Relatório Final de Estágio - Atualização e Desenvolvimento da Plataforma IIoT SensLIVE

Mestrado em Engenharia Informática – Computação Móvel

Pedro Miguel Ribeirinho Fernandes

Leiria, setembro de 2024

Relatório Final de Estágio - Atualização e Desenvolvimento da Plataforma IIoT SensLIVE

Mestrado em Engenharia Informática – Computação Móvel

Pedro Miguel Ribeirinho Fernandes

Estágio realizado sob a orientação do Professor Doutor Leonel Filipe Simões Santos e sob supervisão de João Agostinho

Leiria, setembro de 2024

Originalidade e Direitos de Autor

O presente relatório de estágio é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Mestrado em Engenharia Informática – Computação Móvel, no ano letivo 2023/2024 da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos (se aplicável).

Agradecimentos

Um sincero agradecimento à CapTemp e ao Sr. Carlos Domingues pela oportunidade, pela colaboração e pelos recursos disponibilizados para a realização deste estágio.

Ao supervisor João Agostinho por toda a orientação e ajuda constante, contribuindo para o meu desenvolvimento profissional e pessoal durante esta experiência.

Ao Professor Doutor Leonel Santos pela disponibilidade, orientação e ajuda ao longo do estágio.

Aos meus pais e à minha irmã pelo apoio incansável e orientação ao longo do curso.

À Laura, pela ajuda em todos os momentos da minha vida académica e pessoal.

A todos os meus amigos que estiveram sempre presentes quando precisei.

Aos meus colegas de mestrado Tomás Mendes e Diogo Rodrigues pelo percurso feito, pela amizade e companheirismo criado.

Resumo

Este relatório descreve a experiência e o trabalho desenvolvido no estágio inserido no Mestrado em Engenharia Informática – Computação Móvel (MEI-CM) da Escola Superior de Tecnologia e Gestão (ESTG) do Politécnico de Leiria no ano letivo 2023/2024. O estágio decorreu na empresa CapTemp, localizada em Pombal, e teve uma duração de 1400 horas (setembro a junho). Durante este período foi desempenhada a função de *Full-Stack Developer* e foram desenvolvidos módulos para uma plataforma *Industrial Internet of Things* (IIoT) denominada de SensLIVE. O objetivo do estágio foi contribuir para a migração da plataforma para uma nova versão, corrigindo alguns erros ou bugs da versão anterior, incorporar novas tecnologias de *front-end* e *cross-platform* como Vue e Flutter, desenvolver novas funcionalidades e melhorar a sua interface gráfica.

Este estágio foi uma oportunidade para aplicar e solidificar os conhecimentos adquiridos ao longo do período acdémico e serviu como preparação para os desafios futuros na área de Engenharia Informática (EI), que está em constante evolução. Foi uma etapa importante para o enriquecimento pessoal e profissional e, todo o trabalho desenvolvido, contribuirá significativamente para o crescimento da CapTemp.

Palavras-chave: IIoT, monitorização, SensLIVE, CapTemp, Full-Stack

Abstract

This report describes the experience and work carried out during the internship as part of the MEI-CM at ESTG of the Polytechnic of Leiria in the scholar year 2023/2024. The internship took place in CapTemp, located in Pombal, and lasted 1400 hours (September to June). During this period, I worked as a Full-Stack Developer and developed modules for an IIOT platform called SensLIVE. The aim of the internship was to contribute to the migration of the platform to a new version, fixing some errors or bugs from the previous version, incorporate new front-end and cross-platform technologies such as Vue and Flutter, develop new features and improve its graphical interface.

This internship was an opportunity to apply and solidify the knowledge acquired throughout the academic period and served as preparation for future challenges in the field of Computer Engineering, which is constantly evolving. It was an important step for personal and professional enrichment and all the work carried out will contribute significantly to CapTemp's growth.

Keywords: IIoT, monitoring, SensLIVE, CapTemp, Full-Stack

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

AVAC	Aquecimento, Ventilação e Ar Condicionado
AWS	Amazon Web Services
BD	Base de Dados
CEO	Chief Executive Officer
CTO	Chief Technology Officer
DT	Departamento Tecnológico
EI	Engenharia Informática
ESTG	Escola Superior de Tecnologia e Gestão
GPS	Global Position System
HD	Hardware Developer
HTTPS	Hyper Text Transfer Protocol Secure
IA	Inteligência Artificial
IDE	Integrated Development Environment
IIoT	Industrial Internet of Things
IoT	Internet of Things
MEI-CM	Mestrado em Engenharia Informática – Computação Móvel
ML	Machine Learning
MTV	Model Template View
MVC	Model View Controller
NFC	Near Field Communication
OSI	Open Systems Interconnection
RA	Realidade Aumentada
RV	Realidade Virtual
SD	Senior Developer
SGBD	Sistema de Gestão de Base de Dados
SO	Sistema Operativo
TI	Tecnologias de Informação
UPS	Uninterruptible Power Supply

1. Introdução

Com a contínua evolução tecnológica, o mundo que conhecemos torna-se progressivamente mais interligado digitalmente mudando a nossa vida quotidiana e melhorando a comunicação, o acesso à informação, o trabalho, a produtividade, etc. Em 1999, surgiu o termo *Internet of Things* (IoT) e tem sido utilizado para descrever dispositivos informáticos (digitais e/ou mecânicos) interligados, capazes de transmitir dados através de uma rede específica, sem a necessidade de intervenção direta humana. Esta tecnologia junta diversas estratégias como a análise em tempo real, dispositivos sensoriais, *machine learning* (ML) (ramo de Inteligência Artificial - IA), entre outras. A criação de sistemas de segurança residencial e de iluminação inteligente, gestão de produções agrícolas e monitorização de dispositivos são alguns exemplos de aplicações IoT. Estes promovem a automação, a otimização e a conveniência e contribuem para uma experiência mais eficiente e alinhada com as necessidades da sociedade. [1]

Esta evolução tecnológica não se limita apenas ao domínio da conveniência pessoal e estende-se à Indústria, dando origem ao termo *Industrial Internet of Things* (IIoT). No contexto industrial, é explorado o seu potencial para automatizar processos que, consequentemente, permitem melhorar a eficiência operacional, diminuir o desperdício e reduzir o custo de mão de obra. Além disso, a monitorização em tempo real dos equipamentos é uma caraterística fundamental pois permite identificar e resolver problemas (antes de se tornarem críticos), otimizar o desempenho dos equipamentos, prever manutenções necessárias (manutenção preditiva), etc. Esta análise facilita a toma de decisões e contribui para um processo mais eficiente e eficaz. Assim, a combinação da automação e da monitorização transformam significativamente as operações industriais. [2]

1.1. Enquadramento

Este estágio foi realizado no ano letivo 2023/2024, com a duração total de 1400 horas. Teve início no dia 25/09/2023 e foi concluído no dia 18/06/2024.

A empresa que acolheu o estágio denomina-se de CapTemp e a sua sede está localizada em Pombal, Leiria, Portugal. É uma empresa inovadora que se destaca nas áreas de EI e IIoT. Especializa-se no desenvolvimento de sistemas de monitorização, supervisão e de

controlo remoto. Esta empresa produz e fornece sensores e dispositivos IIoT que, em conjunto com a sua plataforma SensLIVE, oferece aos utilizadores a possibilidade de os gerir e monitorizar remotamente de forma eficaz.

A sua missão é fornecer soluções inovadoras mantendo um compromisso contínuo de qualidade com o cliente. Toda a equipa procura atender às necessidades dos clientes, cumprir requisitos regulamentares, promover parcerias produtivas e otimizar constantemente os processos para oferecer serviços de monitorização e supervisão personalizados e de alta qualidade.

O estágio teve como principal objetivo o desenvolvimento de novas funcionalidades para a plataforma SensLIVE, solicitadas pela empresa e pelos clientes. Além disso, foi dedicado tempo à redação do presente relatório.

1.2. Contexto Específico

No contexto da missão da empresa, a estratégia do Departamento Tecnológico (DT) é importante. Este desempenha um papel fundamental no desenvolvimento, implementação e otimização dos sistemas de monitorização, supervisão e controlo remoto. Sendo uma empresa de desenvolvimento de sistemas IIoT à medida, é reconhecida a necessidade de investimento para criar soluções tecnologicamente avançadas e personalizadas.

Durante o período de estágio, foi proporcionada a oportunidade de participar ativamente como *Full-Stack Developer* no projeto SensLIVE que desempenha um papel relevante nas áreas de EI e IIoT. A relevância desta área para o curso de MEI-CM torna-se evidente na medida em que proporciona aos estagiários uma oportunidade única de aplicar os conhecimentos teóricos adquiridos ao longo do curso em cenários práticos do "mundo real".

1.3. Problema e Objetivos

O projeto SensLIVE passa por uma fase de migração para uma versão mais recente. A decisão de modificar esta plataforma surge da constante evolução da tecnologia e da necessidade de manter o projeto alinhado com as necessidades dos clientes e com as exigências do em mercado em termos de estética, escalabilidade, desempenho e segurança.

O objetivo deste estágio é contribuir para a migração da plataforma SensLIVE para uma versão mais atualizada de modo a corrigir alguns erros ou bugs da versão anterior, incorporar

tecnologias mais recentes, desenvolver novas funcionalidades e melhorar a interface gráfica de modo a tornar a sua utilização mais moderna, simples e apelativa possível.

O estágio curricular é uma forma do estudante ser inserido num contexto profissional e colocar em prática os conhecimentos adquiridos ao longo do período escolar. Deste modo, o estudante pode adquirir experiência e vivenciar os desafios do "mundo real".

1.4. Abordagem

O estágio foi realizado de forma presencial e em tempo integral na empresa. As tarefas foram definidas individualmente pelo supervisor, alinhadas com as necessidades da plataforma e com os requisitos funcionais e não funcionais previamente definidos pelo DT. Não havia prazos de entrega específicos e, à medida que as tarefas eram concluídas, novas tarefas eram atribuídas, criando um ciclo de desenvolvimento.

Foram utilizadas várias ferramentas de *software* para facilitar o desenvolvimento. Foi utilizado Git como controlo de versões, Visual Studio Code como *Integrated Development Environment* (IDE), para correr o servidor *web* foi utilizado o Laragon, para executar o código Vue foi utilizado Node.js e para fazer troca de excertos de código foi utilizado o servidor Slack da empresa.

Não foi adotada nenhuma "metodologia formal" durante o estágio nem foram realizados testes "formais" ao longo do desenvolvimento. O trabalho foi realizado maioritariamente de forma individual, promovendo autonomia e responsabilidade individual. Em alguns casos foi utilizada a técnica de *Pair Programming* que fomentou o trabalho de equipa e a troca de conhecimento. O ambiente de trabalho era pequeno e com poucos elementos, o que facilitava a comunicação entre os trabalhadores e, por isso, não houve necessidade de reuniões formais.

1.5. Estrutura do documento

Este relatório visa a materializar e proporcionar uma visão abrangente das atividades desenvolvidas e dos desafios enfrentados ao longo do estágio, destacando a contribuição que teve para o crescimento profissional e pessoal. É pretendido também dar a conhecer o método de trabalho realizado, descrever as tecnologias utilizadas e oferecer uma visão geral da CapTemp.

O presente documento encontra-se dividido em 7 capítulos.

No capítulo 2, é feita uma apresentação da CapTemp destacando alguns pontos como o setor em que se insere, a sua hierarquia interna e o modo de trabalho.

No capítulo 3 é realizada uma análise do estado da arte de IIoT, de algumas aplicações semelhantes ao SensLIVE e descritas algumas tecnologias de *back-end, front-end, cross-platform* e sistemas de gestão bases de dados (SGBD).

No capítulo 4 é descrita a plataforma SensLIVE realçando a sua arquitetura e tecnologias utilizadas.

No capítulo 5 é apresentado o trabalho realizado ao longo do estágio detalhando todas as funcionalidades desenvolvidas.

No capítulo 6 é feita uma análise crítica de todo o estágio, descrevendo os pontos mais desafiantes, e são indicadas algumas propostas de melhorias do modo de trabalho da empresa.

Por fim, no capítulo 7, é realizado um balanço do trabalho desenvolvido ao longo do estágio e apresentadas as notas conclusivas.

2. CapTemp

A CapTemp, cujo nome deriva das iniciais das palavras “Captura” e “Temperatura”, é uma empresa portuguesa que se especializa no desenvolvimento de sistemas à medida de monitorização, supervisão e de controlo. É responsável pelo fornecimento de sensores, coletores de dados e *software* que permite gerir dispositivos e analisar os dados que chegam dos sensores. São sistemas “à medida” pois são planeados e elaborados consoante as necessidades específicas do cliente. Este conjunto de soluções, proporciona uma gestão eficaz e em tempo real das operações, contribuindo para um ambiente mais eficiente e seguro.

Esta empresa assume uma posição proeminente no setor IIoT. Através da junção de Tecnologias de Informação (TI) e das operações industriais, a CapTemp fornece serviços de monitorização que possibilita às empresas melhorar a sua eficiência e produtividade e ajudar na toma de decisões. Sistemas de aquecimento, ventilação e ar condicionado (AVAC), sensores ambientais, sistemas de incêndio, detetor de fugas de água são algumas das soluções que disponibiliza aos seus clientes.

No primeiro subcapítulo são mencionadas as áreas e indústrias envolvidas e no segundo a hierarquia interna da empresa. No último capítulo é descrito o modo de trabalho feito com o supervisor.

2.1. Áreas e indústrias englobadas

2.1.1. Indústria Alimentar e Retalho

Nos **restaurantes**, o serviço fornecido envolve a monitorização de vários equipamentos como frigoríficos, arcas congeladoras e armários, com o objetivo de manter as temperaturas corretas de armazenamento e garantir a qualidade e a segurança alimentar.

Na **produção alimentar**, a monitorização é mais detalhada, abrangendo desde a produção até ao armazenamento, preservando a qualidade e a frescura dos alimentos.

No **catering**, o foco é no controlo rigoroso da temperatura e humidade, para garantir um ambiente seguro na preparação dos alimentos e manter os padrões de segurança e qualidade em todas as etapas.

Já nas **gelatarias**, há uma monitorização constante das arcas de congelação, com alertas que disparam sempre que há alterações de temperatura, permitindo uma ação rápida para minimizar as perdas de produtos.

2.1.2. Saúde

Nos **hospitais**, o serviço prestado consiste na monitorização rigorosa das temperaturas e humidade, tanto em espaços públicos como em áreas de armazenamento. O objetivo é disparar alertas instantâneos quando alguma condição não corresponder aos parâmetros definidos, permitindo uma resposta rápida e eficaz para manter o funcionamento adequado dos equipamentos, a qualidade do ar em conformidade com os padrões exigidos e as temperaturas corretas de armazenamento dos produtos.

Nos **laboratórios**, caso existam condições adversas, são disparados alertas instantâneos, que permitem uma resposta imediata para preservar a integridade dos materiais sensíveis mantendo os padrões exigidos pelos protocolos laboratoriais.

Nas **farmácias**, o serviço envolve a monitorização das temperaturas e humidade de armazenamento. Assim, são asseguradas as condições ideais dos produtos farmacêuticos de modo a preservar a sua eficácia e segurança.

2.1.3. Transporte e Logística

No **armazenamento**, é realizada a monitorização contínua da temperatura e da humidade dos armazéns para proporcionar um ambiente controlado que assegura a qualidade, integridade e segurança dos produtos.

No **serviço de transportes**, existe um sistema de controlo de temperatura e humidade das cargas em tempo real que assegura a integridade dos produtos durante o transporte.

2.1.4. Agricultura e produção animal

Existem soluções avançadas para **estufas** (sistemas de rega adaptados) que contribuem para o sucesso da produção, maximizando o rendimento e a eficiência agrícola. Estas soluções garantem produtos mais saudáveis e de melhor qualidade minimizando a taxa de erro humano.

Na **criação animal**, é feita a monitorização e controlo de pesos, alimentação e algumas variáveis ambientais. Esta monitorização tem o objetivo de garantir as melhores condições

de crescimento e adaptação para cada tipo de animal. É também assegurado um ambiente propício para o bem-estar animal, maximizado o seu desempenho e crescimento

2.1.5. Tecnologias de Informação

Em **Data-Centers**, é feita a monitorização e controlo preciso dos parâmetros de temperatura e humidade em níveis baixos, de modo a evitar o superaquecimento e manter as condições ideais para o funcionamento eficiente dos equipamentos.

Para as **IT-Rooms**, existem soluções abrangentes de monitorização e controlo preciso de energia, temperatura e humidade. Estas soluções permitem o acoplamento de diversos tipos de sensores, como detetores de inundações, assim como a supervisão contínua das *Uninterruptible Power Supply* (UPS), garantindo a proteção e a operação estável dos sistemas críticos.

2.1.6. Gestão de Recursos

Na gestão de recursos, a monitorização da **água** serve para identificar padrões de consumo e detetar potenciais desperdícios de modo a promover a eficiência hídrica e a implementar práticas sustentáveis para a conservação deste recurso essencial.

No caso da **eletricidade**, é feita a gestão e controlo da energia consumida pelos equipamentos (refrigeração, compressores, linhas de produção, sistemas de ar condicionado, etc.) com o objetivo de fornecer dados em tempo real sobre o consumo energético, possibilitando uma gestão eficiente do uso da eletricidade, reduzindo custos e melhorando a sustentabilidade ambiental.

Em relação ao **gás**, é feito o controlo preciso especialmente em dispositivos como caldeiras a vapor e caldeiras de termo fluidos. Desta forma, são identificados padrões de uso e detetadas variações/fugas permitindo otimizar a eficiência energética desses equipamentos e contribuir para sustentabilidade ambiental.

2.2. Hierarquia Interna

De modo a responder a todas as necessidades da empresa existe uma estrutura organizacional que é composta por uma hierarquia eficiente. Esta estrutura está desenhada para promover a colaboração, permitindo que cada membro contribua para alcançar os objetivos da empresa. Sendo um espaço de trabalho pequeno com poucos elementos, o

ambiente é caracterizado por ter uma comunicação constante entre trabalhadores que promove a eficiência operacional.

Na liderança da empresa (Administração) está o *Chief Executive Officer* (CEO) e fundador Carlos Domingues, responsável pela toma de decisões estratégicas direcionando a empresa a cumprir os seus objetivos e assegurando a ligação entre as diferentes hierarquias. Este estabelece uma comunicação direta com os clientes permitindo a criação de uma relação forte e duradoura entre cliente e empresa.

De seguida temos o Departamento de Vendas que promove os produtos e os serviços ao público de modo a conseguir aumentar o número de vendas e o alcance da empresa.

O DT, inclui o *Chief Technology Officer* (CTO), o *Senior Developer* (SD) e supervisor João Agostinho e o *Hardware Developer* (HD). O CTO é responsável por liderar a estratégia tecnológica garantindo o desenvolvimento e implementação de soluções tecnológicas avançadas e que estas estejam de acordo com os objetivos do negócio. O SD que, dada a sua vasta experiência e habilidades técnicas, contribui significativamente para o desenvolvimento de *software* da empresa. É também responsável por orientar os estagiários de *software*, dando-lhes tarefas e indicações de como as concluir. Tanto o CTO como o SD têm contacto direto com os clientes fornecendo suporte e formação. O HD e os Assistentes de Manutenção (Departamento Operacional) são responsáveis pela construção física dos produtos e pela implementação de *hardware*, no entanto, os assistentes destacam-se na realização de assistências, manutenção e montagem dos dispositivos e sensores nas empresas clientes da CapTemp. Tanto o HD e os assistentes estão responsáveis pela orientação dos estagiários de *hardware*.

Na Figura 1 está representada a estrutura organizacional da CapTemp.

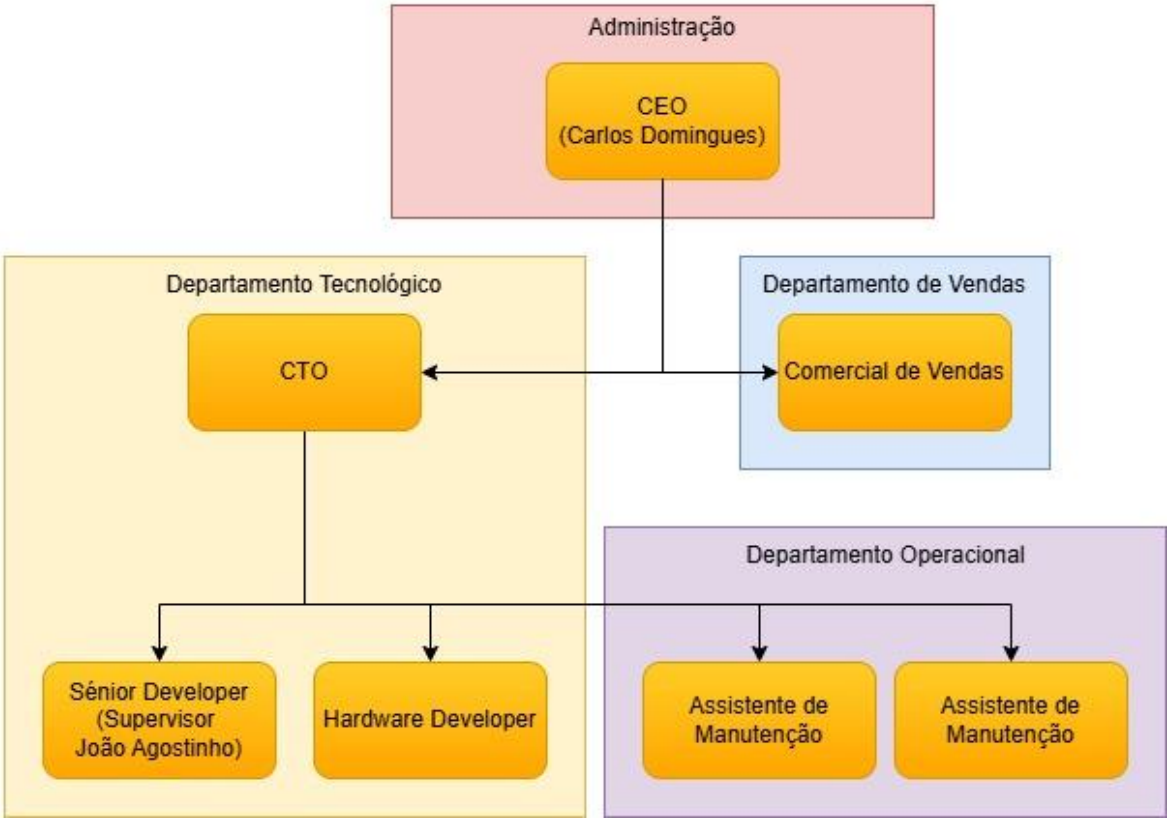

Figura 1 - Organograma da CapTemp

Este estágio está integrado no DT, com destaque no desenvolvimento de *software* para a plataforma SensLIVE. Este departamento, para além do SensLIVE, abrange outros projetos relacionados a EI e às TI.

