IPS Instituto Politécnico de Setúbal

Escola Superior de Tecnologia de Setúbal

TIAGO MIGUEL MEDEIRA CARDOSO

INFRAESTRUTURA PARA DINAMIZAR OS PROCESSOS DA OFICINA LUBAN

Dissertação/Trabalho de Projeto/Relatório de Estágio submetida como requisito parcial para obtenção do grau de **Mestre em Engenharia de Software**

Júri

Presidente: Profª Doutora Rossana Henriques dos Santos, ESTSetúbal/IPS

Orientador: Prof. Doutor José Moinhos Cordeiro, ESTSetúbal/IPS

Vogal: Prof. Doutor Claúdio dos Santos Sapateiro, ESTSetúbal/IPS

Dezembro, 2023

Agradecimentos

À minha família por todo o apoio incondicional que me deram ao longo desta caminhada, com um obrigado especial aos meus pais por todos os esforços que realizaram para me proporcionarem a oportunidade de estar a terminar mais uma fase da minha vida académica.

À minha namorada por toda a sua força e ânimo dados durante esta caminhada, sempre presente para me apoiar nos momentos menos bons, não poderia ter escolhido melhor pessoa para me acompanhar nestes momentos.

Aos meus colegas e amigos por todos os momentos que passamos juntos durante os anos de mestrado.

Ao meu orientador de estágio, Professor Doutor José Cordeiro, que esteve sempre disponível durante todo o trabalho para me ajudar a esclarecer tudo o que fosse necessário, facilitando imensamente o processo no seu todo.

Ao meu coorientador de estágio, Professor José Lucas, responsável pela Luban e que me proporcionou a oportunidade de realizar este trabalho no contexto da sua oficina e do seu trabalho, para além de ter sido sempre acessível para responder a dúvidas e questões no âmbito do trabalho.

Ao corpo docente da escola Superior de Tecnologia de Setúbal que me aconselharam e forneceram os meios para que hoje eu pudesse estar aqui, assim como, a toda a instituição do Politécnico de Setúbal.

Resumo

A Luban é uma oficina, presente no Instituto Politécnico de Setúbal, que tem vindo a desenvolver as suas competências educacionais no âmbito da automação e controlo, através da simulação de sistemas e processos industriais semelhantes aos usados em ambientes reais. No entanto, existem aspirações de continuar a crescer e a preparar a oficina para suportar cada vez mais conceitos utilizados na indústria 4.0, para tal, existem alguns problemas que necessitam de ser resolvidos com o suporte de conhecimentos utilizados no desenvolvimento de aplicações web e para a *"cloud"*. Existe a necessidade de criar uma infraestrutura que seja capaz de responder a vários dos problemas atualmente existentes na Luban, principalmente na gestão dos seus ativos e interações, em tempo real, com os mesmos. Para responder a muitas destas necessidades foi sugerida uma arquitetura de referência de alto nível capaz de suportar diversos elementos lógicos independentes que comunicam entre si e que conseguem, do ponto de vista teórico, resolver os vários problemas mencionados. A implementação abstrata desta infraestrutura preocupou-se em responder aos diversos tópicos suportando as suas razões em implementações concretas de arquiteturas de software para os diferentes elementos lógicos que fizessem sentido para o contexto da Luban, suportada também por aprendizagens obtidas com o desenvolvimento de um protótipo técnico. A flexibilidade desta solução final acabou por ser um dos fatores mais importantes na tomada de decisões.

Palavras-chave: Infraestrutura, Indústria 4.0, Luban, Arquiteturas de Software, Modbus

Abstract

Luban is a workshop at the Polytechnic Institute of Setúbal that has been developing its educational skills in the field of automation and control by simulating industrial systems and processes like those used in real environments. However, there are aspirations to continue growing and preparing the workshop to support more and more concepts used in industry 4.0. To this end, there are some problems that need to be solved with the support of knowledge used in the development of web and cloud applications. There is a need to create an infrastructure that can respond to several of the problems that currently exist at Luban, mainly in the management of its assets and real-time interactions with them. To meet many of these needs, a high-level reference architecture has been suggested that can support several independent logical elements that communicate with each other and that can, from a theoretical point of view, solve the various problems mentioned. The abstract implementation of this infrastructure was concerned with responding to the various topics by supporting its reasons in concrete implementations of software architectures for the different logical elements that made sense for Luban's context, while being supported by lessons learned from the development of a technical prototype. The flexibility of this final solution turned out to be one of the most important factors in the decision-making process.

Keywords: Infrastructure, Industry 4.0, Luban, Software Architectures, Modbus

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

API	*Application Programming Interface*
CRUD	*Create Read Update Delete*
CSR	*Client Side Rendering*
EDA	*Event-driven Architecture*
ESTS	*Escola Superior de Tecnologia de Setúbal*
FC	*Function Codes*
HMI	*Human-machine Interface*
IPS	*Instituto Politécnico de Setúbal*
IIOT	*Industrial Internet of Things*
IOT	*Internet of Things*
IIRA	*Industrial Internet Reference Architecture*
IIS	*Industrial Internet System*
ISA	*International Standard for International Society of Automation*
ISR	*Incremental Static Regeneration*
JWT	*Json Web Token*
LSP	*Language Server Protocol*
MQTT	*MQ Telemetry Transport*
ORM	*Object Relational Mapping*
RAMI	*Reference Architecture Model Industry*
REST	*Representational State Transfer*
RFID	*Radio Frequency Identification*
SCADA	*Supervisory Control and Data Acquisition*

SEO	*Search Engine Optimization*
SPA	*Single Page Application*
SSG	*Static Site Generation*
SSR	*Server-Side Rendering*
PC	*Personal Computer*
PLC	*Programmable Logic Controller*
TCP/IP	*Transmission Control Protocol / Internet Protocol*

Capítulo 1
Introdução

Este capítulo tem como objetivo fornecer uma visão geral dos temas e tópicos que serão abordados ao longo do documento. Será realizada uma contextualização introdutória da oficina Luban, seguindo-se a apresentação dos problemas identificados, que servem de fundamento para a investigação e desenvolvimento técnico. Além disso, serão apresentados os objetivos deste estudo, delineando algumas das metas a serem alcançadas, as opções tomadas para a organização do trabalho e respetiva metodologia de desenvolvimento, dando também algum destaque à estrutura do documento, que auxiliará na organização e apresentação das informações de forma clara e coerente.

Com este capítulo de introdução, espera-se estabelecer uma base sólida para o desenvolvimento dos temas discutidos ao longo do relatório.

1.1. Oficina Luban e Indústria 4.0

Este projeto nasce com o propósito de criar um sistema informático e respetiva estrutura para suportar e agilizar diversas necessidades na oficina Luban, nomeadamente relacionadas com a sua vontade de informatizar muitas das suas atividades e entidades. Instalada na Escola Superior de Tecnologia de Setúbal (ESTSetúbal) do Instituto Politécnico de Setúbal (IPS), com o foco na oferta de um espaço único, agregador de várias áreas disciplinares, em torno do paradigma da Indústria 4.0.

A transição e adaptação das indústrias aos paradigmas para acomodar as integrações de novas tecnologias no âmbito industrial impõem desafios que requerem respostas, apesar de a Luban não ser um ambiente fabril tradicional, as simulações oferecidas no seu espaço pretendem transpor os conhecimentos de um ambiente real e com isso também alguns dos seus problemas. A transição para esta nova era, onde existe uma transformação integral da produção industrial por meio da aglomeração da tecnologia digital e da internet com a indústria convencional, conduzindo a uma conexão total entre todos os equipamentos associados às operações de fabrico, sendo possível monitorizar todos os processos de forma digital.

Atualmente a oficina está equipada de forma a suportar a simulação de diferentes processos industriais, aplicados em diversas estações compostas por diversos componentes físicos e permite aos interessados trabalhar com diversas células de fabrico discreto, semelhantes às existentes na indústria, que utilizam equipamentos de última geração. Esta é reconhecida como uma ferramenta versátil que tem sido amplamente utilizada tanto por estudantes como por professores interessados no desenvolvimento de competências relacionadas com a indústria 4.0 e afins, porém, é visível o potencial que a oficina tem quando conseguir adaptar totalmente os seus processos aos novos paradigmas das indústrias com base na tecnologia digital.

O objetivo final deste trabalho passar por tentar criar uma arquitetura final que possa servir como

uma referência de alto nível para uma futura implementação técnica que consiga conectar todas as entidades físicas entres si, estando qualquer informação sobre estas, acessível através de dispositivos digitais, de preferência num contexto web.

1.2. Problemas Identificados

Existem diversos desafios que surgem naturalmente devido ao facto de a oficina Luban ser um meio que foi inaugurado recentemente, ainda é necessário aprimorar a gestão de seus diversos ativos em vários aspetos, incluindo a documentação associada aos processos, o acesso às informações sobre o estado dos processos e a interação com eles. Diante disso, e olhando para as dimensões atuais da oficina e dos seus processos, centralizar a gestão desses diversos ativos proporcionará um aumento da eficiência nas atividades do dia-a-dia e abre as portas para futuros melhoramentos técnicos.

No entanto, é necessário planear essa gestão de ativos com base no desenvolvimento de uma infraestrutura que permita a aplicação fácil de operações sobre os mesmos, as informações relacionadas aos diferentes recursos e toda a componente de comunicação. Dada a distinção entre os diversos processos, essa estrutura deve proporcionar flexibilidade para a personalização das interfaces associadas ao controlo e monitorização desses processos.

Um requisito extremamente importante é a necessidade de interagir com alguns processos em tempo real, o que motiva uma investigação de diferentes protocolos de comunicação para determinar as melhores abordagens, visando garantir uma comunicação rápida e alinhada com os requisitos pretendidos. Dito isto, é essencial explorar e analisar diversos protocolos de comunicação disponíveis, identificar aqueles que mais se adequam às necessidades da Luban e implementar uma comunicação em tempo real entre os sistemas e processos da oficina. Essa investigação permitirá selecionar e implementar as melhores soluções de comunicação que atendam às necessidades específicas da oficina, garantindo a troca eficiente de informações e o controlo adequado dos processos em tempo real.

Ao considerar essas diferentes questões relacionadas com a gestão dos ativos, personalização das interfaces e comunicação em tempo real, a oficina Luban poderá aprimorar as suas operações, otimizar o seu fluxo de trabalho e melhorar a eficiência geral. Isso contribuirá para um espaço de trabalho mais ágil, produtivo e capaz de responder às necessidades do mercado de forma eficaz. É notório ter em consideração que todos os benefícios aplicados à Luban contribuem para um desenvolvimento educacional de todos os alunos e professores que frequentarem a mesma.

1.3. Objetivos

Como forma de prestar resposta aos problemas identificados na Luban são apresentados um conjunto de objetivos que visam a ser realizados e que servem de suporte para muitos dos fundamentos que a arquitetura de referência final deve conter.

Estes objetivos podem ser agrupados sob um objetivo principal, **definir e implementar uma Arquitetura de Referência para a Luban**.

Ao analisarmos este objetivo e o partirmos em diferentes etapas vamos encontrar as seguintes necessidades:

- Entender os objetivos e os problemas atuais da Luban;
- Perceber a estrutura lógica da Luban e o respetivo vocabulário utilizado no seu contexto;
- Perceber como a estrutura física da Luban está organizada, tanto ao nível dos processos como dos diversos equipamentos;
- Perceber as necessidades associadas à monitorização e controlo dos diversos processos, equipamentos e afins;
- Perceber e investigar os diversos mecanismos de comunicação existentes na Luban;
- Investigar Arquiteturas de Referência no espaço da Indústria 4.0 para perceber o que é praticado;
- Investigar Arquiteturas de Referência e contextualizar os níveis de abstração para os problemas específicos da Luban;
- Investigar a exequibilidade dos requisitos de um ponto de vista técnico e perceber limitações e possíveis pontos de flexibilidade onde existe espaço para crescimento, sugestões e adaptações;

Com este entendimento vamos definir o caminho para o sistema de informação final, definindo por agora a arquitetura de referência a utilizar.

