRandomFloat(0, 60, 0)}, "c": ${
            getRandomFloat(10, 100, 0)}, "h": ${getRandomFloat(30,
            100, 0)}}`;
7
8       return msg;
9   }
```

Listagem 5.10: *Payload* MQTT de confirmação que que o comando para ligar um atuador foi aplicado com sucesso

```
1   {
2       msg.payload = '{"on": "OK"}';
3       return msg;
4   }
```

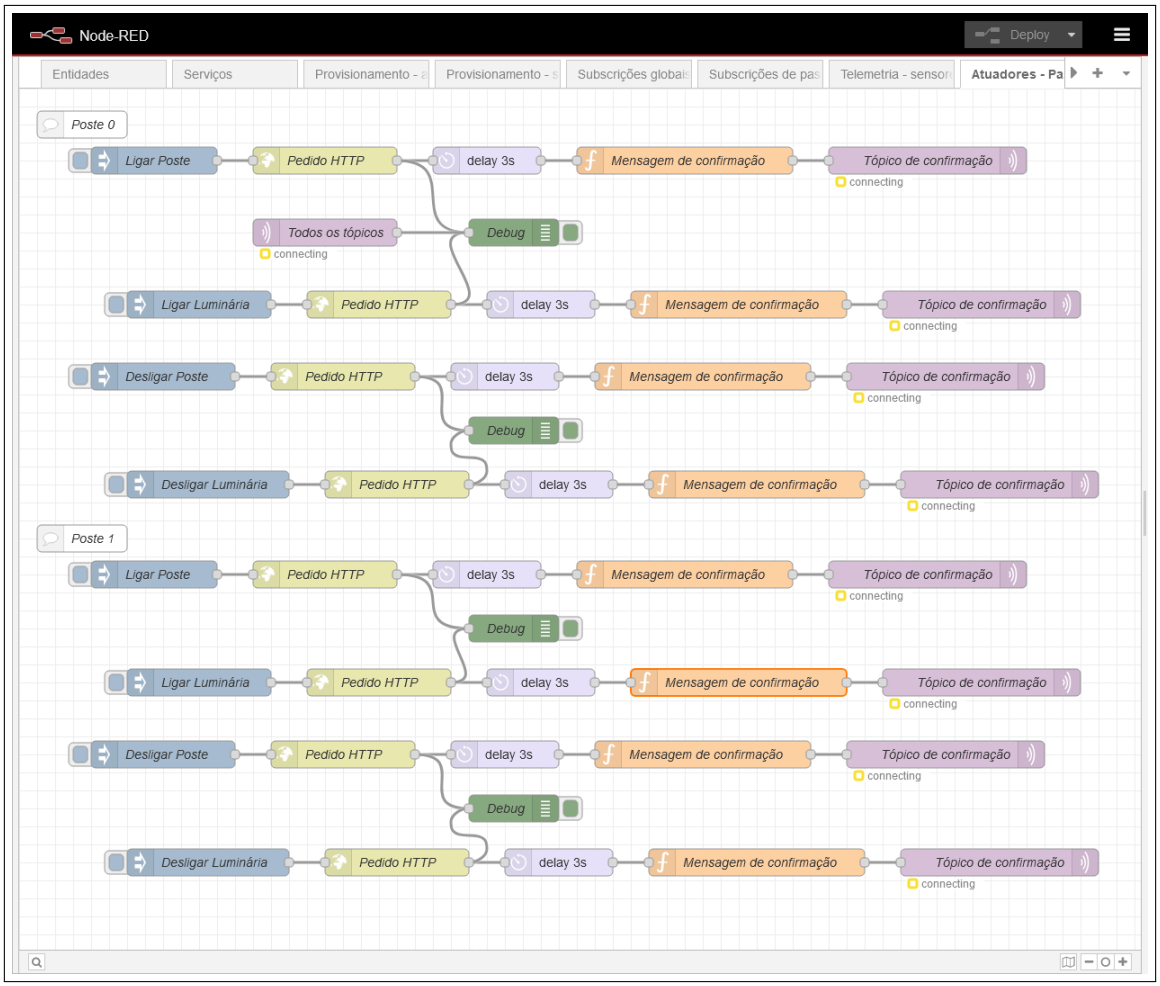

Figura 5.12: Fluxo 8 Node-RED - simulação da alteração do estado dos atuadores

Front-end* da Aplicação *Web Atualizar as informações exibidas nos *dashboards* sempre que ocorrem alterações de contexto, utilizando um mecanismo de atualização orientado a eventos. A não utilização de um *back-end* implicava que, para a atualização das *dashboards*, o *front-end* tivesse de realizar sondagens periódicas ao corretor de contexto Orion-LD para verificar se houve mudanças de contexto, um processo ineficiente.

GDs Atualizar o estado operativo dos atuadores do sistema VALLPASS - poste e luminária - com base nos resultados da execução dos comandos de ligar e desligar.

O *back-end* pode ser visto como um servidor que terá dois tipos de clientes: o corretor de contexto Orion-LD e o *front-end* da aplicação *web*.

5.8.2 Roteamento

Por roteamento entende-se a especificação da maneira como uma aplicação responde a um pedido de um cliente com destino a um determinado *endpoint* e com um determinado

método de pedido HTTP (por exemplo, GET e POST). No caso presente, o *back-end* possui duas rotas para processar pedidos HTTP POST:

`/subscriptions/Pole/:pole` Esta rota processa pedidos HTTP provenientes das subscrições de nível de passadeira, que são criadas para cada sensor e atuador durante o provisionamento automático das passadeiras inteligentes. Quando um sensor reporta novas medições ou o estado de um atuador é atualizado, o corretor de contexto Orion-LD notifica o *back-end* de que o contexto foi alterado. O parâmetro `:pole` identifica poste específico teve o contexto atualizado, permitindo ao *back-end* atualizar apenas as *dashboards* relacionadas com esse poste.

`/context-update/:entity/:command` Provenientes das subscrições globais no Orion-LD - ver Fluxo 5 do Node-RED na Subsecção 5.7.3 - estes pedidos visam que o *back-end* atualize o atributo representativo do estado dos atuadores no modelo de dados semântico, com base nos resultados dos comandos `on` e `off`. O parâmetro `:entity` especifica o tipo de atuador (poste ou luminária), enquanto o parâmetro `:command` indica o comando executado (ou seja, `on` ou `off`).

5.8.3 Comunicação com o *front-end*

A comunicação entre o *back-end* e o *front-end* utiliza o protocolo de comunicação Web-Socket. Este protocolo disponibiliza canais de comunicação bidirecionais (*full-duplex*) sobre uma única ligação TCP [107], permitindo enviar mensagens para o servidor e receber respostas orientadas a eventos sem ser necessário sondar o servidor para uma resposta. Existem diversas ferramentas que utilizam esta tecnologia, como a biblioteca *Socket.IO*, que foi a escolhida. A biblioteca *Socket.IO* é dividida em dois componentes:

- Servidor *Socket.IO* - a ser implementado no *back-end* da aplicação *web*;
- Cliente *Socket.IO* - a ser implementado no *front-end* da aplicação *web*.

Aproveitando o facto de que as subscrições FIWARE foram criadas ao nível dos postes, utilizou-se no *back-end* o conceito de *rooms* presente na biblioteca *Socket.IO* - canais arbitrários em que os *sockets* podem entrar e sair - permitindo que as notificações oriundas do Orion-LD sejam transmitidas apenas a um subconjunto de clientes. Um exemplo prático: no *front-end*, a *dashboard* de um dos postes - cliente - vai entrar apenas na *room* desse poste e, como tal, só vai receber as notificações associadas a esse poste, tornando o processo de atualização da *dashboard*, sempre que se verificam mudanças de contexto, ainda mais eficiente.

Na Fig. 5.13, encontra-se um diagrama que ilustra o funcionamento do *back-end*.

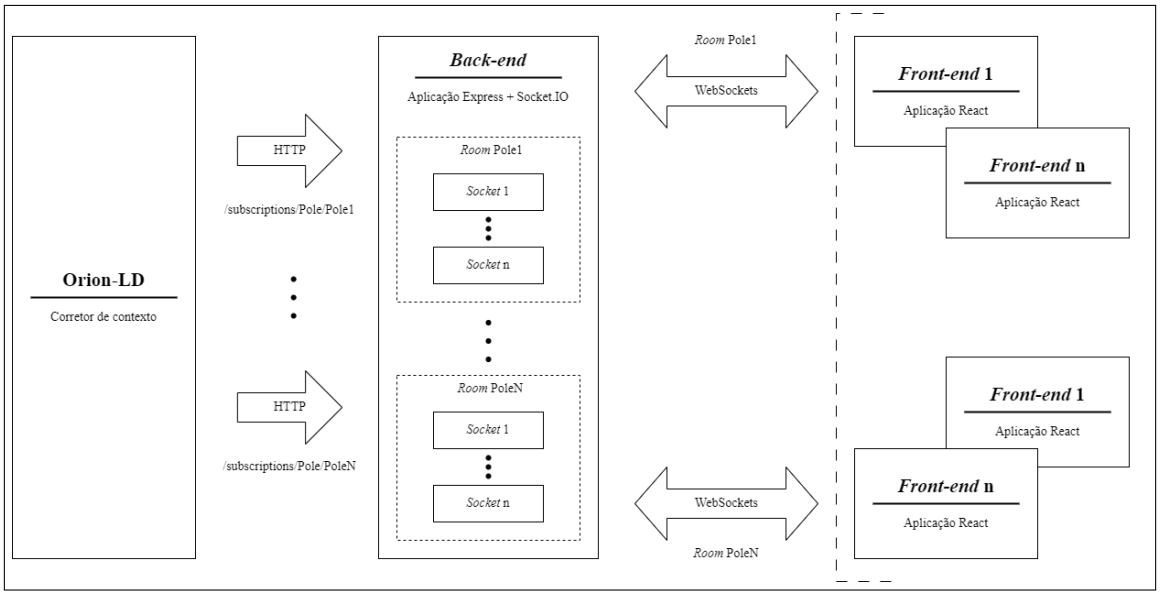

Figura 5.13: Diagrama que ilustra o funcionamento do *back-end* da Aplicação *Web*

5.8.4 Atualização do estado operativo dos atuadores do sistema VALLPASS

Para ilustrar esta funcionalidade do *back-end*, encontra-se na Fig. 5.14 um diagrama de atividades que detalha o processo de ligar remotamente uma luminária segundo uma versão simplificada dos processos FIWARE. Vale ressaltar que o processo descrito neste diagrama é aplicável a qualquer atuador e comando.

5.9 Desenvolvimento do *front-end* da Aplicação *Web*

5.9.1 Introdução e visão geral

Esta Secção descreve o desenvolvimento e a estrutura do *front-end* da aplicação *web* projetada para a plataforma VALLPASS - doravante referida apenas como ”aplicação”. A interface foi concebida utilizando as bibliotecas React e Material UI, em conformidade com os requisitos não funcionais estabelecidos, garantindo uma experiência de navegação intuitiva e responsiva para os utilizadores. Esta Secção está organizada em várias subsecções, abordando os principais componentes funcionais e visuais da aplicação, tais como autenticação, *layout*, dashboards e funcionalidades específicas para a gestão de passadeiras e clientes. Cada Secção detalha a implementação e o propósito funcional dos elementos, proporcionando uma visão abrangente e detalhada da aplicação.

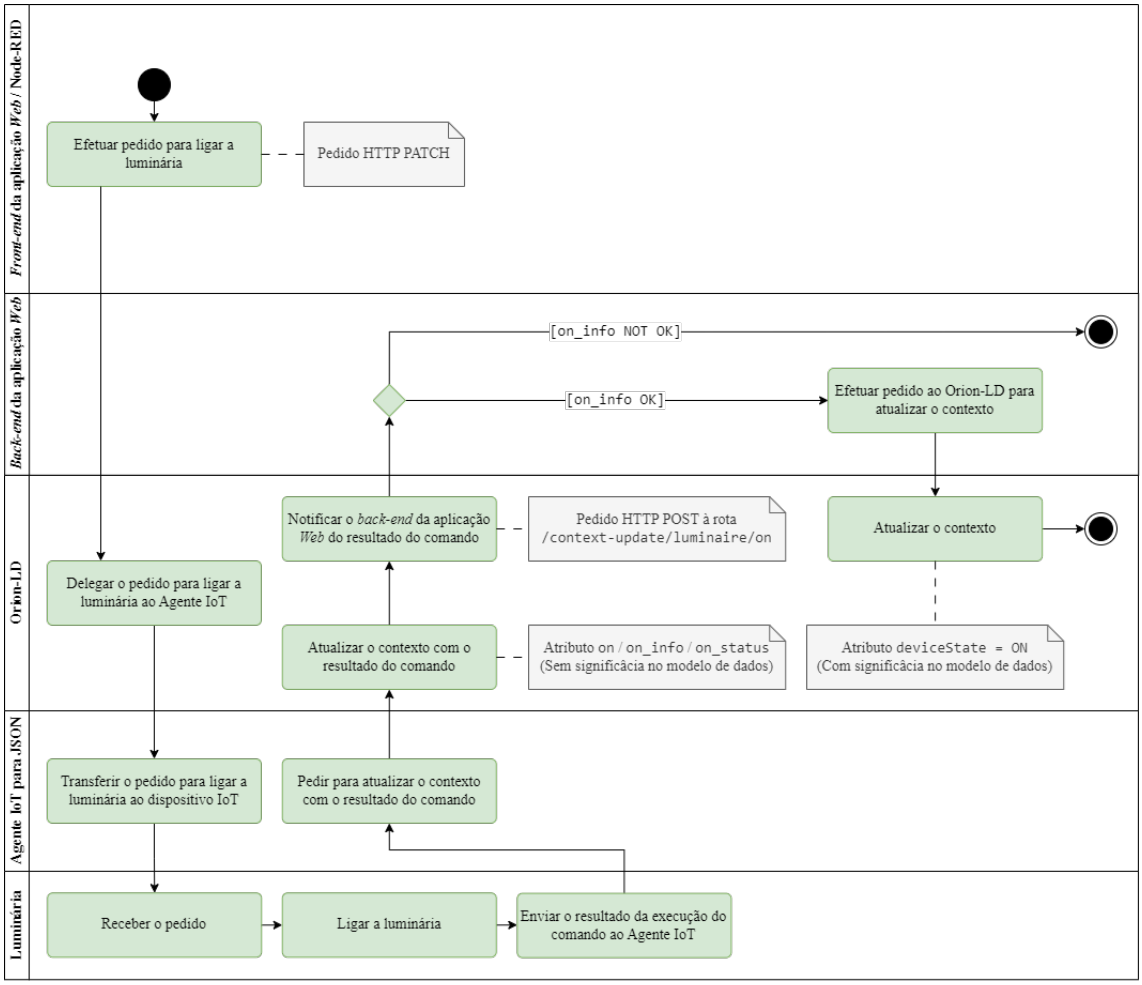

Figura 5.14: Diagrama de atividades que detalha o processo de ligar remotamente uma luminária segundo uma versão simplificada dos processos FIWARE [3]

5.9.2 Autenticação e página de *login*

A autenticação de utilizadores na aplicação é realizada através do protocolo OAuth2, especificamente utilizando o fluxo *Authorization Code*, conforme ilustrado na Fig. 5.15. Este método permite que o utilizador se autentique diretamente numa página fornecida pelo Keyrock, sem que a aplicação tenha acesso às credenciais do utilizador, garantindo assim maior segurança e privacidade.

O processo de autenticação inicia-se na página de *login* da aplicação - Fig. 5.16, onde o utilizador seleciona a opção de iniciar sessão com o Keyrock (única opção disponível). Em seguida, é redirecionado para a página de autenticação do Keyrock - Fig. 5.17, onde insere as suas credenciais. Após uma autenticação bem-sucedida, o utilizador é automaticamente redirecionado para a página inicial da aplicação, estando agora autenticado e apto a utilizar todas as funcionalidades disponibilizadas.

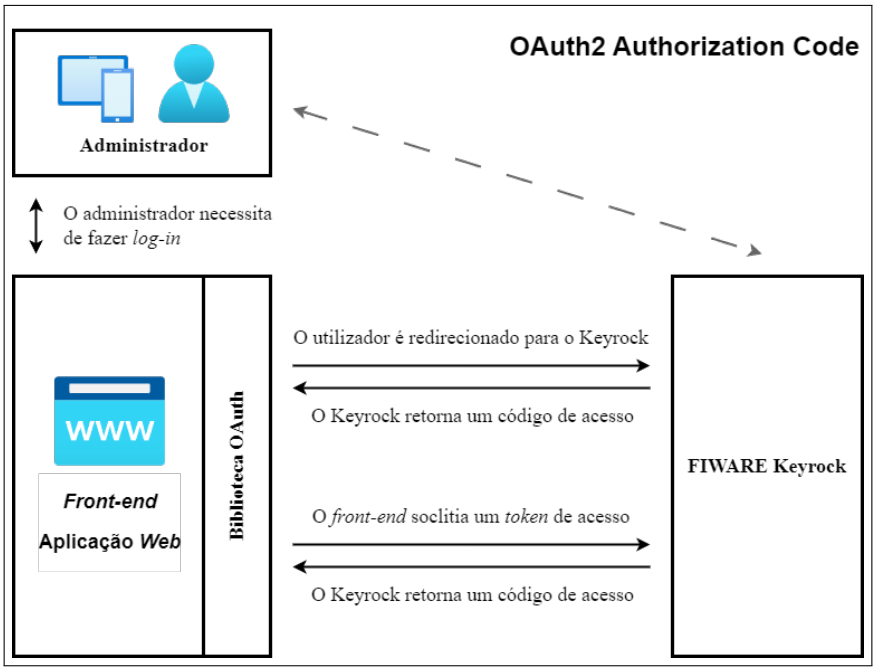

Figura 5.15: Fluxo Authorization Code do protocolo OAuth2

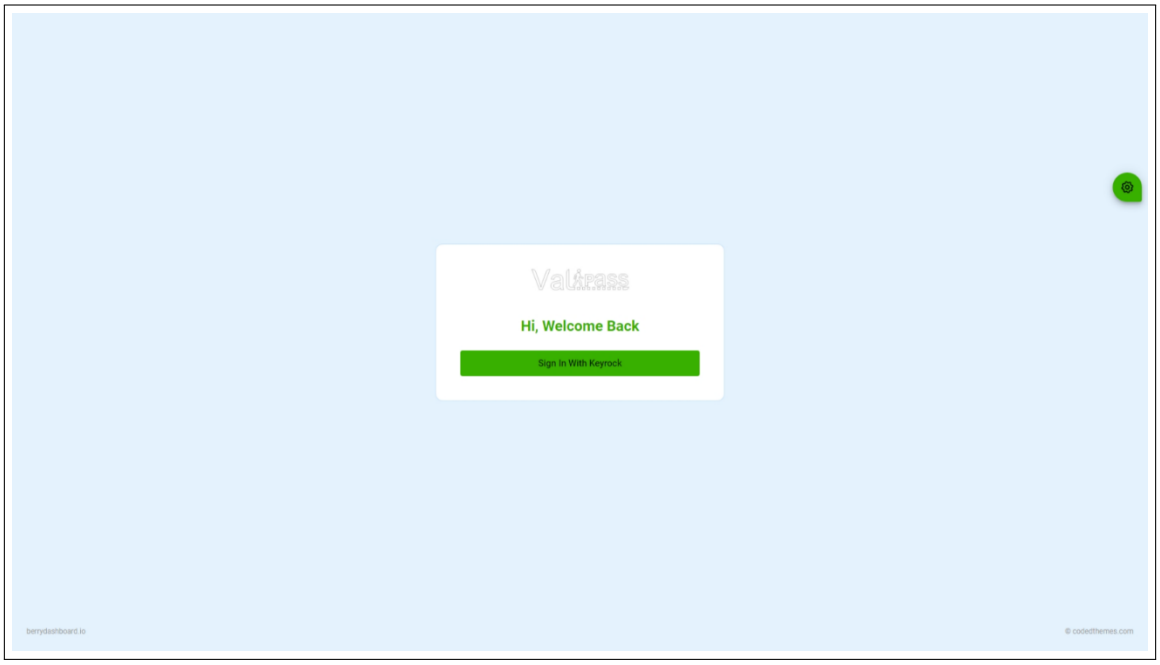

Figura 5.16: Página de *login* da aplicação

5.9.3 *Layout* da Aplicação *Web* e Página Inicial

O *layout* da aplicação foi desenvolvido para proporcionar uma experiência de utilizador intuitiva e eficiente, estruturando-se em três elementos principais: a barra lateral, o cabeçalho e a área de conteúdo, conforme ilustrado pela Fig. 5.18 e pela Fig. 5.19. Estes componentes adaptam-se dinamicamente ao contexto de navegação, melhorando a usabilidade e a acessibilidade da aplicação. Abaixo, descrevem-se detalhadamente esses três