2.3. Modo de trabalho (Supervisor)

Durante o estágio, foi estabelecida uma dinâmica de trabalho colaborativa com o supervisor que teve influência no desenvolvimento e na conclusão das tarefas atribuídas. O supervisor definiu tarefas individuais alinhadas com as necessidades do projeto. Não existia nenhum prazo de entrega logo, cada estagiário tinha autonomia de gerir o seu próprio ritmo de trabalho. À medida que as tarefas eram concluídas, novas tarefas eram adicionadas tornando-se num ciclo até terminar o projeto. Durante este período, o supervisor mostrou-se sempre disponível para prestar apoio, orientação e esclarecimentos caso existissem dúvidas.

Para o projeto SensLIVE, foi criado um repositório no *git* da empresa e, para cada estagiário, uma *branch* e uma cópia da base de dados (BD). Deste modo, cada estagiário possuía a sua própria versão do projeto onde podia implementar as novas funcionalidades. Por fim, o supervisor assumiu a responsabilidade do processo de *merge*, garantindo a

integração de cada contribuição individual numa única versão. Esta abordagem simplificou o fluxo de trabalho individual de cada estagiário, no entanto, dificultou o processo de *merge* (junção do código) pois foi realizado apenas pelo supervisor.

2.4. Síntese

A CapTemp é uma empresa inserida no setor de IIoT e é especializa no desenvolvimento de sistemas à medida de monitorização, encaminhamento e de controlo. Os seus serviços são utilizados em diversas áreas e indústrias como a indústria alimentar, saúde, agricultura, etc. Possui poucos trabalhadores e uma hierarquia desenhada para promover a colaboração e a comunicação. A dinâmica de trabalho com o supervisor é caracterizada por ser simples onde eram definidas novas tarefas à medida que o trabalho era terminado.

3. Estado da arte

Num contexto académico ou profissional, é comum encontrar relatórios ou artigos que fazem uma análise profunda ao "estado da arte". Este termo refere-se ao conjunto mais atualizado de conhecimentos, técnicas, métodos e/ou descobertas realizados numa área específica de estudo [3]. O objetivo é organizar, resumir e analisar essas contribuições feitas por investigadores e profissionais, de modo a criar uma base sólida teórica e compreender os avanços feitos até ao momento da pesquisa. Compreender o que já foi explorado possibilita identificar lacunas no tema, oferecendo aos pesquisadores a oportunidade de contribuir para o avanço do tema na área. Para além disso, a pesquisa das metodologias e abordagens utilizadas em pesquisas anteriores favorece a toma de decisões sobre as melhores práticas a serem adotadas no trabalho.

Neste capítulo é explorado o conceito de IIoT detalhando alguns tópicos como a sua arquitetura e protocolos de comunicação. De seguida, são descritas e comparadas algumas aplicações de monitorização e gestão de dispositivos IIoT e, no final, são apresentadas algumas tecnologias de *back-end, front-end, cross-platform* e SGBD.

3.1. IIoT

Nos últimos anos, a IoT ganhou bastante popularidade relacionada com o aumento de dispositivos potentes de baixo custo como *tag* RFID (Identificação por radiofrequência), sensores, etc. Esta tecnologia refere-se à conexão de objetos pela Internet, muitas vezes equipados com módulos de comunicação, sensores e atuadores. A integração de objetos do quotidiano com dispositivos inteligentes permite a transmissão de dados sem a necessidade de intervenção humana. Desta forma, é possível criar ambientes inteligentes em zonas urbanas e não urbanas. São inúmeras as aplicações e benefícios desta tecnologia como, por exemplo, um sistema de ar condicionado inteligente que varia automaticamente a temperatura consoante a temperatura interior de uma sala. [1]

Atualmente, o setor industrial é cada vez mais alimentado por informação. Grandes quantidades de dados vêm de toda a empresa em tempo real. Surge o termo de Indústria 4.0, que é caracterizado pela junção de várias tecnologias digitais inteligentes (IoT, IA, Big Data, robótica, etc.) no processo de fabrico, permitindo às empresas recolher, analisar, prever, compreender e comunicar esses dados. A aplicação da IoT no setor industrial denomina-se

de IIoT, onde a produção e conectividade está integrada com *cyber-physical systems* (CPS). Estes sistemas desempenham um papel fundamental na ligação entre o mundo virtual e o físico e, através de elementos computacionais, as entidades físicas podem ser controladas. Em IIoT, o foco é tornar as fábricas inteligentes permitindo automatizar processos, melhorar a eficiência operacional e diminuir o desperdício e o custo de mão de obra. [4] [5] [6]

3.1.1. Arquitetura

Várias arquiteturas para sistemas IIoT foram propostas. Sisinni et al. [7], introduzem uma arquitetura de 3 camadas (sensores, redes e serviços). Xu et al. [8], propuseram outra com as camadas física, comunicação e aplicação. Jayalaxmi et al. [9], defendem uma arquitetura de 4 camadas (perceção, rede, suporte e aplicação). Existe também uma arquitetura de 5 camadas (deteção, obtenção de acesso, rede, *middleware* e aplicação aprovada pela União Internacional de Telecomunicações. [10]

Neste subcapítulo será descrita uma arquitetura de 4 camadas pois, para além de ser aplicada em vários sistemas IIoT, inclui componentes essenciais de uma arquitetura de 3 camadas e explica, com uma granularidade mais fina, uma arquitetura de 5 camadas. Na Figura 2 é possível visualizar um esquema da arquitetura IIoT de 4 camadas. [10]

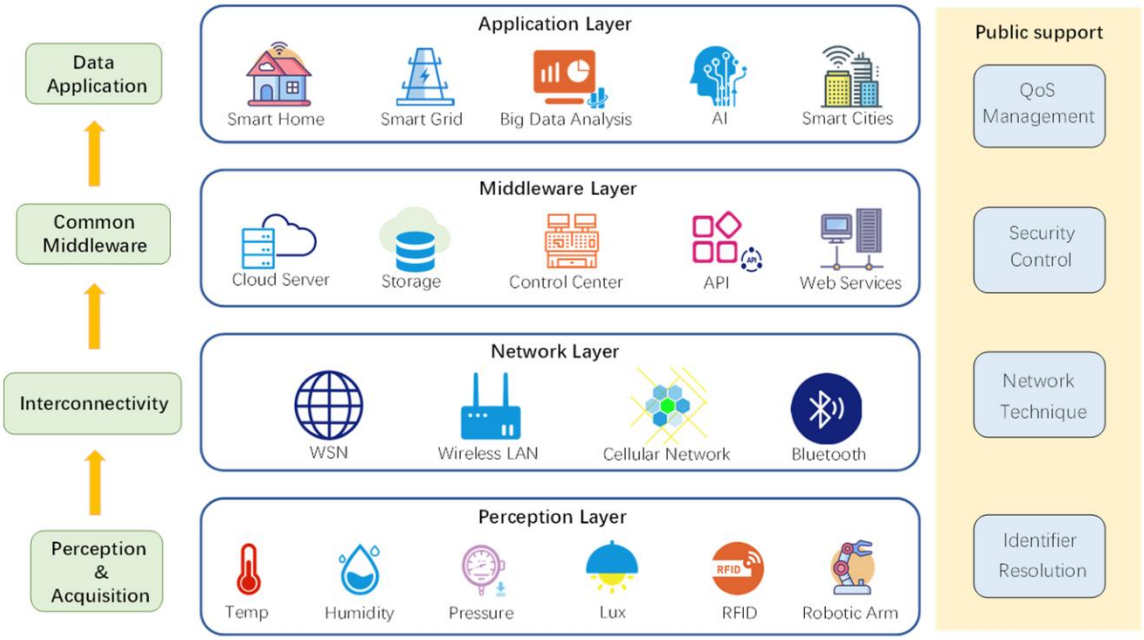

Figura 2 - Arquitetura IIoT de 4 camadas retirado de [10]

A. Perceção

A camada de perceção é considerada a camada mais baixa da arquitetura IIoT que liga o mudo físico com o virtual. Abrange uma variedade de sensores que recolhem e interpretam vários tipos de dados no contexto industrial. Depois de serem transformados em formato digital, os dados são enviados para a camada seguinte (rede). Através de RFID, os dispositivos podem ser reconhecidos, seguidos e monitorizados.

Esta camada utiliza várias tecnologias e dispositivos como *Global Position System* (GPS), etiquetas RFID, *Near Field Communication* (NFC), dispositivos que recolhem imagens, sensores e atuadores. A junção destas tecnologias com os dispositivos permite coletar informação e reconhecer equipamentos inteligentes num ambiente industrial.

B. Rede

Esta camada é responsável pela comunicação entre os dispositivos conectados e os sistemas de *back-end*, ou seja, entre os dispositivos de perceção (sensores, atuadores, etc.) e os sistemas de processamento e análise de dados. Esta ligação envolve protocolos de comunicação como como Ethernet, MQTT, Zigbee, entre outros, dependendo dos requisitos da aplicação e das condições do ambiente industrial. Além disso, esta camada lida com questões de segurança de dados, gestão da largura de banda e encaminhamento de tráfego de modo a garantir uma comunicação eficaz e robusta.

C. Middleware

A terceira camada é também conhecida como a camada de suporte. Oferece bases de dados e serviços *cloud* para a camada de aplicação utilizar. Utiliza técnicas computacionais avançadas para avaliar, processar e armazenar dados. Para analisar e calcular automaticamente as informações recolhidas, podem ser utilizadas tecnologias como *cloud computing* ou análise de grandes volumes de dados (*big data*). É uma camada eficaz para alcançar a interoperabilidade (comunicação e interação) entre sistemas.

D. Aplicação

Esta última camada funciona como ponte entre os utilizadores e a aplicação. Acede aos dados da camada de *middleware* preservando a sua integridade, sigilo e autenticação. Os utilizadores podem aceder aos serviços inteligentes desta camada através de dispositivos

com ligação à Internet como smartphones, tablets, computadores e outros gadgets inteligentes.

3.1.2. Protocolos de comunicação

Os protocolos de comunicação desempenham um papel crucial em IIoT, permitindo conexão e comunicação entre diversos dispositivos, sistemas e aplicações. Por isso, é importante entender a estrutura da comunicação entre sistemas. O Modelo *Open Systems Interconnection* (OSI) é um modelo conceitual que organiza um sistema de comunicação em sete camadas, onde cada camada possui uma função especifica. Estas camadas são:

1. Camada Física – Responsável pela transmissão de dados através de equipamento físico como cabos e computadores;
2. Camada de enlace de dados - Facilita a transferência de dados entre dois dispositivos na mesma rede. Define o formato dos dados;
3. Camada de rede – Facilita a troca de dados entre redes diferentes utilizando endereçamento lógico e encaminhamento de pacotes;
4. Camada de Transporte – Garante a confiabilidade dos dados entre sistemas. Responsável pela gestão de erros da transmissão, controlo do fluxo e de erros.
5. Camada de sessão – Estabelece, mantém e encerra sessões de comunicação entre dois dispositivos;
6. Camada de apresentação – Responsável pela preparação, tradução, criptografia e compactação dos dados que possam ser utilizados na camada de aplicação;
7. Camada de aplicação – Interage diretamente com os dados do utilizador. Inclui interfaces de aplicações e de serviços como navegadores web, envio de emails, etc. [11]

Devido à heterogeneidade dos ambientes industriais e de modo a atender às diferentes necessidades e requisitos, pode ser necessário empregar uma variedade de protocolos como MQTT, Modbus TCP/IP, *Hyper Text Transfer Protocol Secure* (HTTPS), entre outros. De seguida são descritos alguns dos protocolos comuns aos serviços descritos no subcapítulo 3.2.

LoRaWAN

LoRaWAN é um protocolo de comunicação que pertence à camada 2 do modelo OSI [12]. A palavra “LoRaWAN” vem da junção de “LoRa” (l*ong range*) e “WAN” (*Wide Area*

Network). Permite a comunicação segura a longas distâncias e com baixo consumo de energia podendo cobrir cidades inteiras dependendo da topografia. Pode ser aplicada em várias aplicações IoT como sensores e pontos de recolha de dados remotos alimentados a bateria que comuniquem através de mensagens curtas. [13]

Modbus TCP/IP

Pertence às camadas 1,2,3,4 e 7 do modelo OSI [14] e é uma variante da família Modbus utilizado em redes TCP/IP. Permite a comunicação eficiente e fiável dos dispositivos através de uma rede Ethernet que é bastante utilizada nos dias de hoje. Este protocolo "herda" a simplicidade e robustez do Modbus original e a fiabilidade e interoperabilidade do modelo TCP/IP, desta maneira, os dados Modbus tradicionais são encapsulados e transportados através de infraestruturas de rede padrão. [15]

MQTT

Pertence à camada 7 do modelo OSI [16] e é um protocolo de mensagens utilizado em IoT que foi desenvolvido com o objetivo de enviar mensagens de publicação/subscrição. É considerado um protocolo leve e eficiente, ideal para conectar dispositivos remotos que requerem poucos recursos e com uma largura de banda baixa. É utilizado em diversos setores e indústrias como a indústria automóvel, telecomunicações, etc. [17]

HTTPS

Pertence à camada 7 do modelo OSI [18] e é um protocolo de transferência de hipertexto seguro (versão criptografada e segura do HTTP). É considerado o protocolo principal utilizado no envio de dados entre um navegador *web* e um servidor *web*. É importante na transferência de dados sensíveis como dados de início de sessão numa conta de um banco, serviço de email, entre outros. [19]

3.1.3. Monitorização

A monitorização desempenha um papel fundamental em IIoT. A integração de sensores (camada de perceção), coletores de dados (camada de rede) e tecnologias de análise de dados (camada de *middleware* e aplicação) oferecem uma visão em tempo real sobre o funcionamento de equipamentos e processos industriais. Esta monitorização pode ser contínua ou periódica dependendo do tipo de processos ou equipamentos e traz alguns benefícios como:

- **Aumento da Eficiência Operacional** – A recolha de dados em tempo real permite otimizar a produção, reduzir desperdícios e melhorar a alocação de recursos;
- **Redução de Tempo de Inatividade** – A deteção de falhas e a realização de manutenção preditiva ajudam a minimizar paragens nos processos não planeados, promovendo a continuidade das operações;
- **Melhoria de Segurança** – Monitorização de condições perigosas como o nível de gás presente no ar, podem contribuir para um ambiente de trabalho mais seguro;
- **Decisões Baseadas nos Dados** – A recolha em tempo real dos dados, possibilita a toma de decisões informadas por parte dos gestores;
- **Custos Reduzidos** – A otimização dos processos e a redução de falhas resultam na redução dos custos operacionais e de manutenção. [20]

3.1.4. Domínios

Os "domínios" referem-se aos diferentes setores industriais e áreas de aplicação onde as tecnologias IIoT podem ser implementadas. De seguida são explorados alguns dos principais domínios de IIoT. [21]

Saúde

Através de monitoração de sinais vitais, gestão de inventário, manutenção preditiva de equipamentos, entre outros exemplos, a IIoT pode melhorar a qualidade dos cuidados dos pacientes e da eficiência dos serviços de saúde.

Agricultura

Para aumentar a produtividade e sustentabilidade agrícola são utilizados sensores e sistemas de monitorização que recolhem informações que possibilitam gerir as produções agrícolas.

Construção e Infraestrutura

No setor de construção e infraestrutura é realizada a monitorização da integridade dos equipamentos e construções e feita a gestão dos projetos de modo a otimizar a utilização de recursos.

Transporte e Logística

Através da recolha de dados da localização dos veículos e cargas é possível analisar e fazer uma gestão mais eficiente das rotas. A monitorização das condições de transporte permitem fazer o controlo de cargas mais sensíveis.

3.2. Trabalho Relacionado

Neste subcapítulo serão mencionadas algumas aplicações de monitorização semelhantes ao SensLIVE.

3.2.1. SkySpark

SkySpark é um *software* de monitorização desenvolvido pela empresa SkyFoundry (especializada em soluções de *software* de IoT). Pode ser usado para armazenar, analisar e visualizar uma variedade de dados de várias indústrias/setores como a agricultura e a saúde e/ou locais comos edifícios inteligentes e instalações industriais. [22] [23]

A SkySpark obtém dados de diversas formas:

- Ligação em tempo real a um sistema de automação;
- Ligação a uma BD SQL;
- Importação de dados num ficheiro Excel.

Estes dados são armazenados numa BD Folio. Esta BD de alta performance permite armazenar uma grande quantidade de dados e criar modelos para edifícios, equipamentos, dispositivos e sensores. É utilizado também o Project Haystack (*open-source*) para criar "tags" nos dados. Com uma marcação adequada as aplicações podem melhorar a performance e encontrar padrões para identificar falhas e desvios, ajudando a garantir o correto funcionamento dos sistemas.

Para a visualização dos dados, a plataforma divide-se em algumas aplicações:

1. Spark App – Apresenta os dados sob a forma de linhas de tempo e gráficos que indicam o momento, duração, frequência e o custo dos problemas;
2. KPI App – Calcula os indicadores-chave e apresenta os resultados em gráficos de vela;
3. Energy App – Análise de recursos energéticos (consumo, custo, utilização de água e gás);

4. Historian App – Combinação de dados para criar visualizações específicas;
5. Weather App – Lê previsões locais e os dados meteorológicos para ajudar a avaliar o desempenho dos equipamentos;
6. Rule App – Cria regras personalizadas para identificar falhas, problemas, desvios e anomalias.

Na Figura 3 é possível visualizar um *dashboard* da plataforma SkySpark.

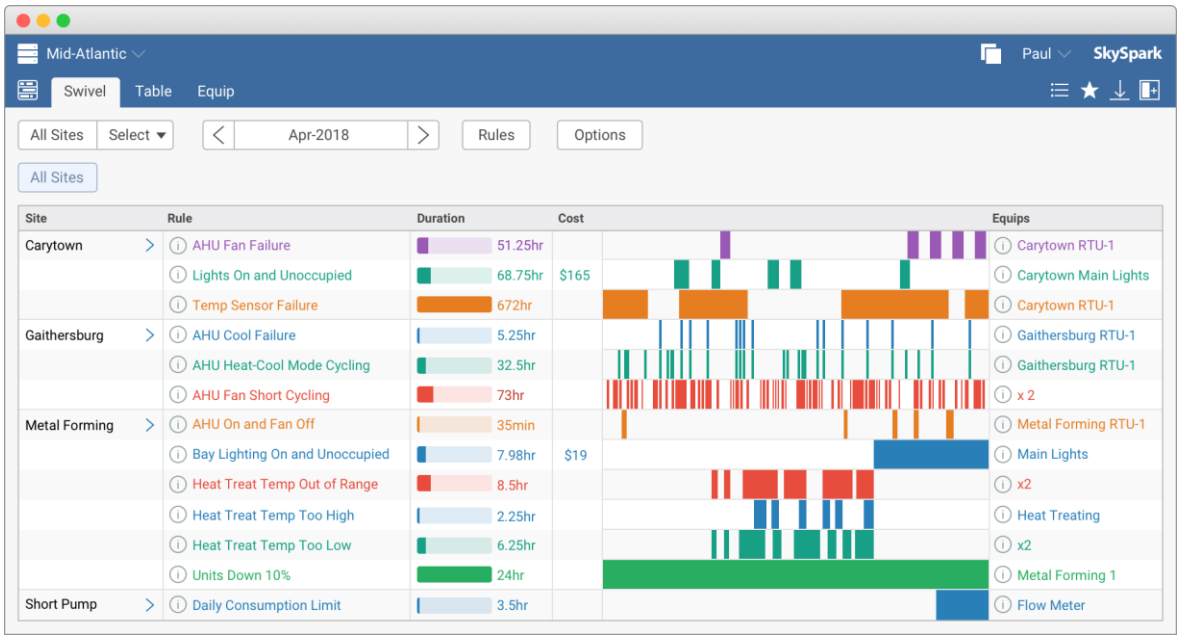

Figura 3 - Dashboard da plataforma SkySpark retirado de [22]

3.2.2. AWS IoT Device Management

A AWS IoT Device Management é um serviço fornecido pela Amazon Web Services (AWS) que ajuda na gestão de dispositivos IoT em grande escala. Neste serviço é feito o registo dos dispositivos que podem, mais tarde, ser organizados em grupos e hierarquias para simplificar a sua manutenção e gestão. Relativamente à monitorização, o estado dos dispositivos pode ser analisado em tempo real, incluindo informações sobre conectividade, integridade e desempenho e ajudar na gestão do ciclo de vida dos dispositivos. Para uma análise posterior, existe a possibilidade de criar relatórios e análises dos dispositivos ao longo do tempo. [24]

Com a integração de outros serviços como a AWS IoT Core é fornecida uma solução completa. Este serviço é responsável pela conexão, oferecendo uma infraestrutura necessária para a comunicação segura entre dispositivos IoT na nuvem. [25]

Ao utilizar AWS IoT Device Management, as organizações podem escalar facilmente as suas soluções IoT, mantendo um controlo eficaz sobre os seus dispositivos em diferentes cenários, como a indústria, saúde, transporte e outros setores que utilizem tecnologias IoT.

A Figura 4 mostra um *dashboard* da plataforma AWS IoT Device Management.