1.4. Metodologia e Contribuição

Ao longo de todo o processo de desenvolvimento deste trabalho, optou-se por não seguir nenhuma metodologia especifica dado que o trabalho do ponto de vista temporal teve bastantes atrasos e contratempos o que levou a que o desenvolvimento tivesse inúmeras pausas dificultando a decisão de escolher uma metodologia especifica. Assim sendo, ao invés de escolher e seguir uma única metodologia de trabalho, tentou-se adaptar o fluxo de trabalho consoante o progresso que ia sendo possível fazer, seguindo, sempre que fizesse sentido no contexto do projeto, algumas boas práticas retiradas de algumas metodologias, especialmente a metodologia Iterativa e Incremental e a metodologia baseada em protótipos. Foram definidos objetivos a atingir e esta ideia repetia-se até ser possível ter um elemento de entrega que proporciona-se uma discussão teórica e técnica para os passos seguintes. Dito isto, este trabalho focou-se imenso na ideia de juntar duas realidades diferentes, automação e controlo e engenharia de software, para tal, foi necessário criar várias pontes para diferentes conceitos teóricos e muitas dessas pontes foram constituídas graças ao desenvolvimento de um protótipo, dentro de uma ótica iterativa à medida que diversos requisitos eram validados de acordo com a sua exequibilidade. A junção de conceitos das duas metodologias mencionadas permitiu que o fluxo de trabalho seguisse uma lógica coerente para atingir os objetivos finais delineados.

Numa ótica de contribuição, nomeadamente no valor que a realização deste trabalho traria para a Luban, podemos focar-nos meramente na resposta aos problemas identificados. Estamos perante um ambiente ambicioso que quer prosperar no contexto da indústria 4.0, desse ponto de vista está inteiramente subtendido aquilo que os avanços tecnológicos significam para esta indústria, com isso, a grande contribuição fica especialmente na criação de uma arquitetura de referência adaptada as necessidades da Luban com potencial para futuras expansões, preparada para resolver os seus diversos problemas com uma implementação técnica suportada com as melhores práticas de arquiteturas de software modernas e eficientes na resolução dos requisitos específicos da Luban. Dentro dos problemas identificados, focamo-nos especialmente na gestão, monitorização e respetiva comunicação em tempo real com os seus diversos processos, equipamentos e recursos associados.

1.5. Estrutura do Documento

Este trabalho irá seguir uma estrutura clara, dividindo cada capítulo de acordo com os tópicos considerados relevantes para o seu contexto:

- Capítulo 1, introduz o contexto do trabalho do ponto de vista teórico e temporal, para além de dar a conhecer os problemas que se querem mitigar.
- Capítulo 2, dá a conhecer o principal âmbito do projeto, a oficina Luban, complementando a introdução apresentada no capítulo anterior.
- Capítulo 3, é onde entramos nos fundamentos teóricos e tecnológicos relevante ao contexto do projeto, são exploradas algumas arquiteturas de referência no âmbito das necessidades da Luban, assim como, alguns dos protocolos necessários para a troca de informação na Luban.
- Capítulo 4, é introduzido o protótipo, dividido em 3 fases, é onde são feitos todos os testes técnicos para perceber se todos os requisitos eram exequíveis. Com isto o protótipo cresceu o suficiente para funcionar como elemento base para uma infraestrutura final.
- Capítulo 5, utilizando o conhecimento alcançado no capítulo anterior, e tendo em conta os problemas identificados e os requisitos a alcançar é dada a conhecer uma arquitetura de referência que ambiciona responder a todos estes elementos.
- Capítulo 6, dedicado a reflexões e conclusões finais da estrutura sugerida e de todo o trabalho realizado.

Capítulo 2

Oficina Luban

Este capítulo foca-se inteiramente em introduzir a oficina Luban de forma mais detalhada, com destaque para uma apresentação do contexto de onde nasceu a oficina, as razões que levaram à escolha da localização da mesma e os objetivos que pretende alcançar. Posteriormente, é dado a conhecer de mais detalhada o que é um processo no contexto da Luban, de que são compostos e para que servem. Esta introdução mais detalhada aos processos é importante pois este irá ser um tópico mencionado várias vezes durante o decorrer deste trabalho.

2.1. Contextualização da Luban

A oficina Luban nasce no âmbito de uma parceria com o governo da província chinesa de Tianjin, e é a única a ser instalada em Portugal e a sexta no mundo, depois da Índia, Reino Unido, Indonésia, Tailândia e Paquistão. As oficinas *Lu Ban*, que levam o nome do carpinteiro, engenheiro e inventor chinês da dinastia Zhou, admirado como um Leonardo da Vinci do Oriente, são plataformas de colaboração tecnológica entre a China e os países destinatários, inscrevendo-se na estratégia de internacionalização do país. A escolha do Instituto Politécnico de Setúbal para a instalação da única oficina Luban em Portugal, vem na sequência de um processo de estudo e avaliação de várias instituições de ensino superior nacionais de renome, e deveu-se essencialmente ao estatuto de referência da instituição nesta área do conhecimento, grande proximidade à indústria da região, disponibilidade de espaço físico adequado para a construção e montagem de equipamentos e afinidade ao nível de cursos e de métodos pedagógicos.

As sugestões para o desenvolvimento do sistema e respetiva infraestrutura surgem de um conjunto de ideias e visões consideradas pelo Prof. José Lucas, coordenador e principal responsável pela oficina. A sua experiência e conhecimento aprofundado no campo da Automação, Robótica e Controlo Industrial servem de alicerce para elevar e preparar a oficina para os desafios da Indústria 4.0 e adiante. O principal objetivo passa por desenvolver um ambiente altamente capacitado para investigação e aprendizagem de diversas competências.

2.2. Processos Existentes

A oficina Luban é reconhecida como uma ferramenta versátil e o seu propósito reside na simulação de processos industriais que representam casos reais, sendo o seu principal objetivo o de impulsionar o crescimento e a aquisição de conhecimentos em todos os indivíduos que interagem com a mesma. A existência de diferentes processos é fundamental para que possam ser exploradas diferentes vertentes de ensino numa ótica de controlo e automação.

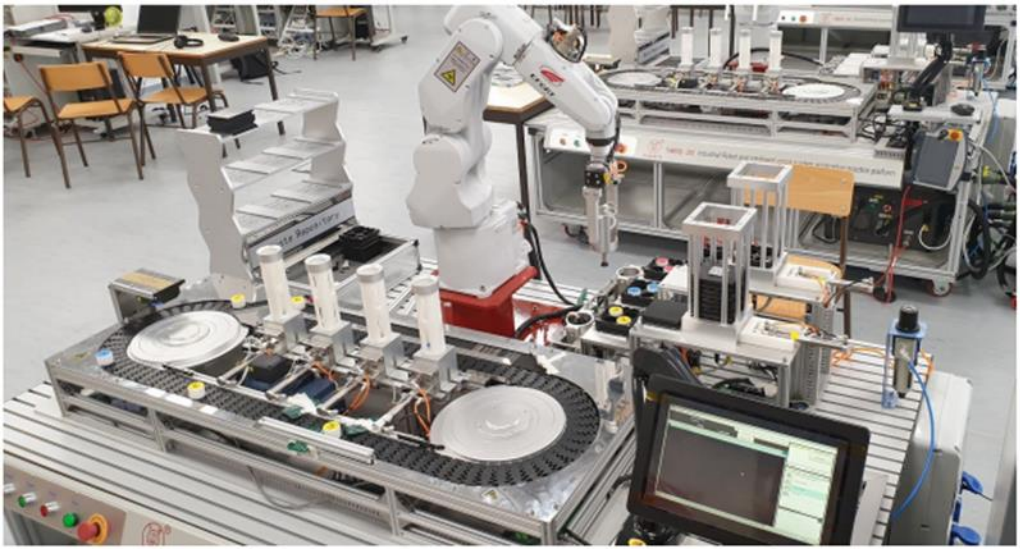


Figura 2.1 - Estação de Processo na Luban

Atualmente a oficina está equipada de forma a suportar 4 tipos de processo, aplicados em diferentes grupos de estações compostas por diversos componentes físicos.

A disposição e organização destes componentes físicos vai de acordo com o contexto industrial proposto para a estação, como exemplo, existe um processo para a indústria farmacêutica, um processo de inteligência com recurso a um sistema de visão e componentes RFID, armários de eletricidade com motores trifásicos, entre outos. O enfâse na adaptabilidade é evidente nos exemplos fornecidos, nos quais podemos destacar o processo destinado à indústria farmacêutica, que aborda as necessidades particulares desse setor altamente regulamentado e sensível. Além disso, a incorporação de sistemas de visão e RFID demonstra a aplicação de tecnologias avançadas para melhorar a inteligência e a monitorização de processos. Por fim, a presença de armários de eletricidade equipados com motores trifásicos evidencia a atenção às necessidades especificas de potência e controlo elétrico em contexto industrial diversificado.

Internamente, os componentes dos processos comunicam por TCP/IP através de switches, estes estão conectados a um router principal que trata o reencaminhamento para o exterior. Na Figura 2.2 - Representação dos processos e as suas ligações, é possível observar uma visão mais técnica e detalhada da composição dos diferentes processos e das suas conexões.

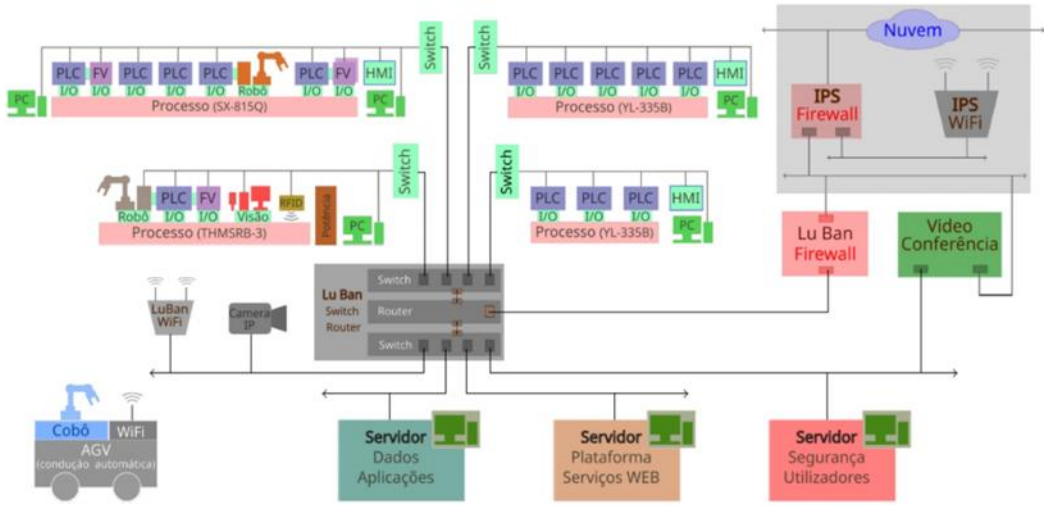


Figura 2.2 - Representação dos processos e as suas ligações

A estrutura representada na figura exibe uma visão geral da Oficina Luban, porém, é fundamental salientar um aspeto essencial: a flexibilidade. As conexões e ligações visíveis são facilmente ajustáveis, permitindo a inclusão ou exclusão de elementos conforme necessário. Essa adaptabilidade e versatilidade são fundamentais para manter a oficina atualizada nos paradigmas da indústria 4.0, possibilitando a interoperabilidade quando exigida.

Um dos componentes de hardware frequentemente presente nos processos da oficina e que é facilmente identificado na Figura 2.2 - Representação dos processos e as suas ligações, é o PLC (*Programmable Logic Controller*). Trata-se de um controlador utilizado para monitorizar e controlar máquinas e processos industriais, independentemente do nível de complexidade. Os PLCs são acessíveis por intermédio de software, que pode ser programado pelos utilizadores ou desenvolvido pelos fabricantes dos controladores. Essa programação permite a interação com um conjunto de entradas e saídas, que desempenham um papel crucial na monitorização, gestão e controlo dos processos. Além disso, cada processo conta com um PC/HMI (*Human-Machine Interface*), que oferece uma interface visual amigável para os operadores interagirem com o sistema, permitindo a realização de ações de controlo e gestão.