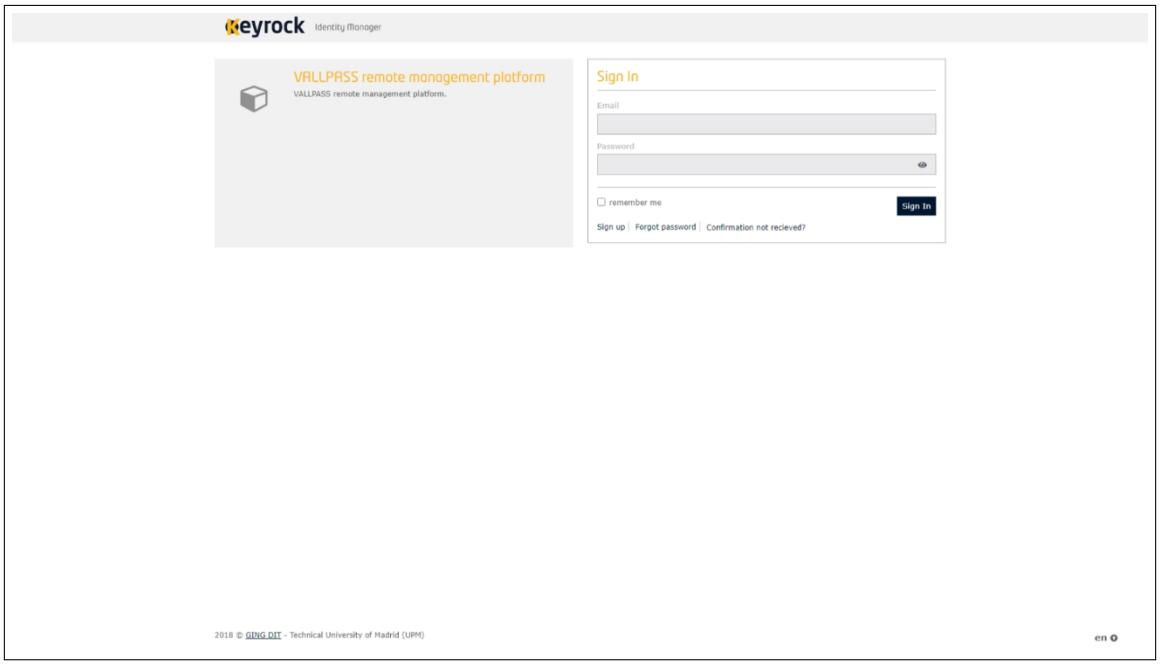

Figura 5.17: Página de *login* do FIWARE Keyrock

elementos.

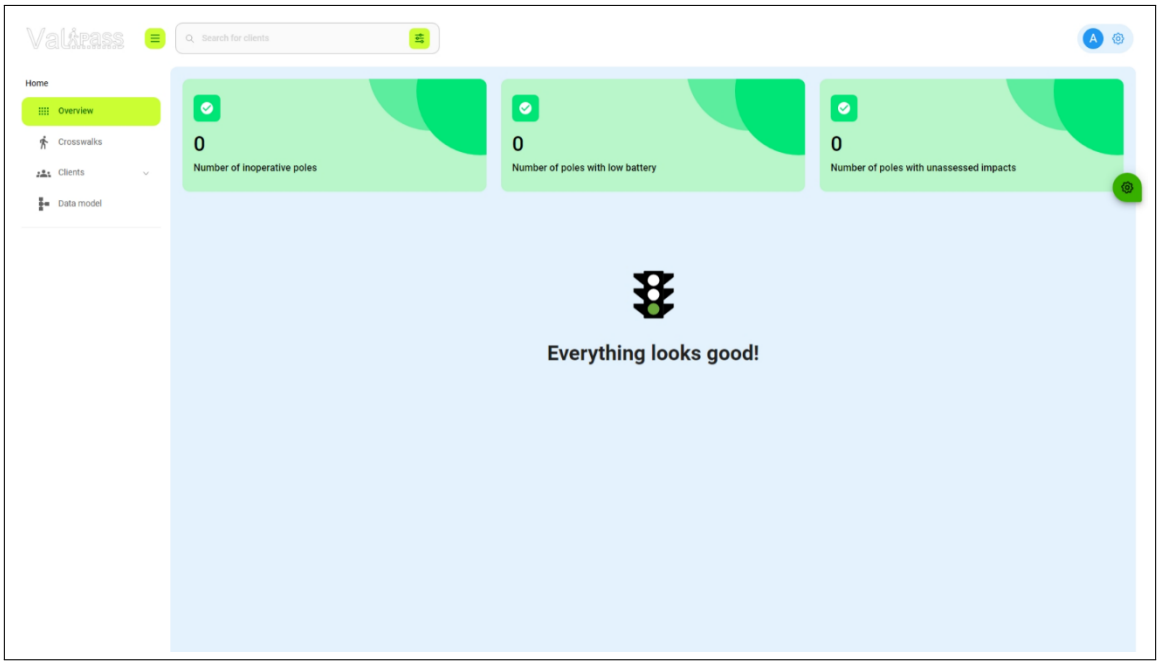

Figura 5.18: Página inicial da aplicação - sem avisos

Barra lateral dinâmica A barra lateral é o principal elemento de navegação, sendo adaptável ao contexto da aplicação. Na página inicial, exibe ligações para as secções principais: ”Overview”(visão geral), ”Crosswalks”(gestão de passadeiras) e ”Clients”(gestão de clientes). Este componente é expansível, permitindo ao utilizador ocultar ou mostrar as

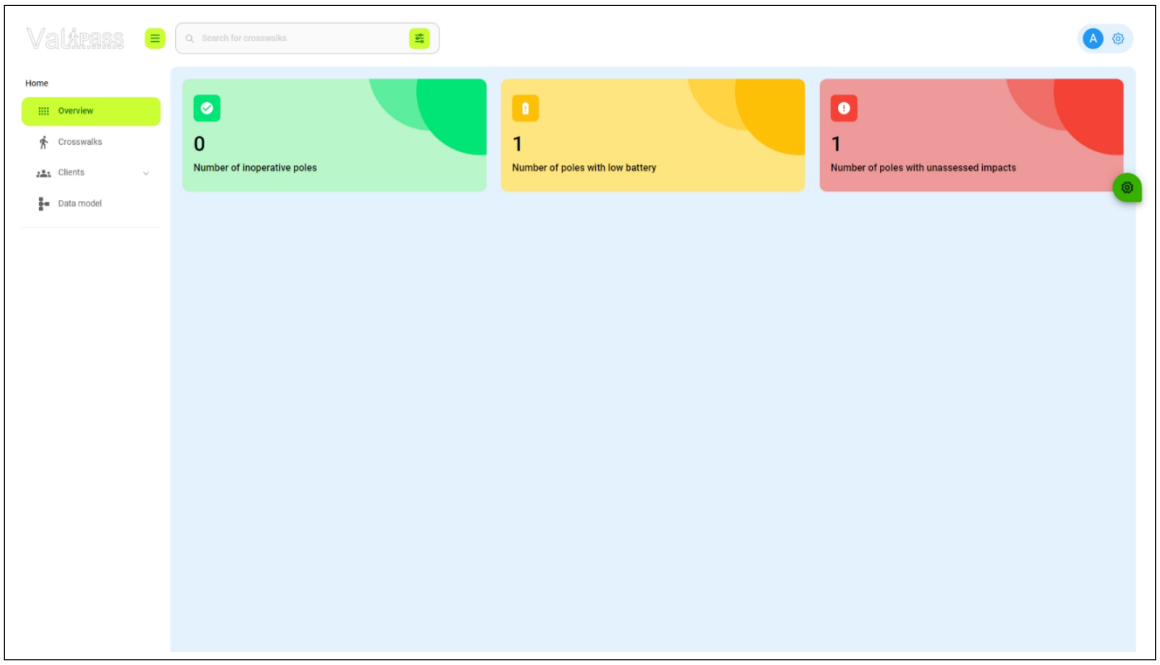

Figura 5.19: Página inicial da aplicação - com avisos

opções de navegação conforme necessário, promovendo uma experiência de uso flexível e personalizada.

Cabeçalho O cabeçalho inclui uma caixa de pesquisa contextual (permitindo ao utilizador efetuar pesquisas específicas dentro da Secção atual da aplicação) e uma Secção de perfil que pode ser vista na Fig. 5.20. Na página inicial, a pesquisa permite procurar passadeiras pelo nome ou pela localização - Fig. 5.21. Quando o utilizador insere uma localização, o sistema utiliza o serviço Nominatim para georreferenciar o endereço, convertendo-o em coordenadas geográficas (associadas aos GDs das passadeiras) e devolvendo todas as passadeiras num raio de 40 km.

Área de conteúdo Adapta-se ao contexto, exibindo a *dashboard* principal ou as páginas de gestão específicas de passadeiras e clientes. Na página inicial, uma *dashboard* apresenta uma visão geral do estado do sistema VALLPASS nas suas variáveis mais críticas, através de cartões informativos, que exibem o número de postes inoperativos, postes com bateria baixa e postes com impactos por avaliar.

Adicionalmente, a aplicação inclui um Floating Action Button, ou Botão de Ação Flutuante (FAB) que, ao ser selecionado, abre uma barra lateral direita - Fig. 5.22. Esta funcionalidade permite ao utilizador personalizar, em tempo real, certos aspetos da UI, como a família tipográfica e o raio das bordas dos elementos, ajustando a experiência de utilização às suas preferências.

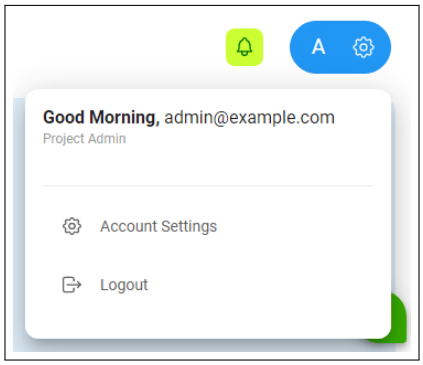

Figura 5.20: Secção de perfil presente no cabeçalho da aplicação

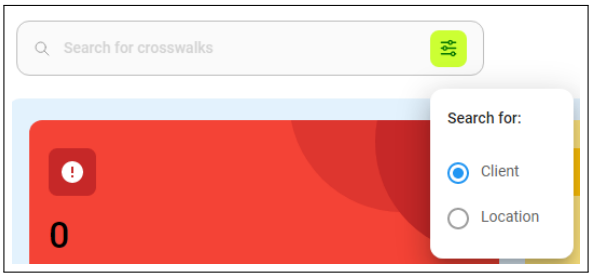

Figura 5.21: Caixa de pesquisa da página inicial da aplicação

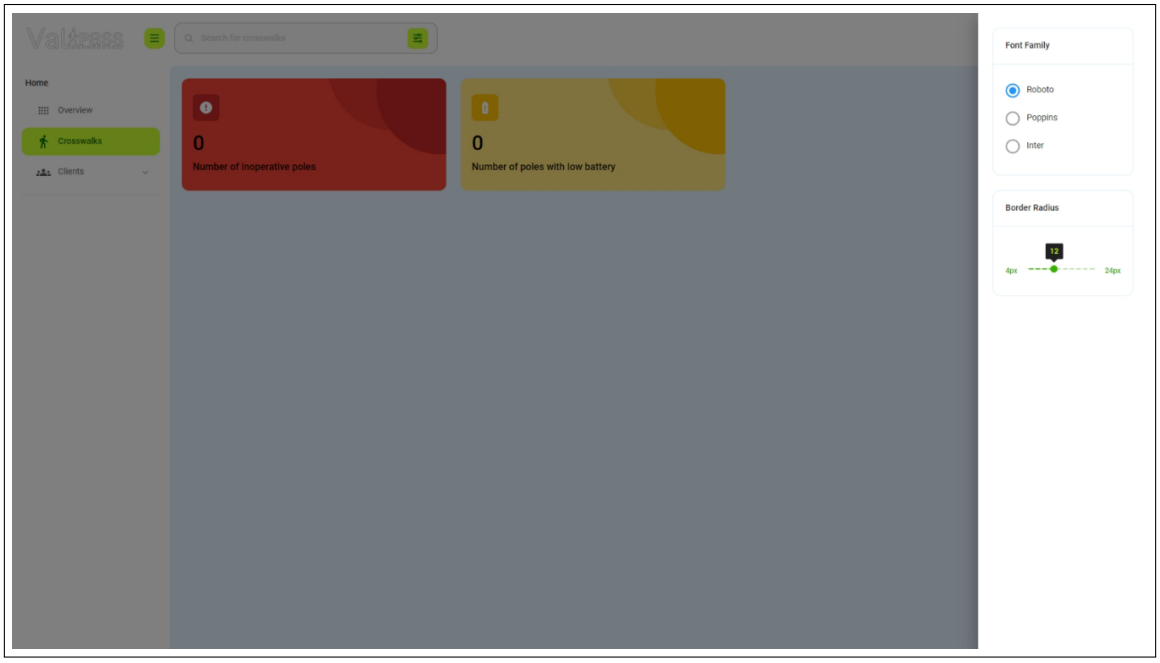

Figura 5.22: Barra lateral de personalização da UI da aplicação

5.9.4 Especificação OpenAPI do modelo de dados semântico VALLPASS

A aplicação integra uma página dedicada à visualização da especificação OpenAPI do modelo de dados semântico NGSI-LD desenvolvido para o sistema VALLPASS - Fig. 5.23.

Esta página proporciona ao administrador uma visão detalhada do modelo de dados que suporta o sistema, sendo uma ferramenta útil para a compreensão aprofundada da arquitetura de dados e para auxiliar em processos de resolução de problemas ou otimização do sistema.

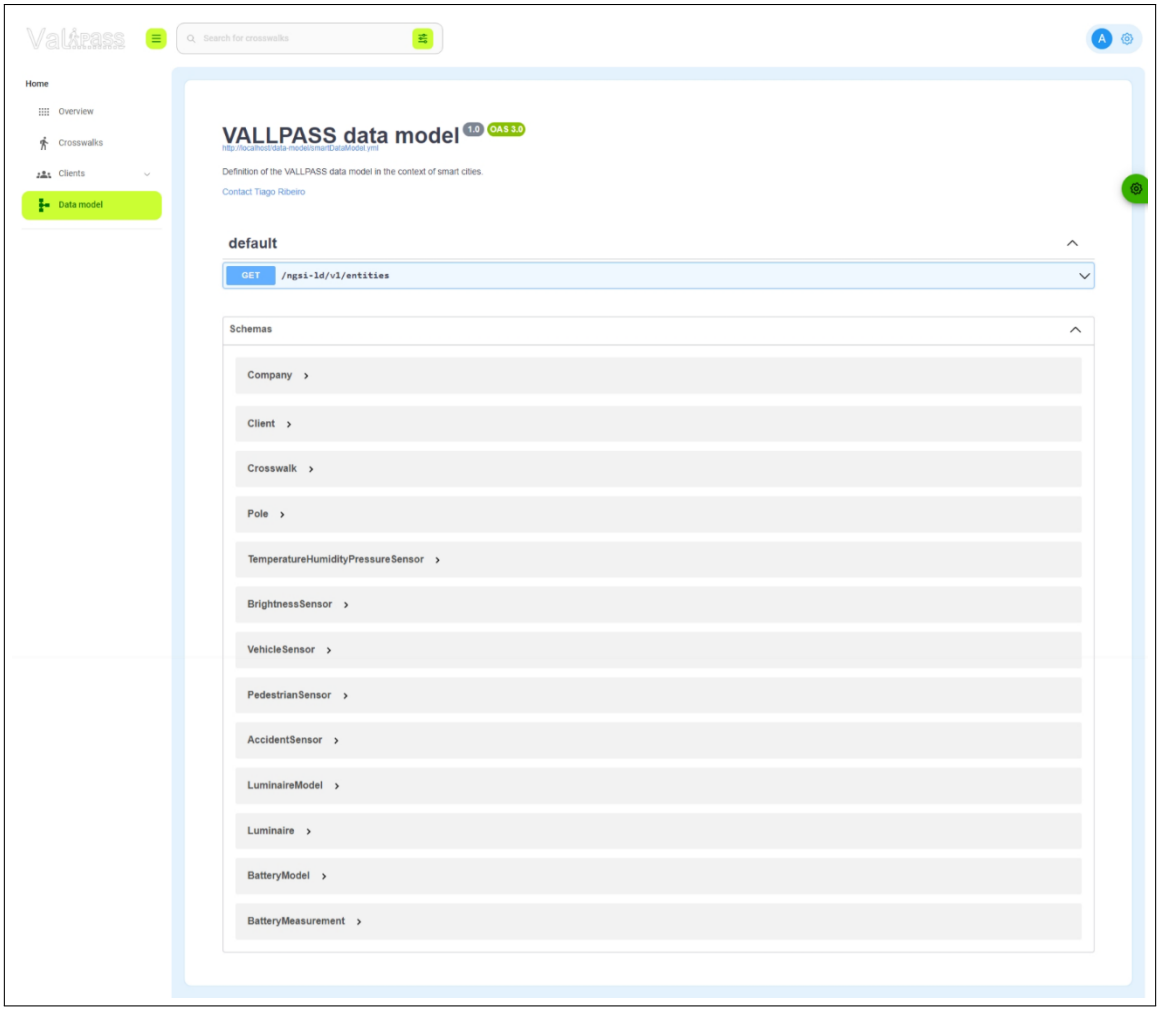

Figura 5.23: Visualização da especificação OpenAPI do modelo de dados semântico na aplicação

5.9.5 Gestão de passadeiras

A gestão de passadeiras é uma funcionalidade central da aplicação *web*, permitindo ao administrador supervisionar e controlar as passadeiras inteligentes do sistema VALLPASS.

A página de listagem de passadeiras - Fig. 5.24, acessível através da barra lateral, exibe todas as passadeiras registadas numa tabela que inclui funcionalidades avançadas de filtragem e ordenação, permitindo aplicar filtros específicos a cada coluna. Na Fig. 5.25 encontra-se o pormenor da aplicação de filtros às colunas desta tabela. A caixa de pesquisa contextual, à semelhança do que acontece na página inicial, continua a permitir a pesquisa

de passadeiras pelo seu nome ou localização.

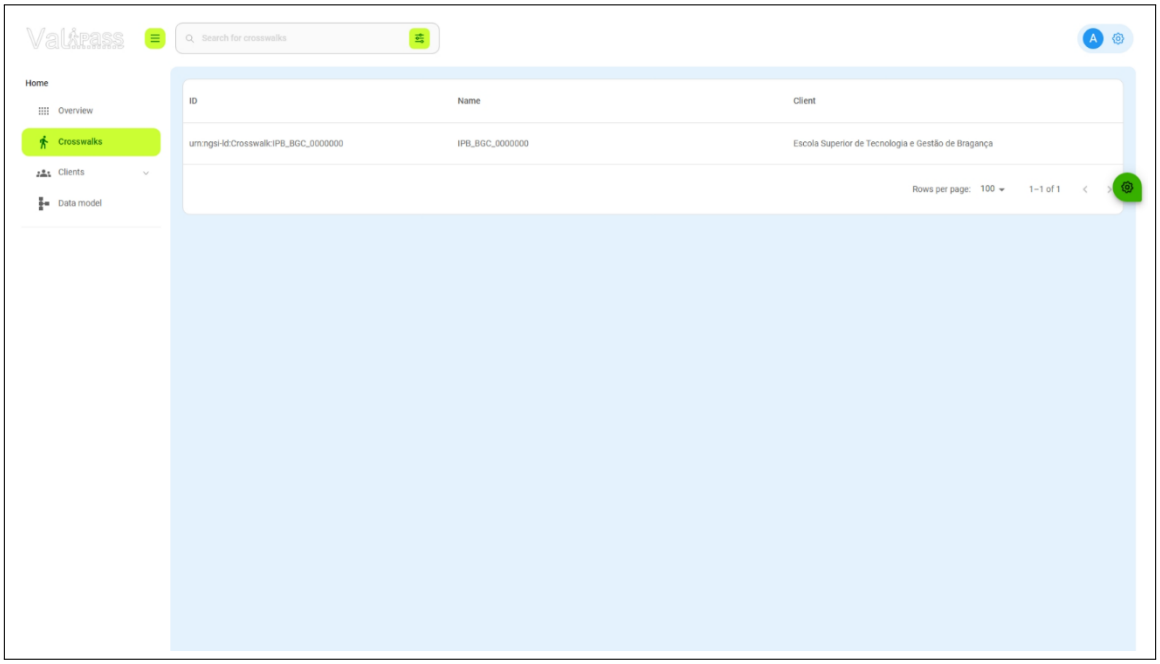

Figura 5.24: Página de listagem de passadeiras

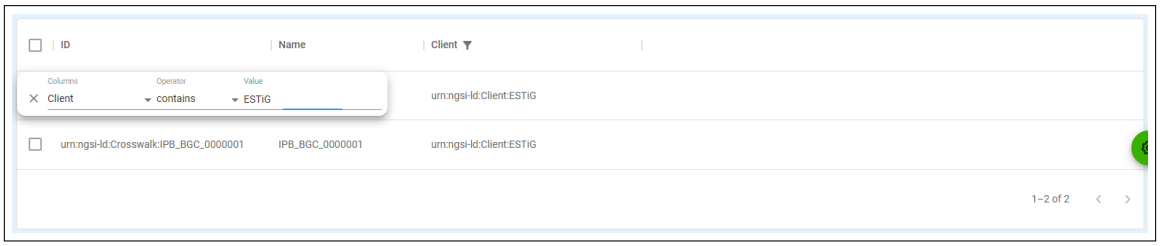

Figura 5.25: Pormenor da aplicação de filtros na listagem de passadeiras na aplicação

Ao selecionar uma passadeira na lista, o utilizador é direcionado para a página de gestão específica dessa passadeira. Nesta página, é apresentada uma *dashboard* detalhada - Fig. 5.26, dividida em duas secções principais:

Secção superior Esta Secção centra-se no estado operacional da passadeira, integrando cartões informativos que exibem o nome da passadeira e um mapa interativo que localiza a sua posição exata. Adicionalmente, apresenta o número de postes inoperativos, postes com bateria em estado crítico e postes que sofreram impactos ainda não avaliados. Inclui ainda um botão de controlo que permite ativar ou desativar a passadeira, com o estado visível através de cores intuitivas para facilitar a identificação.

Secção inferior Exibe dados detalhados sobre variáveis climáticas e de tráfego, incluindo o número de peões e veículos detetados, as velocidades máxima e média dos veículos, pressão atmosférica, temperatura ambiente, humidade relativa e intensidade luminosa.

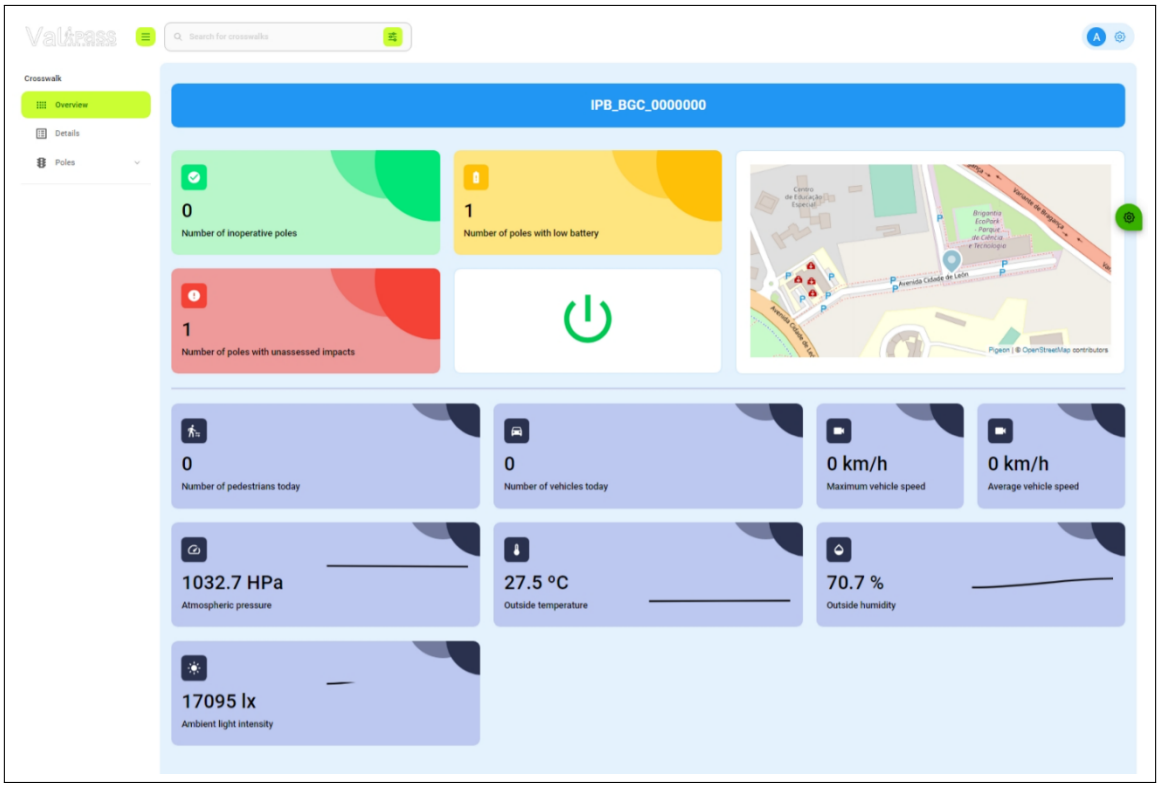

Figura 5.26: Página de gestão de uma passadeira

A página de detalhes da passadeira - Fig. 5.27 oferece ao administrador a possibilidade de visualizar e editar informações específicas da passadeira, como o nome, proprietário e coordenadas de localização. Além disso, inclui a opção de eliminar a passadeira, caso seja necessário.

As páginas de gestão dos postes associados a cada passadeira - Fig. 5.28 apresentam uma visão detalhada de cada poste, seguindo um *layout* semelhante ao da página das passadeiras. Estas páginas exibem informações sobre o estado operacional do poste, medições de variáveis climáticas e dados de tráfego. Adicionalmente, possibilitam o controlo de funcionalidades específicas, como ativar ou desativar o poste e ajustar o estado da luminária manualmente (*override*).

5.9.6 Gestão de clientes

A aplicação disponibiliza funcionalidades completas para a gestão de clientes, essenciais para associar passadeiras inteligentes a entidades específicas e manter um registo organizado dos respetivos proprietários ou responsáveis.

A página inicial da gestão de clientes - Fig. 5.29, acessível através da barra lateral, apresenta todos os clientes registados numa tabela interativa semelhante em funcionalidade à tabela que apresenta a listagem de passadeiras. Além disso, neste contexto (gestão de clientes), a caixa de pesquisa contextual permite procurar cliente pelo seu nome ou

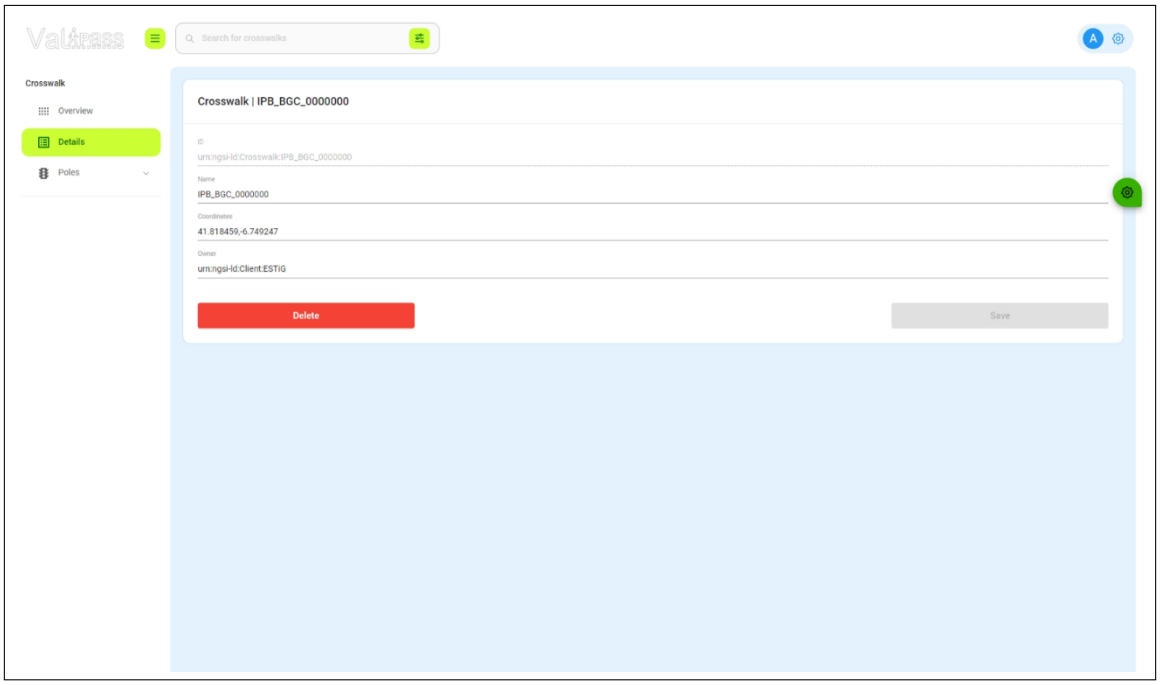

Figura 5.27: Página de detalhes de uma passadeira

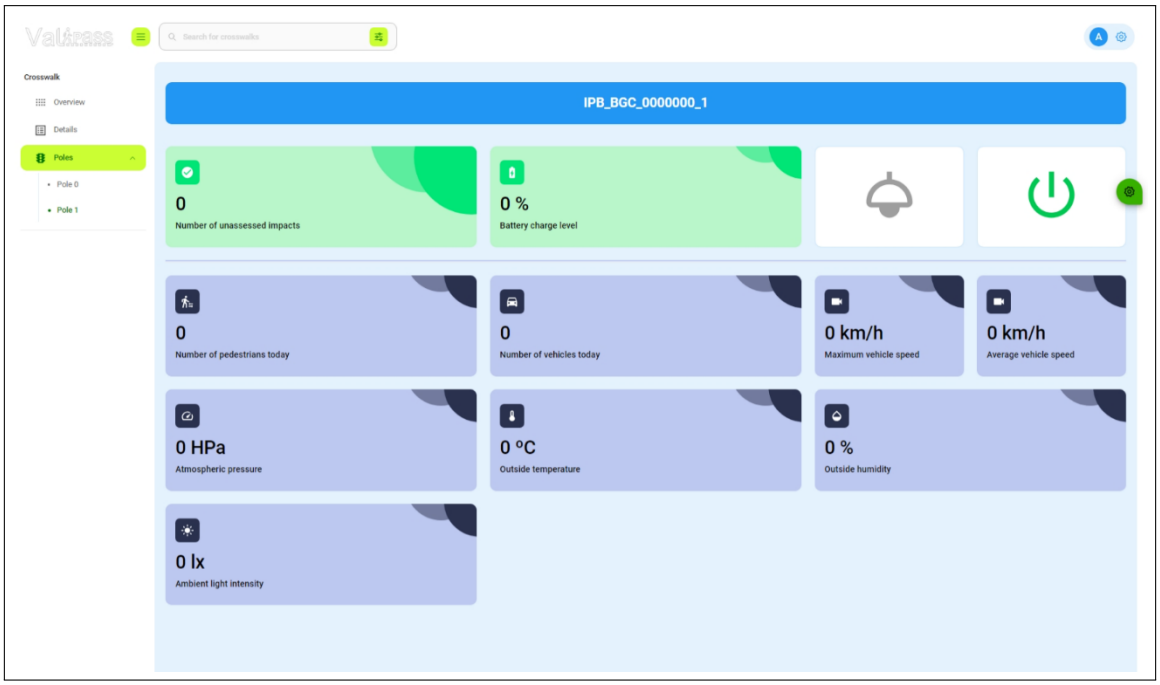

Figura 5.28: Página de gestão de um poste de uma passadeira

número de identificação fiscal - Fig. 5.30.