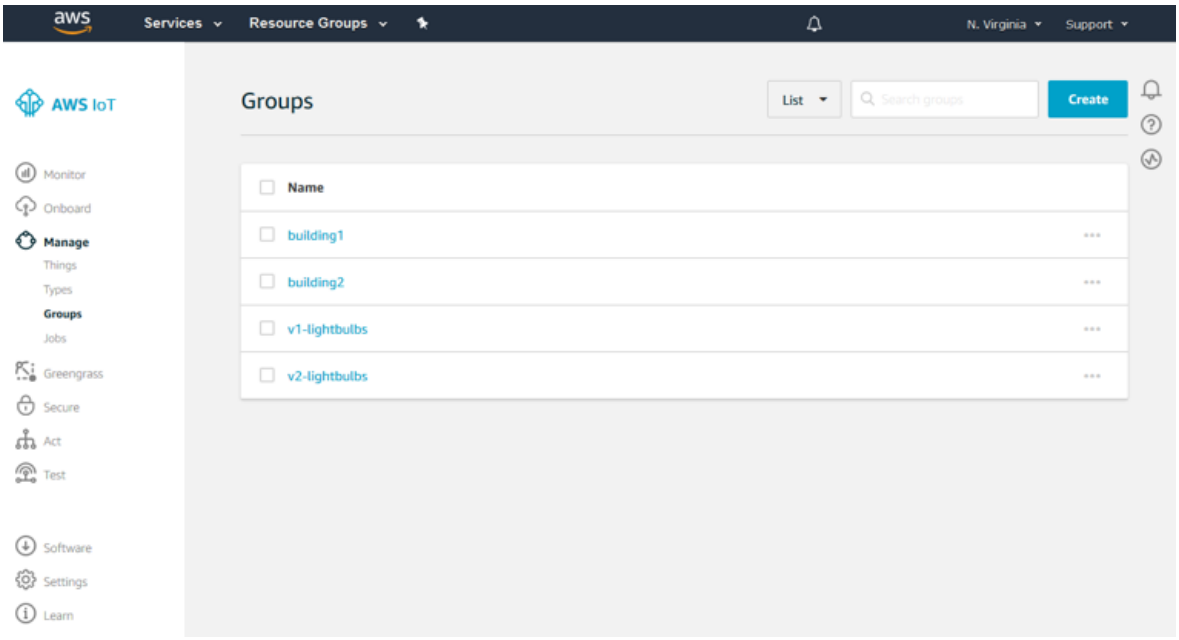

Figura 4 - Dashboard da plataforma AWS IoT Device Management retirado de [26]

3.2.3. Azure IoT Central

A Azure IoT Central é um serviço fornecido pela Microsoft que facilita a criação, implementação e gestão de soluções IoT. [27]

Principais características:

1. Registo, monitorização e gestão de dispositivos IoT permitindo verificar o seu estado, atualizações e segurança.
2. Modelos pré-configurados que simplificam o processo de criação e de configuração de soluções IoT.
3. Interface gráfica intuitiva e personalizável para interagir e monitorizar os dispositivos IoT.
4. Possibilita um grande número de dispositivos na mesma aplicação (200.000). [28]
5. Oferece a possibilidade de integrar com outros serviços Azure como Azure Stream Analytics, Azure Machine Learning e Azure IoT Hub.

Da mesma forma que a AWS IoT Core está para a AWS IoT Devices Management, a Azure IoT Hub possibilita uma conexão segura na Azure IoT Central. Esta permite configurar identidades e credenciais individuais para cada dispositivo de modo a manter a confidencialidade da comunicação bidirecional (dispositivo-*cloud* e *cloud*-dispositivo). [29]

Na Figura 5 é possível visualizar um *dashboard* da Azure IoT Central.

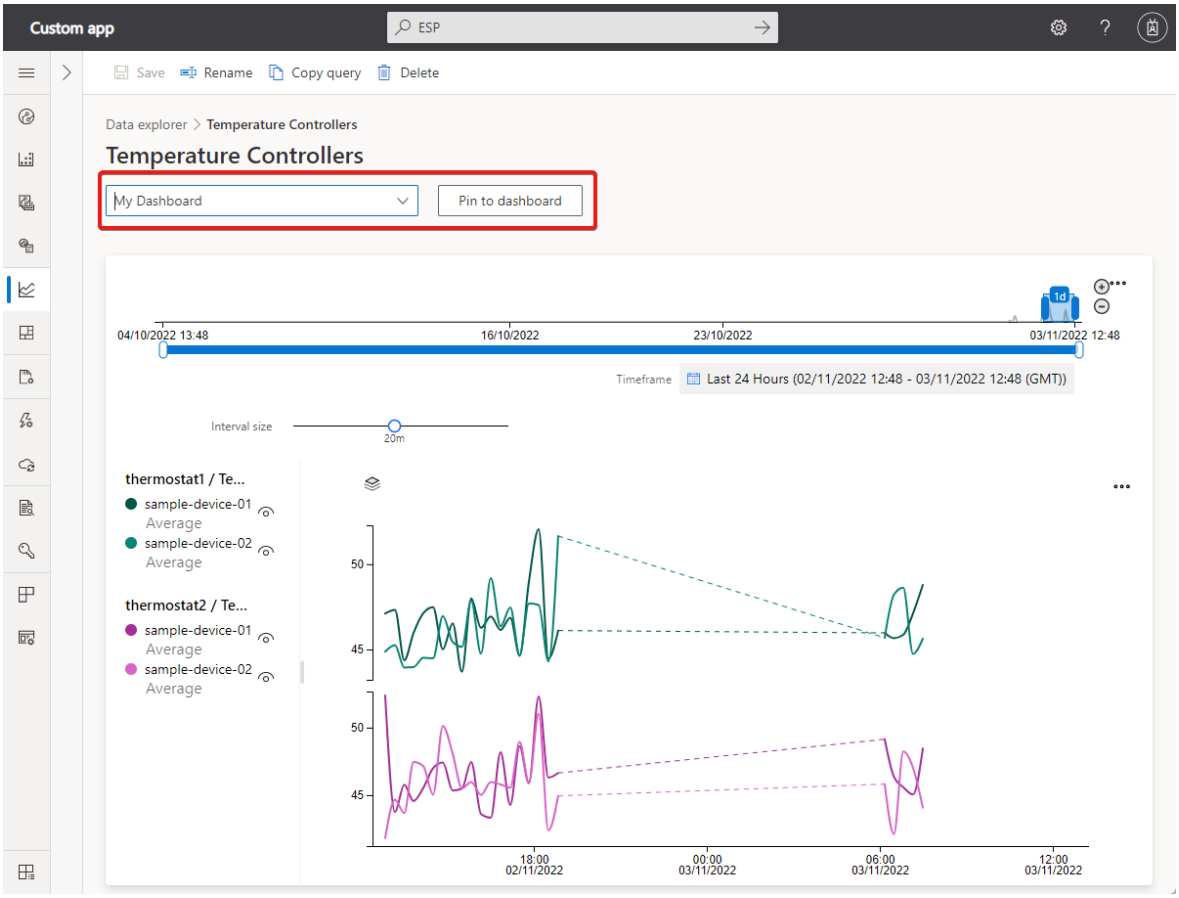

Figura 5 - Dashboard da plataforma Azure IoT Central retirado de [30]

3.2.4. SensLIVE (versão antiga)

O SensLIVE é uma plataforma web IIoT desenvolvida pela CapTemp cujo seu objetivo principal é a monitorização e gestão de sensores e dispositivos. Este sistema permite a gestão de vários dispositivos utilizando o protocolo PUSH XML (XML Encryption Transport Layer) que facilita a segurança da transmissão de documentos XML entre dois *endpoints*. [31] Além disso, é desenvolvido com tecnologias de encriptação de dados para garantir a segurança das informações. A plataforma permite o acesso aos dados a partir de qualquer dispositivo com acesso à internet, como computadores, tablets e smartphones.

Este *software* está certificado e validado de acordo com a norma EN12830:2018 e o GUIA WELMEC 7.2 – 2018. Estes são documentos normativos no contexto de controlo e medição de temperatura. [32]

No capítulo 4, são fornecidos detalhes mais aprofundados da nova versão desta plataforma.

3.2.5. Comparação dos serviços

Na Tabela 1 podemos ver as funcionalidades comuns e não comuns dos serviços descritos no presente subcapítulo (3.2) e da plataforma SensLIVE, assim como os protocolos de comunicação que utilizam.

Tabela 1 - Comparação de funcionalidades dos três serviços

Funcionalidades	SkySpark	AWS IoT Devices Management	Azure IoT Central	SensLIVE (versão antiga)
Registo de Dispositivos	X	X	X	X
Gestão de Dispositivos	X	X	X	X
Monitorização	X	X	X	X
Modelos Pré-Configurados	Não	Não	X	Não
Conectividade com Dispositivos	X	X	X	X
Comunicação Bidirecional	X	X	X	X
Comunicação Segura	X	X	X	X
Protocolos de Comunicação	BACnet IP, Modbus TCP, Obix, Haystack, SNMP, OPC UA e MQTT [33]	MQTT, HTTPS, WebSockets e LoRaWAN [34]	MQTT, HTTPS e AMQP [35]	LoRaWAN, MQTT, HTTP, NB-IoT, SMNP, PROFINET, BACnet, ICMP e Modbus
Integração com outros serviços	X	X	X	X
Interface Gráfica	X	X	X	X
Personalização	X	X	X	X

Como é possível verificar, todas as aplicações oferecem várias funcionalidades essenciais para a gestão e monitorização de dispositivos industriais. Todas suportam registo, gestão, monitorização e conectividade de dispositivos, comunicação bidirecional e segura,

bem como integração com outros serviços e interfaces gráficas intuitivas. Azure IoT Central destaca-se por oferecer modelos pré-configurados, facilitando a configuração inicial. Em termos de protocolos de comunicação, estes variam de plataforma para plataforma exceto MQTT que é comum a todas.

Comparando com a plataforma SensLIVE, os outros serviços apresentam funcionalidades bastante semelhantes. Como as funcionalidades essenciais e a capacidade de gestão de dispositivos IIoT são comuns a todos os serviços, a escolha da melhor solução pode depender de necessidades específicas como a preferência por determinados protocolos de comunicação ou a conveniência dos modelos pré-configurados oferecida pelo Azure IoT Central. Em suma, estas soluções permitem gerir e monitorizar dispositivos IIoT, cada uma com suas particularidades e vantagens distintas.

3.3. Tecnologias

Antes de desenvolver qualquer tipo de *software* é essencial realizar uma vasta pesquisa sobre as tecnologias a usar. Esta escolha deve ser feita com base em diversos fatores como os requisitos funcionais e não funcionais, experiência da equipa, custos, entre outros. As tecnologias escolhidas definem o funcionamento interno do *software* e podem ser cruciais para o sucesso do projeto.

Nos próximos subcapítulos, serão analisadas algumas tecnologias e *frameworks* de *back-end, front-end, cross-platform* e de SGBD.

3.3.1. Frameworks de Back-end

O *Back-end* é uma parte fundamental para qualquer *software*. É responsável pela parte lógica da aplicação, ou seja, o processamento dos dados, os pedidos à BD, gestão de dados, etc. De seguida serão descritas três das *frameworks* de *back-end* mais populares. [36]

Laravel

Laravel é uma *framework* PHP *open-source* (código fonte público) criada em 2011 para desenvolver aplicações *web* em escala. Possui algumas características como:

- Linha de comando Artisan - ajuda na migração de dados, gestão de BD, criar controladores, modelos, etc.;
- Autenticação nativa;

- Padrão arquitetural Model-View-Controller (MVC) - ajuda aos programadores a ter uma estrutura organizada e consistente do código. [37] [38]

Algumas das empresas que usam Laravel são 9GAG, UNICEF, Pfizer. [39]

Django

É uma *framework open-source* grátis desenvolvida para criar o *back-end* de aplicações *web* utilizando a linguagem de programação Python. Baseia-se na arquitetura *Model Template View* (MTV) e facilita o desenvolvimento de aplicações complexas e escaláveis. É uma escolha popular para programadores que procuram eficiência e rapidez no desenvolvimento de *back-end*, mantendo boas práticas de segurança. [40]

ASP.NET

ASP.NET é uma *framework open-source* criada pela Microsoft. Está desenhada para desenvolver aplicações *web* e serviços .NET utilizando C#, F# e Visual Basic. Acrescenta também algumas funcionalidades à plataforma .NET como:

- *Framework* base para processar pedidos *web* em C# ou F#;
- Templates para páginas *web*;
- Bibliotecas para padrões *web* comuns (MVC);
- Sistema de autenticação que inclui bibliotecas, BD, páginas *template*;
- Extensões de edição que ajudam a completar código, realçar a sua sintaxe através de cores, etc. [41]

Algumas das empresas que utilizam ASP.NET são SpaceX, GrubHub, Alibaba Travels, entre outros. [42]

Na Tabela 2 estão apresentadas algumas vantagens e desvantagens das *frameworks back.end* assim como alguns exemplos de aplicações que as utilizam.

Tabela 2 - Vantagens e desvantagens de algumas *frameworks* de *Back-end*

Nome	Vantagens	Desvantagens	Aplicações
Laravel [43]	- Código Simples; - Escalável; - Eficiente na migração de dados.	- Suporte limitado; - Performance baixa com grandes quantidades de dados.	- 9GAG; - UNICEF; - Pfizer.
Django [44] [45]	- Rápido; - Escalável; - Seguro.	- Não é bom para projetos simples; - É necessário ter conhecimento vasto da estrutura antes de começar a usá-lo.	- Instagram; - Spotify; - Youtube.
ASP.NET [46]	- Diminui a quantidade de código necessária para desenhar grandes aplicações; - Boa performance.	- Acesso limitado e controlo sobre o HTML; - Ciclos de vida complexos; - Pouco suporte para testes.	- SpaceX; - Roblox; - Alibaba.

A Tabela 2 compara três *frameworks* de desenvolvimento de *back-end*: Laravel, Django e ASP.NET. Laravel é conhecido pelo seu código simples, escalabilidade e eficiência na migração de dados, mas apresenta limitações de suporte e desempenho com grandes volumes de dados. Django oferece rapidez, segurança e escalabilidade, contudo, requer um conhecimento profundo da sua estrutura e não é ideal para projetos simples. Já o ASP.NET destaca-se por reduzir a quantidade de código necessária para grandes aplicações e ter bom desempenho, mas tem controlo limitado sobre o HTML e ciclos de vida complexos. Empresas como o Instagram, SpaceX e UNICEF utilizam estas tecnologias.

3.3.2. Frameworks de Front-end

O *Front-end* faz a ligação entre os utilizadores e a parte da lógica e do processamento de um produto de *software*. Refere-se a tudo aquilo o que o utilizador "vê" fornecendo uma experiência visual e interativa através de formulários, botões, imagens, etc. De seguida serão apresentadas as três *frameworks* de *front-end* mais populares. [47]

Vue.js, React e Angular JS são três *frameworks open-source* JavaScript para o desenvolvimento de interfaces em *Single-Page Applications* (SPA), ou seja, a aplicação carrega apenas uma página e o seu conteúdo é atualizado quando o utilizador interage com a mesma.

Vue.js

Vue.js possui boa documentação, sendo acessível independentemente da experiência do programador. A sua abordagem progressiva permite uma fácil integração com outras tecnologias, facilitando a expansão e a manutenção dos projetos. No entanto, como é uma *framework* relativamente "recente", Vue.js pode enfrentar desafios devido à falta de programadores experientes no mercado. A sua flexibilidade, embora seja uma vantagem em muitos casos, pode ser uma desvantagem em projetos maiores, onde uma estrutura precisa e consistente é essencial para a manutenção a longo prazo. [48] [49] [50]

React

React é reconhecido pela sua flexibilidade, permitindo criar interfaces de utilizador personalizadas e dinâmicas. Tem uma comunidade grande e ativa que oferece suporte, promovendo uma colaboração eficaz entre programadores. Através do uso de React Native, é possível criar aplicações móveis, aumentando assim as suas possibilidades de uso. Ainda assim, tem uma grande curva de aprendizagem que pode ser desafiadora para novos programadores. A documentação oficial, embora útil, pode ser considerada reduzida em comparação com outras *frameworks*, dificultando a implementação de soluções complexas. Além disso, a integração de React com outras tecnologias pode ser complexa, exigindo conhecimento das ferramentas e bibliotecas envolvidas. [51] [52]

Angular JS

Angular JS, assim como React, é valorizado pela sua comunidade grande. Possui uma arquitetura MVC que facilita a organização e a manutenção do código. Além disso, como é uma das *frameworks* mais antigas no mercado, tem uma base sólida de utilizadores e de recursos. No entanto, o suporte oficial foi encerrado, resultando em problemas de segurança e compatibilidade para projetos atuais. É uma *framework* complexa que pode exigir bastante tempo para ser dominada. Em termos de desempenho, o Angular tende a ter uma menor performance em comparação com outras *frameworks*. [53] [54] [55]

Na Tabela 3 é possível visualizar as vantagens e desvantagens de cada uma assim como alguns exemplos do mundo real.

Tabela 3 - Vantagens e desvantagens de algumas *frameworks* de *Front-end*

Nome	Vantagens	Desvantagens	Aplicações
Vue.js	- Boa documentação; - Acessível; - Progressiva (facilidade em integrar com outros projetos).	- *Framework* recente; - Falta de programadores experientes; - Muito flexível (desvantagem para projetos grandes).	- Trivago; - Glovo; - GitLab.
React	- Flexível; - Comunidade grande e ativa; - Pode ser usado para criar aplicações móveis (React Native).	- Grande curva de aprendizagem; - Documentação oficial reduzida; - Integração complexa com outras tecnologias.	- Facebook; - Airbnb; - Dropbox.
Angular JS	- Comunidade grande; - Possui arquitetura MVC; - Mais tempo de mercado.	- Suporte oficial acabado; - Complexo; - Menor performance comparando com outras *frameworks*;	- Microsoft (Microsoft Office); - Paypal; - Google (Gmail).

A Tabela 3 compara três *frameworks* de desenvolvimento de *front-end*: Vue.js, React e AngularJS. Vue.js é conhecido pela sua boa documentação, acessibilidade e facilidade de integração, mas é recente, tem falta de programadores experientes e pode ser demasiado flexível para grandes projetos. React destaca-se pela sua flexibilidade, grande comunidade e capacidade de criar aplicações móveis (React Native), mas tem uma curva de aprendizagem elevada e integração complexa com outras tecnologias. Já AngularJS tem uma grande comunidade e tem mais tempo no mercado, mas o suporte oficial terminou e apresenta menor desempenho comparado com outras *frameworks*. Empresas como Trivago, Facebook e Microsoft utilizam estas tecnologias.

3.3.3. Frameworks Cross-Platform

Frameworks Cross-Platform ou multiplataforma são ferramentas de desenvolvimento de *software* que, a partir de um único código base, permitem criar aplicações para uma variedade de SO, ou seja, o mesmo código é executado em vários sistemas como Android, iOS, MacOS, Linux, Windows e até mesmo na Web. De seguida serão apresentados as *frameworks cross-platform* mais utilizadas. [56] [57]

Flutter

É uma *framework open-source* criada pela Google em 2017. Suporta o desenvolvimento de aplicações para 6 plataformas: Web, Windows, Linux, MacOS, iOS e Android. Utiliza Dart como linguagem para o seu desenvolvimento de *User Interface* (UI) e oferece uma vasta coleção de *widgets* (componentes) de modo a tornar a experiência do utilizador consistente. [58]

React Native

É uma *framework open-source* baseada em JavaScript criada pelo Facebook em 2015. É desenhada para desenvolver aplicações de alta performance para iOS, Android, macOS, Windows, tvOS, Web e Skia. Assim como Flutter, fornece componentes nativos que são mapeados diretamente para os blocos de construção de UI. [59] [60]

Ionic

É também uma framework open-sourse baseada em Javacript e foi criada em 2013. Permite e integração de Angular, React e Vue e, à semelhança das outras tecnologias, também possui componentes que permitem a construção rápida de UI. [61]

Na Tabela 4 é possível visualizar as vantagens e desvantagens de cada uma assim como alguns exemplos do mundo real.

Tabela 4 - Vantagens e desvantagens de algumas *frameworks Cross-Platform*

Nome	Vantagens	Desvantagens	Aplicações
Flutter [62]	- Alta performance; - Curva de aprendizagem baixa.	- Ficheiros da aplicação de grandes dimensões; - Falta de *third-party libraries*.	- Nubank; - eBay; - Google Pay.
React Native [63]	- Alta performance; - Processo de desenvolvimento móvel simples.	- Pouca maturidade por ser uma tecnologia recente; - Complexidade no *debug*.	- Instagram; - Airbnb; - Uber.
Ionic [64]	- Flexível; - Muitos plugins que permitem a utilização do smartphone (câmara, leituras biométricas, etc.).	- Baixa Performance; - Plugins nativos não são estáveis e podem entrar em conflito uns com os outros.	- JustWatch; - IMB; - T-Mobile.

A Tabela 4 compara três *frameworks cross-platform*: Flutter, React Native e Ionic. Flutter tem alta performance e uma curva de aprendizagem baixa, mas as aplicações têm ficheiros grandes e falta de bibliotecas de terceiros. React Native também proporciona alta performance e um processo de desenvolvimento móvel simples, mas é uma tecnologia recente, com pouca maturidade. Ionic destaca-se pela sua flexibilidade e pela quantidade de plugins que permitem o uso de funcionalidades do smartphone, mas tem baixa performance, e os plugins nativos podem ser instáveis e causar conflitos. Empresas como Nubank, Instagram e JustWatch utilizam estas tecnologias.

3.3.4. Sistema de Gestão de Base de Dados

Os SGBD são essenciais num produto de *software* pois permitem gerir o armazenamento, manipulação e pesquisa de dados dentro de uma BD. De seguida serão apresentados alguns exemplos.