Um exemplo concreto de equipamento presente na oficina, representado na Figura 2.3 - Processo SX-815Q, é o SX-815Q. Esse equipamento desempenha um papel fundamental na automação de processos relacionados à produção e armazenamento de materiais granulares. As suas funcionalidades incluem a alimentação automatizada, o enchimento de frascos vazios, a inspeção e triagem de materiais, entre outras atividades que podem ser aplicadas em diferentes contextos. O SX-815Q exemplifica a diversidade de contextos presente nos processos da Luban, assim como, o realçar de equipamentos com tecnologias avançadas e ponta que vêm garantir uma aprendizagem moderna e dentro daquilo que é a realidade nas indústrias atuais.

Esses componentes e equipamentos mencionados são apenas alguns exemplos do conjunto de recursos e tecnologias presentes na Oficina Luban.

A adoção de tais sistemas de automação e controlo proporcionam benefícios significativos, incluindo o aprimoramento da eficiência operacional, redução de erros e otimização dos processos fabris.

Apesar de a Luban ser uma oficina mais focada na componente educacional e de investigação, a atenção ao detalhe permitirá simular experiências muito semelhantes ao que é aplicado na indústria atualmente, e em muitos casos inclusive estar à frente, pois, apesar da indústria 4.0 ser cada vez mais um standard de eficiência nos dias atuais existem muitas indústrias que ainda não conseguiram dar o salto e seguir muitos dos seus paradigmas, não sendo estes totalmente adotados como uma norma global.

Figura 2.3 - Processo SX-815Q

Capítulo 3

Fundamentos Teóricos e Tecnológicos

Este capítulo tem como foco a análise e discussão de Infraestruturas/Arquiteturas de Referência relevantes no contexto da Oficina Luban. Serão abordadas as principais estruturas tecnológicas utilizadas por empresas em contextos similares e na indústria em geral para otimizar processos, melhorar a gestão de recursos e a eficiência operacional. Será realizada uma revisão detalhada da literatura disponível, explorando trabalhos e pesquisas relacionadas com estas arquiteturas para que, caso faça sentido, possam servir de inspiração para as diferentes fases da infraestrutura final a sugerir.

Outro tópico relevante a ser explorado são os protocolos de comunicação adequados ao contexto da Luban e que acabam por ser um dos fatores mais influentes na estrutura/arquitetura final sugerida.

A análise destas referências servirá como base para o desenvolvimento de uma infraestrutura tecnológica sólida e adequada às necessidades específicas da oficina Luban, visando melhorar as suas operações dentro dos ideais propostos pelo Prof. José Lucas.

3.1. Infraestruturas/Arquiteturas de Referência

Como foi referido anteriormente, o principal objetivo passa por criar uma arquitetura que vá de acordo com as necessidades da Luban, como tal, é importante investigar e perceber quais as arquiteturas de referências utilizadas no âmbito da indústria 4.0.

As arquiteturas/infraestruturas que foram consideradas relevantes para o contexto da oficina Luban são:

- IIRA (Industrial Internet Reference Architecture)
- RAMI 4.0 (Reference Architecture Model Industry 4.0)
- ISA 95 (International Standard for International Society of Automation)
- SCADA (Supervisory Control and Data Acquisition)

3.1.1. IIRA

IIRA é uma arquitetura de referência para a indústria com foco na internet, é especificamente adaptada para abordar os desafios únicos e requisitos do setor industrial, visando aproveitar os benefícios da Internet das Coisas (IoT) e da Internet Industrial das Coisas (IIoT). Esta arquitetura serve como um modelo para a criação de sistemas conectados e inteligentes dentro de um ambiente industrial, promovendo a comunicação e interoperabilidade contínua entre vários componentes e dispositivos. Como o seu nome indica, é possível dividir-se IIRA em 2 conceitos principais:

- Internet Industrial
- Arquitetura de Referência

Internet Industrial, é um conjunto de instrumentos e operações inteligentes utilizados na indústria, que se baseia numa avançada análise de dados para aprimorar os processos e atividades de um sistema. Atualmente, existem diversos sistemas que fazem uso de sensores, processadores e atuadores integrados, proporcionando capacidades operacionais e comerciais. Cada vez mais, observamos estes dispositivos conectados através da internet, o que expande de forma exponencial os seus casos de uso. A Internet Industrial engloba uma variedade de setores, como energia, cuidados de saúde, indústria transformadora, setor público, transportes e indústrias relacionadas. Consequentemente, muitos desses sistemas operam em ambientes críticos, onde os padrões de qualidade, segurança e resiliência devem ser elevados. Para que seja efetivo, um Sistema de Internet Industrial (IIS) exige níveis significativos de desempenho, escalabilidade e eficiência.

Arquitetura de Referência, é um conjunto de diretrizes destinadas a orientar o processo de desenvolvimento de um sistema ou arquitetura de aplicações. Proporciona definições comuns e consistentes no sistema, permitindo a decomposição de vários padrões de design a serem aplicados, criando assim um vocabulário compartilhado que facilita a discussão das implementações e possíveis alterações futuras. Uma arquitetura de referência estabelece um quadro comum que possibilita discussões mais detalhadas. Definida num nível elevado de abstração permite a identificação e compreensão das questões e padrões mais importantes nas suas aplicações. Ao evitar especificações rígidas, uma arquitetura de referência permite que os projetos subsequentes sigam normas sem a necessidade de restrições arbitrárias.

Após uma reflexão sobre os conceitos que compõem a IIRA, podemos identificar essa arquitetura como uma abordagem aberta baseada em padrões para sistemas industriais de Internet (IISs). Com o objetivo de maximizar o seu valor, a IIRA tem ampla aplicabilidade na indústria, visando impulsionar a interoperabilidade, mapear tecnologias aplicáveis e orientar o desenvolvimento de normas e tecnologias.

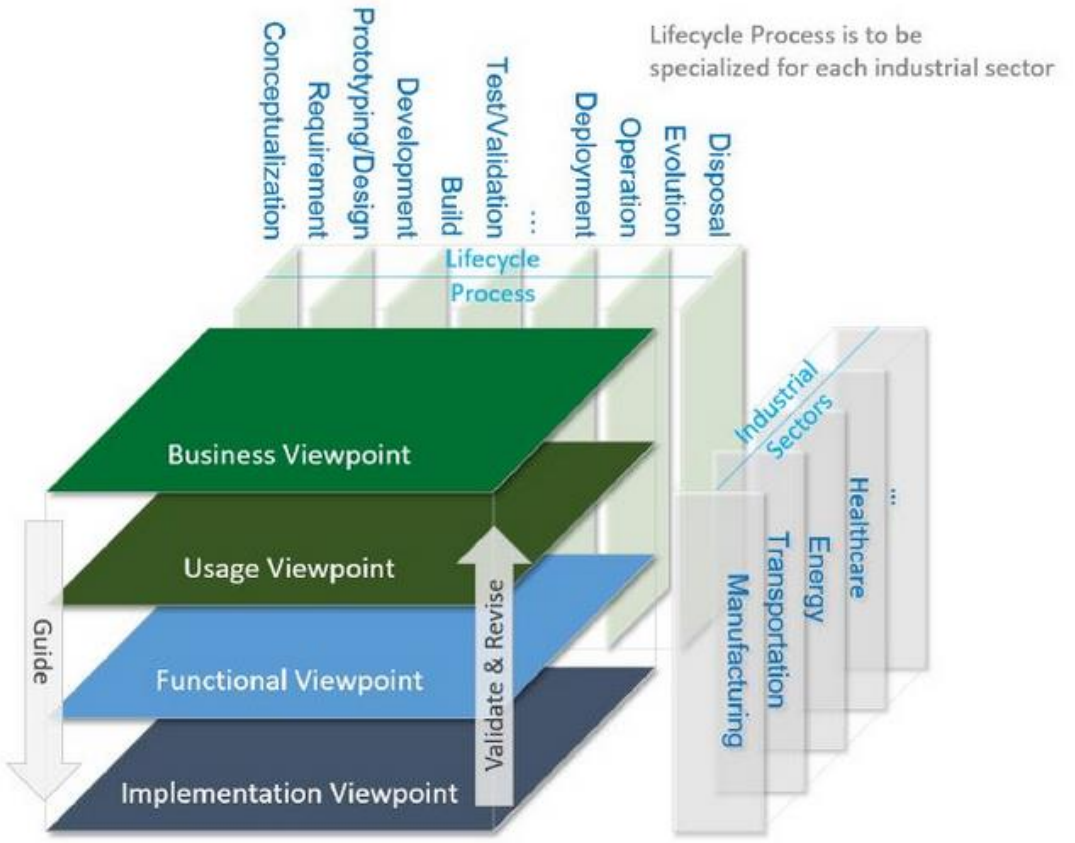


Figura 3.1 – Framework IIRA

O desenho da IIRA, representado na Figura 3.1 – Framework IIRA, é elaborado a partir de quatro pontos de vista, cada um com a sua relevância que varia de acordo com as preocupações dos diferentes *stakeholders*. Destina-se a transcender as tecnologias atualmente disponíveis, permitindo desta forma a identificação de lacunas tecnológicas com base nos requisitos estabelecidos. Esta abordagem, por sua vez, irá impulsionar os esforços e puxar pela inovação na comunidade da Internet industrial para que seja possível colmatar as lacunas identificadas.

No que toca à componente da implementação, vários sistemas industriais de Internet (IISs) coerentes seguem certos padrões arquiteturais bem definidos, entre eles destacam-se os seguintes:

- *Three-tier Architecture Pattern*
- *Gateway-Mediated Edge Connectivity and Management Architecture Pattern*

Three-tier Architecture Pattern, como o seu nome indica, é um padrão de arquitetura dividido em três áreas distintas de ação:

- *Edge*
- *Platform*
- *Enterprise*

O nível ***Edge***, enquadra a área mais próxima do ambiente físico do sistema, onde se encontram os nós responsáveis por recolher e enviar dados para as restantes camadas. Tradicionalmente estes nós referem-se a sensores, atuadores, entre outros.

O nível ***Platform***, recebe, processa e encaminha os comandos de controlo enviados do nível arquitetural acima (*Enterprise*). Consolida processos e analisa a transmissão de dados dos restantes níveis. Oferece funções de gestão para os diversos dispositivos.

O nível ***Enterprise***, implementa aplicações especificas do domínio, sistemas de suporte à decisão e interfaces de utilizador para que estes possam interagir com as restantes camadas. Recebe os dados provenientes dos níveis ***Edge*** e ***Platform*** e é deste nível que partem os comandos de controlo.