Página de detalhes e edição de clientes Ao selecionar um cliente, o administrador é direcionado para a página de detalhes do cliente - Fig. 5.31, onde pode visualizar e editar informações como nome, número de identificação fiscal, email, morada, código postal, localidade e país. Esta funcionalidade garante que os dados dos clientes se mantêm sem-

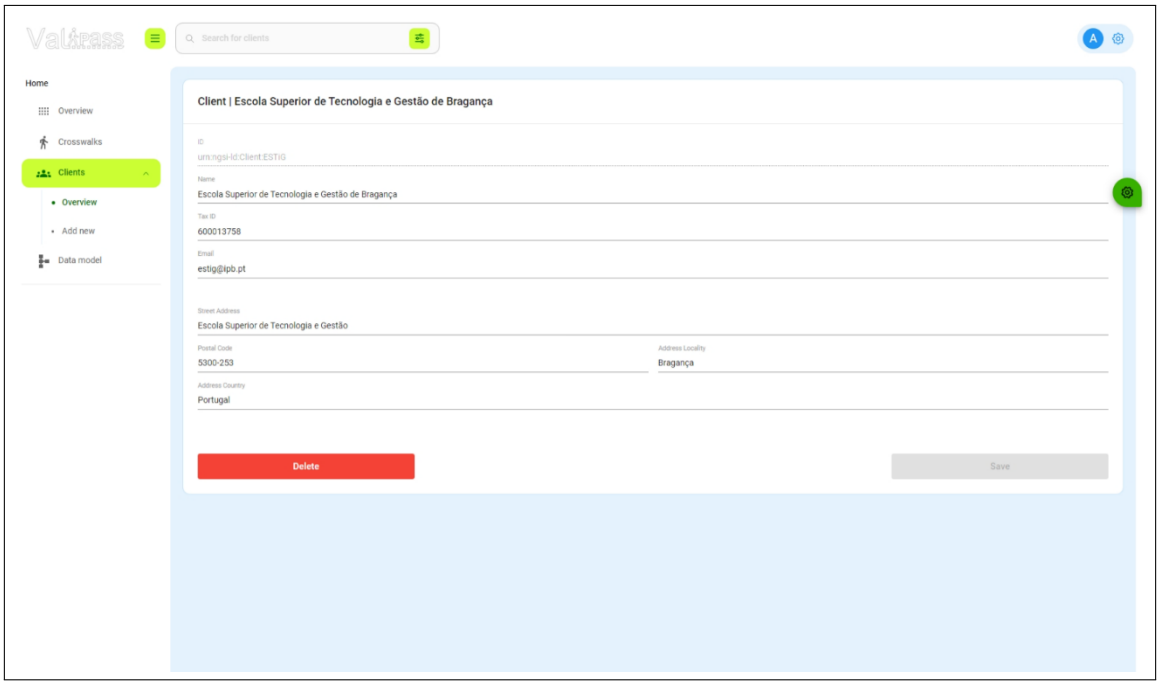

Figura 5.29: Página de listagem de clientes

Figura 5.30: Caixa de pesquisa afeta à gestão de clientes na aplicação

pre atualizados e corretos. Adicionalmente, é possível eliminar um cliente diretamente a partir desta página, caso necessário.

Registo de novos clientes Para o registo de novos clientes, foi implementada uma página específica com um formulário semelhante ao utilizado para edição, que permite ao administrador introduzir todos os dados do cliente, incluindo um identificador único pré-formatado e parcialmente preenchido (parte imutável do formato de acordo com a especificação NGSI-LD) - Fig. 5.32. Esta configuração assegura a consistência e uniformidade dos registos no sistema, facilitando o processo de inclusão de novos clientes.

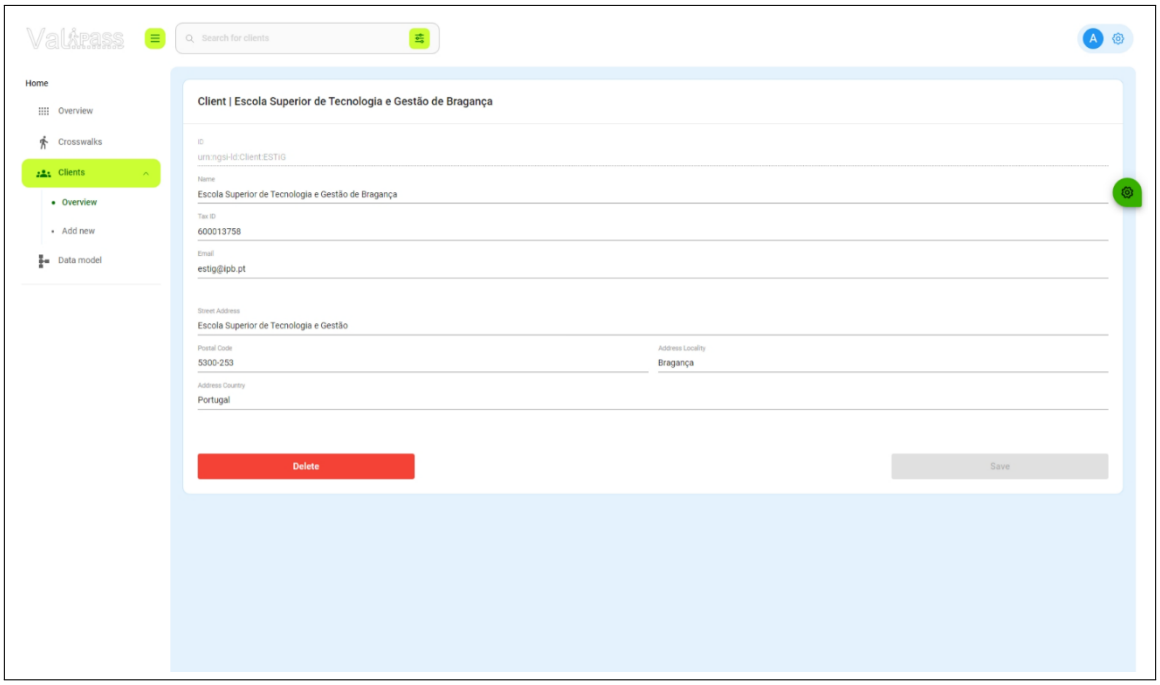

Figura 5.31: Página de detalhes de um cliente na aplicação

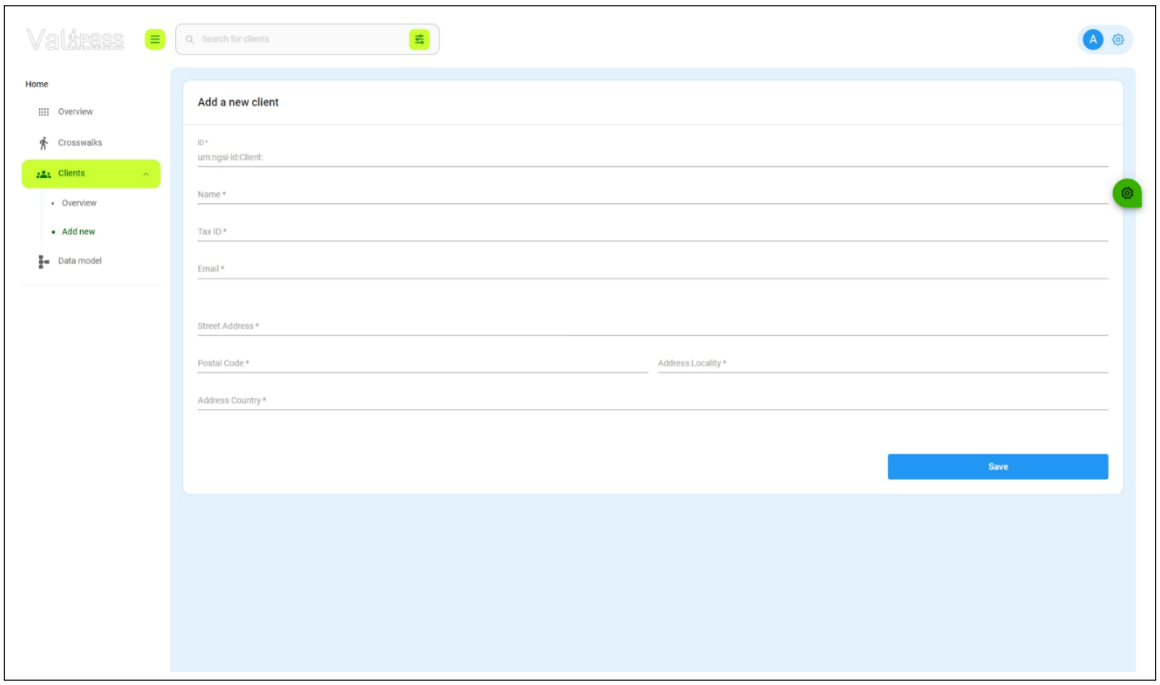

Figura 5.32: Página de registo de um cliente na aplicação

5.10 Conclusão

Neste Capítulo, foi apresentada a contribuição para o projeto VALLPASS, nomeadamente a implementação de uma plataforma de gestão remota *Powered by FIWARE*, sustentada por um modelo de dados semântico orientado às cidades inteligentes. Iniciou-se com uma

análise pormenorizada dos requisitos do sistema, incluindo a caracterização dos utilizadores e a especificação dos requisitos funcionais e não funcionais, complementada pelo diagrama de casos de uso que sintetiza as interações principais.

Desenvolveu-se o modelo conceptual de dados através do diagrama ER e da descrição detalhada das entidades, estabelecendo uma base sólida para a definição do modelo de dados semântico NGSI-LD. Foram selecionados e ajustados modelos de dados de referência, culminando na geração do ficheiro NGSI-LD `@context`, essencial para garantir a interoperabilidade semântica e a conformidade com os padrões das cidades inteligentes.

A arquitetura da plataforma de gestão remota foi definida, detalhando os componentes, as interações entre eles e os *endpoints* do *proxy* reversa (NGINX). Esta arquitetura modular e escalável assegura a robustez e a eficiência necessárias para a operacionalização do sistema VALLPASS.

A prototipagem da plataforma validou a arquitetura proposta e o modelo de dados desenvolvido. A implementação e configuração dos serviços, aliadas à simulação do sistema VALLPASS, permitiram testar as funcionalidades e garantir o alinhamento com os requisitos definidos.

No desenvolvimento do *back-end* da aplicação *web*, foram implementados o roteamento, a comunicação eficaz com o *front-end* e a lógica para a atualização do estado operativo dos atuadores. O *front-end* incorporou funcionalidades de autenticação, um *layout* intuitivo, a integração da visualização da especificação OpenAPI do modelo de dados semântico e ferramentas de gestão de passadeiras e clientes.

Conclui-se que a plataforma de gestão remota desenvolvida cumpre os objetivos delineados, proporcionando uma solução eficiente, escalável e com premissas fortes de segurança para o controlo e monitorização do sistema VALLPASS. A utilização do *framework* FIWARE e a adoção de um modelo de dados semântico orientado às cidades inteligentes contribuem para a interoperabilidade e potencial de integração futura com outros sistemas e plataformas, reforçando o alinhamento com as tendências atuais em soluções urbanas inteligentes.

Capítulo 6 Experiência prática laboratorial de comunicação com VLC

6.1 Introdução

Este Capítulo apresenta uma experiência prática laboratorial de comunicação utilizando VLC, com o objetivo de investigar a viabilidade e a eficiência do uso da VLC para comunicação direta entre postes de iluminação pública, integrados em sistemas de transportes inteligentes. No âmbito do experimento, optou-se por simular a transmissão de informações específicas sobre o tráfego de veículos, utilizando um ambiente experimental controlado que replica o funcionamento de sensores de tráfego em cenários urbanos. Esta abordagem permitiu analisar o desempenho da VLC na transmissão de dados telemétricos, mantendo o foco na aplicação da tecnologia em contextos de mobilidade urbana inteligente.

Inicialmente, propõe-se a arquitetura do sistema, detalhando o cenário de estudo e a configuração física, que inclui o circuito emissor e o circuito recetor. Seguidamente, descreve-se a arquitetura de comunicação, abordando as camadas física (PHY) e de controlo de acesso ao meio (MAC), com referência ao padrão IEEE 802.15.7, além das adaptações realizadas para o contexto experimental. Posteriormente, apresentam-se as implementações de *hardware* e *software*, destacando os componentes utilizados no circuito emissor e recetor, assim como os fluxogramas dos programas desenvolvidos para o microcontrolador de ambos os circuitos.

Este estudo pretende demonstrar a viabilidade da comunicação por VLC como uma solução eficiente para cenários urbanos, assegurando a transmissão de dados de forma eficaz e evidenciando o potencial da VLC para aliviar a pressão sobre o espectro de (RF).

6.2 Proposta de sistema e arquitetura

6.2.1 Cenário de estudo

O presente estudo tem como objetivo investigar a aplicação da VLC na transmissão de informações específicas sobre o tráfego de veículos. Para tal, será desenvolvido um ambiente experimental controlado que possibilitará a análise do desempenho da VLC na comunicação de dados telemétricos, reproduzindo o funcionamento de sensores de tráfego em contextos urbanos.

No sistema VALLPASS, os dados capturados pelos sensores são transmitidos para a nuvem utilizando o protocolo MQTT. No cenário experimental proposto neste estudo, pretende-se replicar esta dinâmica de forma precisa, transmitindo as informações referentes ao tráfego de veículos na forma de uma *payload* JSON, estruturada em tópico e mensagem, com a organização do tópico e a formatação da mensagem a seguirem exatamente o padrão adotado pelo sistema VALLPASS. Com base na arquitetura de alto nível do VALLPASS, o poste sem conectividade LoRa, sempre que operasse como emissor (ao enviar dados dos sensores, por exemplo), transmitiria uma *payload* com uma estrutura semelhante (tópico e mensagem), cabendo ao poste *gateway* a responsabilidade de processar a *payload* e publicar a mensagem no *broker* MQTT. A *payload* utilizada neste estudo segue o formato apresentado na Listagem 6.1, em que os campos representam:

- i: intensidade do tráfego (número de veículos desde a última medição);
- a: velocidade média dos veículos;
- l: velocidade mínima registada;
- h: velocidade máxima registada.

Listagem 6.1: *Payload* JSON utilizada no estudo da VLC