Oracle Database e MySQL

São *open-source* e foram ambas desenvolvidas pela empresa Oracle. Segundo a DB-Engines [65], a Oracle Database encontra-se no primeiro lugar a MySQL no segundo lugar

de SGBD mais utilizados. São ambas SGBD relacionais (armazenam e fornecem acesso a pontos de dados relacionados entre si [66]).

PostgreSQL

É uma SGBD relacional *open-source* e foi desenvolvida pela PostgreSQL Global Development Group. Encontra-se no quarto lugar de SGBD mais utilizadas. [67]

A Tabela 5 mostra algumas vantagens e desvantagens da SGBD mencionadas assim como alguns exemplos de aplicações que as utilizam.

Tabela 5 - Vantagens e desvantagens de alguns exemplos de SGBD

Nome	Vantagens	Desvantagens	Aplicações
Oracle Database [68] [69]	- Confiabilidade e segurança; - Escalabilidade e performance.	- Gestão complexa e difícil.	- FedEx; - DropBox; - Mazda.
MySQL [70]	- Focado em aplicações *web*; - Grande comunidade.	- Não estão tão "madura" comparado com outros SGBD. - Na prática não é 100% *open-source*, tem módulos proprietários e de código fechado.	- Facebook; - Netflix; - Uber.
PostgreSQL [71] [72]	- Suporta objetos geográficos (armazenamento de dados geoespaciais); - Fácil de usar; - Necessita de pouca manutenção e administração.	- Alterações para melhorar a velocidade querem mais trabalho; - Mais lento que outras tecnologias.	- Apple; - IMDB; - Reddit.

A Tabela 5 compara três SGBD: Oracle Database, MySQL e PostgreSQL. A Oracle Database é conhecida pela sua confiabilidade, segurança, escalabilidade e performance, mas a sua gestão é complexa. MySQL destaca-se por se focar em aplicações web e por possuir

uma grande comunidade, mas não apresenta tanta "maturidade" como os outros SGBD. PostgreSQL oferece suporte a objetos geográficos, é fácil de usar e requer pouca manutenção, porém, as melhorias de velocidade exigem mais trabalho e é mais lento que outras tecnologias. Empresas como FedEx, Facebook e Apple utilizam estes SGBD.

3.4. Síntese

IIoT refere-se à conexão de objetos pela Internet (na área industrial), muitas vezes equipados com módulos de comunicação, sensores e atuadores. Existem várias propostas de arquiteturas para a IIoT, no entanto, a arquitetura de 4 camadas (perceção, rede, *middleware* e aplicação) destaca-se, pois, é utilizada em vários sistemas e possui as características das restantes propostas. A monitorização realizada através de aplicações IIoT permitem aumentar a eficiência e reduzir custos e englobam várias áreas como saúde, agricultura, etc.

Existem diversas aplicações IIoT que permitem gerir e monitorizar dispositivos como SkySpark, AWS IoT Device Management e Azure IoT Central. São aplicações semelhantes à SensLIVE fornecida pela CapTemp.

Existem diversas tecnologias e *frameworks* de *front-end*, *back-end*, *cross-platform* e SGBD. Cada uma possui diversas vantagens e desvantagens e por isso é necessário realizar uma pesquisa de modo a determinar a melhor para ser utilizada consoante o tipo de projeto. No estágio, esta análise e decisão foi feita previamente pelo DT, no entanto, é sempre importante realizar uma pesquisa para fazer uma análise critica das decisões tomadas.

4. Plataforma SensLIVE

O SensLIVE é uma plataforma IIoT desenvolvida pela CapTemp cujo seu objetivo principal é a monitorização e gestão de sensores e dispositivos. Esta solução apresenta diversas funcionalidades como:

- **Controlo de Acessos de Utilizador** – Gestão de permissões e acessos dos utilizadores, garantindo que apenas as pessoas autorizadas possam visualizar e interagir com a plataforma;
- **Notificações em tempo real do estado dos Equipamentos (Mail ou SMS)**;
- **Mapa Geográfico da Localização dos dispositivos**;
- **Gestão de Alertas**;
- **Atualização em tempo real das últimas ocorrências**;
- **Visualização de Gráficos** – Permite visualizar as informações dos sensores através de gráficos facilitando a sua análise;
- **Exportação de Relatórios** – Exportação de relatórios em formato PDF de históricos de alertas, gráficos, consumos, etc.
- **Sistema de *Widgets*** – Dispõe de vários *widgets* que podem ser colocados de acordo com as preferências e necessidades do utilizador. Estes *widgets* podem mostrar valores de sensores atualizados, gráficos ou mapas.

Este capítulo está dividido em quatro subcapítulos. O primeiro descreve a arquitetura da plataforma. O segundo refere alguns aspetos relativos à segurança e privacidade dos dados. O terceiro apresenta os protocolos de comunicação utilizados e, no último subcapítulo, são mencionadas as principais diferenças entre a versão antiga e a nova versão do SensLIVE.

4.1. Arquitetura

A arquitetura serve como base num produto de *software* que especifica como os componentes estão interligados e interagem entre si. A escolha da arquitetura e as tecnologias a utilizar dependem de projeto para projeto.

Relativamente ao SensLIVE, é importante explorar a sua arquitetura apesar de ter sido desenhada antes do início do estágio. De seguida são apresentadas as tecnologias utilizadas

para o *back-end*, o *front-end*, *cross-platform* e o SGBD e, por fim, é demonstrada uma visão geral da arquitetura.

4.1.1. Framework Back-End

Laravel [73] é a *framework* utilizada para o desenvolvimento do *back-end*. Oferece um vasto conjunto de ferramentas, recursos e funcionalidades como encaminhamento, validações, armazenamento de ficheiros, etc., para criar aplicações *web* modernas. Utiliza PHP, uma linguagem *open-source* que se foca principalmente em *scripting* do lado do servidor. Esta *framework* possui uma vasta documentação de fácil interpretação e uma comunidade ativa. Segue a arquitetura MVC que é um padrão arquitetural utilizado em diversas plataformas como Windows, MacOS e Web. Esta arquitetura isola a "lógica de domínio" da interface do utilizador e é dividida em três camadas:

- ***Model*** (Modelo): Responsável pela Gestão dos dados;
- ***View*** (Vista): Gera o conteúdo visualizado pelo utilizador (geralmente HTML);
- ***Controller*** (Controlador): Recebe pedidos (*requests*) do cliente e interage com o modelo para exibir os dados da vista. [74]

Na Figura 6 é possível visualizar um esquema da arquitetura MVC.

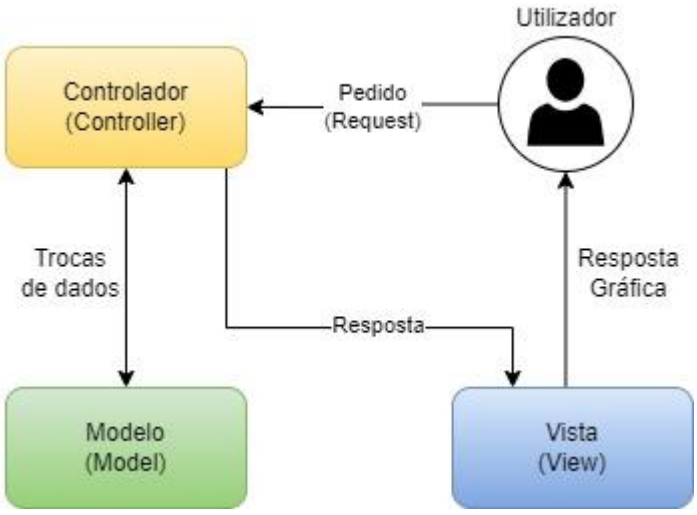

Figura 6 - Arquitetura MVC do Laravel

No subcapítulo 5.2.4 é aprofundada a criação de modelos e controladores.

4.1.2. Frameworks Front-End

Vue.js é uma *framework open-source* JavaScript desenhada para desenvolver interfaces para SPA. Através da utilização de HTML, CSS e JavaScript, é fornecido um modelo de programação declarativo baseado na criação de componentes que ajuda a desenvolver interfaces de qualquer complexidade. Estes componentes são escritos num ficheiro único (*Single-File Component*) que engloba a sua lógica (JavaScript), o modelo (HTML) e o seu estilo (CSS). [75]

Neste projeto são criados vários componentes (formulários, botões, *dropdowns*, etc.) que são reutilizados várias vezes, diminuindo a replicação de código. Sendo uma SPA apenas é carregada uma página e o seu conteúdo é atualizado. Na Figura 7 é possível ver um esquema da página principal.

- **"SideMenu"** – *Navigation bar* lateral que contém botões que mudam o conteúdo da página;
- **"SideMenu2"** – Barra na parte superior da página que contém botão das notificações e do perfil;
- **Conteúdo atualizado** – Mostra todos o conteúdo das páginas (um componente por página).

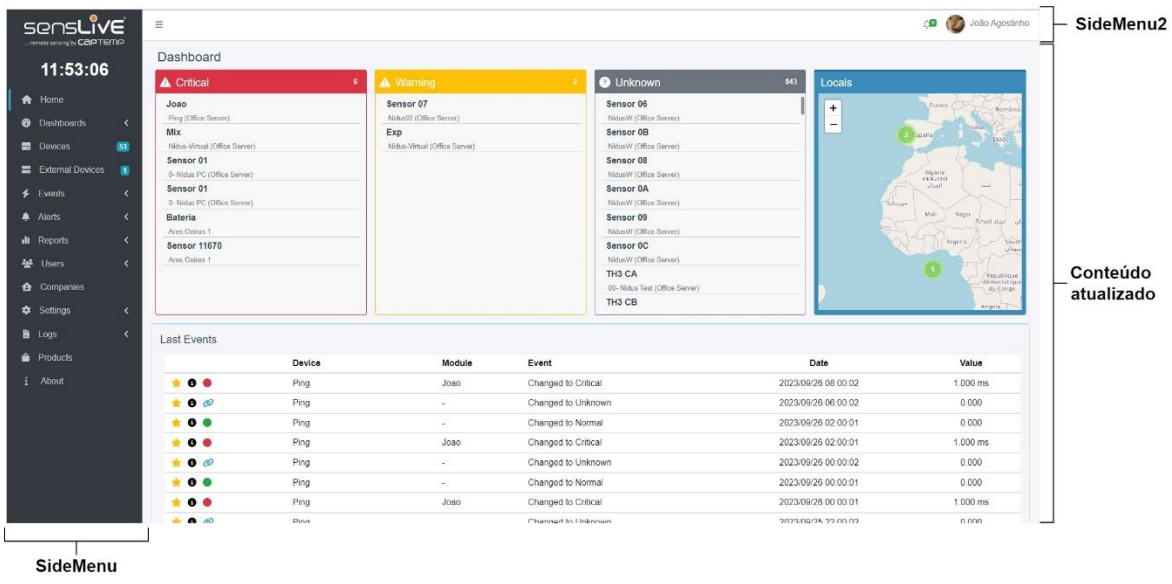

Figura 7 - Esquema com os conteúdos da página principal do SensLIVE Vue

4.1.3. Framework Cross-Platform

Flutter é uma *framework open-source* desenvolvida pela Google. A sua maior vantagem é a criação interfaces de utilizador UI de uma aplicação para várias plataformas usando

apenas uma base de código. Suporta o desenvolvimento de aplicações para 6 plataformas: Web, Windows, Linux, MacOS, iOS e Android. Desta forma, é possível construir aplicações nativas de alta qualidade e consistentes sem necessidade de manter códigos separados. Utiliza Dart como linguagem para o seu desenvolvimento de UI e oferece uma vasta coleção de *widgets* (semelhante aos componentes em Vue) que oferecem uma experiência consistente aos utilizadores. [58]

A Figura 8 apresenta a estrutura das diretorias da aplicação Flutter. As diretorias "android", "ios", "linux", "macos", "web" e "windows" contêm os ficheiros específicos e configurações necessárias para construir a aplicação em cada plataforma. Na diretoria "assets" são guardados todos os ficheiros estáticos como imagens, ficheiros JSON, etc. A maioria do código, incluindo *widgets* personalizados e o ficheiro de inicialização do projeto ("main.dart") encontram-se na diretoria "lib". Como o nome indica, a diretoria "tests" é onde são escritos os testes unitários, de integração, etc. e. por fim, o ficheiro "pubspec.yaml" contém as dependências e a localização dos *assets*.

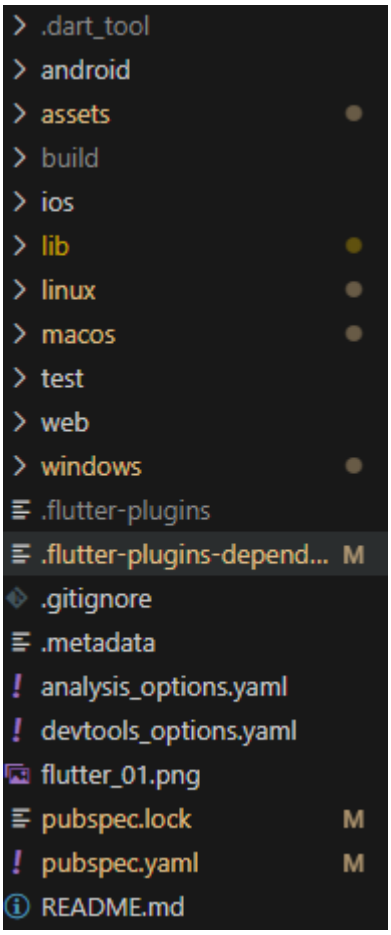

Figura 8 - Estrutura das diretorias do SensLIVE Flutter

4.1.4. Sistema de Gestão Base de Dados

PostgreSQL foi utilizado como SGBD para o projeto SensLIVE. É um SGBD objeto-relacional *open-source* e utiliza a linguagem SQL. Possui mais de 35 anos de desenvolvimento ativo e ganhou bastante reputação pela sua arquitetura, fiabilidade, integridade dos dados e robustez. [76]

Uma BD objeto-relacional é semelhante a uma BD relacional que utiliza recursos de modelagem orientada a objetos. Isto permite a utilização de dados complexos como ficheiros XML e JSON. Foram desenvolvidas para lidar com a complexidade dos dados de aplicações modernas que muitas vezes requerem mais flexibilidade do que os modelos relacionais. [77]

No subcapítulo 5.2.3 é descrita a BD e as tabelas utilizadas.

4.1.5. Visão Geral e Infraestrutura

Na Figura 9 está representado um esquema da arquitetura da plataforma SensLIVE.

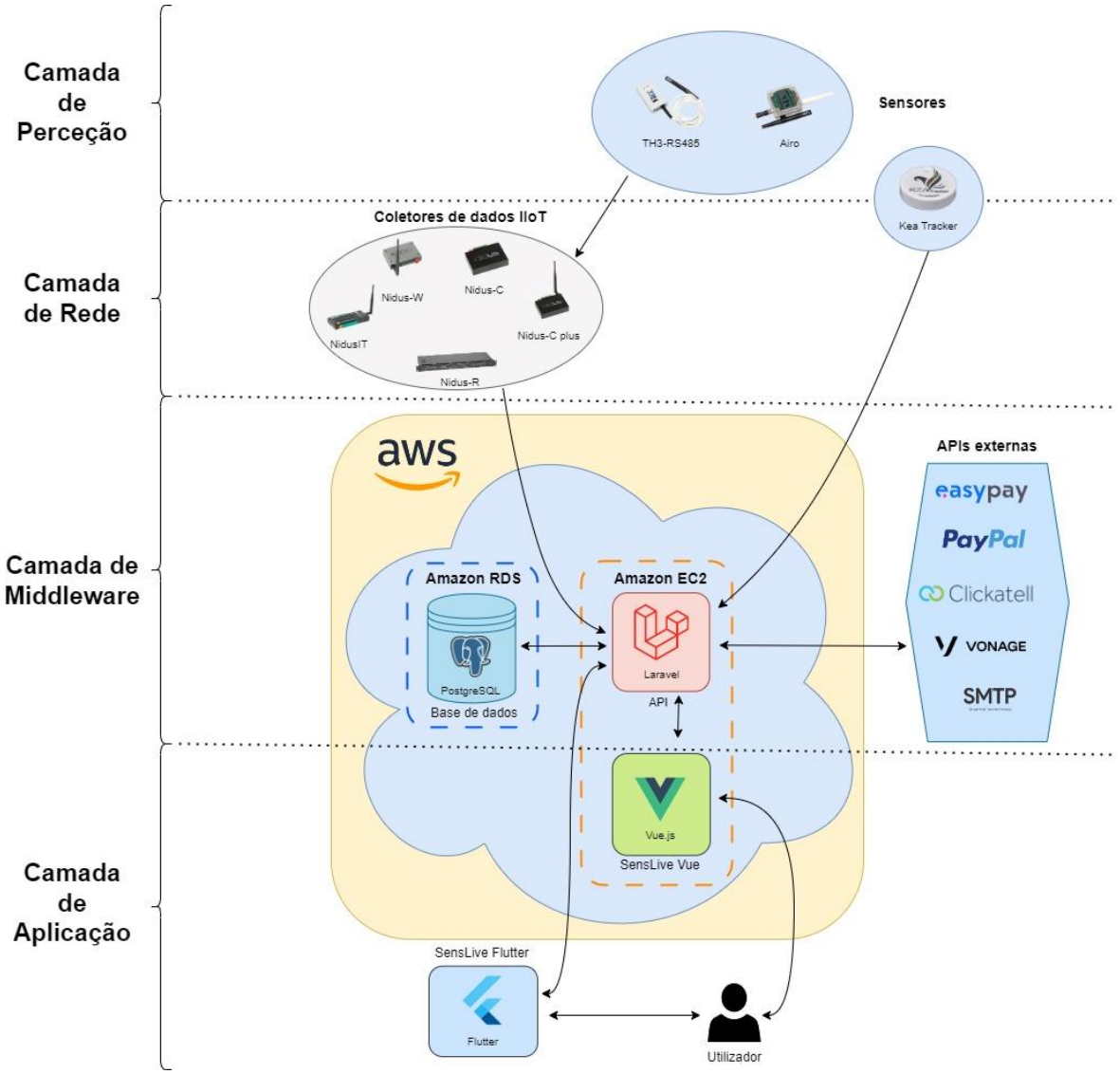

Figura 9 - Arquitetura da plataforma SensLIVE

Como é possível visualizar, a arquitetura da plataforma está dividida pelas camadas da arquitetura IIoT: perceção, rede, *middleware* e aplicação. Esta arquitetura garante uma estrutura organizada e escalável para a plataforma, facilitando a sua manutenção desenvolvimento e expansão futura.

Se seguida são descritas as camadas aplicadas à arquitetura da plataforma.

Camada de Perceção

É composta por sensores (wireless ou com fio) que recolhem dados (temperatura, humidade, gases, etc.) em tempo real e enviam-nos para a camada de Rede. Todos os sensores são desenhados e produzidos pela CapTemp.

Existe um sensor (Kea Tracker) que recolhe e envia os dados para a camada de *Middleware,* ou seja, pertence simultaneamente à camada de Perceção e de Rede.

Camada de Rede

É constituída pelas "Nidus" que são coletores de dados IoT. Estas recebem os dados recolhidos pelos sensores e encaminham-nos para a camada de *Middleware* para serem processados. São essenciais para garantir a transmissão segura dos dados através de protocolos de comunicação. Todas as "Nidus" são desenhadas e produzidas pela CapTemp e possuem características diferentes, como protocolos de comunicação, número de inputs e outputs, etc.

Camada de Rede

É nesta camada que são armazenados e processados os dados. A BD PostgreSQL, a API do Laravel e o SensLIVE Vue estão hospedados na AWS. Dentro da AWS existe a máquina virtual Amazon RDS (serviço de BD relacional) que contém a BD PostgreSQL e a máquina virtual EC2 que contém a API Laravel e o SensLIVE Vue (não faz parte desta camada). Fora da AWS existem APIs externas que são essenciais para algumas das funcionalidades do SensLIVE como efetuar pagamentos, receber/enviar emails ou mensagens, entre outros.

Camada de Aplicação

Inclui as aplicações SensLIVE Vue (hospedada máquina virtual EC2 dentro da AWS) e SensLIVE Flutter. Ambas estabelecem contacto com o utilizador fornecendo interfaces interativas e funcionalidades de visualização e manipulação dos dados. Mantêm uma comunicação bidirecional com a API Laravel.

4.2. Segurança e privacidade

A segurança e privacidade são elementos essenciais da aplicação. Esta possui várias medidas de segurança robusta de modo a garantir a integridade e confidencialidade das informações para proteger os utilizadores e dispositivos contra acessos não autorizados e outras ameaças.

O *middleware* da aplicação atua como uma camada intermediária entre as aplicações e os serviços, assegurando o acesso às APIs apenas por pessoas autorizadas. No SensLIVE, existem vários *middlewares* consoante as permissões dos utilizadores que controlam se este

pode criar, editar visualizar e eliminar certos dados. Por exemplo, um trabalhador da empresa X não pode eliminar sensores da empresa Y. Esta abordagem, complementada com *passwords* encriptadas, minimiza o risco de acessos indevidos e ações não autorizadas dentro do sistema.

Além de *middlewares*, a plataforma realiza verificações de dados para assegurar a integridade e fiabilidade dos mesmos. Estas verificações são feitas na entrada (*front-end*) e na receção (*back-end*) dos dados, garantindo que apenas as informações corretas e esperadas sejam aceites pelo sistema. Estas validações ajudam a prevenir ataques de injeção de código e outras formas de manipulação de dados.

Este conjunto de medidas garantem um ambiente seguro e fiável para a monitorização e gestão de sensores e dispositivos, protegendo as informações dos utilizadores.

4.3. Protocolos de comunicação

São utilizados diversos protocolos de comunicação para assegurar a interoperabilidade e a flexibilidade do sistema. Cada protocolo oferece vantagens que, quando combinadas, tornam a solução mais eficiente, operacional e segura. Os protocolos utilizados pela plataforma são: LoRaWAN, MQTT, HTTP, NB-IoT, SMNP, PROFINET, BACnet, ICMP e Modbus.