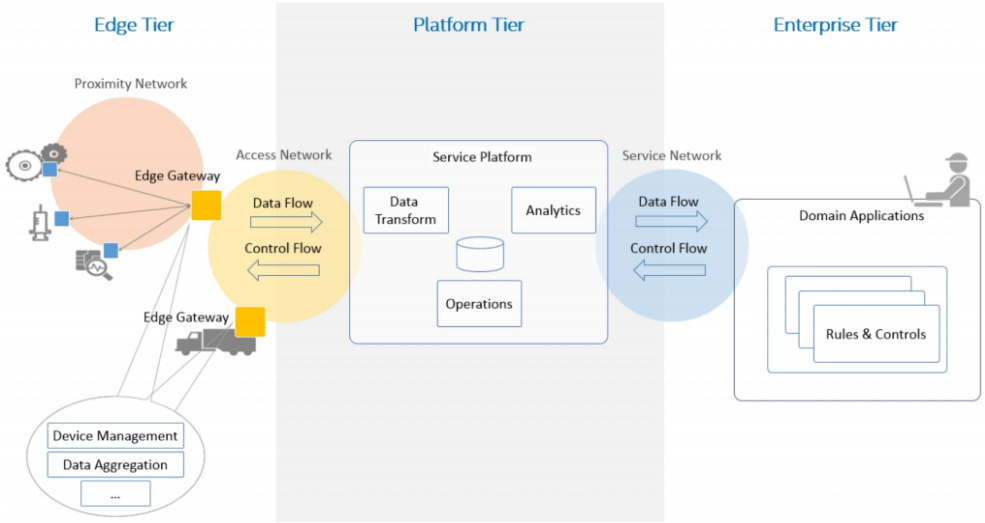


Figura 3.2 - Three-tier Architecture Pattern

Gateway-Mediated Edge Connectivity and Management Architecture Pattern, este padrão abrange o comportamento associado ao nível ***Edge*** dentro de uma rede local, utilizando um gateway que serve como ponte para uma área mais ampla da rede. Esse gateway atua como um ponto de acesso para a rede mais abrangente, isolando, dessa forma, a rede de dispositivos Edge (sensores, atuadores, etc.). O principal objetivo desta arquitetura é simplificar a complexidade dos IISs, permitindo que sejam escalados tanto em termos de número de dispositivos, assim como, em termos de infraestrutura de rede. Ao utilizar gateways, os dispositivos de Edge podem ser conectados de forma eficiente à rede central, facilitando a troca de dados e a gestão das operações.

Esta abordagem é especialmente útil em ambientes industriais, onde a conectividade e a gestão de dispositivos Edge são fundamentais para garantir a eficiência, a segurança e o desempenho do sistema como um todo. Ao aplicar-se o padrão ***Gateway-Mediated Edge Connectivity and Management***, os IISs podem ser dimensionados de maneira mais flexível e adaptável, possibilitando uma melhor integração e controlo dos dispositivos na rede industrial.

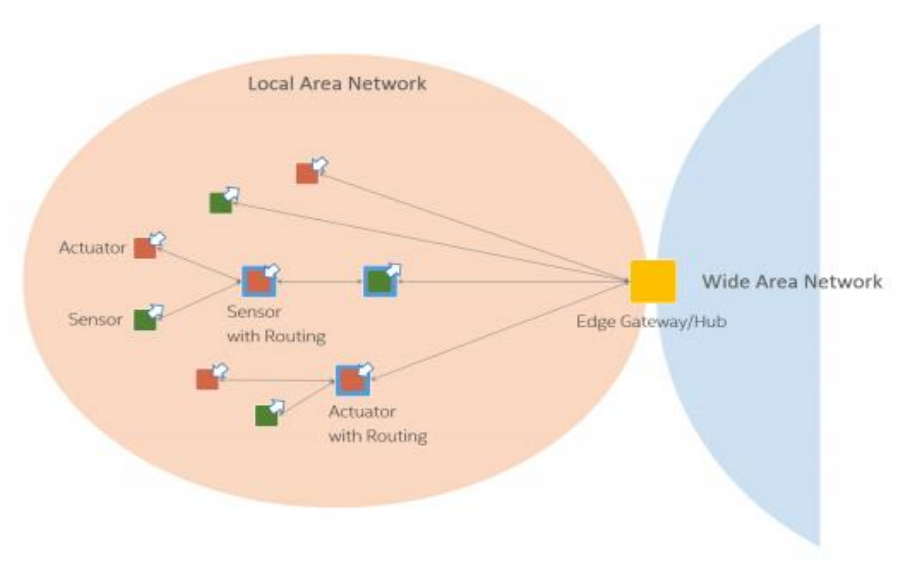


Figura 3.3 - Gateway-Mediated Edge Connectivity and Management Architecture Pattern

3.1.2. RAMI 4.0

Reference Architectural Model Industry 4.0, é um mapa tridimensional que destaca os aspetos mais importantes da Indústria 4.0 e assegura que todos os participantes envolvidos compartilhem uma perspetiva comum durante o desenvolvimento. Trata-se de uma orientação que permite estabelecer os requisitos dos setores em conjunto com as normas nacionais e internacionais para definir e desenvolver a Indústria 4.0.

As diferentes camadas do RAMI 4.0 são as seguintes:

- Camada de Ativos (*Asset*), inclui componentes físicos, documentos, software e atores humanos envolvidos no sistema industrial.
- Camada de Integração (*Integration*), representa a versão digital dos componentes presentes na camada de Ativos, proporcionando o controlo informatizado do processo técnico.
- Camada de Comunicação (*Communication*), oferece a comunicação para os serviços, eventos e dados da camada de Informação, fornecendo comandos de controlo para a camada de Integração.

- Camada de Informação (*Information*), descreve os serviços e eventos/lógica de dados de um componente no que diz respeito ao seu papel num sistema da Indústria 4.0.
- Camada Funcional (*Functional*), é o ambiente de execução para aplicações e funcionalidades técnicas que permitem a realização de tarefas de suporte aos processos de negócio.
- Camada de Negócio (*Business*), contém funções que estabelecem conexões entre diferentes processos empresariais, apoiando os modelos de negócio sob a perspetiva legal, levando em conta as restrições regulamentares.

O RAMI 4.0 proporciona uma estrutura clara para a compreensão e o desenvolvimento de sistemas da Indústria 4.0, integrando e alinhando diferentes elementos para promover uma abordagem coesa e eficiente na indústria moderna.

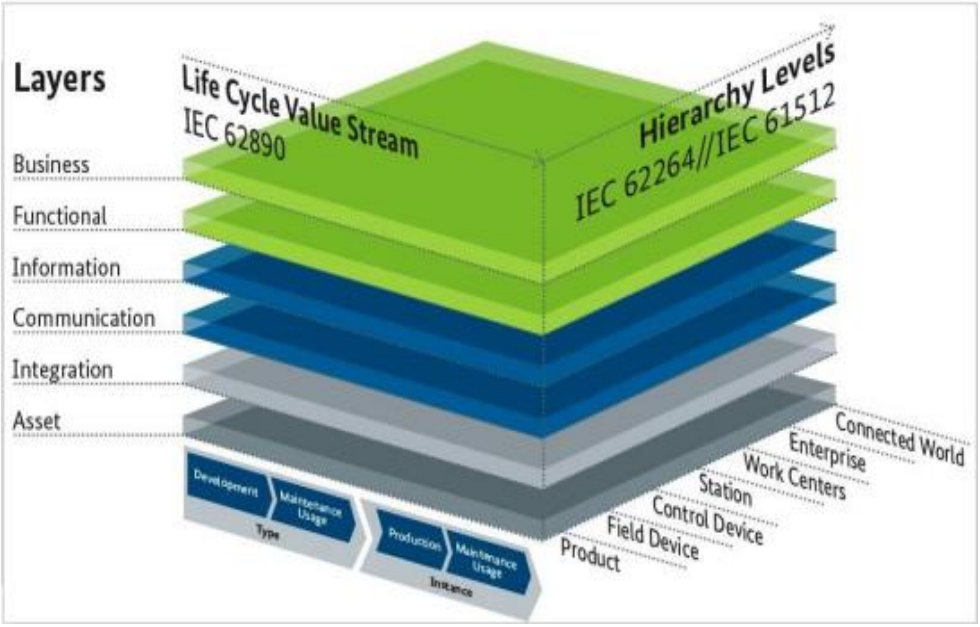


Figura 3.4 - Estrutura do RAMI 4.0

3.1.3. ISA 95

International standard for International Society of Automation, é um standard internacional para o desenvolvimento de interfaces automatizadas entre sistemas de controlo e gestão.

Os seus objetivos passam por:

- Estabelecer uma terminologia consistente que sirva como base para a comunicação entre fornecedores e fabricantes.
- Fornecer modelos de informação consistentes.
- Oferecer modelos de operações consistentes que sirvam como base para esclarecer a funcionalidade da aplicação e a forma como as informações devem ser utilizadas.

Resumindo, este padrão procura garantir a uniformidade e a coerência nas interfaces utilizadas em sistemas de automação, proporcionando uma base sólida para a comunicação e a troca de informações entre diferentes sistemas e atores no contexto da automação industrial.

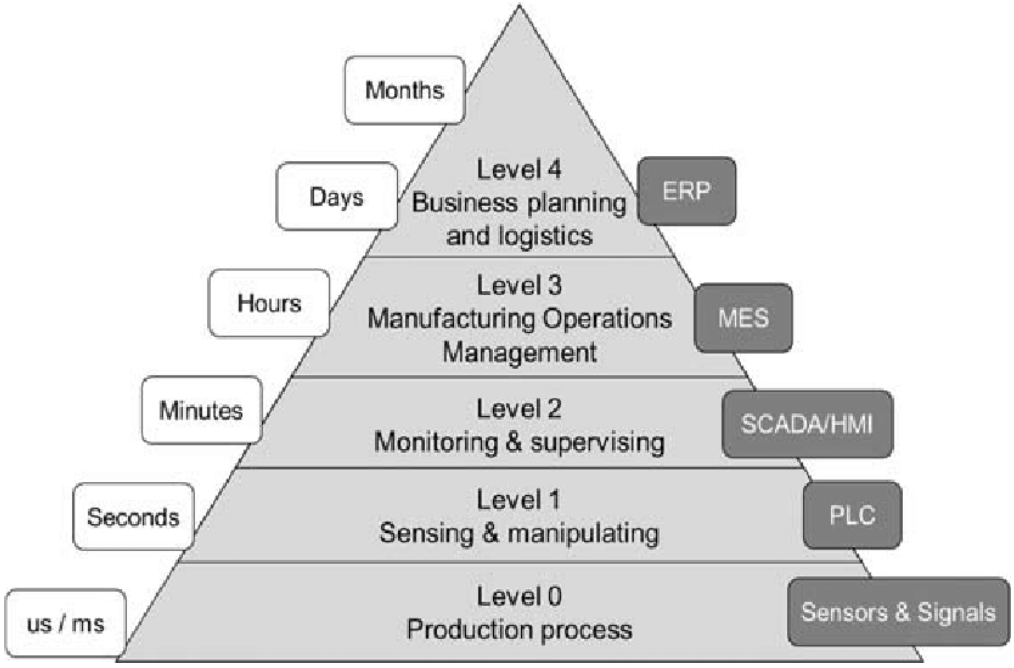


Figura 3.5 - Pirâmide para o ISA 95

A Figura 3.5 - Pirâmide para o ISA 95 oferece uma representação muito importante para o trabalho atual, em cada nível da pirâmide é possível observar como os diferentes elementos deste contexto são quantificados em termos temporais.

3.1.4. SCADA

Supervisory Control and Data Acquisition, é uma arquitetura que define o processo de comunicação e como o controlo pode ser realizado num ambiente industrial. O conceito de SCADA pode ser aplicado a pequenos sistemas de controlo ou a grandes operações. Tipicamente, um sistema SCADA é um componente de uma infraestrutura mais complexa, caracterizada por um alto nível de abstração.

Estes sistemas são geralmente uma combinação de elementos de software e hardware e visam criar interfaces homem-máquina (HMI), que são observáveis em diversos softwares, como:

- Tesla SCADA
- ScadaBR

Figura 3.6 - Software Tesla SCADA

Os softwares mencionados fornecem um ambiente para interagir com os diversos controladores industriais e realizar ações de gestão e controlo sobre eles. No entanto, possuem algumas limitações, especialmente pelo fato de não serem compatíveis com web browsers, o que restringe o potencial de integração com certos sistemas da indústria 4.0.

Resumindo, o SCADA é uma arquitetura fundamental para o controlo e monitorização de sistemas industriais, mas é importante considerar as limitações dos softwares no momento de integrar com as tecnologias avançadas da indústria 4.0.

3.2. Protocolos de Comunicação

No contexto do trabalho a troca de informação é um elemento essencial para o sucesso, e a forma como esta é realizada tem um impacto no momento de tomar decisões sobre a arquitetura a desenvolver, com base neste argumento foram considerados para o projeto dois protocolos de comunicação que se ajustam melhor ao enquadramento técnico da Luban.