```
{
    "topic": "/json/wmnshtp4yfjysxs62oju/IPB_BGC_0000000_0_V/
       attrs",
    "payload": {
        "i": 157,
        "a": 65.3,
        "l": 18.7,
        "h": 59.4
    }
}
```

A *payload* será criada com valores aleatórios para os campos mencionados anteriormente e transmitida em intervalos regulares, com o intuito de simular o comportamento dinâmico do sistema VALLPASS no contexto da monitorização de veículos.

A comunicação será realizada de forma unidirecional, consistindo na transmissão de dados de um emissor para um recetor sem a necessidade de retorno, com o objetivo de simplificar o teste e concentrar a análise na fiabilidade e velocidade da transmissão. Adicionalmente, para facilitar a implementação, os dados não serão encriptados durante o teste, permitindo um processo mais simples e eficiente na avaliação preliminar. Este ambiente experimental visa analisar a viabilidade da comunicação por VLC como uma solução eficiente para cenários urbanos, assegurando a transmissão de dados com baixa taxa de erro e demonstrando potencial para reduzir a pressão sobre o espectro de RF.

6.2.2 Arquitetura física

A arquitetura experimental foi concebida para simular a VLC entre dois postes localizados numa passadeira, utilizando dois circuitos distintos e totalmente independentes: o Circuito Emissor e o Circuito Recetor.

Circuito Emissor

O Circuito Emissor, que representa o poste desprovido de conectividade LoRa, é estruturado em torno de um microcontrolador responsável pelo controlo de um *laser*, o qual atua como fonte de luz para a comunicação via VLC.

Embora possa parecer intuitivo utilizar um LED como fonte luminosa no Circuito Emissor, replicando a configuração existente no sistema VALLPASS (onde os postes possuem luminárias LED), a escolha de um *laser* como fonte de luz foi motivada por razões específicas, que se justificam ao analisar os seguintes fatores:

Eficiência energética A ativação da luminária dos postes durante o período diurno não se revela eficiente do ponto de vista energético. Em contrapartida, um *laser* apresenta um consumo energético substancialmente inferior, constituindo uma alternativa mais sustentável para a comunicação durante o dia.

Redução da interferência ambiental Fatores ambientais como luz solar intensa, nevoeiro, chuva e poluição atmosférica podem impactar de forma significativa a VLC. No entanto, um *laser* emite um feixe focado e coerente, tornando-o menos suscetível à dispersão e interferência em comparação com fontes de luz mais difusas, como as luminárias LED instaladas nos postes. Esta emissão concentrada do *laser* aumenta a fiabilidade do sinal transmitido, mesmo sob diferentes condições atmosféricas.

O microcontrolador realiza a codificação e modulação dos dados antes de ativar o *laser*. Os sinais de luz resultantes desta modulação correspondem aos dados previamente codificados, convertendo assim os sinais elétricos em sinais óticos para transmissão através do espaço livre entre o Circuito Emissor e o Circuito Recetor.

Circuito Recetor

Simulando o poste *gateway*, o Circuito Recetor tem como função capturar e descodificar os sinais óticos transmitidos. Para tal, também integra um microcontrolador dedicado ao processamento dos dados. Um fotodíodo, ligado ao microcontrolador, atua como sensor ótico. A escolha do fotodíodo em detrimento de outros componentes de deteção de luz, como o Light Dependent Resistor, ou Resistor Dependente de Luz (LDR), justifica-se pela sua capacidade de resposta mais rápida e pela maior precisão na deteção de variações de intensidade luminosa:

Alta capacidade de resposta Os fotodíodos apresentam tempos de resposta extremamente rápidos, permitindo-lhes detetar variações súbitas na intensidade luminosa. Esta propriedade é essencial para capturar de forma precisa os sinais de luz modulados, que podem alterar o seu estado a frequências elevadas, possibilitando assim a obtenção de taxas de transmissão superiores.

Precisão melhorada Os fotodíodos possuem uma sensibilidade superior, permitindo-lhes identificar variações subtis nos níveis de luminosidade. Esta elevada precisão contribui para uma maior fiabilidade na receção de dados, particularmente em cenários onde a luz ambiente apresenta variações constantes.

O fotodíodo capta os sinais de luz modulados emitidos pelo *laser* do Circuito Emissor e converte-os novamente em sinais elétricos. Posteriormente, estes sinais são processados pelo microcontrolador de forma a recuperar os dados originais transmitidos. Um ecrã LCD está ligado ao Circuito Recetor para apresentar a informação descodificada em tempo real, funcionando como um meio de validação da transmissão bem-sucedida e como interface de utilizador para monitorizar o desempenho do sistema. Adicionalmente, será integrado um LED vermelho para sinalizar eventuais anomalias na transmissão.

A comunicação entre o Circuito Emissor e o Circuito Recetor é realizada exclusivamente através do canal VLC estabelecido pelo par *laser* e fotodíodo. O *laser* do Circuito Emissor projeta a luz modulada em direção ao fotodíodo do Circuito Recetor, o qual deve estar rigorosamente alinhado para assegurar uma receção de sinal ideal. Este alinhamento preciso é especialmente crucial devido ao feixe estreito e altamente direcionado do *laser*, que exige uma posição exata para ser detetado de forma eficiente. Dado que os circuitos funcionam de forma totalmente independente, sem qualquer ligação física ou sem fios que

não o próprio trajeto de luz, a configuração reflete fielmente o cenário realista associado à arquitetura de alto nível do sistema VALLPASS.

O diagrama de blocos correspondente à arquitetura física do cenário de estudo encontra-se representado na Fig. 6.1.

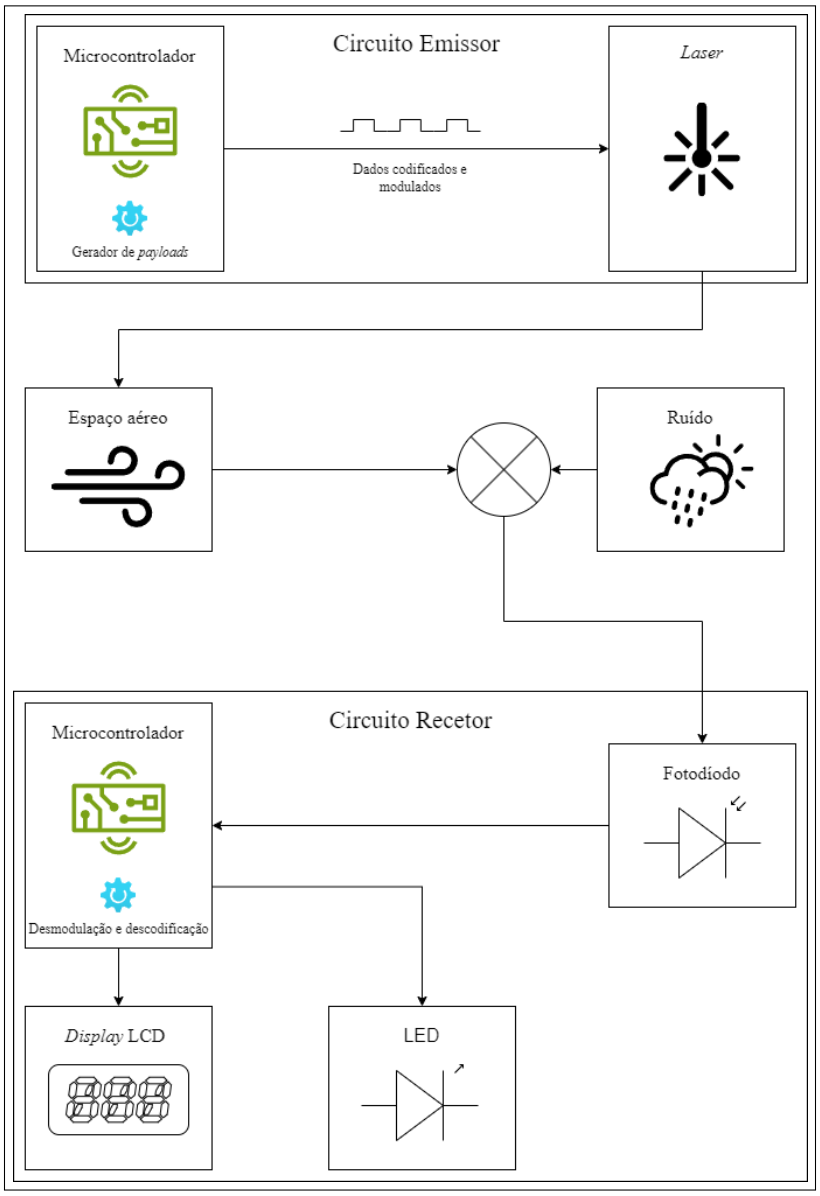

Figura 6.1: Diagrama de blocos afeto à arquitetura física do cenário de estudo proposto

6.2.3 Arquitetura de comunicação - PHY

A camada PHY da arquitetura de comunicação foi concebida tendo em consideração alguns aspetos especificados no padrão IEEE 802.15.7 [4] e orientada para um cenário de implementação real no sistema VALLPASS, sempre que exequível, ou seja, desde que seja viável no contexto do cenário em estudo.

O padrão IEEE 802.15.7 especifica seis tipos PHY, cada um com um ambiente de utilização previsto (interior/exterior), esquemas de modulação e taxas de transferência máximas [4]. Para o presente cenário, foi selecionado o tipo PHY I como o mais adequado, uma vez que é destinado a ambientes exteriores e a aplicações com baixo débito de dados, permitindo alcançar taxas de transmissão na ordem das dezenas a centenas de kbps [4]. Relativamente aos modos de operação do tipo PHY I, serão utilizados os seguintes parâmetros, conforme definidos na Tabela 76 do padrão IEEE 802.15.7 [4]:

- **Modulação:** OOK
- **Run-Length Limited:** Manchester
- **Taxa de relógio ótico (kHz):** 200
- **Forward Error Correction, ou Correção de Erro Direta (FEC):** Nenhum
- **Taxa de transferência de dados (kbps):** 100

No que concerne à sincronização, será implementado um mecanismo simplificado, em que os quadros MAC serão delimitados por um pseudo Start Frame Delimiter, ou Delimitador de Início de Quadro (SFD) e um pseudo End Frame Delimiter, ou Delimitador de Fim de Quadro (EFD), ambos compostos por uma sequência de 8 bits iguais a 1. Importa salientar que nem o SFD nem o EFD serão submetidos à codificação Manchester. Esta abordagem é eficaz, uma vez que o quadro MAC, devido à codificação Manchester, não pode conter uma sequência de 8 bits iguais a 1.

6.2.4 Arquitetura de comunicação - camada MAC

Topologia Os dispositivos VLC utilizados no presente cenário de estudo - emissor e recetor - enquadram-se na classe Infraestrutura, conforme definido na Tabela 1 do padrão IEEE 802.15.7 [4]. Em termos de topologia, o sistema aproxima-se da topologia *broadcast*, uma das três topologias MAC estabelecidas no padrão IEEE 802.15.7, dado que a comunicação é unidirecional e não requer endereçamento [4]. Nesta configuração, o emissor VLC desempenhará o papel de coordenador - controlador central único [4].

Dimming e mitigação de cintilação Considerando que o emissor VLC utiliza um *laser* (empregado exclusivamente para VLC e não para iluminação ambiente) não será implementado qualquer método de *dimming* ou mitigação de cintilação descrito pelo padrão IEEE 802.15.7 [4].

Utilização de *beacons* e estrutura *superframe* No contexto do presente cenário de estudo, e para simplificar a implementação, não serão utilizados *beacons* de rede nem estruturas *superframe* previstos no padrão IEEE 802.15.7 [4].

Modelo de transferência de dados O sistema adota o tipo de transferência de dados definido na Secção 4.5.2.2 do padrão IEEE 802.15.7, caracterizado pela transmissão de dados a partir do coordenador [4]. Dado que não são utilizados *beacons*, o emissor VLC transmite os dados recorrendo ao método de acesso aleatório não segmentado [4]. No cenário de estudo, esta transmissão ocorre a cada 60 segundos com valores aleatórios gerados para as variáveis presentes na *payload*.

Controlo de erros e verificação de integridade Dada a natureza unidirecional da comunicação, não será possível implementar frames de confirmação. Assim, será utilizado apenas o método Cyclic Redundancy Check, ou Verificação de Redundância Cíclica (CRC) para verificação de integridade. A *payload* será apresentada no *display* LCD presente no Circuito Recetor apenas se o CRC for validado corretamente no recetor. Caso contrário, o LED vermelho acenderá durante 3 segundos.

Segurança No presente cenário de estudo, não será aplicada qualquer técnica de encriptação. Importa referir, no entanto, que, apesar das redes VLC apresentarem características físicas distintas de outras redes sem fios (por exemplo, a impossibilidade de propagação através de paredes), o padrão IEEE 802.15.7 contempla a utilização de algoritmos de segurança [4].

Estrutura dos quadros Da estrutura de quadros definida no padrão IEEE 802.15.7, serão utilizados apenas *data frames*, conforme a estrutura apresentada na Subsecção seguinte - 6.2.5.

6.2.5 Estrutura dos quadros MAC

A Tabela 6.1 apresenta o formato geral adotado para o quadro MAC, definido de acordo com a estrutura de *data frame* especificada no padrão IEEE 802.15.7 [4]. A seguir, é fornecida uma descrição sucinta conforme o referido padrão:

- **Frame Control** – define, entre outros, o tipo de quadro e os campos de endereçamento. O significado dos bits associados encontra-se descrito na Tabela 6.2.
- **Frame Payload** – o formato deste campo, também designado como MAC Service Data Unit, ou Unidade de Dados de Serviço MAC (MSDU), é apresentado na Tabela 6.3.
- **Sequence Number** – indica o número de sequência do quadro.
- **Destination Address** – o valor de 16 bits iguais a 1 neste campo representa o endereço curto de *broadcast*.

- **Source Optical Wireless Personal Area Network, ou Rede Pessoal Óptica Sem Fios (OWPAN) Identifier** – especifica o identificador único OWPAN do originador do quadro, que deve ser incluído sempre que o subcampo *Source Addressing Mode* do campo *Frame Control* - consultar Tabela 6.2 - for diferente de zero. Por simplicidade, foi atribuído o valor hexadecimal `0x0`.

- **Frame Check Sequence, ou Sequência de Verificação de Quadro (FCS)** – contém o valor gerado pela aplicação do algoritmo CRC.

Campo	Grupo	Octetos	Valor binário
Frame Control	MAC Header, ou Cabeçalho MAC (MHR)	2	0100000010000101
Sequence number		1	00000000
Destination Address		2	1111111111111111
Source OWPAN Identifier		2	0000000000000000
Frame Payload	*payload* MAC	variável	variável
FCS	MAC Footer, ou Rodapé MAC (MFR)	2	variável

Tabela 6.1: Formato geral adotado para o quadro MAC [4]

Campo	Bits	Valor binário	Significado valor
Frame Version	0-1	01	Indica um quadro compatível com o padrão IEEE 802.15.7-2018
Reserved	2-5	00000	Definido como zero durante a transmissão e ignorado durante a receção
Frame Type	6-8	001	Dados
Security Enabled	9	0	Este quadro não é protegido na subcamada MAC
Frame Pending	10	0	O dispositivo que envia o quadro não tem mais dados para enviar
Acknowledgment Request	11	0	Não é necessário um reconhecimento por parte do dispositivo recetor
Destination Addressing Mode	12-13	01	Quadro de *broadcast* - os campos de endereço de origem e destino não estão presentes no quadro
Source Addressing Mode	14-15	01	

Tabela 6.2: Formato adotado para o campo Frame Control do quadro MAC [4]

Campo	Bits	Valor binário	Significado valor	
Formato utilizado para enviar os dados	0-1	00	Single	↓
Nº de Physical Protocol Data Units, ou Unidades de Dados de Protocolo Físico por quadro	2-7	variável	N/A	
Payload de dados	variável			

Tabela 6.3: Formato adotado para o campo Frame *payload* do quadro MAC [4]

6.2.6 Estrutura dos quadros físicos

No presente cenário de estudo, não será implementada a estrutura de quadros físicos estipulada pelo padrão IEEE 802.15.7 [4]. Cada quadro MAC será delimitado no início e no fim por um SFD e um EFD, respetivamente, conforme descrito na Subsecção 6.2.3.

6.3 Implementação do *hardware*

Os diagramas esquemáticos do Circuito Emissor e do Circuito Recetor estão representados na Fig. 6.2 e na Fig. 6.3, respetivamente. Doravante, serão discutidos apenas os aspetos que possam não estar suficientemente claros nos diagramas ou que mereçam destaque adicional.

6.3.1 Circuito Emissor

Laser

O *laser* selecionado apresenta as seguintes características técnicas:

- Comprimento de onda (nm): 650 (cor vermelha);
- Tensão de entrada DC (V): 4.8 - 5.2;
- Corrente máxima (mA): 35;
- Frequência máxima Transistor-Transistor Logic, ou Lógica Transistor-Transistor (TTL) (kHz): 50.

A modulação/comutação do *laser* são realizadas com recurso à lógica TTL, uma tecnologia de controlo digital que permite ligar e desligar o *laser* a alta velocidade, com uma frequência máxima de até 50 kHz. O pino TTL do *laser* recebe um sinal de controlo digital enviado pela placa de desenvolvimento ESP32, que pode assumir dois estados lógicos: ”baixo”(0 V), em que o laser é desligado, e ”alto”(entre 2 V e 5 V), em que o laser é ativado. Este método possibilita a operação do *laser* com uma frequência máxima

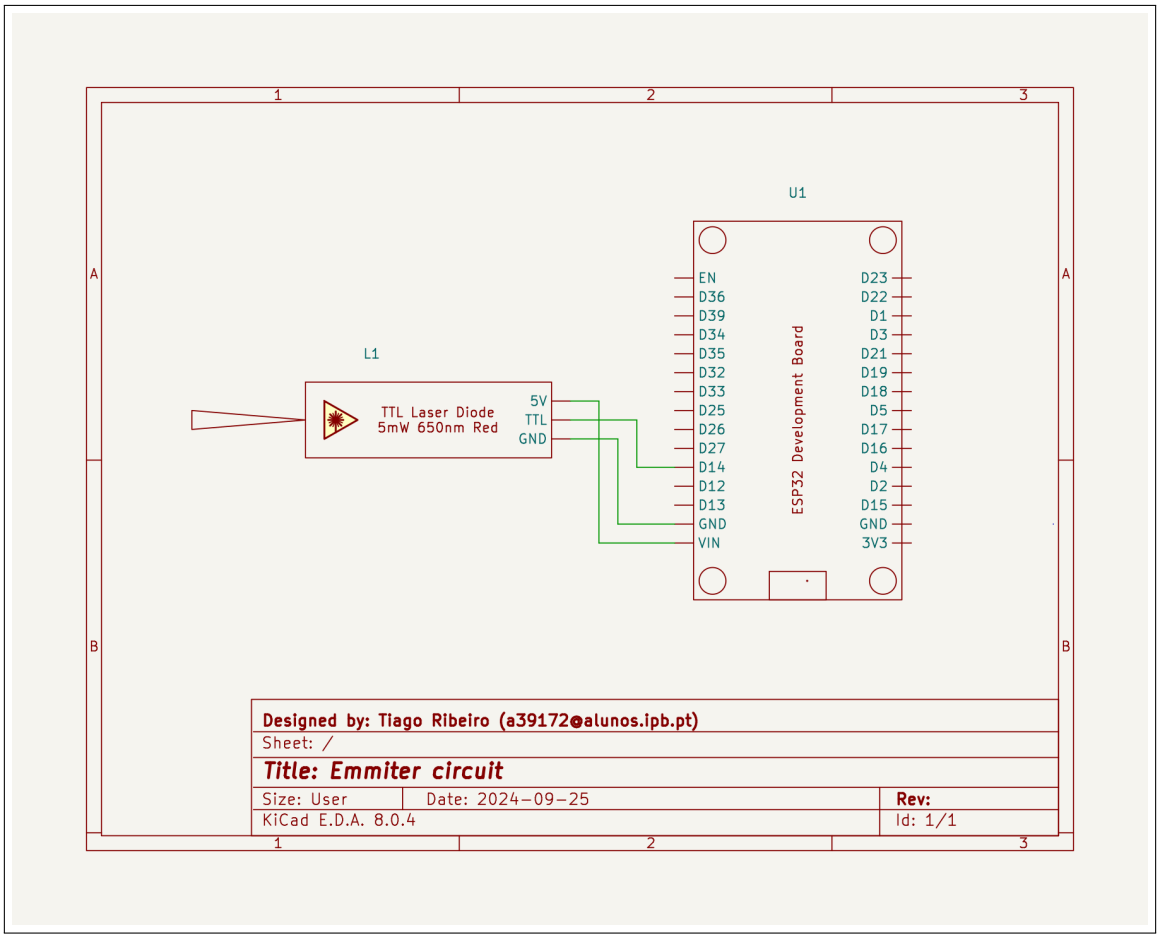

Figura 6.2: Diagrama esquemático do Circuito Emissor

de 50 kHz, o que constitui a primeira limitação identificada neste estudo, dado que esta frequência é consideravelmente inferior à taxa de relógio ótica definida pelo padrão IEEE 802.15.7 [4] e descrita na Subsecção 6.2.3 - 200 kHz.

Alimentação

O Circuito Emissor será alimentado por meio de um *powerbank* conectado à porta micro-USB da placa de desenvolvimento ESP32. A alimentação de 5V do *laser* está conectada ao pino VIN da referida placa, que possui uma ligação direta à porta micro-USB, garantindo assim a disponibilização dos 5V necessários.

6.3.2 Circuito Recetor

Alimentação

O Circuito Recetor será alimentado por um *powerbank* conectado à porta micro-USB da placa de desenvolvimento ESP32.

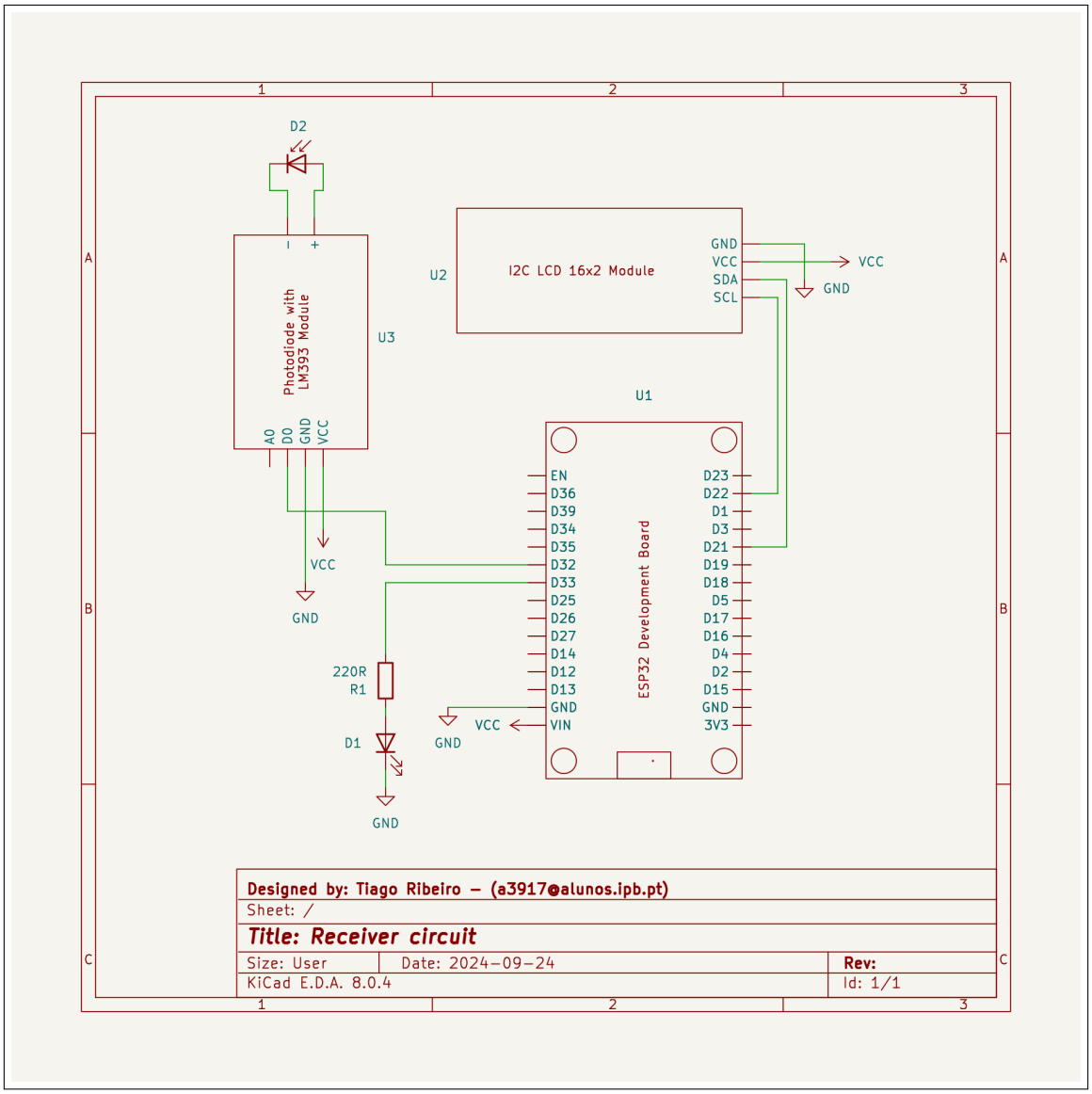

Figura 6.3: Diagrama esquemático do Circuito Recetor

Fotodíodo

O fotodíodo utilizado no sistema está integrado num módulo que incorpora o comparador de tensão LM393. Este módulo foi selecionado devido à sua simplicidade e funcionalidade, permitindo a implementação eficiente do subcircuito de deteção de luz sem a necessidade de desenvolver circuitos adicionais para o condicionamento do sinal. O LM393 atua como um comparador de tensão, comparando o sinal analógico gerado pelo fotodíodo com um valor de referência, que é ajustado por meio de um potenciómetro presente no módulo. Quando a intensidade luminosa captada pelo fotodíodo excede o valor de referência, o LM393 ativa o pino de saída digital, colocando-o em nível lógico ”alto”, detetando assim os pulsos de luz emitidos pelo *laser* através da comutação rápida entre os estados ”alto”e ”baixo”, que correspondem aos diferentes estados operativos do *laser*.

6.3.3 Prototipagem

Para validar os circuitos desenvolvidos e apoiar o processo de desenvolvimento do *software*, ambos os circuitos foram prototipados utilizando *breadboards*. Na Fig. 6.4 é apresentado o protótipo do Circuito Emissor e a Fig. 6.5 ilustra o protótipo do Circuito Recetor.

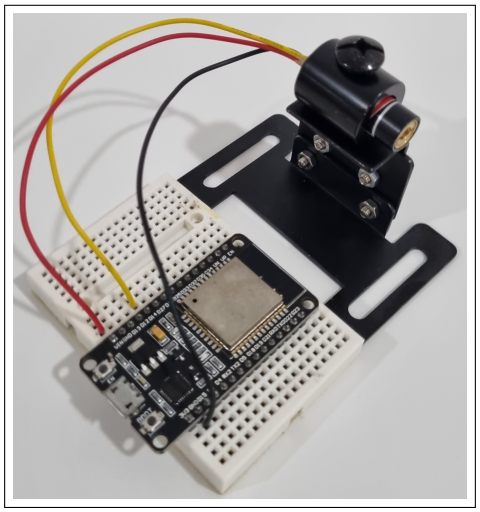

Figura 6.4: Protótipo do Circuito Emissor

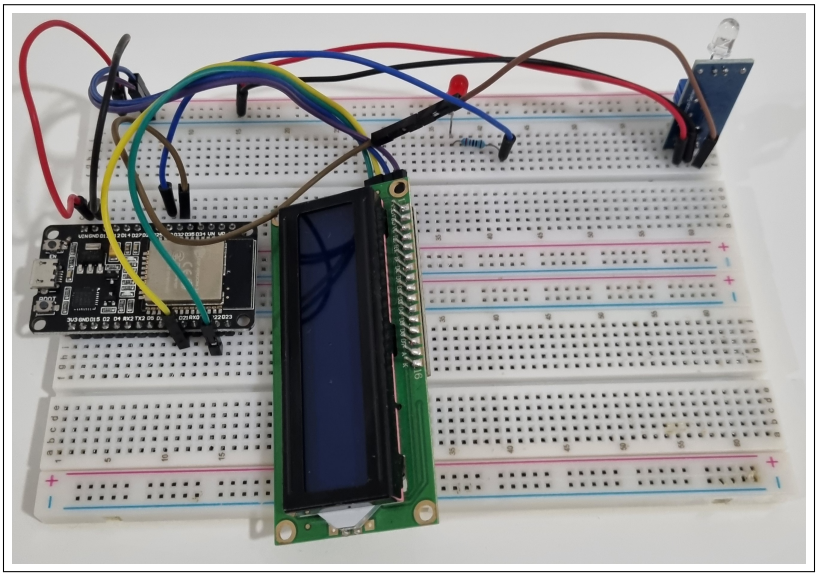

Figura 6.5: Protótipo do Circuito Recetor

No contexto do Circuito Emissor, foi utilizado um suporte ajustável que fixa o *laser* através de um parafuso de aperto e possibilita o ajuste da sua inclinação, com o objetivo de tornar o alinhamento do *laser* com o fotodíodo mais preciso e mais rápido.

6.4 Implementação do *software*

Para o desenvolvimento do *software* do Circuito Emissor e do Circuito Recetor, utilizou-se o Arduino IDE. O código associado ao Circuito Emissor encontra-se disponível no Apêndice A, enquanto o código referente ao Circuito Recetor pode ser consultado no Apêndice B.

Apresentam-se, de seguida, os fluxogramas correspondentes aos programas do Circuito Emissor e do Circuito Recetor. Adicionalmente, serão detalhados alguns pontos que possam não estar suficientemente claros na análise dos fluxogramas ou que requeiram um destaque particular.