4.4.Diferenças do SensLIVE antigo

O SensLIVE passa por uma evolução tecnológica de forma a tornar-se mais eficiente, robusto e "amigo" do utilizador. De seguida são detalhadas as principais diferenças da versão antiga e da nova versão da plataforma.

4.4.1. Tecnologias

A versão antiga era desenvolvida inteiramente em PHP, tanto *back-end* como *front-end*, e não existia nenhuma *framework cross-platform* a ser utilizada. Embora fosse funcional, estava limitada pelas capacidades do PHP em termos de interface e interatividade.

Na nova versão, o *back-end* mantém-se, continuando a utilizar PHP para a lógica, no entanto, o *front-end* foi reescrito utilizando Vue.js e, para chegar a mais plataformas e SO's, foi desenvolvida uma aplicação em Flutter. Ambas as tecnologias (Vue.js e Flutter) proporcionam interfaces mais dinâmicas e responsivas.

4.4.2. Sessões vs. Tokens

Outra melhoria significativa diz respeito à gestão de sessões do utilizador. Na versão anterior era utilizado um sistema de sessões tradicionais para gerir a autenticação e autorização dos utilizadores. Embora funcional, colocava uma carga considerável no servidor que podia afetar o desempenho da aplicação. A autenticação e autorização passaram a ser geridas através de *tokens*, assim, a carga colocada no servidor é menor pois os *tokens* são geridos do lado do cliente, logo, diminui o armazenamento no servidor resultando numa plataforma mais rápida e escalável. [78]

4.4.3. Dashboards reorganizáveis

Uma das limitações da versão antiga era a impossibilidade de reorganizar os *widgets* nos *dashboards*. Os utilizadores estavam restritos a um *layout* fixo, dificultando a personalização e visualização dos dados. Com a utilização das novas tecnologias, foi possível remodelar os *dashboards* permitindo arrastar e mudar o tamanho dos *widgets* conforme as necessidades do utilizador.

4.4.4. Layout mais apelativo

O *layout* do SensLIVE antigo era funcional e simples. A nova versão introduz interfaces mais apelativas e modernas, com um design mais intuitivo e agradável. Estas mudanças melhoram a experiência visual e facilitam a navegação na plataforma.

Na Figura 10 (versão antiga) e Figura 11 (versão nova) é possível verificar a diferença da UI das plataformas.

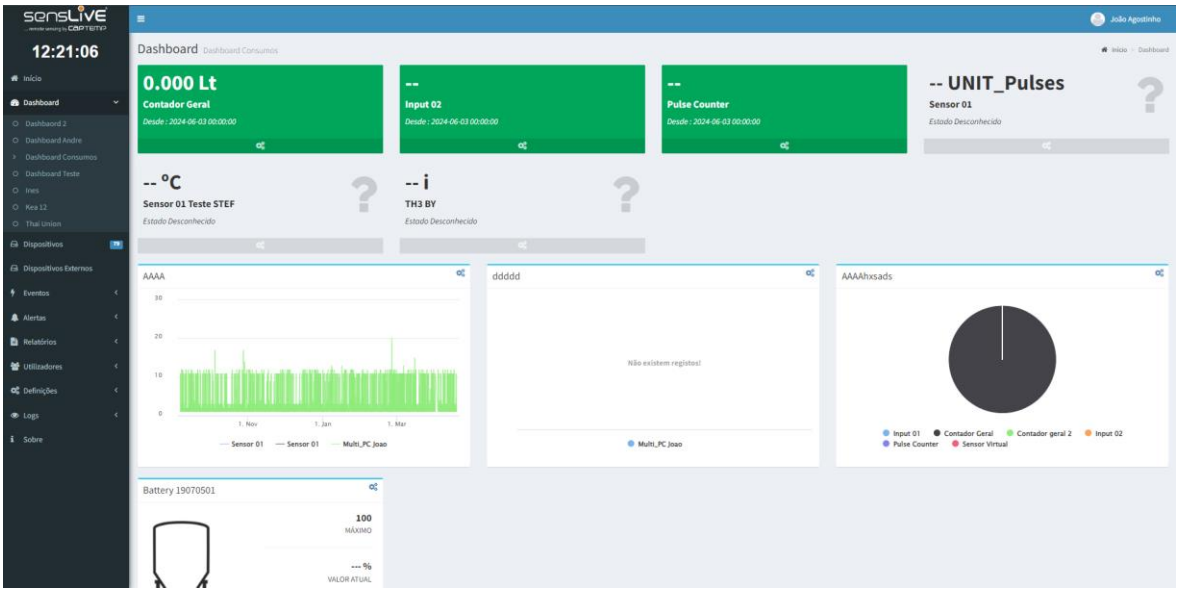

Figura 10 - Página dos Dashboards antiga

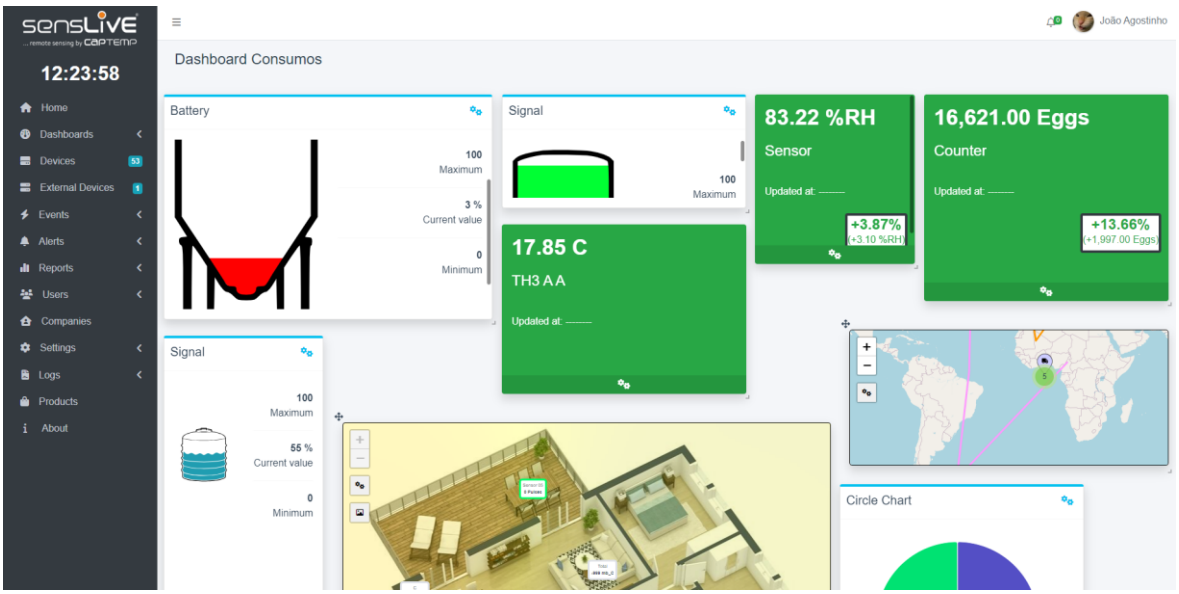

Figura 11 - Página dos Dashboards nova

4.5. Síntese

O SensLIVE é uma plataforma IIoT desenvolvida pela CapTemp. que permite monitorizar e gerir sensores e dispositivos remotamente. Utiliza Laravel como *framework* de *back-end*, Vue.js como *framework* de *front.end*, Flutter como *framework cross-platform* e PostgreSQL como SGBD. Está alojada em máquinas virtuais da AWS e utiliza *middlewares* e verificações para assegurar o acesso aos dados apenas por pessoas autorizadas. Quanto à comunicação, é utilizado uma variedade de protocolos que incluem LoRaWAN, MQTT, HTTP, etc. A sua nova versão tem diversas diferenças relativamente à

antiga, nomeadamente o uso de novas tecnologias, a troca da utilização de sessões por *tokens*, um sistema de *widgets* personalizável e o aspeto geral da aplicação.

5. Trabalho Desenvolvido

O projeto SensLIVE foi o foco do estágio e foi dividido em duas aplicações: Vue (JavaScript) e Flutter (Dart). Em ambas as partes, foram realizadas as mesmas funcionalidades utilizando o mesmo *back-end* Laravel (PHP). Inicialmente, para ficar familiarizado com o código e com a estrutura do projeto, o supervisor atribuiu tarefas simples como alinhamento de CSS e a criação de formulários. Posteriormente a isso, foram realizadas tarefas mais complexas como a criação de PDF e a criação de um sistema de *widgets*. A Tabela 6 mostra o plano de atividades desenvolvidas distribuídas ao longo dos meses.

Tabela 6 – Plano de Atividades Desenvolvidas durante o Estágio

ATIVIDADES DESENVOLVIDAS	**Setembro**	**Outubro**	**Novembro**	**Dezembro**	**Janeiro**	**Fevereiro**	**Março**	**Abril**	**Maio**	**Junho**
- Integração no projeto	X	X								
- Análise dos requisitos solicitados pelos clientes e/ou novas funcionalidades.		X	X	X	X	X	X	X	X	X
- Desenvolvimento das novas funcionalidades - *Back-end* & REST API		X	X	X	X	X	X	X	X	X
- Desenvolvimento das novas funcionalidades – *Front-end*		X	X	X	X	X	X	X	X	X
- Desenvolvimento de uma aplicação *Mobile/Desktop* usando a *framework* Flutter						X	X	X	X	X
- Desenvolvimento de documentação (relatório de estágio, entre outros)							X	X	X	X

Este capítulo está dividido em quatro subcapítulos. O primeiro descreve o ambiente de desenvolvimento. O segundo apresenta algumas notas gerais, ou seja, partes importantes que

são utilizadas ao longo de todo o projeto e boas práticas de programação utilizadas. O desenvolvimento de cada funcionalidade é descrito no terceiro subcapítulo. Por fim, o último capítulo aborda os testes realizados às aplicações.

5.1. Ambiente de desenvolvimento

Para todo o desenvolvimento foi utilizado o seguinte ambiente:

Especificações do computador

1) Processador: 12th Gen Intel(R) Core ™ i5-12400
2) Memória RAM: 8 GB
3) Armazenamento: SSD 256 GB
4) SO: Windows 11 Pro

Software

1) **Back-end**
    a. Tecnologia: Laravel Framework 8.83.27
    b. Linguagem: PHP 8.1.10
2) **Front-End**
    a. Tecnologias: Vue.js 3.3.4
    b. Linguagem: JavaScript
    c. Ferramenta: Node.js 10.1.0
3) **Cross-Platform**
    a. Tecnologia: Flutter 3.22.1
    b. Linguagem: Dart
    c. Ferramenta: Flutter SDK
4) **Servidor Web**
    a. Tecnologia: Laragon 6.0
    b. Componente: Apache 2.4.54
5) **Sistema de BD**
    a. Tecnologias: pgAdmin 7.6
    b. Linguagem: PostgreSQL 12.13
6) **IDE**
    a. Ferramenta: Visual Studio Code 1.89.1

5.2. Notas Gerais

5.2.1. Componentes e *Widgets*

De modo a promover uma estrutura organizada e clara, Vue e Flutter permitem a divisão do código em "blocos". Em Vue esses "blocos" são denominados de componentes e em Flutter de *widgets*. Podem ser reutilizáveis, evitando a replicação do código e tornando a aplicação mais leve e com melhor desempenho. Possibilita também a criação de interfaces complexas a partir de componentes simples mantendo uma consistência visual. De seguida serão mencionados alguns componentes criados.

Dropdowns

Em Vue, é utilizado um componente chamado "CTSelect" (desenvolvido pela CapTemp) que consiste num *dropdown/select* que pode ser personalizado consoante as necessidades da funcionalidade. Por exemplo, permite definir o número de opções escolhidas, pesquisar opções, desselecionar todas as opções, definir um *placeholder*, etc.

Em Flutter, existe uma package chamada "Multi Select Nested" que possui algumas características iguais ao CTSelect. No entanto, para conseguir o mesmo nível de especificação do componente Vue, foi necessário alterar o código fonte do package.

Componentes Formulários

Sendo esta uma aplicação completa onde o utilizador pode criar, modificar, ver e apagar dados, existem inúmeros formulários que o utilizador pode usar. Por isso, tanto em Vue como em Flutter foram criados componentes de formulários para poderem ser reutilizados mantendo o aspeto visual.

Elementos Simples

A mesma técnica é utilizada até para elementos simples (botões, animações de carregamento, alertas ou avisos), mantendo um design parecido em ambas as aplicações. Assim, caso um utilizador deixe de usar uma aplicação para usar outra, a curva de aprendizagem é mais pequena.

5.2.2. Sistema de Traduções

Para que as aplicações pudessem ser utilizadas por pessoas e empresas de diferentes regiões e idiomas, foram utilizadas ferramentas de internacionalização (i18n). Em Vue é

utilizado um plugin chamado de “Vue I18n”, em Flutter um package intitulado de “flutter_i18n” e em Laravel é utilizado a “Localization”. Estas ferramentas funcionam como “arquivos de tradução”, ou seja, são ficheiros JSON cujas chaves correspondem a uma *String* traduzível pela aplicação. Resumindo, é uma forma de traduzir a aplicação em várias línguas.

5.2.3. Base de Dados

Como parte significativa envolveu a interação com BD, foi essencial criar tabelas assim como modificar algumas existentes de modo a desenvolver as funcionalidades. A BD possui 120 tabelas, no entanto, para as funcionalidades desenvolvidas foram utilizadas 20. Na Figura 12 está desenhada a modelação física das tabelas utilizadas da BD. As tabelas azuis já existiam e as brancas foram criadas no estágio. As chaves primárias estão a amarelo e as restantes linhas são chaves estrangeiras.

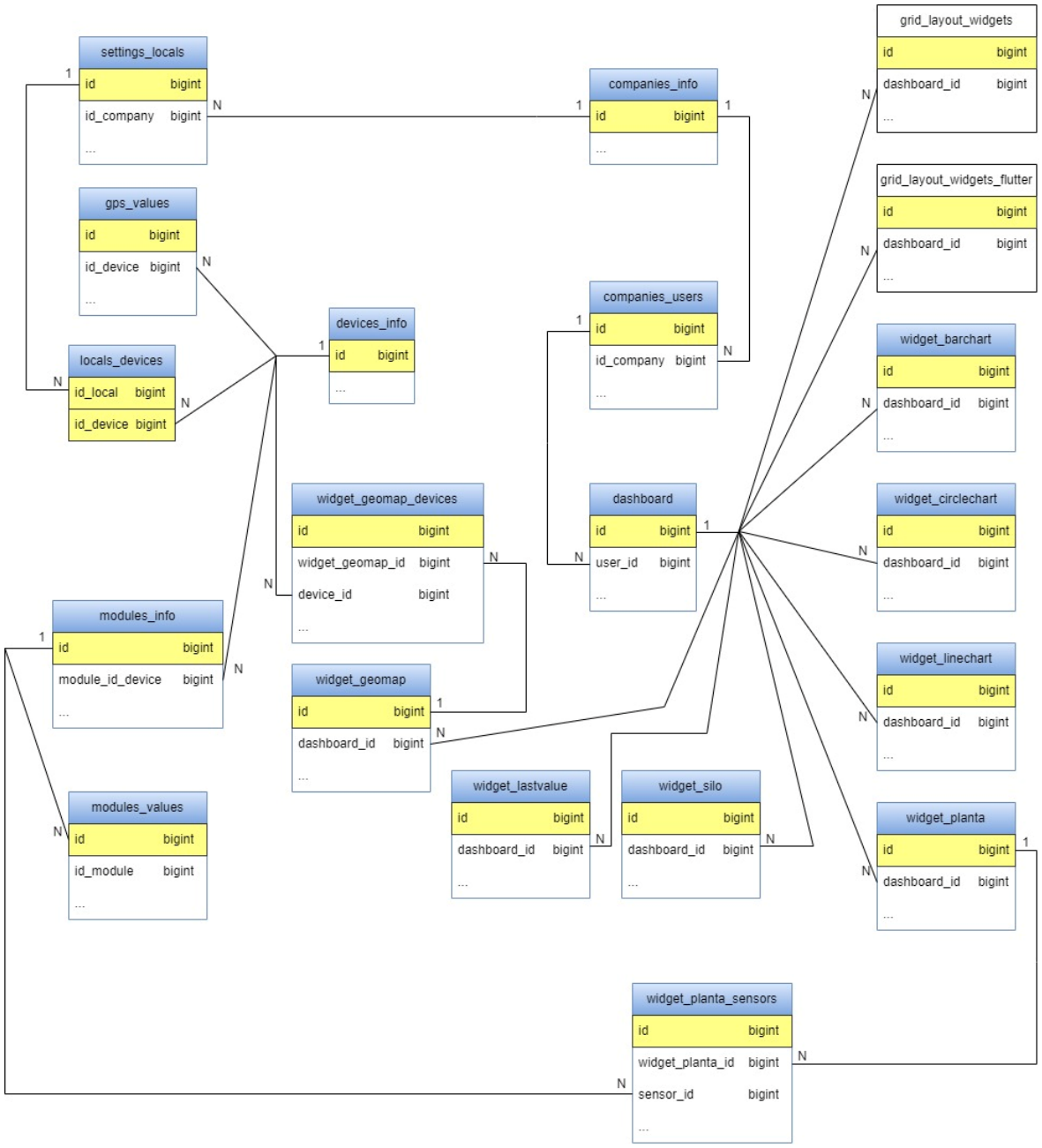

Figura 12 - Modelação Física da BD utilizada

As tabelas fornecem informações das **empresas** (“companies_info”), dos **utilizadores** (“companies_user”), dos ***dashboards*** (“dashboards”), dos ***widgets*** (“widget_lastvalue”, “widget_sile”, “widget_barchart”, “widget_linechart”, “widget_circlechart”, “widget_geomap”, “widget_planta”), dos **sensores** (“modules_info”), dos **dispositivos** (“devices_info”) e dos **locais** (“settings_locals”). A tabela “widget_planta_sensors” é dedicada aos sensores presentes no “Widget Blueprint” associado, a tabela “widget_geomap_devices” refere os dispositivos presentes no “Widget GeoMap” associado, a “gps_values” indica um histórico de coordenadas *gps* dos dispositivos e por fim a tabela

"locals_devices" faz ligação entre a tabela dos locais e dos dispositivos. As informações dos *widgets* são retratadas com mais detalhe no subcapítulo 5.3.3.

Apenas os nomes, chaves primárias e chaves estrangeiras foram adicionadas à modelação pois trata-se de informação sensível da empresa.

Para facilitar a administração da BD, foi utilizada a ferramenta pgAdmin que proporciona uma interface intuitiva que simplifica tarefas como a criação e gestão de tabelas, a execução de consultas SQL, etc.

5.2.4. Modelos, Controladores e Rotas das APIs

Modelos

Cada modelo representa uma tabela da BD como se fosse um objeto ou uma classe em PHP. Assim, foram criados modelos para cada tabela utilizada.

O código abaixo mostra o modelo do "Widget Last Value". Começa por definir o *namespace* (diretoria) onde está guardado o ficheiro e os *imports* necessários para o modelo. De seguida é declarada a classe e a sua extensão ("StoreUserUpdates") e, dentro dela, são definidos os *imports* que são utilizados e algumas propriedades do modelo como: o nome da tabela associada e a utilização de *timestamps*. No final é definida a função "dashboard" que devolve a relação que tem com o modelo "Dashboard". O modelo "StoreUserUpdates" serve para "disparar" eventos quando o modelo é criado, atualizado ou eliminado. Por exemplo, ao criar um "Widget Last Value", as colunas "created_by" e "created_at" são automaticamente preenchidas com o *id* do utilizador e com a data do evento, respetivamente.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use App\Models\Dashboard;

class WidgetAlerts extends StoreUserUpdates


{
    use SoftDeletes;
    use HasFactory;
```

```
    protected $table = 'widget_alerts';
    public $timestamps = true;

    //Relation function

    public function dashboard()
    {
        return $this->belongsTo(Dashboard::class,"dashboard_id",'id');
    }
}
```

Controladores

Agrupam a lógica dos *requests* numa única classe, ou seja, possuem as funções de tratamento de dados dos modelos. Todos os controladores estendem a classe "Controller" incluída no Laravel.

No código abaixo é possível ver o controlador DashboardController. Neste controlador, estão todas as funções relacionadas aos *dashboards* e aos *widgets*, no entanto, no código abaixo apenas são mostradas as declarações das funções relativas ao "Widget Last Value".

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
```

(…)

```
class DashboardController extends Controller
{
```

(…)

```
//WIDGET LAST VALUE
    //Get a single WidgetLastValue info
    public function getWidgetsLastValue(Request $request, int $id)
    {…
    }
    //Edit WidgetLastValue
    public function editWidgetLastValue(Request $request)
    {…
    }
    //Create Widget Last Value
    public function createWidgetLastValue(Request $request)
```

```
    {…
    }
    //Delete WidgetLastValue
    public function deleteWidgetLastValue(Request $request, int $id)
    {…
    }
}
```

Rotas API

As rotas API são projetadas para lidar com os *requests* da aplicação e retornam dados no formato JSON que são utilizados para mostrar informações ao utilizador. Estas rotas estão definidas no ficheiro "api.php".