3.2.1. Modbus

Modbus é um protocolo de comunicação de dados, muito utilizado em sistemas de automação e controlo industrial, desenvolvido em 1970 para a comunicação entre PLC's da *Modicon*. Atualmente é um protocolo gratuito, muito versátil, de rápida velocidade e fisicamente a sua comunicação pode ser feita através de Ethernet ou Comunicação Série e por isso tornou-se um standard nas indústrias para envios de dados discretos ou numéricos (entradas e saídas analógicas).

Este protocolo opera segundo a estrutura de comunicação "Mestre-Escravo", na qual o mestre assume o papel de iniciador das comunicações, estabelecendo conexões com um ou múltiplos escravos e emitindo comandos de leitura ou escrita. A principal distinção entre este método e o equivalente "Cliente-Servidor" reside na responsabilidade pelo início das comunicações em ambos os métodos. Tanto o Mestre como o Cliente desempenham esse papel, contudo, o Mestre detém a primazia em termos de capacidade computacional, em contraste com o método "Cliente-Servidor", no qual a responsabilidade recai sobre o Servidor. Resumindo, o Mestre assemelha-se ao Cliente em termos de iniciação das comunicações, mas deve possuir a capacidade computacional que, no contexto do método "Cliente-Servidor", é atribuída ao Servidor.

Como foi referido anteriormente, o Modbus trabalha com dados discretos ou numéricos e do ponto de vista de estandardização são dadas algumas nomenclaturas próprias aos respetivos blocos de memória e permissões:

Bloco de Memória	**Tamanho**	**Acesso do Mestre**	**Acesso do Escravo**
Coil	*1 bit*	Read/Write	Read/Write
Discrete Input	*1 bit*	Read	Read/Write
Holding Register	*16 bits*	Read/Write	Read/Write
Input Register	*16 bits*	Read	Read/Write

Tabela 3.1 - Tabela de dados do Modbus

Considerando a Tabela 3.1 - Tabela de dados do , podemos analisar que os valores discretos correspondem aos "Coils" e aos "Discrete Inputs", enquanto os valores destinados a receber dados numéricos (geralmente números inteiros) são os "Holding Registers" e os "Input Registers". Além disso, é relevante mencionar que, com as ferramentas apropriadas, é possível manipular valores com tamanho superior a 16 bits, como, por exemplo, 32 bits. Nesse caso, ao efetuar a leitura, serão solicitados dois endereços de 16 bits, que serão convertidos em uma palavra dupla (double word). Durante a escrita, a palavra dupla será novamente dividida em duas partes e armazenada.

Para que seja possível interagir com os diferentes blocos de memória, seja para ler ou escrever, existem um conjunto de instruções definidas através de *function codes* (FC), estes códigos fazem parte dos pedidos feitos aos “escravos” e que acabam por ditar a ação a realizar, é possível observar na Tabela 3.2 - Funções do Modbus, uma listagem dos códigos mais utilizados.

Nome da Função	**Código**	**Tipo de Dados**
Read Discrete Inputs	*2*	*1 bit*
Read Coils	*1*	*1 bit*
Write Single Coil	*5*	*1 bit*
Write Multiple Coils	*15*	*1 bit*
Read Input Registers	*4*	*16 bits*
Read Multiple Holding Registers	*3*	*16 bits*
Write Single Holding Register	*6*	*16 bits*
Write Multiple Holding Registers	*16*	*16 bits*

Tabela 3.2 - Funções do Modbus

3.2.2. MQTT

O MQTT é um protocolo de mensagens relativamente leve, pensado para um contexto onde a comunicação deve ser eficiente entre dispositivos em redes com baixa largura de banda, alta latência ou conexões não confiáveis. Criado no final da década de 1990, o MQTT ganhou considerável popularidade nos últimos anos devido à sua adequação para aplicações de IoT (Internet das Coisas).

O MQTT opera sobre um modelo cliente-servidor e é caracterizado pelos seguintes elementos:

- **Broker:** O componente central de uma rede MQTT é o broker. Este atua como intermediário, recebendo mensagens de clientes, filtrando-as e encaminhando-as para os *subscribers* interessados.
- **Publisher:** Um cliente que gera e envia mensagens para o broker para serem distribuídas pelos respetivos *subscribers*.
- **Subscriber:** Um cliente que demonstra interesse em receber mensagens sobre os tópicos subscritos. Os *subscribers* recebem mensagens do broker que correspondem às suas subscrições.
- **Topics:** As mensagens no MQTT são publicadas para tópicos. Os *subscribers* indicam interesse em tópicos particulares, subscrevendo-se a estes onde a partir desse momento receberam mensagens novas dedicadas aos seus tópicos subscritos.

O MQTT é ideal para cenários onde os dispositivos precisam de trocar informações de forma assíncrona e onde a alta escalabilidade e flexibilidade são necessárias. A sua utilização brilha em aplicações que envolvem redes de sensores, telemetria e monitorização remota. Este é projetado para transmissão eficiente em redes com baixa largura de banda e conexões não confiáveis. Trata de minimizar a sobrecarga e permite mensagens compactas.

Capítulo 4

Protótipo

Neste capítulo será introduzido um dos principais componentes do trabalho, o protótipo, este serviu como principal elemento de ligação entre duas realidades distintas, a componente de Software em contexto Web e a componente de Hardware em contexto de controlo e automação.

Existiram vários momentos durante o projeto onde foi necessário perceber se alguns dos objetivos eram exequíveis, principalmente de um ponto de vista técnico, e foi a partir deste protótipo que diversas ideias e conceitos foram postos em prática através da utilização de uma abordagem incremental e iterativa, apesar do produto final não ser uma aplicação finalizada com todos os detalhes e elementos que a Luban desejaria, é com base nas aprendizagens realizadas ao longo das diversas fases do protótipo que as sugestões para a aplicação e respetiva infraestrutura finais foram tomadas.

O desenvolvimento do protótipo pode ser repartido principalmente por 3 fases:

- Fase de Introdução/Aprendizagem
- Fase de Complementaridade
- Fase de Ajustes Finais

Durante este capítulo, cada fase será introduzida do ponto de vista teórico, onde serão explicadas as motivações, quais as abordagens de alto nível que se pretende atingir, assim como, do ponto de vista técnico onde serão exploradas todas as tecnologias usadas com suporte a pequenos excertos de código onde fizer sentido bem como dos respetivos diagramas de alto nível para ajudar a partilhar as ideias pretendidas.

4.1. Fase de Introdução/Aprendizagem

Esta fase ditou o início do processo, após algumas conversas iniciais onde foram levantados vários requisitos, muitos acabariam por necessitar de passar por um processo de “tradução”, tal como foi mencionado anteriormente, o contexto da Luban no mundo da automação e controlo necessita de software num contexto web para que se possa modernizar, esta ideia acaba por levar a uma automatização de vários processos tornando-os mais eficientes, no entanto, quando se está na fase inicial existe uma ponte entre realidades diferentes que necessita de ser construída, ou pelo menos perceber se é exequível, foi então esse requisito que serviu como ponto de partida, levando assim à criação do conceito de um protótipo.

Durante as conversas iniciais um dos tópicos mais relevantes, e fundamental para o sucesso do projeto, foi perceber como seria possível comunicar diretamente com os PLC’s presentes nas diversas estações de processo na Luban, especialmente num tempo útil com uma latência relativamente baixa, esse acabou por se tornar noutro requisito bastante desafiante. Para atingir esta comunicação entre o browser e o PLC foi necessário implementar uma camada intermédia que permitisse enviar pedidos utilizando o protocolo de comunicação Modbus através de JavaScript, neste caso, o interesse não era o de reinventar a roda e tentar ganhar tempo utilizando alguma implementação já existente, como tal, foi realizada uma investigação para perceber as bibliotecas que existiam dentro do ecossistema do NodeJS que permitissem enviar pedidos ModBus sobre o protocolo TCP/IP, algumas destas bibliotecas foram:

- *modbus-tcp*
- *modbus-ws*
- *jsmodbus*
- *modbus-stream*
- *modbus-tcp-ip*

Dentro das bibliotecas mencionadas, foram feitos vários testes para perceber a viabilidade de cada para o contexto pedido, no fim acabou por se optar por utilizar a biblioteca *jsmodbus*, que demonstrou ser a mais estável e com mais suporte para além de conter diversas funcionalidades nativas do ModBus, de mencionar que o principal objetivo destas bibliotecas era o de conseguir instanciar clientes Modbus (escravos) que se pudessem conectar ao servidor (mestre) utilizando TCP/IP, esta comunicação entre os diversos elementos ocorria utilizando sockets.

Após conseguir-se ter uma comunicação com Modbus utilizando uma implementação em JavaScript estamos num contexto de NodeJS e a partir desse momento as possibilidades são inúmeras, porém, dado que o objetivo principal era o de conseguir interagir com um PLC através do browser, começou-se a testar as diversas funções (códigos de funções, FC) para interagir com os *coils*, *registers* e inputs. Os testes realizados foram suportados por uma ferramenta muito importante, TeslaScada, este software acabou por permitir simular interações com um PLC e visualmente observar luzes a ligar/desligar, números a incrementar/decrementar e até mesmo à escrita de texto, estas operações eram primitivas essenciais para garantir que o fluxo da comunicação estaria a

funcionar e era importante que esta base estivesse sólida antes de prosseguir para uma aplicação Web mais complexa. Este software foi muito importante pois nesta fase inicial não existia uma interface de utilizador no browser, apenas a componente de servidor NodeJS que funcionava como um cliente modbus.

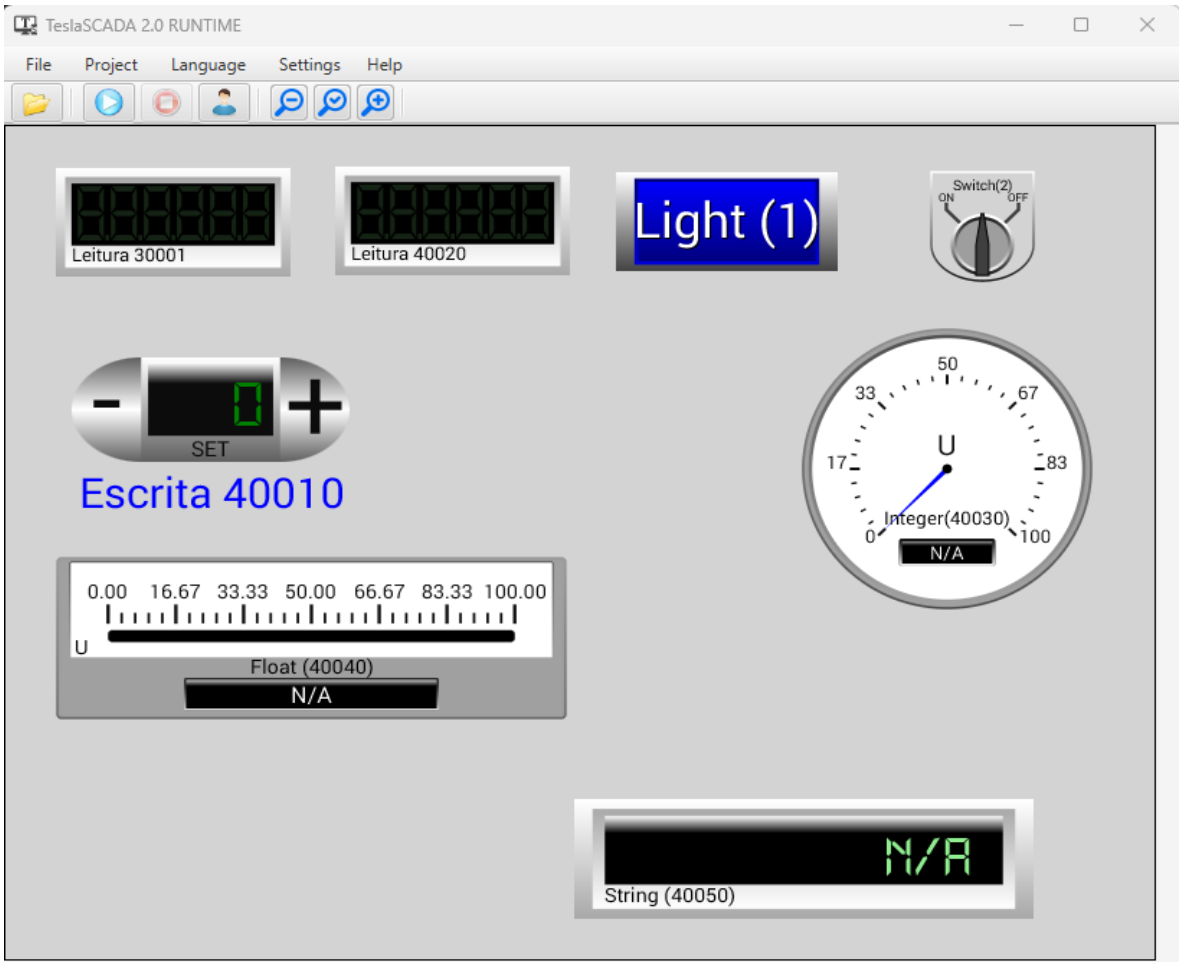


Figura 4.1 - Interface Visual de Testes no Tesla Scada

A Figura 4.1 - Interface Visual de Testes no Tesla Scada, expressa visualmente os diversos componentes que estavam a ser testados, todos estes dispositivos de uma maneira ou outra necessitavam que os dados fossem tratados de forma diferente, seja como valores números inteiros, reais ou em texto (*string*). Para que esta comunicação fosse atingida foi necessário desenvolver uma abstração que permitisse facilmente converter qualquer tipo primitivo (números, *strings*, etc) em JavaScript para um buffer binário ou hexadecimal, pois a biblioteca que tratava da comunicação em Modbus necessitava que os dados passassem por essa transformação para os respetivos formatos corretos antes de serem enviados.