6.4.1 Circuito Emissor

A Fig. 6.6 apresenta o fluxograma que ilustra a visão geral do programa desenvolvido para o Circuito Emissor. Os fluxogramas dos processos predefinidos "Gerar quadro MAC" e "Transmitir sinal", presentes nas Fig. 6.6, estão representados nas Fig. 6.7 e Fig. 6.8, respetivamente.

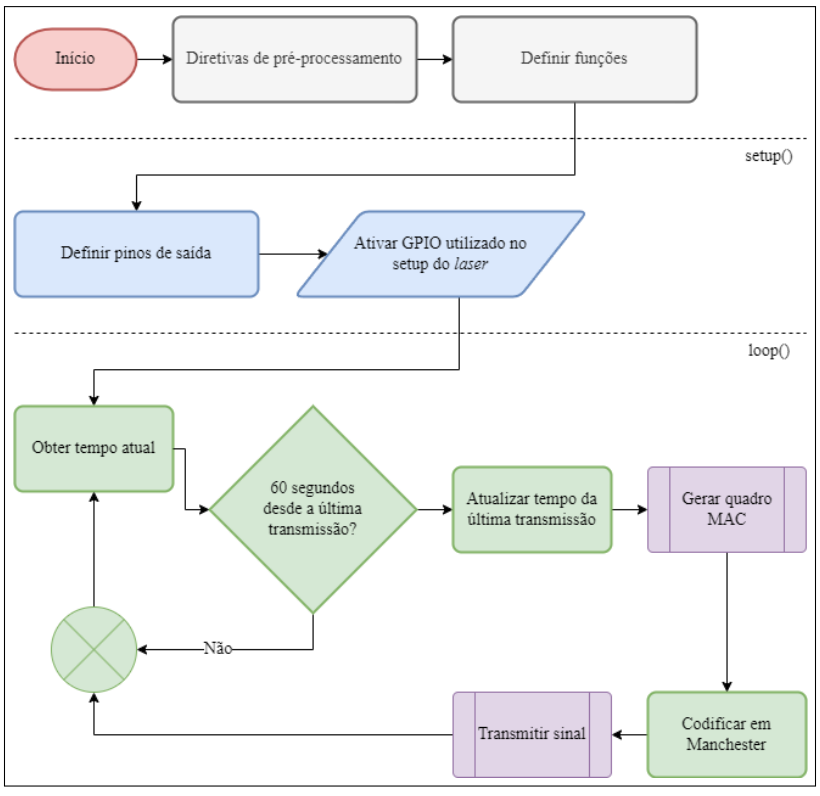

Figura 6.6: Fluxograma do programa desenvolvido para o Circuito Emissor - visão geral

Na função `setup()`, foi configurado um pino que permanecerá continuamente ativo, permitindo a sua utilização para alinhar o *laser* com o fotodíodo durante a montagem física do cenário experimental, bastando para tal conectar o fio TTL do *laser* a esse pino.

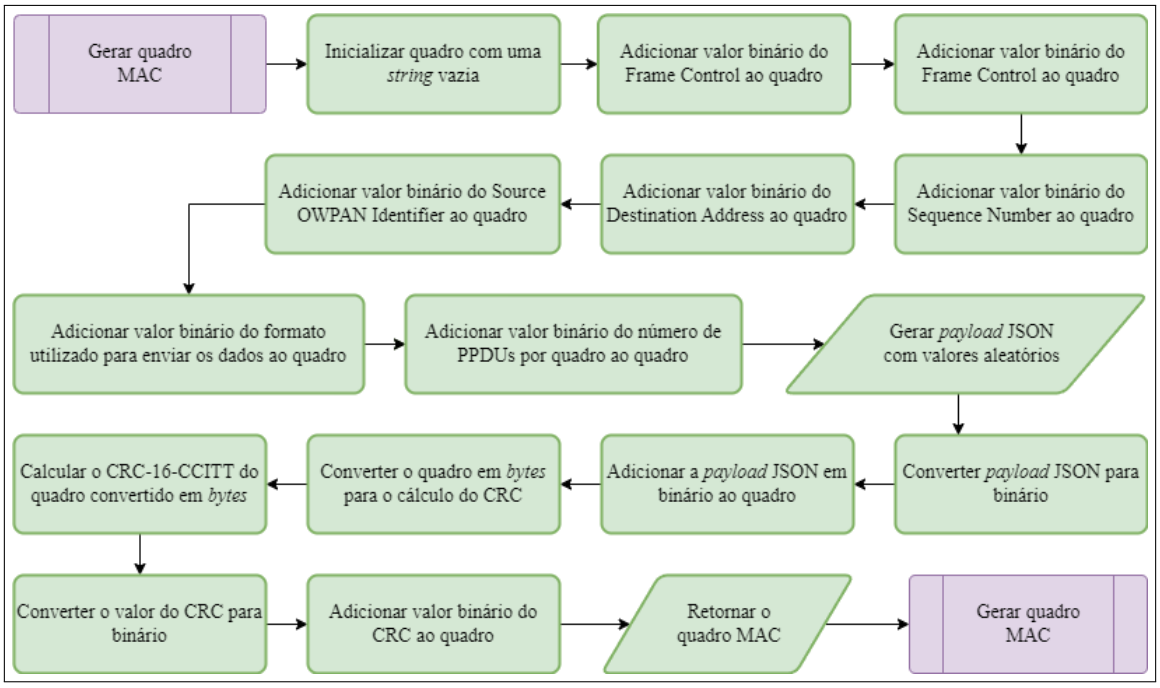

Figura 6.7: Fluxograma do programa desenvolvido para o Circuito Emissor - "Gerar quadro MAC"

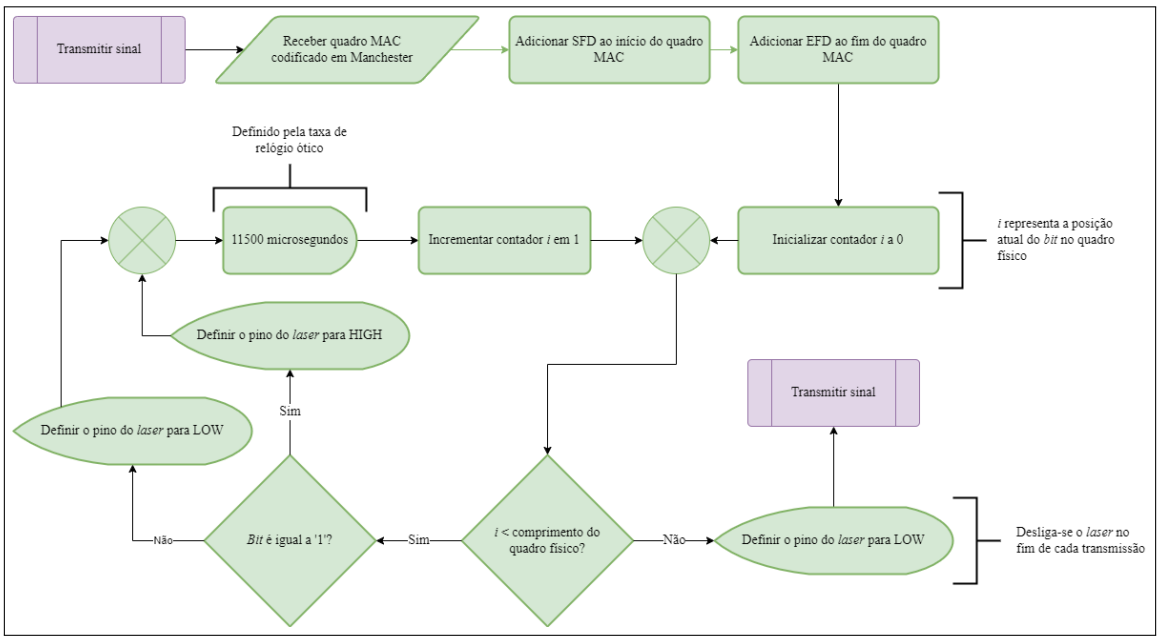

Figura 6.8: Fluxograma do programa desenvolvido para o Circuito Emissor - "Transmitir sinal"

6.4.2 Circuito Recetor

A Fig. 6.9 apresenta o fluxograma que ilustra a visão geral do programa desenvolvido para o Circuito Recetor. Os fluxogramas dos processos predefinidos "Receber sinal" e "Processar dados", presentes nas Fig. 6.9, estão representados nas Fig. 6.10 e Fig. 6.11,

respetivamente.

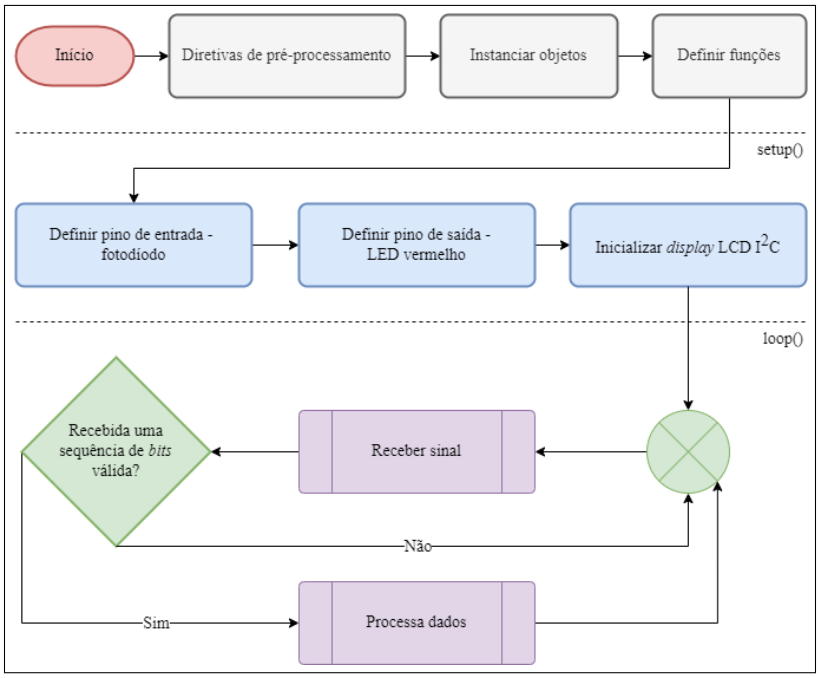

Figura 6.9: Fluxograma do programa desenvolvido para o Circuito Recetor - visão geral

6.5 Conclusão

Neste Capítulo, foi conduzida uma experiência laboratorial prática de comunicação utilizando VLC, com o intuito de explorar o uso desta na comunicação direta entre postes de iluminação pública, integrados em sistemas de transportes inteligentes. Propôs-se um cenário de estudo em que são transmitidos dados telemétricos em formato JSON, simulando o funcionamento de sensores de tráfego no sistema VALLPASS.

A arquitetura física do sistema foi detalhada, incluindo o Circuito Emissor e o Circuito Recetor, justificando a escolha de um *laser* como fonte de luz no Circuito Emissor e de um fotodíodo no Circuito Recetor, considerando fatores como eficiência energética e minimização de interferências ambientais. Descreveu-se a arquitetura de comunicação, com foco nas camadas PHY e MAC, conforme o padrão IEEE 802.15.7 [4], além das adaptações realizadas para o cenário experimental, incluindo a modulação empregue, a estrutura dos quadros e os mecanismos de sincronização.

Foram descritas as etapas de implementação do *hardware*, com a apresentação dos diagramas esquemáticos dos circuitos e o processo de prototipagem, bem como a implementação do *software*, com os fluxogramas dos programas desenvolvidos para o Circuito Emissor e o Circuito Recetor. Abordaram-se ainda aspetos como a geração e transmissão dos quadros MAC e a receção e processamento dos sinais.

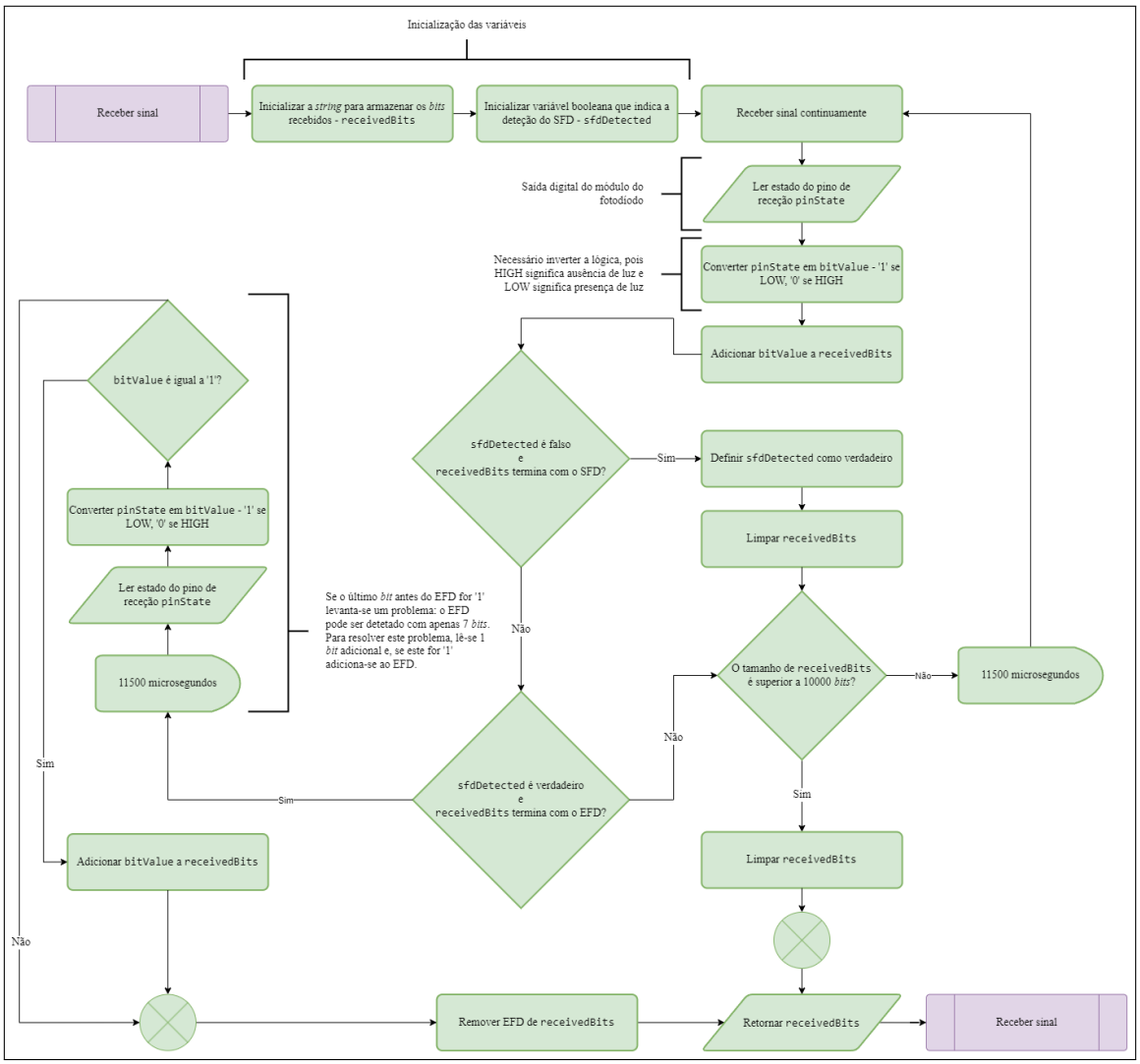

Figura 6.10: Fluxograma do programa desenvolvido para o Circuito Recetor - “Receber sinal”

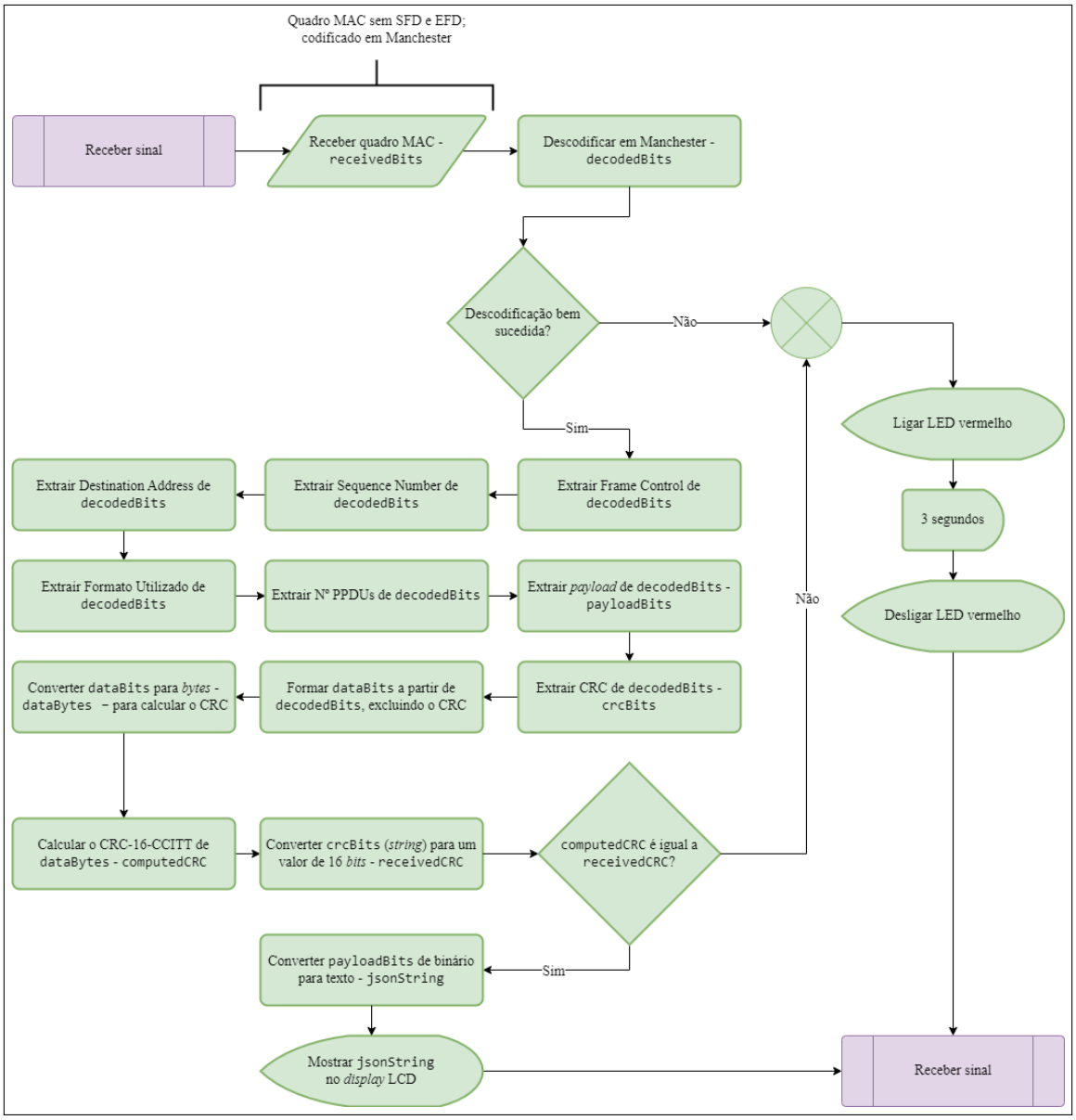

Figura 6.11: Fluxograma do programa desenvolvido para o Circuito Recetor - “Processar dados”

Capítulo 7 Análise e discussão de resultados

7.1 Introdução

Neste Capítulo, apresentam-se e discutem-se os resultados obtidos ao longo desta dissertação, organizados em duas partes principais, alinhadas com os objetivos delineados inicialmente. A primeira parte centra-se na contribuição para o projeto VALLPASS, cobrindo a integração do modelo de dados semântico, a implementação da plataforma de gestão remota e o desenvolvimento da Aplicação *Web* de gestão. A segunda parte é dedicada ao estudo da VLC, explorando a sua viabilidade e eficiência na comunicação direta entre postes de iluminação pública.

Este Capítulo visa, assim, fornecer uma visão abrangente das contribuições e inovações alcançadas, detalhando os métodos e resultados obtidos para cada objetivo específico da dissertação.

7.2 Contribuição para o projeto VALLPASS

7.2.1 Modelo de dados semântico

A integração do modelo de dados semântico na plataforma de gestão remota VALLPASS foi concluída com sucesso, viabilizando uma representação estruturada e interoperável das entidades e dos relacionamentos do sistema.

Seleção e adaptação de modelos de dados de referência

Identificou-se a disponibilidade de diversos SDMs alinhados com os requisitos do sistema VALLPASS. A utilização destes modelos de referência, providos pela Smart Data Models Initiative, simplificou o processo de modelação, reduzindo significativamente o esforço necessário para definir as entidades e as suas propriedades.

Apesar da adequação geral dos SDMs aos requisitos do VALLPASS, foram necessárias adaptações pontuais para ajustar o modelo de dados às especificidades do

sistema. Foi necessário, por exemplo, adicionar propriedades específicas às entidades Empresa e Cliente, como telefone e e-mail, usando o esquema "ContactPoint"do Schema.org [98, 99]. Adicionalmente, foram incluídas propriedades adicionais para estabelecer relações entre entidades, como a propriedade `refProvider` na entidade Cliente, que permite referenciar a organização fornecedora do sistema VALLPASS.

Desafios na integração de atuadores

Embora a maioria das entidades tenha sido integrada com sucesso, surgiram desafios específicos na integração das entidades que representam os atuadores - Poste e Luminária - devido à forma como são processados os comandos pelo Orion-LD e pelo Agente IoT para JSON [15]. Em particular, o atributo `deviceState`, que representa o estado operacional do Poste, não pôde ser configurado diretamente como um comando. Esta limitação deve-se a duas razões principais: em primeiro lugar, essa abordagem não é recomendada (os comandos devem ser atómicos); em segundo lugar, ele seria automaticamente expandido em dois atributos sem significado direto no modelo de dados: `deviceState_info` - o resultado real do comando - e `deviceState_status` - o estado do comando [15].

Para contornar esta limitação, definiram-se dois novos comandos "auxiliares" - `on` e `off` - que permitem controlar o estado destes atuadores de forma atómica. Além disso, o serviço de subscrição do Orion-LD foi aproveitado para estender a funcionalidade do *back-end* da Aplicação *Web*, onde um novo *endpoint* foi implementado para processar os atributos `on_info` e `off_info` resultantes dos comandos auxiliares, permitindo a atualização do atributo `deviceState` conforme o resultado dos comandos. Esta solução assegurou a coerência do modelo de dados e a funcionalidade correta dos atuadores no sistema.

Avaliação dos resultados

A utilização de SDMs existentes e a sua adaptação ao contexto do VALLPASS resultaram num modelo de dados semântico robusto e interoperável. A abordagem adotada permitiu uma implementação eficiente, reduzindo significativamente o tempo de desenvolvimento e promovendo a compatibilidade com outras aplicações e sistemas baseados em NGSI-LD. Os desafios encontrados, especialmente na integração dos atuadores, representaram uma oportunidade para aprofundar a compreensão dos mecanismos internos do Orion-LD e do Agente IoT para JSON. As soluções implementadas não só resolveram os problemas imediatos, como também enriqueceram a arquitetura do sistema, tornando-o mais flexível e escalável. Em suma, os resultados confirmam a viabilidade e a eficácia da abordagem para a modelação de dados semânticos em sistemas IoT, como o VALLPASS.

7.2.2 Plataforma de gestão remota

A plataforma de gestão remota implementada para o sistema VALLPASS resultou numa solução robusta e escalável, suportada por uma arquitetura de microsserviços que promove a modularidade e independência no desenvolvimento de componentes.

A adoção do *framework* FIWARE e do modelo semântico NGSI-LD favoreceu a interoperabilidade e a integração com outros serviços e aplicações, respondendo a um dos principais desafios da IoT - interoperabilidade. O uso de GEs FIWARE, como o Orion-LD para gestão de contexto e o QuantumLeap para persistência de dados históricos, incrementou a funcionalidade da plataforma e garantiu conformidade com os padrões emergentes na IoT. Este design modular permite flexibilidade na adaptação a requisitos futuros e facilita a integração de novos serviços.

A orquestração de contentores Docker com Docker Compose simplificou a implementação e a escalabilidade horizontal dos serviços, facilitando a replicação do ambiente em diversos cenários.

A segurança, outro desafio crítico na IoT, foi uma prioridade desde a conceção até à implementação. Mecanismos de autenticação e autorização providos pelo Keyrock e pelo Wilma asseguraram um controlo de acesso robusto. O uso de `secrets` Docker para armazenar credenciais sensíveis reforçou a proteção de dados nas comunicações internas do sistema.

A prototipagem do sistema VALLPASS validou tanto o modelo de dados semântico desenvolvido quanto a arquitetura proposta. A simulação das passadeiras inteligentes e dos dispositivos associados demonstrou a eficácia da plataforma no provisionamento, monitorizaçã e controlo dos dispositivos IoT.

Em suma, a plataforma de gestão remota implementada oferece uma base sólida para o sistema VALLPASS, combinando modularidade, escalabilidade e segurança. A abordagem adotada facilita futuras integrações e adaptações, posicionando o sistema VALLPASS de forma competitiva no contexto das cidades inteligentes e na gestão avançada de infraestruturas urbanas.

7.2.3 Aplicação *Web*

A implementação da Aplicação *Web* para o sistema VALLPASS resultou numa solução funcional e eficiente, composta por um *back-end* desenvolvido em Express.js e um *front-end* construído com as bibliotecas React e Material UI. A aplicação cumpre os requisitos funcionais e não funcionais estabelecidos, proporcionando uma interface intuitiva e responsiva para os utilizadores.

Back-end

O *back-end* desempenha um papel crucial na atualização dinâmica das informações exibidas no *front-end*, utilizando um mecanismo orientado a eventos que elimina a necessidade de sondagens periódicas ao Orion-LD. Esta abordagem melhorou significativamente a eficiência da aplicação, reduzindo a carga sobre o corretor de contexto e proporcionando atualizações em tempo real aos utilizadores.

Além disso, o *back-end* foi fundamental na atualização do estado operativo dos atuadores do sistema VALLPASS, baseando-se nos resultados da execução dos comandos auxiliares de ligar e desligar. Esta funcionalidade assegura a coerência dos dados e a sincronização entre os dispositivos físicos e o sistema de gestão.

A comunicação entre o *back-end* e o *front-end* foi implementada utilizando o protocolo WebSocket, através da biblioteca *Socket.IO*. Esta solução permitiu estabelecer canais de comunicação bidirecionais e eficientes, melhorando a responsividade da aplicação e proporcionando uma experiência de utilizador mais fluida. A utilização de *rooms* no *Socket.IO* para segmentar as notificações por postes específicos revelou-se eficaz, garantindo que apenas os clientes interessados recebam as atualizações relevantes. Este mecanismo contribuiu para a otimização da comunicação e para a escalabilidade da aplicação.

Front-end

O *front-end* da aplicação *web* foi desenvolvido com foco na usabilidade e na experiência do utilizador. A adoção das bibliotecas React e Material UI permitiu criar uma interface moderna e responsiva, alinhada com as melhores práticas de *design* de interfaces.

A aplicação oferece funcionalidades abrangentes para a gestão das passadeiras inteligentes e dos clientes, incluindo visualizações detalhadas dos dados operacionais, controlos para gestão remota dos dispositivos e ferramentas avançadas de filtragem e pesquisa. A inclusão de elementos como a personalização da interface através do FAB enriqueceu a experiência do utilizador, permitindo ajustes conforme as preferências individuais.

A implementação do fluxo de autenticação Authorization Code do protocolo OAuth2 com o FIWARE Keyrock facilitou o desenvolvimento do *front-end* e reforçou a premissa de escalabilidade da arquitetura da plataforma, garantindo, em expansões futuras, uma experiência de *login* uniforme e centralizada na plataforma VALLPASS.

A integração da especificação OpenAPI do modelo de dados semântico NGSI-LD na aplicação permite aos administradores uma compreensão aprofundada e intuitiva da arquitetura de dados, podendo revelar-se útil em processos de resolução de problemas ou otimizações.

7.2.4 Avaliação dos resultados

A Aplicação *Web* desenvolvida cumpriu os objetivos propostos, resultando numa plataforma sólida que serve como um bom ponto de partida para futuras expansões e melhorias. A modularidade do código e a utilização de tecnologias amplamente suportadas facilitam a manutenção e a evolução do sistema.

A segurança foi uma premissa fundamental em todo o processo de desenvolvimento. Desde a implementação de mecanismos de autenticação robustos até à proteção das comunicações entre os componentes do sistema, as medidas adotadas garantem a integridade dos dados e a fiabilidade da aplicação.

A abordagem orientada a eventos e a utilização de tecnologias como o WebSocket contribuíram para a eficiência e a escalabilidade da aplicação, permitindo o processamento em tempo real dos dados provenientes dos dispositivos IoT.

Em suma, a Aplicação *Web* do sistema VALLPASS alcançou resultados positivos, demonstrando a viabilidade e a eficácia das soluções técnicas adotadas. A plataforma está preparada para integrar novas funcionalidades e adaptar-se a futuros desafios no contexto das Cidades Inteligentes e da gestão de infraestruturas urbanas.

7.3 VLC

Para uma distância de 50,5 cm, foi estabelecida uma comunicação fiável utilizando uma frequência de relógio ótico de 86,96 Hz, o que resulta numa taxa de transmissão de 43,48 bps, em virtude da utilização da codificação Manchester. A frequência de relógio ótico atingida - 86,96 Hz - é consideravelmente inferior ao valor estipulado pelo padrão IEEE 802.15.7 para o tipo PHY selecionado (PHY I), que se encontra fixado nos 200.000 Hz [4]. Importa referir, contudo, que foram obtidas frequências de relógio ótico ligeiramente superiores, próximas dos 90 Hz, embora, nesse cenário, a maioria dos quadros físicos fosse descartada. Em determinadas circunstâncias, os quadros eram eliminados numa fase anterior ao cálculo do CRC, ocorrendo o descarte durante o processo de descodificação Manchester.

Adicionalmente, não foi possível gerar um gráfico que correlacionasse a taxa de relógio ótico com o número de erros, uma vez que o experimento não apresentou uma variação gradual e previsível da taxa de erros em função da frequência de relógio. Com efeito, para a taxa de 86,96 Hz, após se garantir um alinhamento preciso entre o *laser* e o fotodíodo e ajustar o potenciómetro de forma adequada, não houve descarte de quadros. No entanto, para frequências superiores, a comunicação revelou-se extremamente instável, resultando na validação dos quadros apenas em situações muito esporádicas e irregulares.

Apesar de não terem sido avaliadas distâncias superiores a 50,5 centímetros - principalmente devido aos desafios associados ao alinhamento do *laser* com o fotodíodo - considera-se que, com a frequência de relógio ótico de 86,96 Hz, seria viável alcançar distâncias substancialmente maiores, ultrapassando aquelas habitualmente observadas entre dois postes no sistema VALLPASS (equivalente à largura de uma passadeira de peões).

Embora a taxa de relógio ótica obtida tenha sido baixa, considera-se o experimento um sucesso, especialmente considerando os seguintes fatores:

Utilização de *hardware* não especializado Apesar de, no Circuito Emissor, o *laser* utilizado estar, desde o início, limitado a uma taxa de relógio ótica bastante inferior à preconizada pelo padrão IEEE 802.15.7 [4], considera-se que o Circuito Recetor teria beneficiado mais de um *hardware* mais sofisticado. Em particular, a utilização de um fotodíodo de melhor desempenho e, eventualmente, a substituição do módulo que o integra por um subcircuito desenhado especificamente para o experimento poderiam ter proporcionado melhores resultados. A utilização de uma *breadboard* e de *jumper wires* pode também ter contribuído negativamente, sobretudo devido à qualidade das ligações (elevada resistência e capacitância).

Ausência de implementação de mecanismos de correção de erros Não foram implementados mecanismos de correção de erros, como o FEC, especificado no padrão IEEE 802.15.7 [4], que, por definição, poderia ter tornado a comunicação significativamente mais robusta.

Limitações impostas pelo Teorema de Nyquist Outro fator que contribuiu para a baixa taxa de transmissão foi a não conformidade com o Teorema de Nyquist. No experimento, o Circuito Recetor amostrava o sinal exatamente à mesma frequência com que o Circuito Emissor transmitia. Esta abordagem, adotada para simplificar o sistema, pode ter limitado a taxa de transmissão alcançada. De acordo com o Teorema de Nyquist, para uma reconstrução precisa do sinal sem perda de informação, a frequência de amostragem no recetor deve ser, no mínimo, o dobro da frequência do sinal transmitido [108]. Ao não satisfazer esta condição, é provável que tenham ocorrido fenómenos de *aliasing*, dificultando a deteção precisa dos dados e impedindo o aumento da frequência de relógio ótico para valores mais elevados.