O bloco de código mostra uma parte do ficheiro "api.php" onde, novamente, inicializa com os *imports* de classes. Logo de seguida é definido o grupo de rotas através do "Route::group(..)". Dentro desse grupo é determinado o *throttle* (número de pedidos por minuto) e definidas as rotas, que neste caso são rotas para criar e receber dados de um "Widget Last Value". Cada rota define o método (GET, POST, PUT ou DELETE) do *request*, o *url*, o controlador da função, o nome da função, *middleware* utilizado e o nome da rota.

```
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::group(['prefix' => '/v1', 'namespace' => 'v1'], function () {
$trotle='throttle:500,1';  //500 request per minute and need wait one minute
after
```

(…)

```
//---LAST VALUE WIDGET---
//Create Widget Last Value
Route::post('/dashboards/createWidgetLastValue',[DashboardController::class,
'createWidgetLastValue'])-
>middleware(['auth:sanctum','apiPermissionGlobal','devicesView','apiManager'
,$trotle])->name("createWidgetLastValue");
```

```
//Get Widget Last Value
Route::get('/dashboards/widgetsLastValue/{id}',[DashboardController::class,'
getWidgetsLastValue'])-
>middleware(['auth:sanctum','apiPermissionGlobal','devicesView','apiManager'
,$trotle])->name("GetWidgetsLastValue");

}
```

Por exemplo, para a rota “Get Widget Last Value” (última rota do bloco de código acima) podemos definir:

- **Método:** GET;
- **URL:** ‘/dashboards/widgetsLastValue/{id}’;
- **Controlador:** DashboardController;
- **Função do Controlador:** getWidgetsLastValue;
- **Middleware:** 'auth:sanctum', 'apiPermissionGlobal', 'devicesView', 'apiManager', $trotle;
- **Nome da rota:** GetWidgetsLastValue;

5.2.5. Boas práticas

Para todo o desenvolvimento foram adotadas “boas práticas” aprendidas ao longo do curso de modo a garantir a qualidade, usabilidade e manutenção do *software*. De seguida são apresentadas algumas dessas “boas práticas”.

Responsividade

Capacidade da aplicação se adaptar a diferentes tamanhos de ecrã e dispositivos, desde ecrãs de computador até smartphones. Garante aos utilizadores uma experiência consistente independentemente do dispositivo que estejam a utilizar.

Utilização de cores Web-Safe

É uma palete de 216 cores formadas pela combinação de vermelho, verde e azul (RGB – *Red Green Blue*) em valores hexadecimais. A utilização destas cores garante uma aparência consistente em qualquer plataforma evitando discrepâncias visuais que possam ocorrer na reprodução de cores.

Verificações e Mensagens de Erro

A implementação de verificações e de mensagens de erro melhora a usabilidade e a robustez da aplicação. As verificações ajudam a prevenir a introdução de dados incorretos e as mensagens de erro ajudam o utilizador a corrigir esses dados, reduzindo a sua "frustração" e minimizando o risco de erros que possam afetar o bom funcionamento da aplicação.

Na Figura 13 é possível visualizar um alerta exemplo de sucesso da aplicação Flutter.

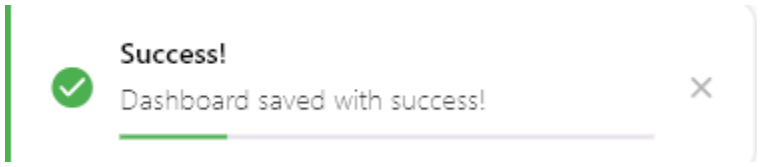

Figura 13 - Alerta exemplo da aplicação Flutter

Epoch Time

É um formato simples e uniforme que facilita a manipulação, comparação e armazenamento de datas. É independente de fuso horários, eliminando complicações associadas a diferentes regiões. É suportado em várias linguagens de programação e sistemas garantindo a compatibilidade e portabilidade do código.

Comentários

Em todo o código implementado foram escritos comentários simples de modo a manter a sua clareza e compreensão, ajudando futuros programadores a entender rapidamente a sua lógica. Esta prática facilita a manutenção e expansão futura da aplicação.

Eliminação de dados

Em termos de eliminação de dados, é sempre necessário ter cuidado. No *front-end*, quando o utilizador tenta apagar algo é sempre exibida uma mensagem de confirmação de modo a prevenir eliminações acidentais. No *back-end* são utilizados "Soft Deletes", ou seja, em vez de serem apagadas linhas das tabelas da BD, são preenchidas as colunas *deleted_at* e *deleted_by* com a data e o id de quem apagou, respetivamente. Desta forma, os dados nunca são "apagados" e podem ser recuperados se necessário.

Funções do Controlador (*Back-End* Laravel)

Para todas as funções dos Controladores, são realizadas verificações para garantir que os dados recebidos são válidos antes de serem processados. Para qualquer alteração, criação

ou eliminação de dados na BD, é usado um método do Laravel chamado “DB Transactions” que permite fazer múltiplas operações na mesma unidade. Caso ocorra algum erro, todas as operações são revertidas, prevenindo dados incompletos na BD. Todas as tabelas possuem campos *created_at*, *created_by*, *updated_at*, *updated_by*, *deleted_at* e *deleted_by* para fazer o acompanhamento de quem e quando criou, alterou ou eliminou dados.

O bloco de código abaixo permite criar um “Widget Last Value”. Começamos o código com a validação da permissão que o utilizador tem para fazer a operação. De seguida, através da função *validate*, são feitas as verificações dos dados que vêm pelo *request*. Por fim, temos a transação das operações necessárias para criar o *widget* na BD. Dentro da transação, podemos verificar a utilização dos campos *created_at* e *created_by* para registar quem criou e quando foi criado o *widget*. É possível verificar ao longo do código alguns comentários que ajudam a explicar o mesmo.

```
//Create Widget Last Value
    public function createWidgetLastValue(Request $request)
    {

      //Validate user Dashboard
        $dashboard = Dashboard::findOrFail($request->idDashboard);
        if ($dashboard->user_id != $request->user()->id) {
            //check if the person has permissions to edit shared dashboard
            $validated = $request->validate([
                'id' =>
'required|exists:dashboard_symlink,dashboard_id,user_id,' . $request-
>user()->id, "shared_enable,1",
            ]);
        }


        //Validate Data
        $validated = $request->validate([
            'idSensor' => 'required',
            'idDashboard' => 'required|exists:dashboard,id',
        ]);


        DB::transaction(function () use ($request) {

            //Set Sizes
```

(…)

```
            //Create Widget
```

```
            $widget = new WidgetLastValue;
```

(...)

```
            $widget->created_at = time();
            $widget->created_by = $request->user()->id;
             $widget->save();

            //Create Grid Item
            $this->addGrid(…);

            //Log Message
            Log::info("WidgetLastValue created by user " . $request->user()->id);
        });
    }
```

5.3. Funcionalidades

5.3.1. Formulário "Dashboard Settings"

Consiste num formulário simples que muda dois dados na BD: intervalo de consumo e o intervalo de atualização. Para selecionar ambas as opções são utilizados *dropdowns* (CTSelect). Na Figura 14 é possível visualizar os formulários em Vue (lado direito) e em Flutter (lado esquerdo).

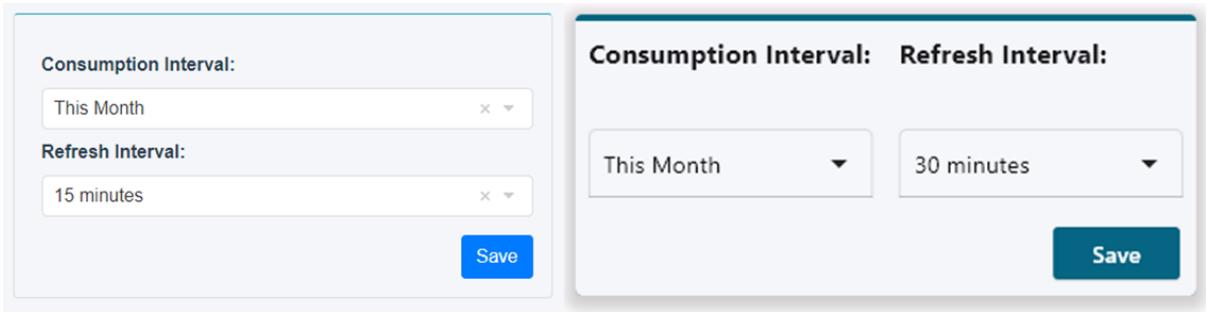

Figura 14 - Dashboard Settings Vue.js (direita) e Flutter (esquerda)

Estes dados definem o intervalo de atualização e de consumo dos dados dos sensores presentes nos *dashboards*.

5.3.1. Exportação de PDF sobre histórico de alertas

É uma funcionalidade que permite exportar para o formato PDF o histórico de alertas. Através de um formulário, é pedido ao utilizador um intervalo de datas e um conjunto de dispositivos. Ao carregar no botão de "Export to PDF", os dados são enviados para o *back-*

end e é criado um PDF com essas informações. O design do documento é desenhado num ficheiro "blade.php" e utiliza HTML, CSS e PHP. Esse ficheiro é convertido em PDF utilizando uma biblioteca PHP pública (laravel-dompdf). Assim que o PDF é criado, é feito o download do ficheiro.

No Apêndice 1 é possível visualizar o resultado de um PDF exportado com valores aleatórios da BD. A estrutura do documento possui: imagem da empresa, intervalo de tempo, uma tabela com a quantidade de alertas agrupado por local e uma tabela para cada local onde cada linha corresponde a um dispositivo. É de referir que o Apêndice 1 apenas mostra a primeira página e uma página exemplo que contenha um alerta.

Na Figura 15 é possível visualizar os formulários desenvolvidos para esta funcionalidade (Vue na parte de cima e Flutter na parte de baixo).

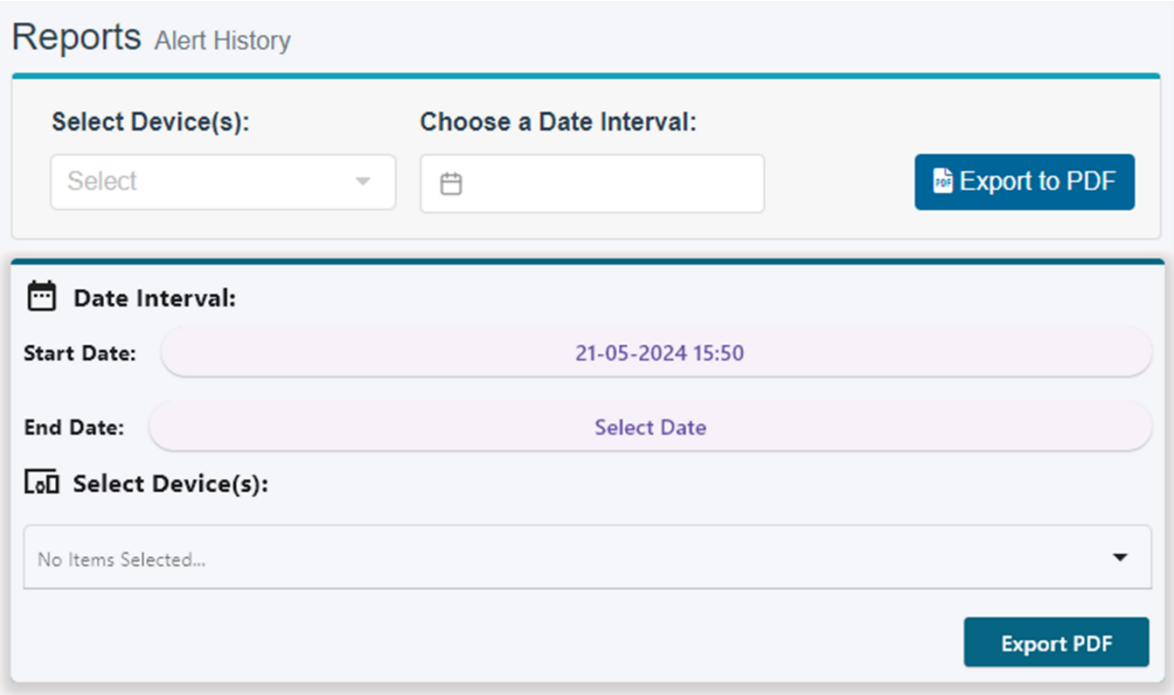

Figura 15 - Formulários para exportar em PDF o histórico de alertas

5.3.2. Formulário para mudar a imagem da empresa

Cada empresa tem a opção de mudar a sua imagem que é utilizada na exportação de PDF de alertas (Apêndice 1) e outros documentos. Esta funcionalidade é acessível através de um "pop-up" que permite ao utilizador escolher um ficheiro local para alterar a imagem da empresa. Após o envio deste ficheiro para o *back-end*, é realizada uma verificação do seu tipo, garantindo que a aplicação aceite apenas ficheiros do tipo JPG, JPEG ou PNG. Depois desta verificação, é gerado um novo nome para a imagem que é guardado na BD e o respetivo

conteúdo guardado na *storage* da aplicação. Na Figura 16 é possível ver o formulário (lado esquerdo) e o pop-up de confirmação (lado direito) com a nova imagem.

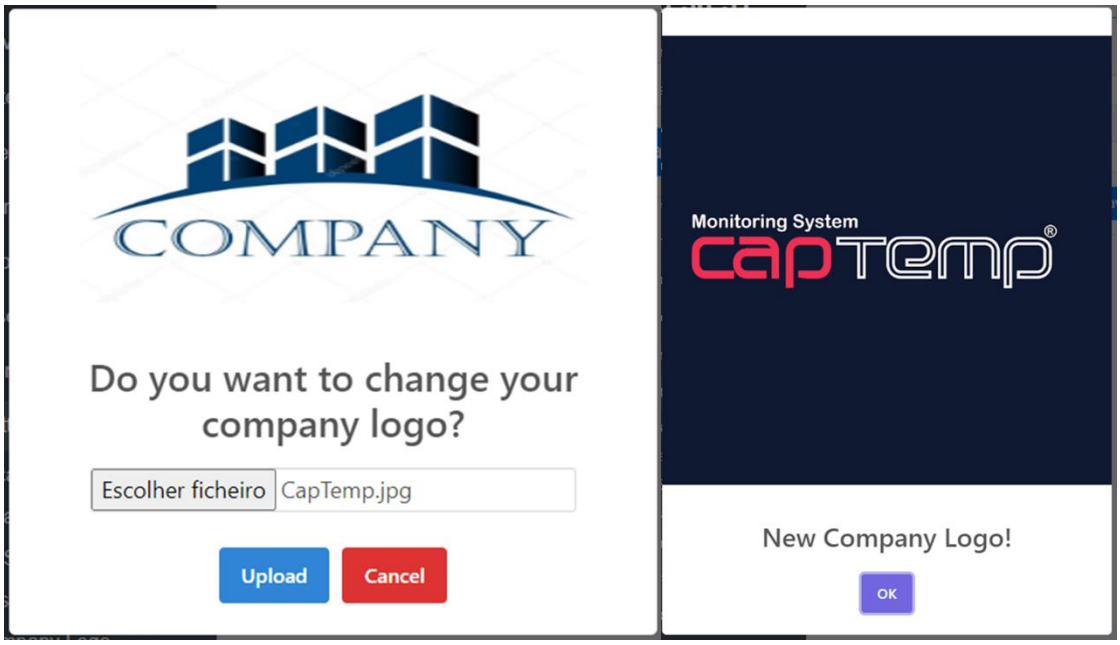

Figura 16 - Formulário para alterar a imagem da empresa

5.3.3. Sistema de Widgets

Consiste num *dashboard* personalizável e configurável onde os utilizadores podem adicionar, remover e organizar diversos *widgets* de acordo com as suas necessidades e preferências. Os *widgets* disponíveis pela aplicação incluem gráficos, mapas ou janelas que exibem valores de sensores atualizados em tempo real. Os *dashboards* são responsivos e guardam as posições e tamanhos dos *widgets* consoante o tamanho do ecrã.

Devido à complexidade da funcionalidade, serão descritos em primeiro lugar todos os *widgets* desenvolvidos de modo a tornar a leitura e a compreensão mais simples.

Widget Last Value

Este o *widget* limita-se a mostrar alguns dados de um sensor: valor atual, unidade de medida, nome, percentagem e valor exato da diferença entre a última e a penúltima leitura. É possível ver um exemplo na Figura 17.

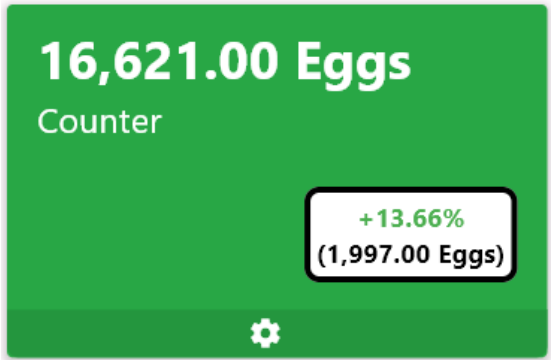

Figura 17 - Widget Last Value

Widget Silo

O "Widget Silo" representa a capacidade atual de um sensor. Mostra o seu valor máximo, mínimo e a percentagem do valor. Esta percentagem é demonstrada através de uma imagem representativa. Apenas sensores com determinadas unidades (percentagens, quilogramas, euros, etc.) podem ser representados através deste *widget*. Na Figura 18 é demonstrado um "Widget Silo" para cada imagem representativa (bateria, silo ou tanque de água, respetivamente).

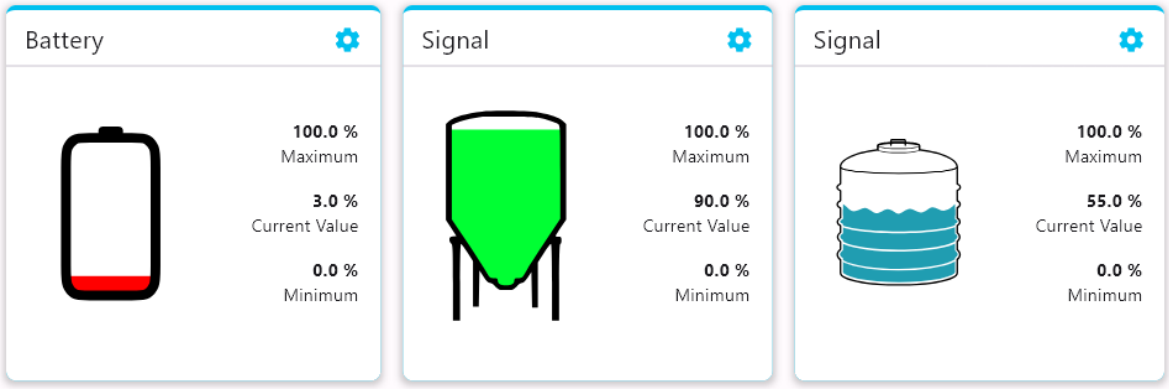

Figura 18 - Exemplos de Widget Silo

Widget Charts

Os utilizadores têm acesso a três tipos *widgets* de gráficos: circular, barras e linhas. Este conjunto permite a visualização clara e intuitiva dos valores dos dados dos sensores. Para visualizar os gráficos, os utilizadores selecionam, num formulário, os sensores, um período e intervalos de tempo (no caso do gráfico de barras). O gráfico circular mostra proporções e distribuições de dados, o gráfico de barras possibilita comparar valores em intervalos de tempo contantes ao longo de um período e o de linhas permitem comparar valores de sensores no mesmo período contínuo. Cada gráfico oferece uma perspetiva diferente dos

dados, permitindo fazer uma análise mais detalhada dos valores fornecidos pelos sensores. Informações mais detalhadas podem ser visualizadas ao passar o rato ou o dedo (no caso de ser num telemóvel/tablet) por cima do gráfico.

Na Figura 19 podemos ver exemplos do *widgets* que contêm gráfico circular (lado esquerdo), gráfico de barras (centro) e gráfico de linhas (lado direito). Os *widgets* da parte de cima (Vue) apresentam um design diferente dos da parte de baixo (Flutter) pois foram utilizadas bibliotecas/packages diferentes. Em Vue foi utilizado a biblioteca "Highcharts" e em Flutter a package "Syncfusion".

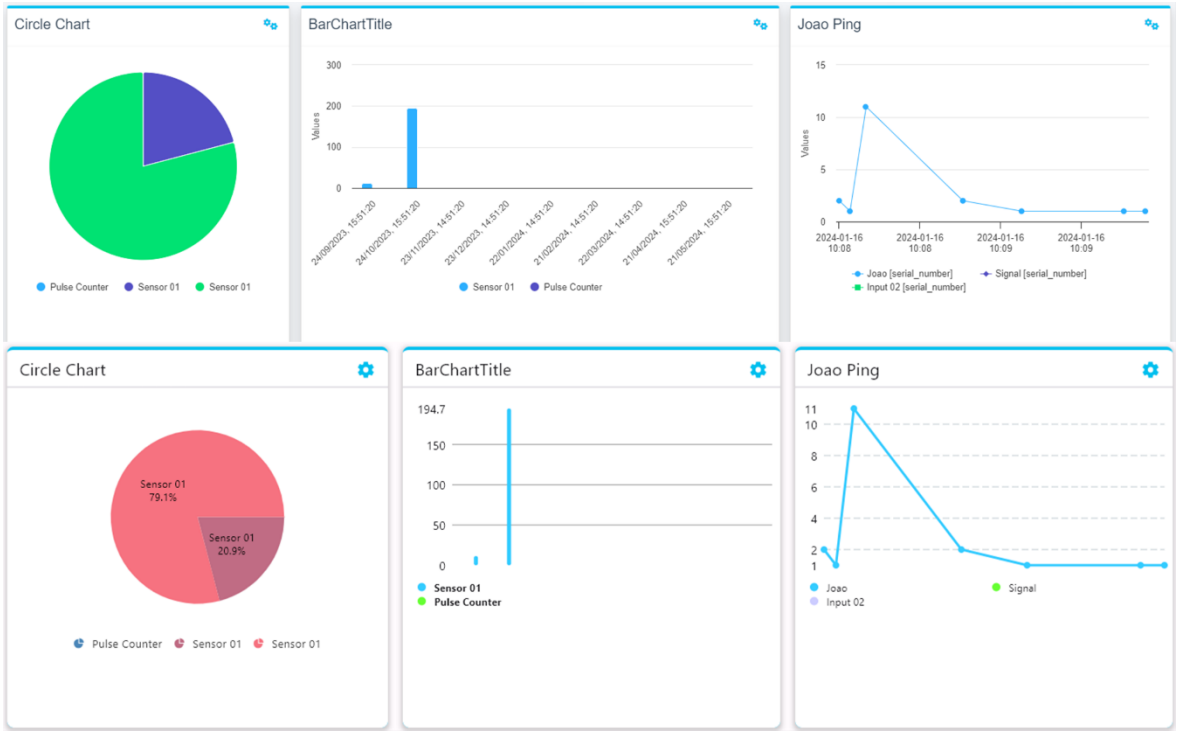

Figura 19 - Widgets Charts nas duas aplicações (Vue e Flutter)

Widget GeoMap

O "Widget GeoMap" fornece a visualização geográfica dos locais e/ou dos dispositivos que possuem informações de localização GPS. No mapa (Figura 21), são exibidos marcadores/pins que representam esses locais (pin simples azul) e dispositivos (pins coloridos com ícone de carrinha). Inclui também a funcionalidade de histórico de GPS, ou seja, existem linhas traçadas no mapa (Figura 20) que indicam o percurso do dispositivo ao longo do tempo. Desta maneira, os utilizadores podem analisar a movimentação dos dispositivos facilitando a gestão e a toma de decisão baseada na localização geográfica.