Após garantir que a comunicação era possível passou-se para o passo seguinte, era essencial ter uma página HTML simples onde fosse possível interagir com o nosso servidor NodeJS e através de inputs partilhar informação que fosse transformada e enviada por Modbus para o nosso

PLC de teste, para efeitos de simplicidade, o PLC de teste foi simulado em Python durante uns testes realizados na oficina Luban e tinha o objetivo de funcionar como o servidor Modbus (mestre), era aqui que os blocos de memória eram escritos e lidos e depois as respetivas atualizações eram visualizadas nos clientes (escravos). Este servidor em Python foi temporário e usado meramente neste contexto local para testes e foi alcançado através da biblioteca pyModbusTCP, era também neste servidor onde se podia facilmente através de *scripting* escrever dados nos blocos de memória para que se conseguisse testar a sua leitura nos restantes clientes.

A última experiência a ser realizada neste contexto inicial foi a de tentar efetivamente substituir a forma como a comunicação entre o browser e o servidor NodeJS estava a ser realizada, para os testes iniciais foi criada uma RESTful API. Cada ação do Modbus a testar foi implementada seguindo os princípios REST, onde as diversas formas de leitura e escrita do modbus estavam implementadas em diversos *endpoints* independentes, associados aos seus respetivos verbos HTTP. Esta comunicação funcionava, mas não era eficiente e estava longe de garantir os níveis de eficiência esperados para o contexto da oficina Luban, como foi falado anteriormente, era bastante importante garantir uma comunicação com baixa latência e que não estivesse dependente de ações manuais do utilizador no browser. Foi neste momento que se tornou relevante a utilização de sockets num contexto web (websockets), estes permitiram uma comunicação bidirecional constantemente ativa e que funcionava sob a forma de eventos onde o servidor NodeJS estava responsável por enviar as atualizações relevantes a todos os utilizadores que estavam conectados. Para este caso foi novamente realizada uma pesquisa para entender as melhores bibliotecas no ecossistema NodeJS que permitiam facilmente comunicar com sockets, foram realizados alguns testes de compatibilidade com as seguintes bibliotecas:

- *socket-io/socket-io-client*
- *pusher-js*
- *ws*

Durante os testes a biblioteca que mais facilmente se adaptou às necessidades foi o *socket-io*, como tal, acabou por ser a biblioteca usada durante os momentos iniciais do protótipo. Dito isto, estavam garantidos os elementos primários para que fosse possível implementar uma aplicação e respetiva estrutura, a partir deste momento começava-se a explorar outros requisitos principalmente numa componente de abstração.

4.1.1. Arquitetura

Após as várias experiências com as diversas tecnologias mencionadas é possível resumir a estrutura geral do protótipo nesta fase introdutória na seguinte imagem.

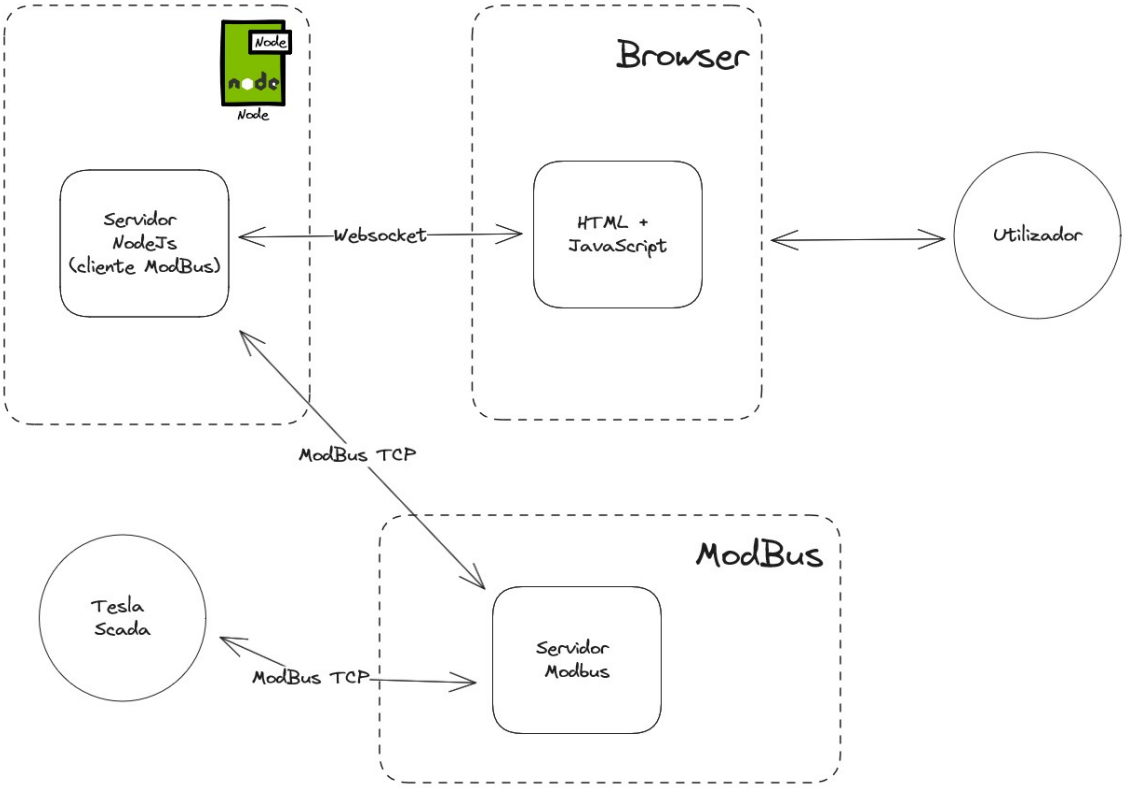


Figura 4.2 - Diagrama de Arquitetura para a Fase de Introdução

Apesar de ainda estarmos numa fase inicial, o objetivo destas experiências é o de conseguir simular a comunicação através de modbus, tentando sempre correlacionar com a realidade da Luban. Neste caso, o servidor modbus utilizado tem como principal objetivo simular um PLC presente na oficina, e o servidor NodeJS está a comunicar diretamente com este utilizando o protocolo modbus, todas as ações do utilizador são realizadas através de uma API existente no NodeJS e que faz uso do adaptador criado para modbus para comunicar com o *"*PLC*"*. Graças ao Tesla Scada, conectado ao servidor modbus por TCP, é possível confirmar que as ações de escrita e leitura, nos diversos blocos de memória, iniciadas pelo utilizador através do browser funcionam de forma esperada, sendo este fluxo de comunicação uma das primitivas para a infraestrutura a sugerir.

4.1.2. Tecnologias

Existem diversos modos de comunicação a acontecerem de forma sequencial, porém, o principal acaba por ser a comunicação entre o browser e o servidor NodeJS através de **Websockets**. Estes sockets são usados no contexto de aplicações web em que existe a necessidade de comunicar com um servidor em tempo real de forma bidirecional. A sua comunicação ocorre sobre o protocolo TCP/IP e apresenta algumas características, nomeadamente na forma como o mecanismo de *handshake* entre as partes envolventes acontece, que os distingue dos sockets tradicionais que acabam por ser mais genéricos, mas por sua vez mais poderosos para outros contextos. Neste caso, a implementação do servidor websocket foi conseguida utilizando a biblioteca *socket-io*, esta biblioteca facilita a criação de um servidor websocket integrado com NodeJS, abstraindo várias primitivas na comunicação sob a forma de uma API mais simpática de consumir.

```
const allSockets = [];
export default (io) => {
    io.on('connection', (socket) => {
        console.log('User Connected...');
        allSockets.push(socket)

        socket.on('modbus_readAddress', (data) => {
            const client = new ModbusClient(data.ip, data.port);
            await client.initializeConnection();
            const result = await client.readHoldingRegisters(data.address);
            socket.emit('readAddress', convertBufferIntoNumber(result?.response.data));
        })

        // ...other event handlers

        socket.on('disconnect', () => {
            allSockets = allSockets.filter((s: Socket) => s.id != socket.id);
            console.log('User Disconnected.');
        });
    });
}
```

Figura 4.3 - Excerto da Implementação do servidor Websocket

O excerto de código em cima demonstra como a componente websocket é criada em NodeJS, tendo sido implementados diversos eventos que fazem uso do *ModbusClient*, o adaptador que faz a ponte entre JavaScript e as diversas funções do modbus.

```
export default class ModbusClient {
  #ipAddress
  #port
  #options
  #connection

  async #connect() {
    return new Promise((resolve, reject) => {
      modbus.tcp.connect(
        this.port,
        this.ipAddress,
        this.options,
        (err, connection) => {
          if (err) reject(err)
          resolve(connection)
        }
      )
    })
  }

  async readHoldingRegisters(address) {
    try {
      if (this.connection !== undefined) {
        return new Promise((resolve, reject) => {
          this.connection?.readHoldingRegisters({ address }, (err, info) => {
            if (err) reject(err)
            resolve(info)
          })
        })
      }
    } catch (err) {
      throw new Error('No Connection Found ... ')
    }
  }

  // ... other modbus methods
}
```

Figura 4.4 – Excerto da Implementação do cliente Modbus em JavaScript

Na Figura 4.4 – Excerto da Implementação do cliente Modbus em JavaScript, temos um excerto do cliente modbus, previamente mencionado como o elemento intermédio responsável por permitir comunicar com dispositivos (PLCs) através de modbus dentro do contexto do NodeJS. Esta abstração foi criada em cima da biblioteca *modbus-stream* e oferece uma API mais conveniente de consumir e é um ponto flexível para quaisquer possíveis alterações que sejam necessárias, para além de permitir facilmente expandir as funcionalidades disponíveis.

Ambos os elementos foram implementados com a ideia de se complementarem neste dado contexto.