7.4 Conclusão

Em síntese, os resultados apresentados confirmam a eficácia das abordagens adotadas nesta dissertação. No âmbito do projeto VALLPASS, a integração do modelo de dados semântico resultou numa representação estruturada e interoperável das entidades e relacionamentos do sistema, facilitando a gestão remota e promovendo a compatibilidade

com outras aplicações baseadas em NGSI-LD. A plataforma de gestão remota revelou-se robusta e escalável, beneficiando de uma arquitetura modular suportada pelo *framework* FIWARE. A Aplicação *Web* proporcionou uma interface intuitiva e eficiente, satisfazendo aos requisitos funcionais e não funcionais estabelecidos.

Relativamente ao estudo da VLC, os resultados evidenciaram o potencial desta tecnologia na comunicação entre postes de iluminação pública. Apesar dos desafios enfrentados na implementação prática, nomeadamente limitações de *hardware* e a ausência de mecanismos de correção de erros, o experimento validou a viabilidade da VLC em distâncias pertinentes ao sistema VALLPASS.

Desta forma, a dissertação atinge os objetivos propostos, contribuindo com soluções inovadoras para a gestão eficiente de passadeiras inteligentes e explorando novas abordagens de comunicação no contexto das cidades inteligentes.

Capítulo 8 Conclusões e trabalhos futuros

8.1 Introdução

Este Capítulo apresenta as conclusões finais da dissertação, dividindo-se em duas partes principais que refletem as áreas distintas abordadas ao longo do trabalho. A primeira parte foca-se na contribuição para o projeto VALLPASS, onde se desenvolveu uma plataforma de gestão remota para o projeto VALLPASS, utilizando o *framework* FIWARE e um modelo de dados semântico NGSI-LD. A segunda parte aborda o estudo da tecnologia VLC, explorando a sua viabilidade para comunicação direta entre postes de iluminação pública.

Inicialmente, é feita uma síntese do trabalho desenvolvido em cada área, destacando-se as principais contribuições e resultados obtidos. Em seguida, discutem-se as limitações e desafios enfrentados durante o desenvolvimento. Por fim, são propostas direções para trabalhos futuros que visam aprimorar e expandir as soluções apresentadas.

8.2 Contribuição para o projeto VALLPASS

8.2.1 Síntese do trabalho desenvolvido

Nesta dissertação, desenvolveu-se uma plataforma de gestão remota para o projeto VALLPASS, orientada para o contexto de cidades inteligentes, recorrendo a tecnologias emergentes como o *framework* FIWARE e explorando a integração de modelos de dados semânticos conforme a especificação NGSI-LD. O trabalho incluiu a definição de um modelo de dados semântico customizado para as necessidades específicas do sistema VALLPASS, a conceção de uma arquitetura modular e escalável baseada em microsserviços, a prototipagem da plataforma de gestão remota e o desenvolvimento de uma Aplicação *Web* para a gestão eficiente de passadeiras inteligentes.

Esses elementos foram projetados para aprimorar a monitorização e o controlo dos dispositivos associados, promovendo a interoperabilidade entre componentes do sistema e a gestão de dados em tempo real, fatores cruciais para o desenvolvimento e sustentabi-

lidade de infraestruturas urbanas inteligentes.

8.2.2 Principais contribuições

As principais contribuições deste trabalho incluem:

Desenvolvimento de um modelo de dados semântico NGSI-LD Criação de um modelo de dados semântico robusto e interoperável, utilizando modelos de dados de referência da Smart Data Models Initiative, adaptados às especificidades do sistema VALLPASS. Esta abordagem facilita a integração com outros sistemas e aplicações baseados em NGSI-LD, promovendo a interoperabilidade no contexto da IoT aplicada a cidades inteligentes.

Implementação de uma plataforma de gestão remota Desenvolvimento de uma plataforma com arquitetura modular baseada em microsserviços, utilizando o *framework* FIWARE, o que garante modularidade, escalabilidade e independência entre componentes. A utilização de contentores Docker e do Docker Compose permitiu uma orquestração eficiente dos serviços, facilitando a replicação e a escalabilidade horizontal da plataforma.

Desenvolvimento de uma Aplicação *Web* intuitiva e responsiva Criação de uma Aplicação *Web* com *back-end* em Express.js e *front-end* em React com Material UI, oferecendo funcionalidades abrangentes para a gestão das passadeiras inteligentes e dos clientes. O uso de tecnologias modernas e a implementação de mecanismos de comunicação eficientes, como o protocolo WebSocket, proporcionam uma experiência de utilizador fluida e atualizações em tempo real.

Integração de mecanismos de segurança robustos Implementação de mecanismos de autenticação e autorização, utilizando o FIWARE Keyrock e *proxies* PEP Wilma, assegurando um controlo de acesso rigoroso e proteção dos dados sensíveis. A gestão de credenciais através de `secrets` do Docker reforçou a segurança nas comunicações internas da plataforma.

Validação do sistema através de prototipagem e simulação Simulação do sistema VALLPASS, incluindo a criação de cenários de teste e o provisionamento automático de dispositivos, permitiu validar a eficácia da plataforma de gestão remota e da Aplicação *Web*, demonstrando a sua capacidade de monitorização e controlo em tempo real.

8.3 Limitações e Desafios

Durante o desenvolvimento deste trabalho, identificaram-se algumas limitações e desafios:

Integração de atuadores no modelo NGSI-LD A integração dos atuadores - poste e luminária - apresentou desafios devido às restrições no processamento de comandos pelo Orion-LD e pelo Agente IoT para JSON. Foram necessárias soluções alternativas, como a criação de comandos auxiliares e a extensão da funcionalidade do *back-end* da Aplicação *Web*, para assegurar a coerência do modelo de dados e o funcionamento adequado dos atuadores.

Complexidade na gestão de subscrições A configuração e gestão das subscrições no Orion-LD, essenciais para garantir notificações eficientes e atualizações em tempo real, exigiram um planeamento rigoroso e uma compreensão aprofundada dos mecanismos internos do FIWARE.

Escalabilidade e desempenho Embora a arquitetura de microsserviços favoreça a escalabilidade e o protótipo da plataforma de gestão remota não tenha apresentado problemas de desempenho, a gestão eficiente de recursos e a otimização do desempenho em cenários com um grande volume de passadeiras VALLPASS instaladas continuam a ser desafios a abordar em desenvolvimentos futuros.

8.4 Trabalhos Futuros

Com base nos resultados alcançados e nas limitações identificadas, sugerem-se os seguintes trabalhos futuros:

Desenvolvimento de uma API para dados públicos Implementar uma API que disponibilize dados públicos sobre o tráfego e as condições meteorológicas recolhidos pelas passadeiras inteligentes. Esta funcionalidade pode beneficiar outros sistemas e aplicações, promovendo a integração e fomentando a criação de novos serviços para cidades inteligentes.

Melhorias na Aplicação *Web*: Embora a Aplicação *Web* forneça uma base sólida, existe potencial para adicionar mais opções de personalização, como a definição do raio de pesquisa das passadeiras e a configuração de mais preferências de visualização. Estas melhorias visam aumentar a usabilidade e adequação da aplicação às necessidades dos utilizadores.

Expansão das funcionalidades de gestão de utilizadores Implementar diferentes perfis de utilizador com níveis de acesso distintos, permitindo que clientes possam gerir apenas as suas próprias passadeiras. Esta funcionalidade aumentaria a flexibilidade do sistema e poderia suportar modelos de negócio mais complexos.

Otimização da escalabilidade e desempenho Realizar testes de carga e otimizações para garantir que a plataforma mantém um desempenho consistente com o aumento do número de dispositivos e utilizadores.

Avaliação de segurança e privacidade Conduzir análises aprofundadas de segurança para identificar potenciais vulnerabilidades e assegurar a proteção dos dados e da infraestrutura. Adicionalmente, abordar questões de privacidade na recolha e processamento de dados, assegurando conformidade com regulamentações aplicáveis.

8.5 Considerações Finais

Este trabalho contribuiu significativamente para o desenvolvimento de soluções inteligentes para a gestão de infraestruturas urbanas. A plataforma de gestão remota e a Aplicação *Web* desenvolvidas evidenciam a viabilidade e eficácia do uso de tecnologias emergentes, como o *framework* FIWARE e modelos de dados semânticos NGSI-LD, na criação de sistemas IoT escaláveis e interoperáveis.

Os resultados alcançados demonstram o potencial da plataforma de gestão remota VALLPASS para aumentar a segurança e a eficiência das passadeiras inteligentes, fornecendo ferramentas avançadas de monitorização e controlo. Este trabalho estabelece uma base sólida para expansões e melhorias futuras, promovendo a evolução contínua das Cidades Inteligentes e contribuindo diretamente para a qualidade de vida dos cidadãos.

8.6 VLC

8.6.1 Síntese do trabalho desenvolvido

No contexto da VLC, investigou-se a viabilidade e eficiência da utilização desta tecnologia para comunicação direta entre postes de iluminação pública, tomando como referência o padrão IEEE 802.15.7. A proposta surgiu no âmbito do projeto VALLPASS, com o objetivo de explorar a VLC como uma solução complementar ao LoRaWAN, visando simplificar a infraestrutura de comunicação, reduzir custos, aumentar a eficiência energética e operacional, e libertar o espectro de RF para outras aplicações.

Para atingir esse objetivo, desenvolveu-se um cenário experimental que simulou a transmissão de informações específicas sobre o tráfego de veículos, replicando o funcionamento de sensores de tráfego em ambientes urbanos. Projetaram-se e implementaram-se dois circuitos inspirados pela arquitetura de alto nível do sistema VALLPASS: um Circuito Emissor, representando um poste sem conectividade LoRaWAN, e um Circuito Recetor, simulando o poste *gateway* com ligação à nuvem. A comunicação foi realizada de forma unidirecional, utilizando um *laser* como fonte emissora e um fotodíodo como

recetor. A arquitetura de comunicação seguiu, sempre que possível, as especificações do padrão IEEE 802.15.7 para as camadas PHY e MAC, adotando-se algumas simplificações para fins experimentais.

8.6.2 Discussão dos resultados

Os resultados obtidos demonstraram a possibilidade de estabelecer uma comunicação fiável por VLC em ambiente controlado. Em condições experimentais, foi alcançada uma comunicação estável para uma distância de 50,5 cm, utilizando uma frequência de relógio ótico de 86,96 Hz, o que resultou numa taxa de transmissão efetiva de 43,48 bps, considerando a codificação Manchester empregada.

A frequência de relógio ótico obtida foi significativamente inferior ao valor estipulado pelo padrão IEEE 802.15.7 para o tipo PHY I (200 kHz) [4]. As seguintes limitações foram identificadas como possíveis fatores para a baixa taxa de transmissão alcançada:

Utilização de *hardware* não especializado No Circuito Emissor, o *laser* utilizado estava, desde o início, limitado a uma taxa de relógio ótica bastante inferior à estipulada pelo padrão IEEE 802.15.7 [4]. Contudo, considera-se que o Circuito Recetor poderia ter beneficiado mais significativamente de um *hardware* mais sofisticado, como um fotodíodo de alto desempenho. Além disso, a utilização de uma *breadboard* e *jumper wires* pode ter introduzido resistência e capacitância parasitas indesejadas nas ligações, prejudicando a qualidade da transmissão.

Ausência de implementação de mecanismos de correção de erros Não foram implementados mecanismos de correção de erros, como o FEC, especificado no padrão IEEE 802.15.7 [4], que, por definição, poderia ter tornado a comunicação significativamente mais robusta.

Limitações impostas pelo Teorema de Nyquist A frequência de amostragem no recetor era igual à frequência de transmissão, o que não cumpre o requisito do Teorema de Nyquist [108]. Esta situação pode ter conduzido a fenómenos de *aliasing*, dificultando a reconstrução precisa dos sinais e limitando a possibilidade de aumentar a taxa de transmissão.

No entanto, considerando os resultados obtidos e o impacto teórico das limitações identificadas, considera-se que a experiência realizada comprovou a viabilidade da VLC como uma solução eficaz para a transmissão de dados em cenários urbanos, demonstrando o potencial desta tecnologia para complementar as redes de comunicação existentes e reduzir a pressão sobre o espectro de RF. Os resultados obtidos fornecem uma base robusta para futuras investigações e desenvolvimentos no campo da VLC, sugerindo a integração

desta tecnologia em sistemas de monitorização e gestão de tráfego em Cidades Inteligentes.

8.6.3 Trabalhos futuros

Com base nos resultados obtidos e nas limitações identificadas, sugerem-se as seguintes direções para trabalhos futuros:

Utilização de *hardware* especializado Substituir o *laser* e o fotodíodo por componentes com capacidades superiores e considerar o desenvolvimento de circuitos dedicados para o condicionamento do sinal.

Implementação de mecanismos de correção de erros Integrar mecanismos de correção de erros, como o FEC, para aumentar a robustez da comunicação.

Teorema de Nyquist Assegurar que o sistema respeite o Teorema de Nyquist, implementando uma frequência de amostragem igual ou superior ao dobro da frequência de transmissão, minimizando assim o risco de *aliasing*.

Implementação de comunicação bidirecional Expandir o sistema para suportar comunicação bidirecional e implementar confirmações de receção.

Integração com sistemas existentes Investigar a integração da VLC com outras tecnologias de comunicação, como o LoRaWAN, numa arquitetura híbrida, com o objetivo de maximizar as vantagens de cada tecnologia e proporcionar uma solução mais completa para aplicações em Cidades Inteligentes.

Segurança e criptografia Implementar mecanismos de segurança, incluindo encriptação dos dados transmitidos, de modo a garantir a confidencialidade e integridade das comunicações.

Análise de desempenho em ambientes reais Testar o sistema em cenários reais, avaliando o impacto de fatores ambientais como iluminação natural, condições meteorológicas e interferências óticas externas.

8.6.4 Considerações finais

Este trabalho contribuiu para a exploração da VLC como uma alternativa viável para comunicação entre dispositivos em sistemas de Cidades Inteligentes.

Os resultados obtidos são promissores e indicam fortemente que, com otimizações técnicas e adoção de componentes mais apropriados, a VLC pode desempenhar um papel

significativo na melhoria da infraestrutura de comunicação do projeto VALLPASS e em sistemas análogos.

Os trabalhos futuros propostos visam superar as limitações identificadas, contribuindo para o avanço no desenvolvimento de soluções práticas e eficientes de VLC.

Conclui-se, portanto, que o estudo efetuado fornece uma base sólida para futuras investigações e desenvolvimentos no campo da VLC, contribuindo para o avanço das tecnologias de comunicação e para a construção de cidades mais inteligentes e sustentáveis.

8.7 Conclusão

Neste Capítulo, foram resumidas as principais atividades e resultados da dissertação. Na primeira parte, destacou-se o desenvolvimento de uma plataforma de gestão remota para o projeto VALLPASS, utilizando tecnologias como o *framework* FIWARE e modelos de dados semânticos NGSI-LD. Abordaram-se as contribuições realizadas, os desafios enfrentados e sugestões para trabalhos futuros.

Na segunda parte, apresentou-se o estudo sobre VLC, investigando a sua aplicação para comunicação entre postes de iluminação pública. Foram discutidos os resultados obtidos, as limitações encontradas e propostas para futuras investigações.

De forma geral, este trabalho contribuiu para o avanço de soluções inteligentes na gestão de infraestruturas urbanas, demonstrando o potencial das tecnologias emergentes para melhorar a eficiência e a segurança nas cidades inteligentes.

Bibliografia

[1] ANSR/MAI, "Acidentes de viação com vítimas: total e por tipo de acidente - continente," 2024, fonte: PORDATA, Última actualização: 2024-05-10. Dados obtidos de https://www.pordata.pt a 27-05-2024.

[2] "FIWARE Catalogue – FIWARE," Jun. 2023, acesso em: 2024-10-22. [Online]. Available: https://www.fiware.org/catalogue/

[3] "IoT Agent (JSON) - NGSI-LD Smart Farm Tutorials," acesso em: 2024-10-26. [Online]. Available: https://ngsi-ld-tutorials.readthedocs.io/en/latest/iot-agent-json.html

[4] "Ieee standard for local and metropolitan area networks–part 15.7: Short-range optical wireless communications," *IEEE Std 802.15.7-2018 (Revision of IEEE Std 802.15.7-2011)*, pp. 1–407, April 2019.

[5] "EU: pedestrian traffic fatalities," acesso em: 2024-09-14. [Online]. Available: https://www.statista.com/statistics/1197292/pedestrian-traffic-fatalities-in-the-eu/

[6] J. S. Gracias, G. S. Parnell, E. Specking, E. A. Pohl, and R. Buchanan, "Smart cities—a structured literature review," *Smart Cities*, vol. 6, no. 4, pp. 1719–1743, 2023. [Online]. Available: https://www.mdpi.com/2624-6511/6/4/80