O *widget* conta com botões de aumento/diminuição do zoom (botão "+" e "-") e um botão com ícone de "engrenagem" (semelhante aos outros *widgets*) onde o utilizador pode editar ou eliminar o mesmo. Na edição, existe a possibilidade de escolher vários sensores, definir um intervalo de tempo, selecionar se pretende mostrar os locais e selecionar a "área ignorada". Esta "área ignorada" permite ignorar os pontos GPS seguidos numa certa área que se torna útil em situações onde um dispositivo permaneça no mesmo sítio por um período prolongado. Devido à margem de erro do GPS, isso resultaria num aglomerado de pontos muito próximos no mapa, tornando a visualização confusa e desordenada. Com esta funcionalidade, o mapa substitui esse "aglomerado" de pontos por um único ponto eliminando o ruído visual.

Figura 20 - Widget GeoMap (Vue)

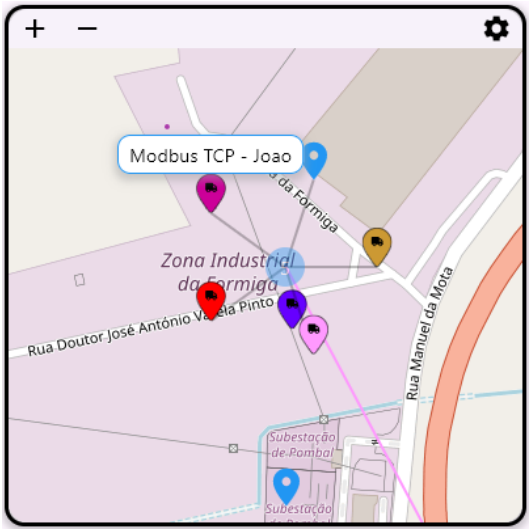

Figura 21 - Widget GeoMap aproximado (Flutter)

Para o desenvolvimento deste *widget* foi utilizado uma biblioteca chamada Leaflet para a aplicação Vue e uma package chamada "flutter_map" para Flutter, ambas desenvolvidas

pela mesma comunidade de programadores. Como a lógica das ferramentas era semelhante, a implementação e a adaptação não foram difíceis.

Widget Blueprint

O "Widget Blueprint" é o *widget* mais complexo. Permite aos utilizadores visualizar e organizar sensores numa imagem de fundo representativa de um ambiente físico. Estes sensores são exibidos como retângulos que podem ser arrastados e reposicionados pelos utilizadores, permitindo uma disposição personalizada. Cada retângulo pode conter o nome do sensor, o seu valor atualizado e uma borda colorida que revela o estado do sensor: desconhecido (cinzento), normal (verde), aviso (amarelo) ou critico (vermelho). Este *widget* possui também a funcionalidade de "Heatmap" onde o utilizador define um valor máximo e mínimo de temperatura e todos os sensores de temperatura no estado normal influenciam o padrão de cores no fundo.

Relativamente ao "Heatmap", como é possível verificar na Figura 22 e na Figura 23, o design é diferente pois são usadas ferramentas diferentes. Em Flutter é utilizado um plugin (flutter_map_heatmap) compatível com a package "flutter_map" e em Vue é importado um ficheiro JavaScript pelo ficheiro "welcome.blade.php".

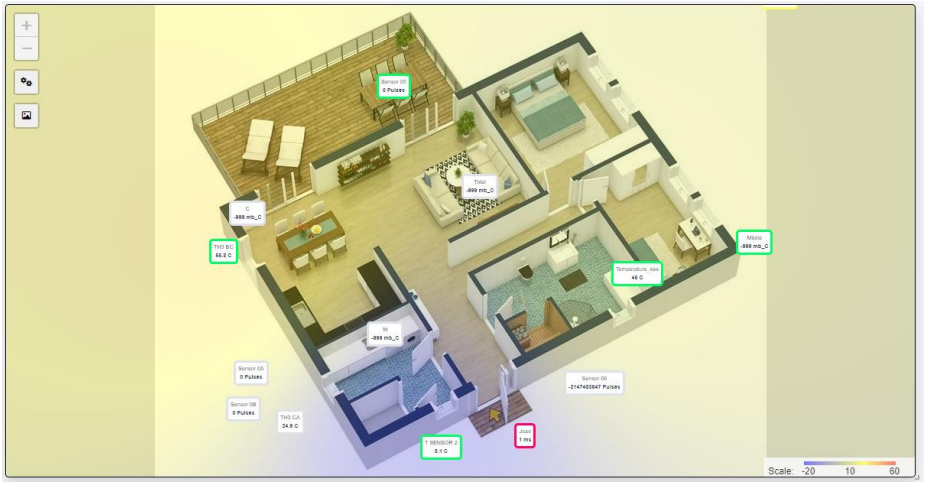

Figura 22 - Widget Blueprint (Vue)

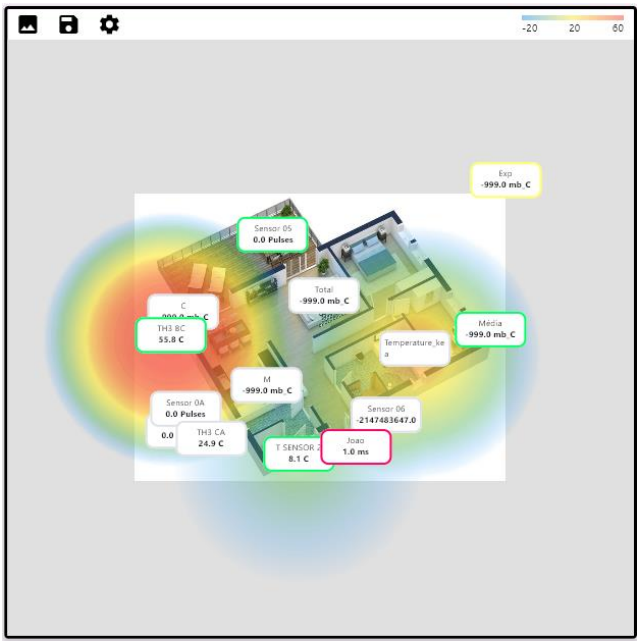

Figura 23 - Widget Blueprint (Flutter)

Ao carregar num sensor, aparece um pop-up com um gráfico de linhas que demonstra o histórico do sensor. Na Figura 24 podemos ver o histórico de valores do sensor “T SENSOR 2”, a linha preta mostra os valores do sensor, a zona amarela mostra a zona de aviso e a zona vermelha a zona critica (excede o valor máximo/mínimo somado com a taxa de erro).

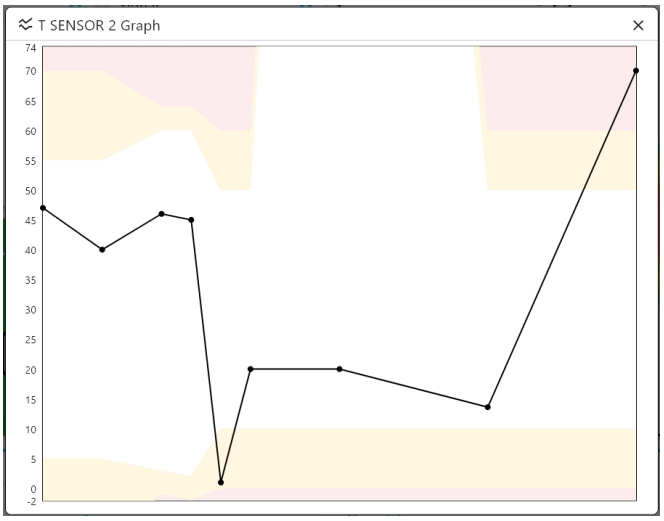

Figura 24 - Gráfico de linhas de um sensor presente num Widget Blueprint

Foi utilizada a mesma biblioteca e package que o "Widget GeoMap" mas, ao invés de um mapa é utilizada uma imagem no background e os sensores são marcadores/pins modificados. Neste *widget*, ao contrário do "Widget GeoMap", é utilizado o sistema de coordenadas "CRS.Simples" que utiliza coordenadas X e Y (diferente do habitual sistema de longitude e latitude) para posicionar os sensores.

Dashboard

O Dashboard é uma estrutura que divide a interface em colunas e linhas, formando uma grelha. Dentro dessa grelha, os itens, neste caso os *widgets*, podem ser posicionados e redimensionados de acordo com as preferências do utilizador. Esta abordagem traz várias vantagens:

- **Organização Visual** – Os *widgets* estão alinhados e ordenados, tornando o *dashboard* visualmente agradável;
- **Flexibilidade e Customização** – Permite uma personalização completa da interface onde cada utilizador pode criar uma disposição de *widgets* que se adeque aos seus objetivos;
- **Responsividade** – As posições e tamanhos dos *widgets* são guardadas consoante o tamanho do ecrã, ou seja, o utilizador pode guardar uma ordem de *widgets* com o ecrã com dimensões semelhantes à de um telemóvel e guardar outra ordem de *widgets* com o tamanho de ecrã superior (*tablet*, ecrã de computador, etc.).

As posições e os tamanhos dos *widgets* são guardados na BD. Existe uma tabela para cada aplicação, ou seja, mesmo que o tamanho do ecrã seja o mesmo em ambas as aplicações, o *layout* pode ser diferente.

Para reunir todas estas características, foi utilizada a biblioteca "Vue Grid Layout" para a aplicação "Vue" e a package "dashboard" para a aplicação Flutter. Na Figura 25 e na Figura 26 é possível ver o exemplo do mesmo *dashboard* em aplicações diferentes.

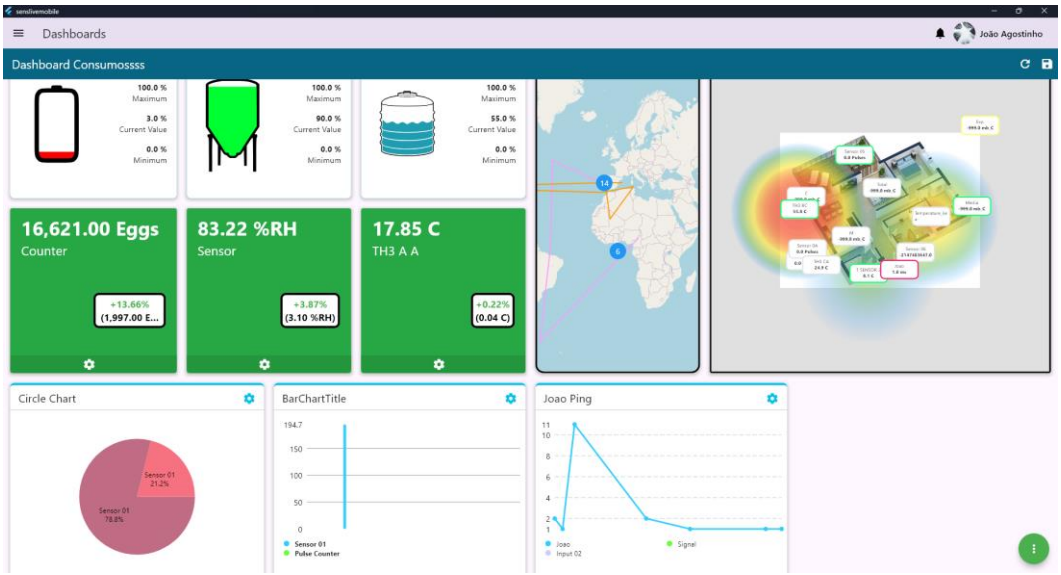

Figura 25 - Dashboard Flutter

Figura 26 - Dashboard Vue

Os *dashbords* possuem várias opções (Figura 27):

- "Lock Drag and Drop" – Permite bloquear a posição e o tamanho dos *widgets*;
- "More Information" – Mostrar o código QR do *dashboard*;

- "Replicate Dashboard" – Replicar o *dashboard* (não implementado);
- "Add Widget" – Abre o formulário para adicionar novos *widgets*;
- "Delete Dashboard" – Eliminar o *dashboard* e todos os seus *widgets*;
- "Edit Dashboard" – Editar o nome do *dashboard*;
- "Add Dashboard" – Adicionar um novo *dashboard*;

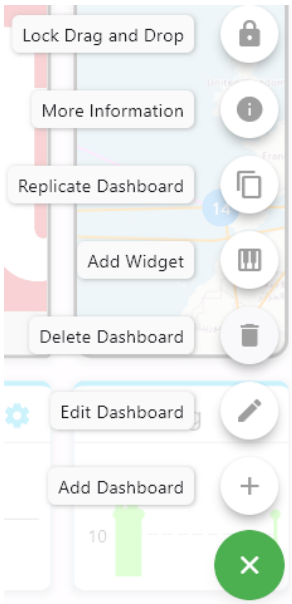

Figura 27 - Opções dos Dashboards

O código QR (Figura 28) serve para fazer login numa aplicação externa dedicada apenas à visualização de *dashboards*. Ao ler o código, é devolvido o id e o *token* (S*tring* aleatória hexadecimal de 30 caracteres) do *dashboard*.

Figura 28 - QR code de um Dashboard

CRUD

Para cada *widget* e *dashboard* foi realizado o CRUD (*Create, Read, Update, Delete*), ou seja, funcionalidades de criação de novos *widgets/dashboards*, visualização dos existentes, atualização das suas configurações e a sua eliminação.

A Figura 29 mostra o exemplo de um formulário de edição/eliminação de um "Widget Blueprint". Ao abrir o formulário, todos os dados atuais são pré-preenchidos de modo a ajudar o utilizador a ver os valores existentes sem precisar de os inserir novamente.

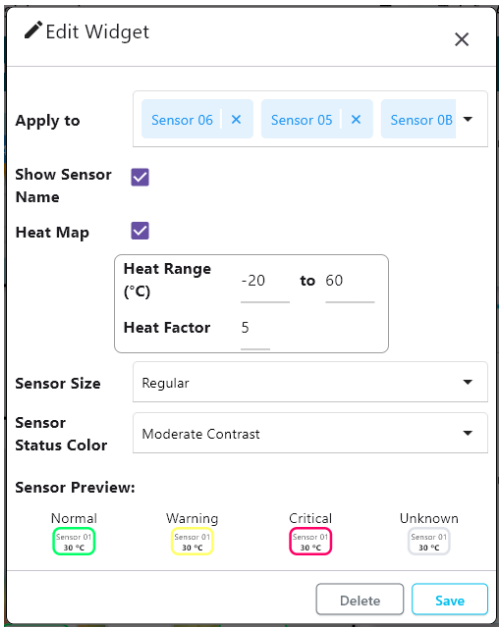

Figura 29 - Formulário de edição e eliminação de um Widget Blueprint

5.4. Testes

Durante o estágio, não foram realizados quaisquer tipos de testes formais (unitários, integração, sistema, etc.). A única forma de verificação foi uma testagem "informal" ao longo do desenvolvimento, onde cada funcionalidade foi testada manualmente para garantir que o código funcionava conforme o esperado. Esta abordagem foi uma decisão do supervisor e, embora tenha ajudado a identificar e corrigir alguns problemas, não proporciona a mesma confiabilidade que um conjunto de testes automáticos poderia oferecer.

5.5.Síntese

O projeto SensLIVE divide-se em duas aplicações: Flutter e Vue.js. Para ambas as aplicações foram realizadas as mesmas tarefas que incluem formulários de edição e criação de dados, exportação de dados para PDF e um sistema de *widgets* personalizáveis que permitem determinar o tamanho e a posição de cada *widget*. Este sistema foi a funcionalidade mais complexa e ocupou a maior parte do tempo do estágio. Ao longo do desenvolvimento foram criados componentes de modo a diminuir a replicação de código, utilizado um sistema de traduções, criadas tabelas na BD e criados modelos, controladores e rotas da API no Laravel. Não foram realizados quaisquer testes "formais" (unitários, integração, etc.) à plataforma, no entanto, foram utilizadas "boas práticas" de modo a garantir a qualidade, usabilidade e manutenção do *software*.

6. Análise crítica e proposta de melhorias

É fundamental refletir sobre as experiências e dificuldades enfrentadas ao longo do estágio pois permitem obter uma visão clara dos obstáculos e dos pontos a melhorar, quer a nível pessoal ou profissional. Esta reflexão é crucial para propor melhorias que visam a promover o desempenho global de uma equipa.

No primeiro subcapítulo são descritos os momentos mais desafiantes e as dificuldades e do estágio e, no segundo, é realizada uma análise crítica dos métodos de trabalho da empresa.

6.1. Dificuldades

Ao longo do estágio, foram encontradas algumas dificuldades. No início, foi necessária uma fase de adaptação ao código existente devido à sua extensão e complexidade. A primeira semana foi utilizada para estudar a estrutura e a lógica do projeto e, para auxiliar este "estudo" inicial, foram atribuídas tarefas simples que permitiram a adaptação ao ambiente de desenvolvimento.

Outra grande dificuldade foi a implementação de *widgets* complexos, em particular o "Widget Blueprint". Exigia a manipulação de uma imagem de fundo com sensores e valores sobrepostos que podiam ser arrastados. Implementar esta funcionalidade necessitou de tempo e de estudo para compreender a ferramenta utilizada assim como o sistema de coordenadas a utilizar.

Encontrar bibliotecas que atendessem às especificações dos *dashboards* (responsividade, guardar posições e mudar tamanhos dos *widgets*) foi outro grande desafio. Para a sua implementação, foi necessária uma vasta pesquisa e testagem de várias bibliotecas. A ferramenta que foi utilizada fornecia exemplos incorretos que não funcionavam como suposto, o que elevou a dificuldade do desenvolvimento.

O maior desafio foi a implementação de todas as funcionalidades em Flutter, uma tecnologia à qual não existia experiência. Além da complexidade de algumas funcionalidades, foi necessário tempo de adaptação a novas práticas de desenvolvimento e uma linguagem diferente (Dart).

6.2.Propostas de melhorias

Apesar da pouca experiência a nível profissional, foram identificados alguns problemas que podem afetar a eficiência e a organização do trabalho da equipa na empresa. Com base nessas observações, são propostas as seguintes melhorias que podem contribuir positivamente para aprimorar a qualidade do trabalho desenvolvido.

6.2.1. Implementação de uma Metodologia de Trabalho

Problema

Ausência de uma metodologia definida com base no projeto para guiar o seu desenvolvimento.

Solução

Adotar uma metodologia adequada para o projeto.

A escolha correta de uma metodologia auxilia a criação de *software*. Através da definição de etapas, tarefas e datas de entrega é possível estimar custos e recursos, aumentando a previsibilidade de um projeto. Outra vantagem é a melhoria da comunicação e colaboração entre os membros da equipa, assegurando que todos estejam alinhados com os objetivos do projeto.

6.2.2. Utilização do Git ou outro controlo de Versões

Problemas

Pouca utilização do controlo de versões e processo de "merge" feito por uma pessoa e com trabalho acumulado de várias.

Solução

O uso frequente de Git ou de outro controlo de versões é essencial para melhorar a organização e a colaboração no desenvolvimento de *software* pois promove o trabalho em paralelo através da criação de *branches* onde cada programador realiza o seu trabalho. Optar por estratégias como "Feature Branch" (criar uma *branch* por cada funcionalidade) oferece várias vantagens como:

- **Isolamento de Funcionalidades** – Cada funcionalidade é desenvolvida por um *branch* separado;

- **Entrega Contínua** – Funcionalidades completas e testadas podem ser integradas no *branch* principal, permitindo um ciclo de desenvolvimento ágil e eficiente;
- **Redução do esforço durante o "merge"** – As mudanças são feitas em *branches* separados e menores, o que reduz a complexidade do "merge";
- **Histórico de Desenvolvimento Claro** – O projeto mantém um histórico claro de alterações por funcionalidade.

6.3. Síntese

Existiram algumas dificuldades ao longo do estágio como a implementação de código numa linguagem nova e a criação de componentes complexos. Foram desafios que foram ultrapassados com sucesso e proporcionaram uma experiência de aprendizagem significativa.

Apesar da pouca experiência profissional, foram identificados alguns problemas no desenvolvimento de *software* como a ausência de uma metodologia e a pouca utilização de um controlo de versões.

7. Conclusão

Neste estágio, foi possível fazer parte das operações diárias do DT da CapTemp, colaborar com profissionais experientes e contribuir para a migração da plataforma SensLIVE. Foram desenvolvidos vários módulos para a plataforma onde se destacou a criação de um sistema de *widgets* responsivo e personalizável. Posto isto, é possível afirmar que os objetivos estabelecidos ao longo dos meses foram concluídos com sucesso e os desafios que vieram com eles ultrapassados.

A realização deste estágio num ambiente empresarial e com desafios do "mundo real" proporcionou uma oportunidade de aplicar e solidificar o conhecimento adquirido ao longo dos anos e serviu de preparação para os desafios futuros neste campo em constante evolução. Foi uma etapa importante no enriquecimento a nível pessoal e profissional e todo o trabalho desenvolvido poderá, sem dúvida, ajudar no crescimento da CapTemp.

Para o futuro, existem várias áreas promissoras que podem ser integradas com a plataforma para expandir as suas funcionalidades. Ao utilizar IA e ML é possível prever falhas e realizar a manutenção preditiva em sensores, arcas frigorificas, áreas de armazenamento, etc. Através de Realidade Virtual (RV) podem ser desenvolvidas simulações de cenários para testar diferentes condições ambientais como, por exemplo, testar diferentes temperaturas num local de armazenamento de comida. Por fim, a integração de Realidade Aumentada (RA) pode ajudar na instalação de sensores remotamente.