4.2. Fase de Complementaridade

Após ter garantido diversos elementos introdutórios, nomeadamente na componente da comunicação, era momento de consolidar algumas das tecnologias web, com um foco grande na *framework* a utilizar. Foram consideradas algumas opções como:

- NextJS (ReactJS)
- SvelteKit (Svelte)
- NuxtJS (VueJS)

Todas as opções mencionadas acima são *frameworks* válidas, modernas e que oferecem características muito idênticas entre ambas, acabando a decisão final por ser mais influenciada pela experiência passada e gosto pessoal. Dito isto, a decisão tomada tendeu-se para NextJS, por ser uma *framework* criada com ReactJS na sua base, utilizado para renderizar as páginas HTML. É de salientar que o objetivo era o de consolidar tecnologias e ao utilizar uma *framework* denominada de *fullstack*, permitiria englobar num único projeto as componentes de *backend* (servidor) e *frontend* (HTML + JavaScript + CSS), sendo que muitos destes princípios estão enraizados nestas tecnologias, o que permitira ter acesso a otimizações interessantes ao invés de ter os elementos separados em projetos distintos. No entanto, antes de começar a desenvolver em NextJS era importante testar alguns dos paradigmas em ReactJS, principalmente na componente da comunicação, por isso a primeira tarefa passou por se desenvolver um componente reutilizável, denominado de *hook*, que permitisse facilmente enviar e receber dados num contexto de websockets. Este hook foi implementado utilizando a biblioteca *socke.io-client* que se complementa bastante bem com a sua versão de servidor e facilita a interação entre ambos.

De seguida, foi o momento de começar a substituir as páginas desenvolvidas para a fase de introdução em HTLM nativo para JSX, utilizado em contexto de React. O foco neste momento estava presente na funcionalidade e utilização de boas práticas e não no aspeto visual ou qualidade da experiência de utilização, garantindo que todas as conexões através de websockets com o servidor NodeJS estavam a funcionar. Na Figura 4.5 – Página Web para simular interação com uma lâmpada, conseguimos observar um exemplo de uma das páginas migradas para a nova *framework*, esta página tem como objetivo simular a interação com um dispositivo presente num PLC, neste caso o dispositivo em causa é uma simples lâmpada, onde o intuito é o de conseguir ligar e desligar a lâmpada, assim como, o de conseguir escutar quaisquer alterações que existam no estado da lâmpada feita por terceiros, é relevante ter em consideração que na oficina Luban continua a existir interação manual com os PLCs, desta forma é importante que a aplicação desenvolvida ao conectar-se a um determinado equipamento consiga representar o seu estado real a qualquer momento. Para este efeito entra novamente em ação a utilização do Tesla Scada, pois permite interagir com os equipamentos funcionando como um bom teste para o caso de uso em questão.

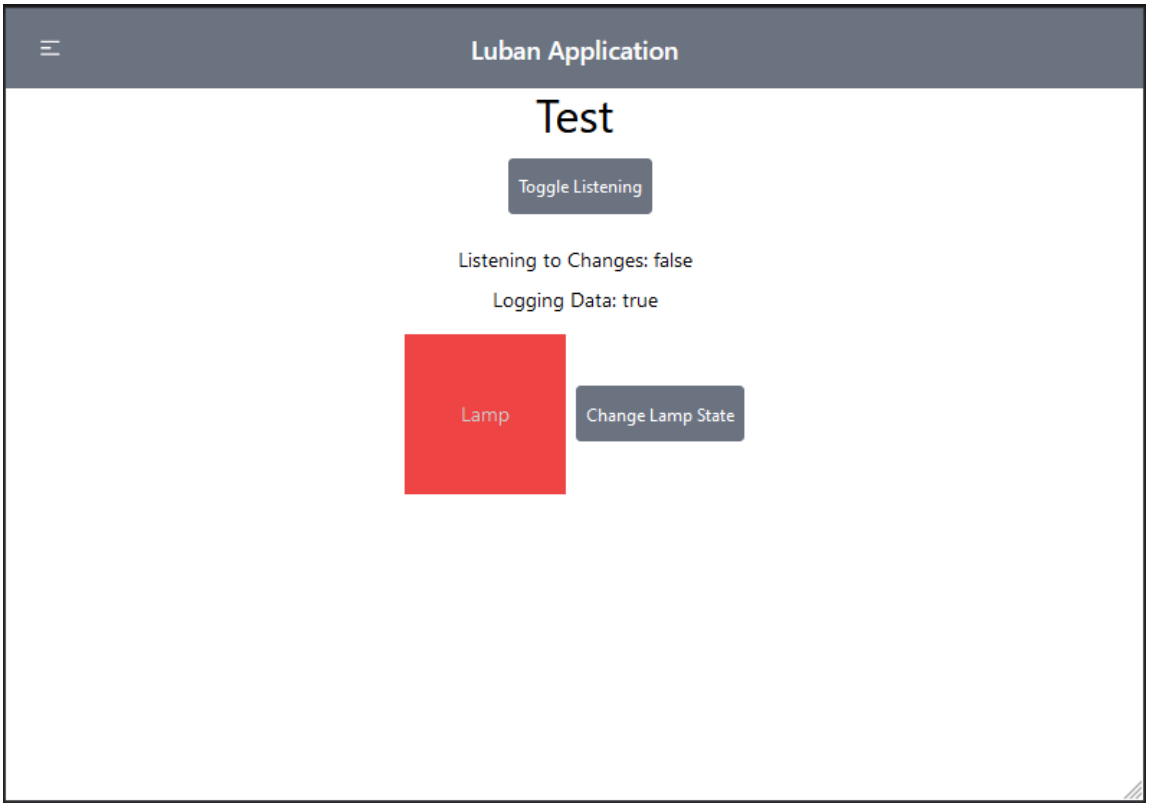


Figura 4.5 – Página Web para simular interação com uma lâmpada num PLC

À medida que estas bases iam ficando cada vez mais sólidas, era o momento de começar a pôr em prática outro requisito para a aplicação, o de ter um sistema de registo de dados para que houvesse um histórico para cada interação dos componentes, na imagem acima é ilusiva a existência de uma variável booleana que indica se um equipamento está a registar dados ou não, por defeito todos os registos de dados estão ativos. Este registo é efetuado em cada momento de leitura ou escrita de dados no componente, os dados a serem guardados englobam o valor processado no momento e obviamente o registo da data e hora, opcionalmente, também é possível registar qual o utilizador que desencadeou o registo. Este último ponto vem numa ótica da de estar preparada para outro requisito que se avizinha, e é naturalmente necessário, autenticação e permissões. Porém, o foco neste momento continua no registo de dados, principalmente no processo de investigação para perceber onde e como os dados seriam guardados, evidentemente entramos na componente de bases de dados. Foi previamente estabelecido, como um requisito, que para a oficina Luban era importante a necessidade de guardar os dados associados aos seus processos, a isto agregamos o registo de dados, no entanto, estamos a falar de tipos de dados diferentes e é relevante perceber se os dados são estruturados e para isso avançamos para uma base de dados relacional ou se os dados são não estruturados e se se adequam melhor a uma base de dados não relacional. A decisão final acabou por ser na utilização de ambas, os tipos de dados e o contexto onde são usados adequam-se a serem usados diferentes tipos de bases de dados, como tal, neste contexto de *logging* optou-se por usar MongoDB para guardar os registos. Estes que acabariam por acontecer com uma carga relevante na escrita de dados, como tal, Mongo acabou por ser uma opção que se adaptaria ao caso

de uso. Para os restantes dados a serem tratados, uma base de dados relacional seria a melhor opção, sendo a decisão tecnológica entre:

- MySQL
- PostgreSQL

Ambas as opções são válidas e bastante idênticas, as suas diferenças expressam-se mais na forma como os seus motores de base de dados estão implementados, sendo PostgreSQL uma opção com mais flexibilidade no tipo de dados que se consegue guardar. Contudo, para o nosso caso de uso, decidiu-se optar por MySQL por ser mais simples, puramente relacional e ser suficiente para o contexto da Luban. Após estarem definidas as bases de dados, começou-se a tratar de criar uma API RESTful dedicada aos CRUDs associados aos processos da Luban, sendo esta uma componente mais de *backoffice* para que seja possível ter a informação dos processos informatizada.

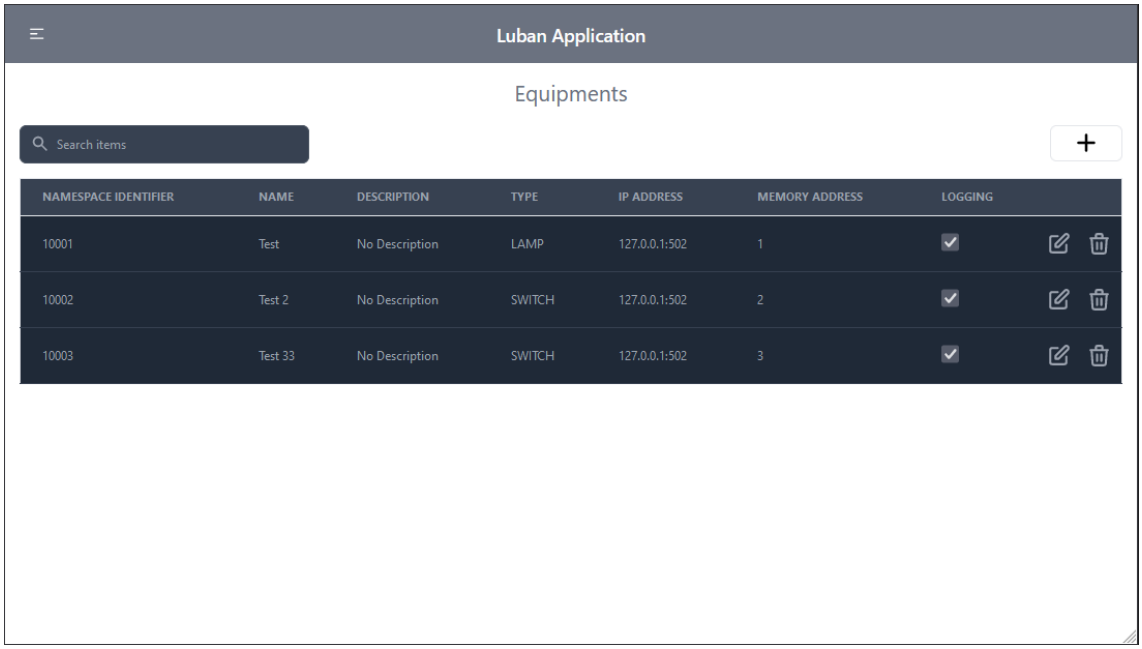


Figura 4.6 – Primeira Implementação para a lista de processos do tipo equipamento

É relevante mencionar que estas listagens foram replicadas para os 4 tipos de elementos semânticos que constituem um processo:

- *Island*
- *Station*
- *Sub-station*
- *Equipment*

Sendo o tópico mais relevante, o equipamento, visto que estes são os elementos mais propícios à comunicação por modbus. Esta distribuição conceptual ocorre com o objetivo de suportar um conceito previamente mencionado, denominado de “espaço de nomes”. Esta noção nasce da necessidade de

abstrair a forma como os processos estão montados na Luban para serem mais fáceis de interpretar, especialmente num contexto de software. Na Figura 4.7 – Primeira Implementação da Interface para adicionar elementos à lista de equipamento, é possível observar uma implementação simples de um *modal* que permite adicionar elementos à respetiva lista, este processo repete-se para qualquer um dos 4 elementos mencionados previamente.