[7] I. Hussain, "Secure, sustainable smart cities and the internet of things: Perspectives, challenges, and future directions," *Sustainability*, vol. 16, no. 4, 2024. [Online]. Available: https://www.mdpi.com/2071-1050/16/4/1390

[8] M. García-Castellano, J. M. González-Romo, J. A. Gómez-Galán, J. P. García-Martín, A. Torralba, and V. Pérez-Mira, "Iterl: A wireless adaptive system for efficient road lighting," *Sensors*, vol. 19, no. 23, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/23/5101

[9] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[10] I. Jawhar, N. Mohamed, and J. Al-Jaroodi, "Networking architectures and protocols for smart city systems," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 26, 2018. [Online]. Available: https://doi.org/10.1186/s13174-018-0097-0

[11] M. M. Kamruzzaman, "Key technologies, applications and trends of internet of things for energy-efficient 6g wireless communication in smart cities," *Energies*, vol. 15, no. 15, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/15/5608

[12] S. Ayub, S. Kariyawasam, M. Honary, and B. Honary, "A practical approach of vlc architecture for smart city," in *2013 Loughborough Antennas & Propagation Conference (LAPC)*. IEEE, 2013, pp. 106–111.

[13] P. H. Pathak, X. Feng, P. Hu, and P. Mohapatra, "Visible light communication, networking, and sensing: A survey, potential and challenges," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2047–2077, 2015.

[14] H. Elgala, R. Mesleh, and H. Haas, "Indoor optical wireless communication: potential and state-of-the-art," *IEEE Communications magazine*, vol. 49, no. 9, pp. 56–62, 2011.

[15] T. Ribeiro, J. P. Coelho, L. Jorge, J. Sardão, J. Gonçalves, and H. Rosse, "Modelling with ngsi-ld: the vallpass project case study," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–8.

[16] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[17] T. Insights, "IoT connected devices worldwide 2019-2030," Dec. 2020, acesso em: 2021-10-12. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[18] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, "Iot architecture challenges and issues: Lack of standardization," in *2016 Future Technologies Conference (FTC)*, Dec 2016, pp. 731–738.

[19] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: a survey," *IEEE Internet of things journal*, vol. 3, no. 1, pp. 70–95, 2015.

[20] G. Paolone, D. Iachetti, R. Paesani, F. Pilotti, M. Marinelli, and P. Di Felice, "A holistic overview of the internet of things ecosystem," *IoT*, vol. 3, no. 4, pp. 398–434, 2022.

[21] J. Zhang, M. Ma, P. Wang, and X. dong Sun, "Middleware for the internet of things: A survey on requirements, enabling technologies, and solutions," *Journal of Systems Architecture*, vol. 117, p. 102098, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121000795

[22] M. A. A. da Cruz, J. J. P. C. Rodrigues, J. Al-Muhtadi, V. V. Korotaev, and V. H. C. de Albuquerque, "A reference model for internet of things middleware," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 871–883, April 2018.

[23] W. Kassab and K. A. Darabkh, "A–z survey of internet of things: Architectures, protocols, applications, recent advances, future directions and recommendations," *Journal of Network and Computer Applications*, vol. 163, p. 102663, 2020.

[24] "IoT Hub | Microsoft Azure," acesso em: 2024-10-18. [Online]. Available: https://azure.microsoft.com/en-us/products/iot-hub

[25] "Gerenciamento de dispositivos IoT | AWS IoT Device Management | Amazon Web Services," acesso em: 2024-10-18. [Online]. Available: https://aws.amazon.com/pt/iot-device-management/

[26] "What Types of IoT Platforms Are There and How to Choose the Right One? | IronFlock Blog," acesso em: 2024-10-18. [Online]. Available: https://www.ironflock.com/blog/what-types-of-iot-platforms-are-there-and-how-to-choose-the-right-one

[27] "Main - CHOREVOLUTION," acesso em: 2024-10-19. [Online]. Available: https://chorevolution.ow2.org/view/Main/

[28] "Home | m2mlabs.com," acesso em: 2024-10-19. [Online]. Available: http://www.m2mlabs.com/

[29] thingsboard, "ThingsBoard — Open-source IoT (Internet of Things) Platform," acesso em: 2024-10-19. [Online]. Available: https://thingsboard.io/

[30] "FIWARE - Open APIs for Open Minds," Feb. 2021, acesso em: 2024-10-19. [Online]. Available: https://www.fiware.org/

[31] I. I. Consortium, "Digital twins for industrial applications: Definition, business value, design aspects, standards, and use cases," Industrial Internet

Consortium, Tech. Rep., 2020, acesso em: 2024-10-10. [Online]. Available: https://www.iiconsortium.org/pdf/IIC_Digital_Twins_Industrial_Apps_White_Paper_2020-02-18.pdf

[32] Gartner, Inc., "Digital twin," acesso em: 2024-10-10. [Online]. Available: https://www.gartner.com/en/information-technology/glossary/digital-twin

[33] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *The International Journal of Advanced Manufacturing Technology*, vol. 94, pp. 3563–3576, 2018.

[34] Q. Qi and F. Tao, "Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison," *Ieee Access*, vol. 6, pp. 3585–3593, 2018.

[35] R. F. El-Agamy, H. A. Sayed, A. M. AL Akhatatneh, M. Aljohani, and M. Elhosseini, "Comprehensive analysis of digital twins in smart cities: a 4200-paper bibliometric study," *Artificial Intelligence Review*, vol. 57, no. 6, p. 154, 2024.

[36] "Smart Cities Market Size to Reach USD 1,427.84 Billion in 2030," acesso em: 2023-01-11. [Online]. Available: https://finance.yahoo.com/news/smart-cities-market-size-reach-163000941.html

[37] V. Albino, U. Berardi, and R. M. Dangelico, "Smart cities: Definitions, dimensions, performance, and initiatives," *Journal of urban technology*, vol. 22, no. 1, pp. 3–21, 2015.

[38] R. E. Hall, B. Bowerman, J. Braverman, J. Taylor, H. Todosow, and U. V. Wimmersperg, "The vision of a smart city," Brookhaven National Lab. (BNL), Upton, NY (United States), Tech. Rep., 2000.

[39] A. Abella, M. O. de Urbina-Criado, and C. De-Pablos-Heredero, "A model for the analysis of data-driven innovation and value generation in smart cities' ecosystems," *Cities*, vol. 64, pp. 47–53, 2017.

[40] S.-R. Oh and Y.-G. Kim, "Security requirements analysis for the iot," in *2017 International Conference on Platform Technology and Service (PlatCon)*. IEEE, 2017, pp. 1–6.

[41] "ETSI GS CIM 006 V1.1.1," ETSI, Tech. Rep., July 2019, acesso em: 2023-06-05. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/006/01.01.01_60/gs_CIM006v010101p.pdf

[42] D. Bees, L. Frost, M. Bauer, M. Fisher, and W. Li, "NGSI-LD API: For Context Information Management," ETSI, Tech. Rep., January 2019, acesso em: 2023-06-05. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp31_NGSI_API.pdf

[43] T. Berners-Lee and M. Fischetti, *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. Harper San Francisco, 1999.

[44] R. Uceda-Sosa, B. Srivastava, and R. J. Schloss, "Building a highly consumable semantic model for smarter cities," in *Proceedings of the AI for an Intelligent Planet*, 2011, pp. 1–8.

[45] B. Rocha, E. Cavalcante, T. Batista, and J. Silva, "A linked data-based semantic information model for smart cities," in *2019 IX Brazilian Symposium on Computing Systems Engineering (SBESC)*. IEEE, 2019, pp. 1–8.

[46] N. Zhang, H. Chen, X. Chen, and J. Chen, "Semantic framework of internet of things for smart cities: Case studies," *Sensors*, vol. 16, no. 9, p. 1501, 2016.

[47] J. Lee, S. Jeong, S. K. Yoo, and J. Song, "Ssf: Smart city semantics framework for reusability of semantic data," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2021, pp. 1625–1627.

[48] B. Villalón, D. M., L. S. Jiménez, M. de la Iglesia, M. E. Campos, and T. D. Fernández, "An iot architecture for smart cities based on the fiware platform," *Revista de Ciencia y Tecnología*, no. 38, pp. 21–30, 2022.

[49] L.-G. Cretu, "Smart cities design using event-driven paradigm and semantic web," *Informatica Economica*, vol. 16, no. 4, p. 57, 2012.

[50] P. Gonzalez-Gil, J. A. Martinez, and A. Skarmeta, "A prosumer-oriented, interoperable, modular and secure smart home energy management system architecture," *Smart Cities*, vol. 5, no. 3, pp. 1054–1078, 2022.

[51] "About FIWARE – FIWARE," October 2021, acesso em: 2023-02-09. [Online]. Available: https://www.fiware.org/about-us/

[52] "FIWARE, the standard that the IoT needs," acesso em: 2023-02-09. [Online]. Available: https://www.tmforum.org/press-and-news/fiware-standard-iot-needs/

[53] "Developers Catalogue – FIWARE," May 2018, acesso em: 2023-02-09. [Online]. Available: https://www.fiware.org/catalogue/

[54] "FIWARE, Information Platform for Implementing Data Utilization Based City Management: NEC Technical Journal — NEC," acesso em: 2023-02-09. [Online]. Available: https://www.nec.com/en/global/techrep/journal/g18/n01/180109.html

[55] "FIWARE – Cities Directory," November 2021, acesso em: 2023-02-09. [Online]. Available: https://www.fiware.org/about-us/smart-cities/cities-directory/

[56] "Smart Data Models – FIWARE," March 2020, acesso em: 2023-02-09. [Online]. Available: https://www.fiware.org/smart-data-models/

[57] "Authentication vs. Authorization | Okta," acesso em: 2023-02-19. [Online]. Available: https://www.okta.com/identity-101/authentication-vs-authorization/

[58] "Administrating users - step-by-step for ngsi-v2," acesso em: 2023-02-19. [Online]. Available: https://fiware-tutorials.readthedocs.io/en/latest/identity-management.html

[59] "Securing application access - step-by-step for ngsi-v2," acesso em: 2023-02-19. [Online]. Available: https://fiware-tutorials.readthedocs.io/en/latest/securing-access.html

[60] "Oauth 2.0 — oauth," acesso em: 2023-02-19. [Online]. Available: https://oauth.net/2/

[61] "What is Docker?" Sep. 2024, acesso em: 2024-10-23. [Online]. Available: https://docs.docker.com/get-started/docker-overview/

[62] "Docker Hub," Oct. 2024, acesso em: 2024-10-23. [Online]. Available: https://docs.docker.com/docker-hub/

[63] "Manage repositories," Sep. 2024, acesso em: 2024-10-23. [Online]. Available: https://docs.docker.com/docker-hub/repos/

[64] "Node-RED," acesso em: 2024-10-22. [Online]. Available: https://nodered.org/

[65] "About : Node-RED," acesso em: 2024-10-22. [Online]. Available: https://nodered.org/about/

[66] "Library - Node-RED," acesso em: 2024-10-22. [Online]. Available: https://flows.nodered.org/

[67] "Express - Node.js web application framework," acesso em: 2024-10-14. [Online]. Available: https://expressjs.com

[68] “Using Express middleware,” acesso em: 2024-10-14. [Online]. Available: https://expressjs.com

[69] “Express routing,” acesso em: 2024-10-14. [Online]. Available: https://expressjs.com

[70] “Using template engines with Express,” acesso em: 2024-10-14. [Online]. Available: https://expressjs.com

[71] “10 Best Nodejs Frameworks for App Development in 2024,” May 2023, acesso em: 2024-10-14. [Online]. Available: https://www.geeksforgeeks.org/best-nodejs-frameworks-for-app-development/

[72] “Virtual DOM and Internals – React,” acesso em: 2024-10-15. [Online]. Available: https://legacy.reactjs.org/docs/faq-internals.html

[73] “Quick Start – React,” acesso em: 2024-10-15. [Online]. Available: https://react.dev/learn

[74] “Writing Markup with JSX – React,” acesso em: 2024-10-15. [Online]. Available: https://react.dev/learn/writing-markup-with-jsx

[75] “Technology | 2024 Stack Overflow Developer Survey,” acesso em: 2024-10-15. [Online]. Available: https://survey.stackoverflow.co/2024/technology#2-web-frameworks-and-technologies

[76] “Overview - Material UI,” acesso em: 2024-10-15. [Online]. Available: https://mui.com/material-ui/getting-started/

[77] M. Bhardwaj, S. Singh, S. Tyagi, A. Tripathi, Y.-C. Hu, R. K. Tewari, and A. Sharma, “A novel architecture for the smart pedestrian crossing in cities using iot-based approach,” *Mathematical Problems in Engineering*, vol. 2023, no. 1, p. 7334013, 2023.

[78] T. d. J. M. Sanguino, J. M. L. Domínguez, M. J. R. González, and J. M. D. Martin, “New approach to intelligent pedestrian detection and signaling on crosswalks,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–0, 2024.

[79] S. Jeong, S. Kim, and J. Kim, “City data hub: Implementation of standard-based smart city data platform for interoperability,” *Sensors*, vol. 20, no. 23, p. 7000, 2020.

[80] O. Hylli, V. Heikkilä, and K. Systä, “Collecting data to fiware,” 2020.

[81] J. Conde, A. Munoz-Arcentales, A. Alonso, S. López-Pernas, and J. Salvachua, "Modeling digital twin data and architecture: A building guide with fiware as enabling technology," *IEEE Internet Computing*, vol. 26, no. 3, pp. 7–14, 2021.

[82] J. Conde, A. Munoz-Arcentales, Á. Alonso, G. Huecas, and J. Salvachúa, "Collaboration of digital twins through linked open data: Architecture with fiware as enabling technology," *IT Professional*, vol. 24, no. 6, pp. 41–46, 2022.

[83] T. Cevik and S. Yilmaz, "An overview of visible light communication systems," *arXiv preprint arXiv:1512.03568*, 2015.

[84] F. Delgado-Rajo, A. Melian-Segura, V. Guerra, R. Perez-Jimenez, and D. Sanchez-Rodriguez, "Hybrid rf/vlc network architecture for the internet of things," *Sensors*, vol. 20, no. 2, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/2/478

[85] S. Shao, A. Khreishah, M. B. Rahaim, H. Elgala, M. Ayyash, T. D. Little, and J. Wu, "An indoor hybrid wifi-vlc internet access system," in *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, 2014, pp. 569–574.

[86] "Openapi specification v3.1.0 — introduction, definitions, & more," acesso em: 2023-03-29. [Online]. Available: https://spec.openapis.org/oas/v3.1.0

[87] G. Privat, "Guidelines for modelling with ngsi-ld," ETSI, Tech. Rep., March 2021, acesso em: 2023-06-05. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp_42_NGSI_LD.pdf

[88] "Organization," acesso em: 2023-03-29. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Organization/Organization/swagger.yaml

[89] "Roadsegment," acesso em: 2023-03-29. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Transportation/RoadSegment/swagger.yaml

[90] "Device," acesso em: 2023-03-29. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Device/Device/swagger.yaml

[91] "Weatherobserved," acesso em: 2023-03-27. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Weather/WeatherObserved/swagger.yaml

[92] “Itemflowobserved,” acesso em: 2023-03-27. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Transportation/ItemFlowObserved/swagger.yaml

[93] “Roadaccident,” acesso em: 2023-03-27. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Transportation/RoadAccident/swagger.yaml

[94] “Streetlightmodel,” acesso em: 2023-03-29. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Streetlighting/StreetlightModel/swagger.yaml

[95] “Streetlight,” acesso em: 2023-03-29. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Streetlighting/Streetlight/swagger.yaml

[96] “Storagebatterydevice,” acesso em: 2023-03-27. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Battery/StorageBatteryDevice/swagger.yaml

[97] “Storagebatterymeasurement,” acesso em: 2023-03-27. [Online]. Available: https://swagger.lab.fiware.org/?url=https://smart-data-models.github.io/dataModel.Battery/StorageBatteryMeasurement/swagger.yaml

[98] S. D. M. Initiative, “Schema.org,” 2022, acesso em: 2023-03-29. [Online]. Available: https://github.com/smart-data-models/data-models/blob/master/schema.org.yaml

[99] “Contactpoint - schema.org type,” acesso em: 2023-03-26. [Online]. Available: https://schema.org/ContactPoint

[100] S. D. Models, “common-schema.json (source code) [github repository],” 2020, acesso em: 2023-03-27. [Online]. Available: https://github.com/smart-data-models/data-models/blob/master/common-schema.json

[101] F. Foundation, “Understanding @context - step-by-step for ngsi-ld,” 2023, acesso em: 2023-03-29. [Online]. Available: https://ngsi-ld-tutorials.readthedocs.io/en/latest/understanding-@context.html

[102] “Securing Microservices - NGSI-v2 Smart Supermarket Tutorials,” acesso em: 2024-10-28. [Online]. Available: https://fiware-tutorials.readthedocs.io/en/latest/pep-proxy.html#securing-an-iot-agent-south-port

[103] “Working with @context - NGSI-LD Smart Farm Tutorials,” acesso em: 2024-10-28. [Online]. Available: https://ngsi-ld-tutorials.readthedocs.io/en/latest/working-with-%40context.html#architecture

[104] “Time-Series Data - NGSI-LD Smart Farm Tutorials,” acesso em: 2024-10-27. [Online]. Available: https://ngsi-ld-tutorials.readthedocs.io/en/latest/time-series-data.html

[105] “Nominatim,” acesso em: 2024-10-28. [Online]. Available: https://nominatim.org/

[106] “Introduction - Fiware-iotagent-json,” acesso em: 2024-10-26. [Online]. Available: https://fiware-iotagent-json.readthedocs.io/en/latest/stepbystep.html#provisioning-multiple-devices-with-a-configuration

[107] W. contributors. (2022, aug) Websocket. Acesso em: 2022-09-24. [Online]. Available: https://en.wikipedia.org/w/index.php?title=WebSocket&oldid=1102718692

[108] S. W. Smith, *The Scientist and Engineer’s Guide to Digital Signal Processing*. California Technical Publishing, 1997.

Apêndice A Código do Circuito Emissor