A integração destas áreas abrirá novas possibilidades inovação que garantem à plataforma a permanência relevante e competitiva no mercado empresarial que se encontra em constante evolução.

8. Referências

[1] R. P. Singh, M. Javaid, A. Haleem e R. Suman, "Internet of things (IoT) applications to fight against COVID-19 pandemic," *Diabetes and Metabolic Syndrome: Clinical Research and Reviews,* vol. 14, nº 4, pp. 521-524, 2020.

[2] A. S. Gillis, "TechTarget," [Online]. Available: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT. [Acedido em 2024 1 12].

[3] "Estado da Arte," Wikipédia, [Online]. Available: https://pt.wikipedia.org/wiki/Estado_da_arte. [Acedido em 26 01 2024].

[4] "Industry 4.0: The Future of Manufacturing," SAP, [Online]. Available: https://www.sap.com/products/scm/industry-4-0/what-is-industry-4-0.html. [Acedido em 08 03 2024].

[5] . S. Ruj, . S. D. Bit e J. Sengupta, "A Comprehensive Survey on Attacks, Security Issues and Blockchain Solutions for IoT and IIoT," *Journal of Network and Computer Applications,* vol. 149, 2020.

[6] S. Munirathinam, "Chapter Six - Industry 4.0: Industrial Internet of Things (IIOT)," em *Advances in Computers*, 2020, pp. 129-164.

[7] E. Sisinni, A. Saifullah, S. Han, U. Jennehag e M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," *IEEE Transactions on Industrial Informatics,* vol. 14, nº 11, pp. 4724 - 4734, 2018.

[8] H. Xu, W. Yu, D. Griffith e N. Golmie, "A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective," *IEEE Access,* vol. 6, 2018.

[9] P. Jayalaxmi, R. Saha, G. Kumar, N. Kumar e T.-H. Kim, "A Taxonomy of Security Issues in Industrial Internet-of-Things: Scoping Review for Existing

Solutions, Future Implications, and Research Challenges," *IEEE Access,* vol. 9, 2021.

[10] J. Leng, Z. Chen, Z. Huang , X. Zhu, H. Su, Z. Lin e D. Zhang, "Secure Blockchain Middleware for Decentralized IIoT towards Industry 5.0: A Review of Architecture, Enablers, Challenges, and Directions," *Machines,* vol. 10, nº 10, p. 858, 2022.

[11] "Modelo OSI | Cloudflare," Cloudflare, [Online]. Available: https://www.cloudflare.com/pt-br/learning/ddos/glossary/open-systems-interconnection-model-osi/. [Acedido em 09 09 2024].

[12] "LoRa vs LoRaWAN," LoRa Alliance, [Online]. Available: https://resources.lora-alliance.org/home/lora-and-lorawan. [Acedido em 09 09 2024].

[13] "LORAWAN - O QUE É?," ZenzorControl, [Online]. Available: https://zenzorcontrol.pt/pt/lorawan-o-que-e. [Acedido em 03 06 2024].

[14] "INTRODUCTION TO MODBUS TCP/IP," Acromag, [Online]. Available: https://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf. [Acedido em 09 09 2024].

[15] S. Sharma, "Understanding Modbus TCP-IP: An In depth Exploration," WEVOLVER, 20 06 2023. [Online]. Available: https://www.wevolver.com/article/modbus-tcp-ip. [Acedido em 03 06 2024].

[16] F. Dewanta, B. Y. Yustiarini e B. I. R. Harsritanto, "A study of secure communication scheme in MQTT: TLS vs AES cryptography," *JURNAL INFOTEL,* vol. 14, nº 4, p. 271, 2022.

[17] "MQTT: The Standard for IoT Messaging," MQTT, [Online]. Available: https://mqtt.org/. [Acedido em 03 06 2024].

[18] "What is OSI Model | 7 Layers Explained | Imperva," Imperva, [Online]. Available: https://www.imperva.com/learn/application-security/osi-model/. [Acedido em 09 09 2024].

[19] "O que é HTTPS? | Cloudflare," Cloudflare, [Online]. Available: https://www.cloudflare.com/pt-br/learning/ssl/what-is-https/. [Acedido em 03 06 2024].

[20] F. CERTI, "IIoT: o que é e qual a importância para a Indústria 4.0," Linkedin, 27 03 2023. [Online]. Available: https://pt.linkedin.com/pulse/iiot-o-que-%C3%A9-e-qual-import%C3%A2ncia-para-ind%C3%BAstria-40-fundacao-certi. [Acedido em 17 06 2024].

[21] "A era da Internet Industrial das Coisas (IIoT) - Mecalux.pt," Mecalux, 02 08 2022. [Online]. Available: https://www.mecalux.pt/blog/iiot-internet-industrial-das-coisas. [Acedido em 01 09 2024].

[22] "SkyFoundry," SkyFoundry, [Online]. Available: https://skyfoundry.com/. [Acedido em 26 01 2024].

[23] "SkyFoundry," SkyFoundry, [Online]. Available: https://skyfoundry.com/product. [Acedido em 26 01 2024].

[24] "AWS IoT Device Management," Amazon Web Services, [Online]. Available: https://aws.amazon.com/pt/iot-device-management/. [Acedido em 26 01 2024].

[25] "AWS IoT Core," Amazon Web Services, [Online]. Available: https://aws.amazon.com/pt/iot-core/. [Acedido em 26 01 2024].

[26] "Recursos do AWS IoT Device Management – Amazon Web Services," Amazon, [Online]. Available: https://aws.amazon.com/pt/iot-device-management/features/. [Acedido em 26 06 2024].

[27] "Azure IoT Central," Microsoft, [Online]. Available: https://azure.microsoft.com/pt-pt/products/iot-central. [Acedido em 02 02 2024].

[28] D. Betts, T. Hopwood e K. Sekhar , "Quotas and limits," Microsoft, 26 10 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/iot-central/core/concepts-quotas-limits. [Acedido em 02 02 2024].

[29] "Azure IoT Hub," Microsoft, [Online]. Available: https://azure.microsoft.com/pt-pt/products/iot-hub. [Acedido em 02 02 2024].

[30] "Criar e gerenciar painéis do Azure IoT Central - Azure IoT Central," Microsoft, [Online]. Available: https://learn.microsoft.com/pt-br/azure/iot-central/core/howto-manage-dashboards. [Acedido em 26 06 2024].

[31] "XML encryption filters," Axway, [Online]. Available: https://docs.axway.com/bundle/axway-open-docs/page/docs/apim_policydev/apigw_polref/encryption_xml/index.html. [Acedido em 07 09 2024].

[32] "Wireless AiroLog Sensor and Airolog Syncronization Technology!," CapTemp, [Online]. Available: https://www.captemp.com/software-senslive. [Acedido em 11 06 2024].

[33] "SkyFoundry," 09 2023. [Online]. Available: https://skyfoundry.com/file/542/SkyFoundry-Insider-August-2023-The-Data--Decarbonization-Connectionpdf.pdf. [Acedido em 02 02 2024].

[34] "Amazon Web Systems," 2024. [Online]. Available: https://docs.aws.amazon.com/pdfs/iot/latest/developerguide/iot-dg.pdf. [Acedido em 02 02 2024].

[35] D. Betts, A. Sérgio Azevedo e A. Jenks, "Device implementation and best practices for IoT central," Microsoft, 06 07 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/iot-central/core/concepts-device-implementation. [Acedido em 02 02 2024].

[36] L. Minvielle, "The 7 Most Popular Backend Frameworks for Developers," WeAreDevelopers, 17 03 2024. [Online]. Available: https://awari.com.br/os-5-

melhores-frameworks-de-backend-para-desenvolvimento-rapido/. [Acedido em 01 09 2024].

[37] "The Laravel Framework," GitHub, [Online]. Available: https://github.com/laravel. [Acedido em 09 02 2024].

[38] A. Jalli, "Waht is Laravel?," builtin, 21 12 2022. [Online]. Available: https://builtin.com/software-engineering-perspectives/laravel. [Acedido em 23 02 2024].

[39] G. Renganathan, "What are some companies that use Laravel as their backend framework for web development?," Quora, 2022. [Online]. Available: https://www.quora.com/What-are-some-companies-that-use-Laravel-as-their-backend-framework-for-web-development. [Acedido em 23 02 2023].

[40] "Django overview | Django," Django, [Online]. Available: https://www.djangoproject.com/start/overview/. [Acedido em 01 09 2024].

[41] "What is ASP.NET?," Microsoft, [Online]. Available: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet. [Acedido em 23 02 2024].

[42] C. Sheladiya, "LIST OF OUTLINES WHY PEOPLE STILL USE ASP.NET WEB FORMS IN 2021?," MetizSoft, 04 08 2021. [Online]. Available: https://www.metizsoft.com/blog/why-people-still-use-asp-dot-net-web-forms-in-2021. [Acedido em 23 02 2024].

[43] B. S. LTD, "The Pros and Cons of Laravel," Linkedin, 15 08 2023. [Online]. Available: https://www.linkedin.com/pulse/pros-cons-laravel-bsuperior-system-ltd-. [Acedido em 04 06 2024].

[44] J. Protasiewicz, "Top 10 Django Apps Examples," Netguru, 26 05 2024. [Online]. Available: https://www.netguru.com/blog/django-apps-examples. [Acedido em 01 09 2024].

[45] M. Deery, "9 Pros and Cons of the Django Framework: A Coder's Guide," CareerFoundry, 30 08 2023. [Online]. Available:

https://careerfoundry.com/en/blog/web-development/django-framework-guide/. [Acedido em 01 09 2024].

[46] "ASP.NET - Advantages & Disadvantages," Eternitech, [Online]. Available: https://eternitech.com/technologies/asp-net/. [Acedido em 04 06 2024].

[47] L. Minvielle, "The Best 7 Frontend Frameworks for Developers in 2024," WeAreDevelopers, 17 03 2024. [Online]. Available: https://www.wearedevelopers.com/magazine/best-frontend-frameworks-for-developers. [Acedido em 28 08 2024].

[48] "Vue.js," Vue.js, [Online]. Available: https://vuejs.org/guide/introduction.html. [Acedido em 09 02 2024].

[49] "Top 10 startups usando Vue," back4app, [Online]. Available: https://blog.back4app.com/pt/startups-usando-vue/. [Acedido em 23 02 2024].

[50] "altexsoft," 28 09 2022. [Online]. Available: https://www.altexsoft.com/blog/pros-and-cons-of-vue-js/. [Acedido em 09 02 2024].

[51] "Top 32 Sites Built With ReactJS," Medium, 10 06 2016. [Online]. Available: https://medium.com/@coderacademy/32-sites-built-with-reactjs-172e3a4bed81. [Acedido em 23 02 2024].

[52] "The Pros and Cons of Using React Today," The New Stack, 04 12 2023. [Online]. Available: https://thenewstack.io/the-pros-and-cons-of-using-react-today/. [Acedido em 09 02 2024].

[53] D. Fleury, "12 Websites Built by Angular: Microsoft, PayPal, Samsung, and More," Trio, 01 03 2022. [Online]. Available: https://www.trio.dev/blog/companies-use-angular. [Acedido em 23 02 2024].

[54] "AngularJS Pros and Cons: A Quick Guide | BootstrapDash," BootstrapDash, 12 04 2023. [Online]. Available: https://www.bootstrapdash.com/blog/angularjs-pros-cons. [Acedido em 09 02 2024].

[55] "Angular JS: Developer Guide: Introduction," Angular JS, [Online]. Available: https://docs.angularjs.org/guide/introduction. [Acedido em 09 02 2024].

[56] F. Magalhães, "Explorando o Desenvolvimento Cross-Platform," Medium, 27 06 2023. [Online]. Available: https://medium.com/@felipesses/explorando-o-desenvolvimento-cross-platform-7a3b0f88f4ff. [Acedido em 14 06 2024].

[57] F. Cunha, "Framework de desenvolvimento: quais os mais usados?," Mestres da Web, [Online]. Available: https://www.mestresdaweb.com.br/tecnologias/framework-de-desenvolvimento-quais-os-mais-usados. [Acedido em 01 09 2024].

[58] "O que é o Flutter? - Explicação sobre a aplicação Flutter - AWS," Amazon, [Online]. Available: https://aws.amazon.com/pt/what-is/flutter/. [Acedido em 04 06 2024].

[59] "Out-of-Tree Platforms," React Native, 22 04 2024. [Online]. Available: https://reactnative.dev/docs/out-of-tree-platforms. [Acedido em 17 06 2024].

[60] "React Native · Learn once, write anywhere," React Native , [Online]. Available: https://reactnative.dev/. [Acedido em 17 06 2024].

[61] "Ionic Framework - The Cross-Platform App Development Leader," Ionic Framework, [Online]. Available: https://ionicframework.com/. [Acedido em 17 06 2024].

[62] K. Beladiya, "What is Flutter App Development? Advantages & Drawbacks of Flutter," The One Technologies, 11 12 2023. [Online]. Available: https://theonetechnologies.com/blog/post/flutter-mobile-application-development. [Acedido em 17 06 2024].

[63] "1. What Is React Native? - Learning React Native [Book]," O'Reilly, [Online]. Available: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html. [Acedido em 17 06 2024].

[64] I. Puzhay, “Advantages and Disadvantages of Ionic Development | UKAD,” UKAD, [Online]. Available: https://ukad-group.com/blog/6-pros-and-3-cons-of-ionic-development/. [Acedido em 17 06 2024].

[65] “Oracle System Properties,” DB-Engines, 06 2024. [Online]. Available: https://db-engines.com/en/system/Oracle. [Acedido em 07 06 2024].

[66] “What is a Relational Database | Oracle,” Oracle, [Online]. Available: https://www.oracle.com/database/what-is-a-relational-database/. [Acedido em 07 06 2024].

[67] “PostgreSQL System Properties,” DB-Engines, [Online]. Available: https://db-engines.com/en/system/PostgreSQL. [Acedido em 07 06 2024].

[68] H. Singh, “Oracle Database Advantages, Disadvantages and Features [Guide 2024],” NineHertz, 24 01 2024. [Online]. Available: https://theninehertz.com/blog/advantages-of-using-oracle-database. [Acedido em 07 06 2024].

[69] “Histórias de sucesso de clientes da Cloud Infrastructure | Oracle Brasil,” Oracle, [Online]. Available: https://www.oracle.com/br/cloud/customers/. [Acedido em 07 06 2024].

[70] J. Hurley, “Secure File Sharing & FTP Hosting for Enterprise | SmartFile,” Smartfile, 03 06 2021. [Online]. Available: https://www.smartfile.com/blog/the-pros-and-cons-of-mysql. [Acedido em 07 06 2024].

[71] R. Peterson, “What is PostgreSQL? Introduction, Advantages & Disadvantages,” GURU99, 16 03 2024. [Online]. Available: https://www.guru99.com/introduction-postgresql.html. [Acedido em 07 06 2024].

[72] J. Romanowski, “ Major Companies Using PostgreSQL: Purposes & Examples | LearnSQL.com,” LearnSQL, 19 05 2020. [Online]. Available: https://learnsql.com/blog/companies-that-use-postgresql-in-business/. [Acedido em 07 06 2024].

[73] "Laravel - The PHP Framework for Web Artisans," Laravel, [Online]. Available: https://laravel.com/. [Acedido em 29 05 2024].

[74] P. Fernandes e R. Neves, "ATEMPHAR 2022 - Público," Leiria, 2022.

[75] "Introduction | Vue.js," Vue.js, [Online]. Available: https://vuejs.org/guide/introduction.html. [Acedido em 04 06 2024].

[76] "PostgeSQL: About," PostgreSQL, [Online]. Available: SGBD objeto-relacional . [Acedido em 06 06 2024].

[77] "Entenda a diferença entre banco de dados relacional e orientado a objetos," Rock content, 25 06 2021. [Online]. Available: https://rockcontent.com/br/blog/diferenca-entre-banco-de-dados-relacional-e-orientado-a-objetos/. [Acedido em 06 06 2024].

[78] P. Pandit, X. Dias e H. Parekh, "Sessões vs tokens: como escolher o melhor método de autenticação," Linkedin, 15 03 2024. [Online]. Available: https://www.linkedin.com/advice/0/what-difference-between-session-token-skills-programming-umcpe. [Acedido em 05 06 2024].

[79] "W3Schools," [Online]. Available: https://www.w3schools.com/whatis/whatis_react.asp. [Acedido em 02 09 2024].

[80] "What is Java technology and why do I need it?," Java, [Online]. Available: https://www.java.com/en/download/help/whatis_java.html. [Acedido em 23 02 2024].

[81] C. McKenzie, "Real world Java applications," TheServerSide, 04 03 2023. [Online]. Available: https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Java-Applications-Uses-Types-Games-Best-Apps-Minecraft-Android-Mobile-Desktop-IoT. [Acedido em 23 02 2024].

[82] "Advantages and disadvantages of Java," JavatPoint, [Online]. Available: https://www.javatpoint.com/advantages-and-disadvantages-of-java. [Acedido em 04 06 2024].

Apêndices

Apêndice 1 - Exemplo de duas páginas de um PDF gerado

COMPANY

History of Alerts

Time Interval: 13/12/2023 17:30 - 24/12/2023 17:30

Local	Alerts
Bruno F	0
Demo Abacus	0
Office Server	6

Local: Bruno F

Local	Device	Sensor	Alerts	Total Duration
Bruno F	00- Nidus-IT+	TH3 A A	No Occurrences	No Occurrences
Bruno F	NB-IoT 3592151032818620	Battery	No Occurrences	No Occurrences
Bruno F	00-NidusINPUT	TH3 0Y	No Occurrences	No Occurrences
Bruno F	00-NidusINPUT	Contador Geral	No Occurrences	No Occurrences
Bruno F	NB-IoT 3592151032818620	Sensor	No Occurrences	No Occurrences
Bruno F	NB-IoT 3592151032818620	Sensor	No Occurrences	No Occurrences
Bruno F	NB-IoT 3592151032818620	Sensor	No Occurrences	No Occurrences
Bruno F	00- Nidus-IT+	TH3 BY	No Occurrences	No Occurrences
Bruno F	00- Nidus-IT+	TH3 AZ	No Occurrences	No Occurrences
Bruno F	00- Nidus-IT+	TH3 BA	No Occurrences	No Occurrences
Bruno F	NB-IoT 3527530930487570	Battery	No Occurrences	No Occurrences
Bruno F	NB-IoT 3527530930487570	Signal	No Occurrences	No Occurrences
Bruno F	NB-IoT 3527530930487570	Sensor	No Occurrences	No Occurrences
Bruno F	NB-IoT 3592151032818620	Sensor	No Occurrences	No Occurrences
Bruno F	NB-IoT 3592151032818620	Sensor	No Occurrences	No Occurrences
Bruno F	NB-IoT 3592151032818620	Sensor	No Occurrences	No Occurrences
Bruno F	00- Nidus-IT+	Sensor 01	No Occurrences	No Occurrences
Bruno F	00- Nidus-IT+	Signal	No Occurrences	No Occurrences

Made by João Agostinho from CapTemp at 17:30:33 Page 1 of 27

Office Server	Kea Tracker E7:C5:69:CB:C7:73	Temperature	No Occurrences	No Occurrences
Office Server	[0] Nidus-W airOs	Sensor 01	No Occurrences	No Occurrences
Office Server	Nidus-C+ [DEMO]	TH3 AY	No Occurrences	No Occurrences
Office Server	Nidus-C+ [DEMO]	Signal	No Occurrences	No Occurrences
Office Server	Nidus-C+ [DEMO]	Battery 01	No Occurrences	No Occurrences
Office Server	Nidus-C+ [DEMO]	Signal	No Occurrences	No Occurrences
Office Server	Nidus-IT [DEMO]	autotemp	1	01:26:04
Office Server	[000] NidusW	Battery	No Occurrences	No Occurrences
Office Server	Nidus-IT [DEMO]	[A] I2	No Occurrences	No Occurrences
Office Server	[000] airO Test	Sensor 01	No Occurrences	No Occurrences
Office Server	[000] NidusW	Signal	No Occurrences	No Occurrences
Office Server	Nidus-IT - PT100	TH3 EA	No Occurrences	No Occurrences
Office Server	[000] NidusW	Battery	No Occurrences	No Occurrences
Office Server	Kea Tracker E7:C5:69:CB:C7:73	Humidity	No Occurrences	No Occurrences
Office Server	Nidus-IT - PT100	TH3 DY	No Occurrences	No Occurrences
Office Server	Nidus-IT - PT100	TH3 JA	No Occurrences	No Occurrences
Office Server	[0] Nidus-W airOs	Input 01	No Occurrences	No Occurrences
Office Server	[0] Nidus-W airOs	Battery	No Occurrences	No Occurrences
Office Server	[0] Nidus-W airOs	Signal	No Occurrences	No Occurrences
Office Server	[0] Nidus-W airOs	Input 02	No Occurrences	No Occurrences
Office Server	Nidus-IT - PT100	Signal	No Occurrences	No Occurrences
Office Server	Nidus-C+ [DEMO]	Battery	No Occurrences	No Occurrences
Office Server	Nidus-IT [DEMO]	autoinput	No Occurrences	No Occurrences
Office Server	[000] airO Test	Sensor 01	No Occurrences	No Occurrences
Office Server	Nidus-IT - PT100	TH3 BZ	No Occurrences	No Occurrences
Office Server	[000] airO Test	Battery	No Occurrences	No Occurrences
Office Server	Nidus-IT [DEMO]	[IR33] Probe1 Val	No Occurrences	No Occurrences
Office Server	[000] airO Test	Sensor 03	No Occurrences	No Occurrences
Office Server	Nidus-IT [DEMO]	[UEM80_4D] Corrente 1	No Occurrences	No Occurrences
Office Server	Nidus-C+ [DEMO]	Input 01	No Occurrences	No Occurrences
Office Server	Nidus-IT - PT100	Battery	No Occurrences	No Occurrences