```
#include <Arduino.h>
#include <esp_crc.h>
#include <stdlib.h>
#include <math.h>

// Definir o pino de saída para o laser - OOK (GPIO 14)
#define PIN_LASER 14

// Definir o pino de saída para ajuste do laser; estará sempre ligado
#define PIN_SETUP 27

// Período de cada transição Manchester em microsegundos
#define BIT_PERIOD_US 11500

// Intervalo de transmissão de 60 segundos (em milissegundos)
#define TRANSMISSION_INTERVAL 60000

// Variável para armazenar o tempo da última transmissão
unsigned long lastTransmissionTime = 0;

// Função para gerar um número aleatório em ponto flutuante com precisão decimal especificada (para utilização na \textit{payload} JSON)
float getRandomFloat(float min, float max, int precision) {
  float random = ((float)rand() / (float)RAND_MAX) * (max - min) + min;
  return round(random * pow(10, precision)) / pow(10, precision);
}

// Função para converter uma string em binário
String stringToBinary(String input) {
  String binaryString = "";

  // Percorrer cada carácter da string
  for (int i = 0; i < input.length(); i++) {
    // Converter o carácter para binário
    char c = input.charAt(i);
    for (int j = 7; j >= 0; j--) {
      binaryString += String((c >> j) & 1);  // Calcular cada bit do carácter e adicioná-lo diretamente à string
```

```
37        }
38      }
39      return binaryString;
40    }
41
42    // Função para gerar o \textit{payload} JSON com valores aleat
          órios dentro dos intervalos especificados
43    String generatePayload() {
44      float i = getRandomFloat(0, 300, 0);  // número de veículos
45      float a = getRandomFloat(0, 80, 1);   // velocidade média de
            todos os veículos
46      float l = getRandomFloat(0, 30, 1);   // velocidade mínima
            registada para os veículos
47      float h = getRandomFloat(30, 80, 1);  // velocidade máxima
            registada para os veículos
48
49      String \textit{payload} = "{␣\"topic\":␣\"/json/
            wmnshtp4yfjysxs62oju/IPB_BGC_0000000_0_V/attrs\",␣\"
            payload\":␣{";
50      \textit{payload} += "\"i\":␣" + String(i, 0) + ",";
51      \textit{payload} += "\"a\":␣" + String(a, 1) + ",";
52      \textit{payload} += "\"l\":␣" + String(l, 1) + ",";
53      \textit{payload} += "\"h\":␣" + String(h, 1);
54      \textit{payload} += "␣}␣}";
55
56      return payload;
57    }
58
59    // Função para codificar uma string binária usando Manchester
60    String manchesterEncode(String input) {
61      String encoded = "";
62
63      // Codificação Manchester: 10 para '1' e 01 para '0'
64      for (int i = 0; i < input.length(); i++) {
65        if (input[i] == '1') {
66          encoded += "10";
67        } else if (input[i] == '0') {
68          encoded += "01";
69        }
70      }
71
72      return encoded;
73    }
74
75    // Função para calcular o CRC-16 usando o algoritmo CRC-16-
          CCITT (XMODEM)
76    uint16_t calcularCRC16(uint8_t* dados, size_t comprimento) {
77      uint16_t crc = 0x0000;         // Inicializar o CRC com zero
78      uint16_t polinomio = 0x1021;  // Polinómio gerador G(x) = x
            ^16 + x^12 + x^5 + 1
79
80      // Para cada byte na sequência de entrada
81      for (size_t i = 0; i < comprimento; i++) {
82        crc ^= (dados[i] << 8);  // XOR do byte atual com os 8
              bits mais significativos do CRC
83
84        // Processar cada um dos 8 bits do byte
```

```
85        for (uint8_t j = 0; j < 8; j++) {
86          // Verificar se o bit mais significativo do CRC é 1
87          if (crc & 0x8000) {
88            // Deslocar o CRC para a esquerda e aplicar o polinó
                 mio
89            crc = (crc << 1) ^ polinomio;
90          } else {
91            // Apenas deslocar o CRC para a esquerda
92            crc <<= 1;
93          }
94        }
95      }
96      return crc;
97    }
98
99    // Função para gerar o quadro MAC (incluindo CRC e \textit{
         payload} aleatória)
100   String generateMacFrame() {
101     String frame = "";
102
103     // Frame control: 16 bits (0100000010000101)
104     frame += "0100000010000101";
105
106     // Sequence number: 8 bits (00000000)
107     frame += "00000000";
108
109     // Destination address: 16 bits (1111111111111111)
110     frame += "1111111111111111";
111
112     // Source OWPAN Identifier: 16 bits (0000000000000000)
113     frame += "0000000000000000";
114
115     // Formato usado para enviar os dados: 2 bits (00)
116     frame += "00";
117
118     // Número de PPDUs por quadro de dados: 2 bits (01) para 1
           PPDU
119     frame += "01";
120
121     // Gerar \textit{payload} aleatória JSON
122     String \textit{payload} = generatePayload();
123
124     // Converter a \textit{payload} aleatória JSON para binário
125     frame += stringToBinary(payload);
126
127     // Converter o quadro para bytes - necessário para calcular
           o CRC
128     uint8_t frameBytes[frame.length() + 1];
129     frame.getBytes(frameBytes, frame.length() + 1);
130
131     // Calcular o CRC-16 e adicioná-lo ao final do quadro
132     uint16_t crc = calcularCRC16(frameBytes, frame.length());
133
134     // Adicionar o CRC, em binário, ao quadro (16 bits)
135     char crcStr[17];
136     for (int i = 15; i >= 0; i--) {
137       crcStr[15 - i] = ((crc >> i) & 1) ? '1' : '0';
```

```
138       }
139       crcStr[16] = '\0';
140       frame += crcStr;
141
142       return frame;
143     }
144
145     // Função para transmitir o sinal codificado para o laser -
          Manchester + OOK
146     void transmitSignal(String ookEncoded) {
147
148       // Pseudo SFD e pseudo EFD - sem codificação Manchester
149       String fd = "11111111";  // 8 bits de '1'
150       ookEncoded = fd + ookEncoded + fd;
151
152       // Transmitir o quadro bit a bit
153       for (int i = 0; i < ookEncoded.length(); i++) {
154         if (ookEncoded[i] == '1') {
155           digitalWrite(PIN_LASER, HIGH);  // Transmitir "1" (HIGH)
                no GPIO 14
156         } else if (ookEncoded[i] == '0') {
157           digitalWrite(PIN_LASER, LOW);   // Transmitir "0" (LOW)
                no GPIO 14
158         }
159         delayMicroseconds(BIT_PERIOD_US);  // Manter cada estado
              por determinado número de us - taxa de clock
160       }
161
162       // Desligar o laser no fim de cada transmissão (caso contrá
            rio ficaria ligado devido ao EFD)
163       digitalWrite(PIN_LASER, LOW);
164     }
165
166     // Configuração inicial do sistema
167     void setup() {
168       Serial.begin(115200);
169
170       // Configurar o GPIO associado ao laser como saída
171       pinMode(PIN_LASER, OUTPUT);
172
173       // Configurar o GPIO utilizado para o setup do laser como sa
            ída
174       pinMode(PIN_SETUP, OUTPUT);
175
176       // Ativar permanentemente o GPIO utilizado para o setup do
            laser
177       digitalWrite(PIN_SETUP, HIGH);
178
179       // Forçar a primeira transmissão imediatamente
180       lastTransmissionTime = millis() - TRANSMISSION_INTERVAL;
181     }
182
183     // Loop principal para transmissão periódica
184     void loop() {
185       // Verificar se o intervalo de transmissão se passou ou se é
            a primeira execução
```

```
186       if (millis() - lastTransmissionTime >= TRANSMISSION_INTERVAL
              ) {
187         // Atualizar o tempo da última transmissão
188         lastTransmissionTime = millis();
189 
190         // Gerar quadro MAC com \textit{payload} aleatória
191         String macFrame = generateMacFrame();
192         Serial.println("Quadro␣MAC␣com␣CRC␣e␣\textit{payload}␣
                Aleatória:␣" + macFrame);
193 
194         // Codificar o quadro em Manchester
195         String manchesterEncoded = manchesterEncode(macFrame);
196         Serial.println("Codificado␣Manchester:␣" +
                manchesterEncoded);
197 
198         // Transmitir o sinal no laser
199         transmitSignal(manchesterEncoded);
200       }
201     }
```

Apêndice B Código do Circuito Recetor

```
#include <Arduino.h>
#include <stdlib.h>
#include <LiquidCrystal_I2C.h>

// Definição de constantes para os pinos e parâmetros de comunicação

// Pino ao qual o fotodíodo está ligado
#define PIN_PHOTODIODE 32
// Pino do LED vermelho que sinaliza falhas de comunicação
#define PIN_COMMUNICATION_FAILURE 33

// Define o período de um bit em microsegundos (19 ms)
#define BIT_PERIOD_US 11500

// Start Frame Delimiter (SFD): 8 bits de '1' que indicam o início do quadro MAC (não codificados em Manchester)
#define SFD "11111111"

// End Frame Delimiter (EFD): 8 bits de '1' que indicam o fim do quadro MAC (não codificados em Manchester)
#define EFD "11111111"

// Comprimento do SFD e EFD em bits (8 bits)
#define SFD_LENGTH 8

LiquidCrystal_I2C lcd(0x27, 16, 2);

// Função para exibir o tópico e o \textit{payload} com scroll individual
void displayScroll(String topic, String payload) {

  // Inicializa variáveis para a posição de scroll de cada linha
  int topicPosition = 0;
  int payloadPosition = 0;
  bool topicScrollComplete = false;
  bool payloadScrollComplete = false;

  // Inicializa o display dentro da função
  lcd.init();
  lcd.backlight();
```

```
38
39      // Calcula a posição final para o scroll de cada linha
40      int topicStopPosition = (topic.length() > 16) ? topic.length
            () - 16 : 0;
41      int payloadStopPosition = (payload.length() > 16) ? payload.
            length() - 16 : 0;
42
43      // Tempo total de scroll (em milissegundos)
44      unsigned long startTime = millis();
45      unsigned long scrollDuration = 12000;  // Tempo de scroll
            (12 segundos)
46
47      while (millis() - startTime < scrollDuration) {
48
49        // Atualiza a linha superior (tópico) com scroll, se ainda
              não alcançou o final
50        if (!topicScrollComplete) {
51          String topicScroll = topic.substring(topicPosition, (
                topicPosition + 16 < topic.length()) ? topicPosition
                + 16 : topic.length());
52          lcd.setCursor(0, 0);
53          lcd.print(topicScroll);
54
55          // Preenche com espaços em branco se necessário
56          if (topicScroll.length() < 16) {
57            lcd.print("␣");
58          }
59
60          // Verifica se o tópico chegou ao ponto de paragem
                calculado
61          if (topicPosition >= topicStopPosition) {
62            topicScrollComplete = true;
63          } else {
64            topicPosition++;  // Incrementa a posição para
                  continuar o scroll
65          }
66        }
67
68        // Atualiza a linha inferior (payload) com scroll, se
              ainda não alcançou o final
69        if (!payloadScrollComplete) {
70          String payloadScroll = payload.substring(payloadPosition
                , (payloadPosition + 16 < payload.length()) ?
                payloadPosition + 16 : payload.length());
71          lcd.setCursor(0, 1);
72          lcd.print(payloadScroll);
73
74          // Preenche com espaços em branco se necessário
75          if (payloadScroll.length() < 16) {
76            lcd.print("␣");
77          }
78
79          // Verifica se a posição do \textit{payload} alcançou o
                ponto de paragem
80          if (payloadPosition >= payloadStopPosition) {
81            payloadScrollComplete = true;
82          } else {
```

```
83            payloadPosition++;  // Incrementa a posição para
                  continuar o scroll
84          }
85        }
86
87        // Controla a velocidade do scroll
88        delay(200);
89
90        // Termina o loop se ambos os scrolls estiverem completos
91        if (topicScrollComplete && payloadScrollComplete) {
92          break;
93        }
94      }
95
96      // Mantém a última exibição antes de limpar
97      delay(3000);  // Pausa de 3 segundos para manter os últimos
            caracteres visíveis
98
99      // Limpa e desliga o display após o scroll
100     lcd.clear();
101     lcd.noBacklight();
102   }
103
104   // Função para processar a \textit{payload} JSON, extraindo o
        tópico MQTT e o conteúdo da payload
105   void processarJson(String jsonString) {
106
107     // Extrai o tópico manualmente
108     int topicStart = jsonString.indexOf("\"topic\":␣\"") + 10;
109     int topicEnd = jsonString.indexOf("\"", topicStart);
110     String topic = jsonString.substring(topicStart, topicEnd);
111
112     // Extrai o \textit{payload} manualmente
113     int payloadStart = jsonString.indexOf("\"payload\":␣{") +
            12;
114     int payloadEnd = jsonString.indexOf("}", payloadStart);
115     String payloadStr = jsonString.substring(payloadStart + 1,
            payloadEnd);
116
117     // Remove espaços e caracteres desnecessários do payload
118     payloadStr.replace("\"", "");
119     payloadStr.replace("␣", "");
120     payloadStr.replace(":", ":␣");
121
122     // Chama a função para exibir no display
123     displayScroll(topic, payloadStr);
124   }
125
126   // Função para converter a string binária associada ao \textit
        {payload} JSON numa string de texto
127   String binaryToString(String binaryInput) {
128
129     // Inicializa a string de texto final e determina o
            comprimento da string binária
130     String text = "";
131     int length = binaryInput.length();
132
```

```
133       // Processa a string binária em grupos de 8 bits
134       for (int i = 0; i < length; i += 8) {
135         String byteString = binaryInput.substring(i, i + 8);  //
              Divide a string em blocos de 8 bits
136         char c = 0;
137
138         // Converte cada grupo de 8 bits para um caractere
139         for (int j = 0; j < 8; j++) {
140           c = (c << 1) | (byteString[j] - '0');
141         }
142         text += c;  // Adiciona o caractere ao texto final
143       }
144       return text;  // Retorna o texto final
145     }
146
147     // Função para receber o sinal, considerando o SFD e EFD
148     String receiveSignal() {
149
150       // Inicializa variáveis para os bits recebidos e o estado do
            SFD
151       String receivedBits = "";
152       bool sfdDetected = false;
153
154       Serial.println("Waiting for transmission...");
155
156       // Loop de receção do sinal
157       while (1) {
158         int pinState = digitalRead(PIN_PHOTODIODE);
159
160         // Inverte a lógica, pois HIGH significa ausência de luz e
              LOW significa presença de luz
161         char bitValue = (pinState == LOW) ? '1' : '0';
162         receivedBits += bitValue;
163
164         // Verifica se o SFD foi detetado no início
165         if (!sfdDetected && receivedBits.endsWith(SFD)) {
166           sfdDetected = true;
167           receivedBits = "";  // Limpa os bits recebidos após o
                SFD
168         }
169
170         // Verifica se o EFD foi detetado no final
171         else if (sfdDetected && receivedBits.endsWith(EFD)) {
172
173           // Resolve o problema de leitura incompleta ao verificar
                um bit adicional
174           delayMicroseconds(BIT_PERIOD_US);
175           pinState = digitalRead(PIN_PHOTODIODE);
176           bitValue = (pinState == LOW) ? '1' : '0';
177           if (bitValue == '1') {
178             receivedBits += bitValue;
179           }
180
181           receivedBits = receivedBits.substring(0, receivedBits.
                length() - 8);  // Remove os 8 bits do EFD
182
183           Serial.println("EFD detected. Frame received.");
```

```
184           break;  // Sai do loop
185         }
186
187         // Reinicializa se muitos bits forem recebidos sem detetar
                o EFD
188         if (receivedBits.length() > 10000) {
189           receivedBits = "";
190           sfdDetected = false;
191           Serial.println("Transmission␣error:␣too␣many␣bits␣
                received.␣Resetting...");
192           break;  // Sai do loop
193         }
194
195         delayMicroseconds(BIT_PERIOD_US);
196       }
197
198       return receivedBits;  // Retorna os bits recebidos
199     }
200
201     // Função para decodificar uma string em Manchester
202     String manchesterDecode(String input) {
203
204       // Inicializa a string para armazenar os bits decodificados
205       String decoded = "";
206       if (input.length() % 2 != 0) {
207         Serial.println("Invalid␣Manchester␣encoding␣length");
208         return "";  // Retorna vazio se o comprimento for inválido
209       }
210
211       // Processa a string em pares de bits
212       for (int i = 0; i < input.length(); i += 2) {
213         String pair = input.substring(i, i + 2);
214         if (pair == "10") {
215           decoded += "1";
216         } else if (pair == "01") {
217           decoded += "0";
218         } else {
219           Serial.println("Invalid␣Manchester␣pair:␣" + pair);
220           return "";  // Retorna vazio se um par inválido for
                encontrado
221         }
222       }
223
224       return decoded;
225     }
226
227     // Função para calcular o CRC-16 utilizando o algoritmo CRC
          -16-CCITT (XMODEM)
228     uint16_t calcularCRC16(uint8_t* dados, size_t comprimento) {
229
230       // Inicializa o CRC e define o polinómio gerador
231       uint16_t crc = 0x0000;
232       uint16_t polinomio = 0x1021;
233
234       // Percorre todos os bytes dos dados
235       for (size_t i = 0; i < comprimento; i++) {
236         crc ^= (dados[i] << 8);  // Faz XOR entre o byte atual e
```

```
237
238     o CRC
239         for (uint8_t j = 0; j < 8; j++) {
240           if (crc & 0x8000) {
241             crc = (crc << 1) ^ polinomio;
242           } else {
243             crc <<= 1;
244           }
245         }
246       }
247
248       return crc;
249     }
250
251     // Função para processar os dados recebidos
252     void processData(String receivedBits) {
253
254       // Decodificação Manchester
255       String manchesterEncodedData = receivedBits;
256       String decodedBits = manchesterDecode(manchesterEncodedData)
              ;
257
258       // Verifica se a decodificação falhou
259       if (decodedBits == "") {
260         Serial.println("Manchester␣decoding␣failed.");
261
262         // Liga o LED vermelho durante 3 segundos para sinalizar
              falha de comunicação
263         digitalWrite(PIN_COMMUNICATION_FAILURE, HIGH);
264         delay(3000);
265         digitalWrite(PIN_COMMUNICATION_FAILURE, LOW);
266
267         return;
268       }
269
270       // Extração dos campos do quadro MAC a partir da string de
              bits decodificados
271       String frameControlBits = decodedBits.substring(0, 16);
272       String sequenceNumberBits = decodedBits.substring(16, 24);
273       String destinationAddressBits = decodedBits.substring(24,
              40);
274       String sourceAddressBits = decodedBits.substring(40, 56);
275       String formatUsedBits = decodedBits.substring(56, 58);
276       String numberOfPPDUBits = decodedBits.substring(58, 60);
277
278       // Os bits de \textit{payload} estão entre a posição 60 e o
              comprimento total - 16 (excluindo os bits de CRC)
279       String payloadBits = decodedBits.substring(60, decodedBits.
              length() - 16);
280
281       // Os últimos 16 bits são reservados para o CRC
282       String crcBits = decodedBits.substring(decodedBits.length()
              - 16);
283
284       // Reconstrução dos bits de dados (excluindo os bits de CRC)
285       String dataBits = decodedBits.substring(0, decodedBits.
              length() - 16);
```

```
286
287       // Converte a string de bits de dados para um array de bytes
              - necessário para calcular o CRC
288       uint8_t dataBytes[dataBits.length() + 1];
289       dataBits.getBytes(dataBytes, dataBits.length() + 1);
290
291       // Calcula o CRC a partir dos bytes dos dados utilizando a
              função de cálculo de CRC-16
292       uint16_t computedCRC = calcularCRC16(dataBytes, dataBits.
              length());
293
294       // Converte os bits de CRC recebidos (em string) para um
              valor de 16 bits (uint16_t)
295       uint16_t receivedCRC = 0;
296       for (int i = 0; i < 16; i++) {
297         if (crcBits[i] == '1') {
298           receivedCRC |= (1 << (15 - i));
299         }
300       }
301
302       // Verificação do CRC: compara o CRC calculado com o CRC
              recebido
303       if (computedCRC != receivedCRC) {
304         Serial.println("CRC verification failed.");
305         Serial.print("Computed CRC: 0x");
306         Serial.println(computedCRC, HEX);
307         Serial.print("Received CRC: 0x");
308         Serial.println(receivedCRC, HEX);
309
310         // Liga o LED vermelho durante 3 segundos para sinalizar
                uma falha de comunicação
311         digitalWrite(PIN_COMMUNICATION_FAILURE, HIGH);
312         delay(3000);
313         digitalWrite(PIN_COMMUNICATION_FAILURE, LOW);
314
315         return;
316       }
317
318       // Converte a \textit{payload} JSON de binário para texto
319       String \textit{payload} = binaryToString(payloadBits);
320
321       // Apresenta a \textit{payload} JSON
322       Serial.println("Received \textit{payload} in JSON format:");
323       Serial.println(payload);
324
325       processarJson(payload);
326     }
327
328     void setup() {
329       Serial.begin(115200);
330
331       // Define o pino de receção (fotodíodo) como entrada
332       pinMode(PIN_PHOTODIODE, INPUT);
333
334       // Define o pino do LED de falha de comunicação como saída
335       pinMode(PIN_COMMUNICATION_FAILURE, OUTPUT);
336
```

```
337       // Inicializa o LCD
338       lcd.init();
339       lcd.clear();
340       lcd.noBacklight();
341     }
342
343     void loop() {
344
345       // Chama a função para receber o sinal e armazena os bits
              recebidos
346       String receivedBits = receiveSignal();
347
348       // Se foram recebidos bits válidos, processa-os
349       if (receivedBits != "") {
350         processData(receivedBits);  // Processa os bits recebidos
                para decodificação e verificação
351       }
352     }
```