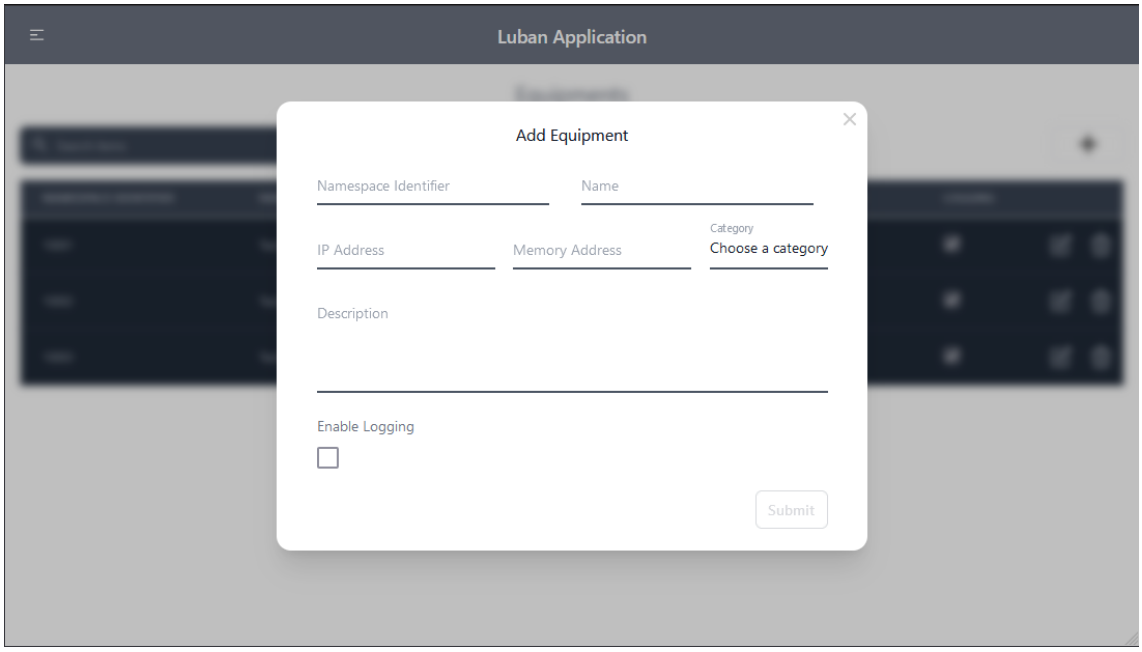


Figura 4.7 – Primeira Implementação da Interface para adicionar elementos à lista de equipamentos

A utilização do conceito de “espaço de nomes” é um dos principais requisitos e funciona como uma forma de categorizar os diversos elementos na oficina. Dando um exemplo concreto, assumindo que uma ilha tem um identificador com o valor 1000, meramente para efeitos de simulação, as estações que possam existir nesta ilha iriam ter um identificador com valores 1100 e 1200, assumindo esta lógica para os restantes elementos, como subestações e equipamentos. Esta estratégia permite mais facilmente identificar as diferentes estruturas hierárquicas dos vários processos, ajudando inclusive num contexto presencial, desta forma até a olho seria possível associar os processos informatizados aos respetivos elementos físicos.

Por fim, a última interface criada nesta fase permite listar todos os diversos elementos existentes no sistema, assim como, de interagir com os mesmos, esta usabilidade é representada na Figura 4.5 – Página Web para simular interação com uma lâmpada num PLC e faz uso de todas os conceitos que foram implementados e testados numa fase anterior, contribuindo para um desenvolvimento relativamente incremental.

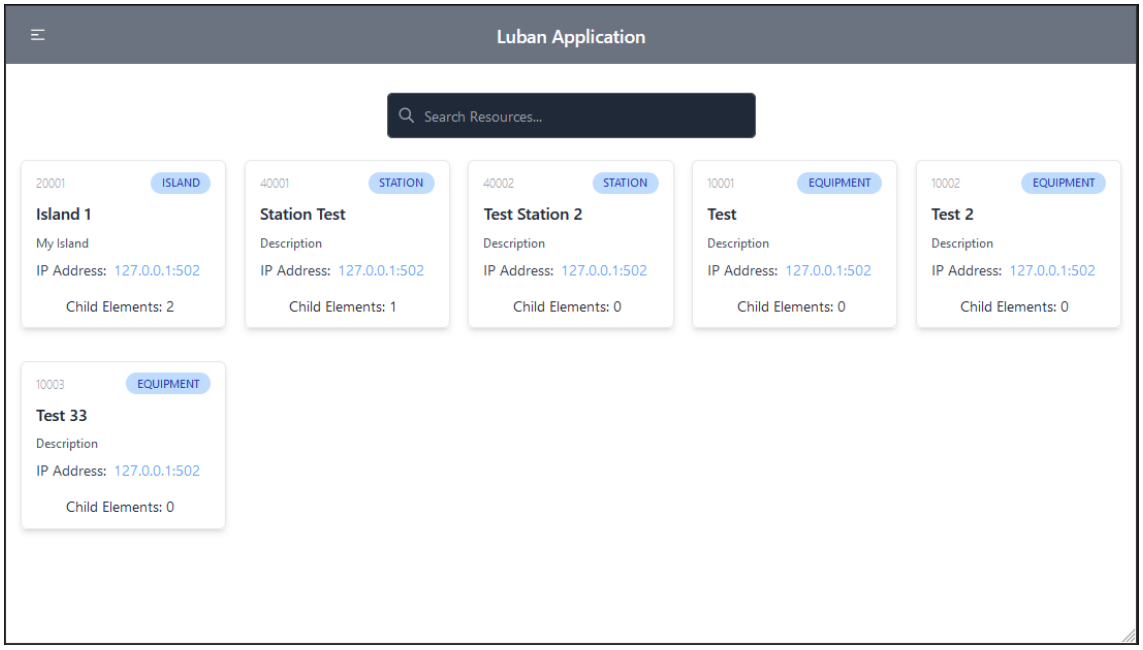


Figura 4.8 – Primeira Implementação da Interface para listar todos os elementos no sistema

4.2.1. Arquitetura

Após algumas introduções de conceitos e tecnologias novas é possível atualizar o diagrama de arquitetura mostrado na Figura 4.9 - Diagrama de Arquitetura para a Fase de Complementaridade, para a seguinte versão.

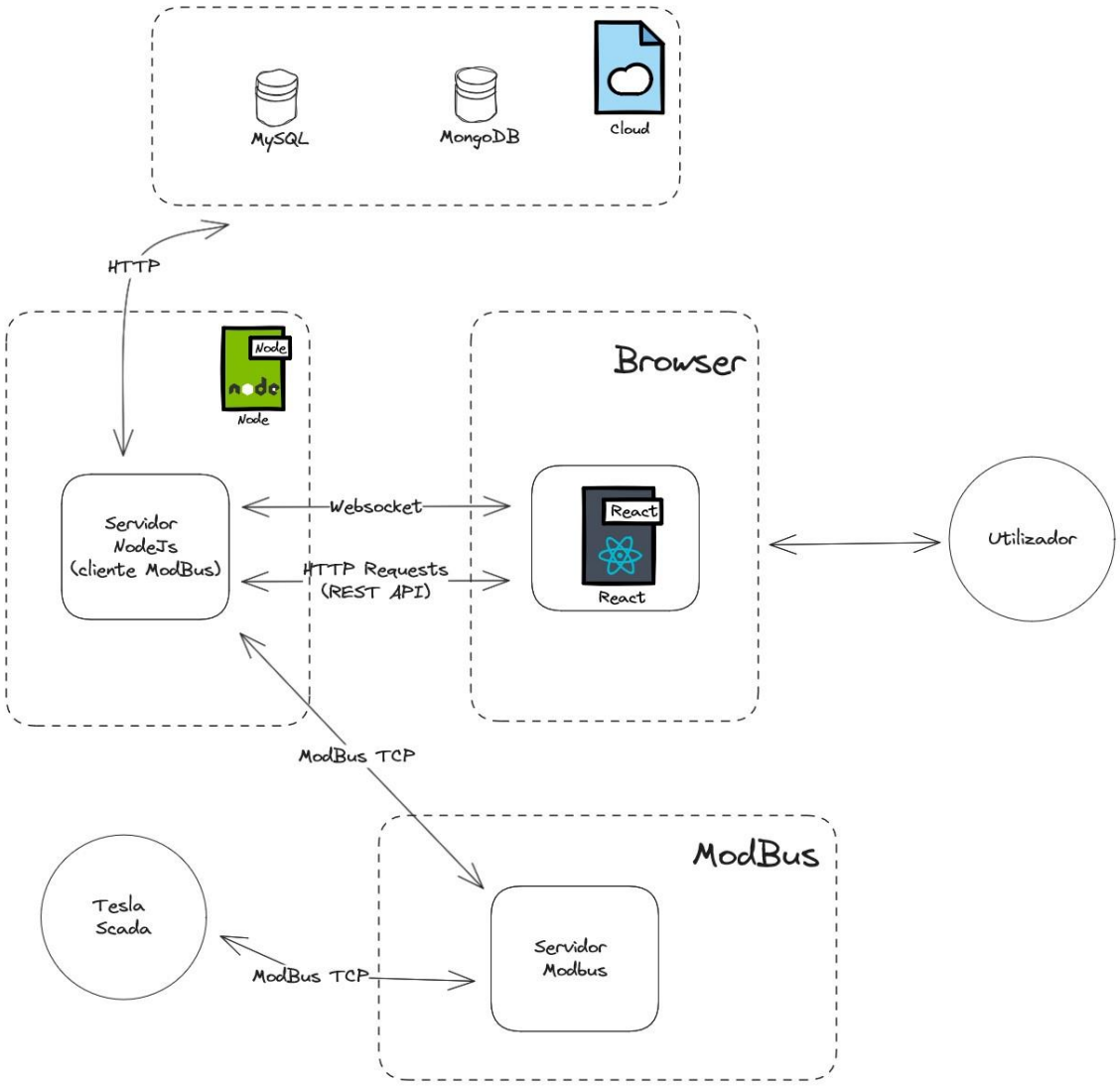


Figura 4.9 - Diagrama de Arquitetura para a Fase de Complementaridade

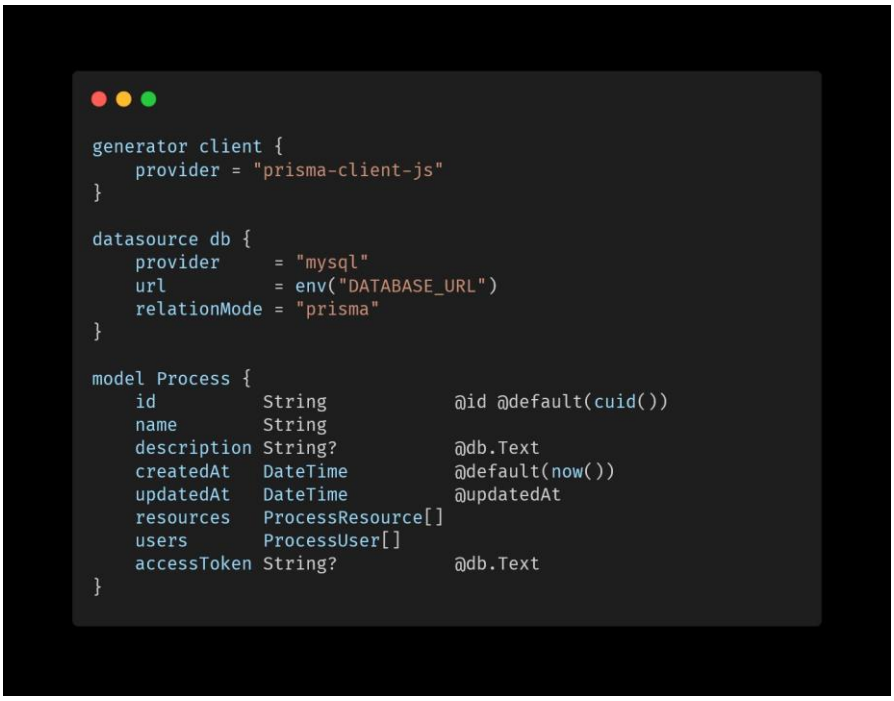
O foco neste diagrama recai maioritariamente sobre a introdução de ReactJS para tratar da componente de *frontend* e das introduções das bases de dados e das respetivas API's para interagir com os dados dos processos.

4.2.2. Tecnologias

Dentro das introduções tecnológicas para esta fase do protótipo, existe um foco muito grande na camada de dados devido à introdução de bases de dados. Tradicionalmente, com a necessidade de comunicar com estas, existem padrões e boas práticas definidas que são recomendados seguir especialmente na integração entre os *endpoints* de uma API (*service layer*) e as operações que se querem realizar na base de dados, estes padrões sugerem a utilização de um ORM (Object Relational Mapping). Historicamente, são usados em contextos de bases de dados relacionais, no entanto, cada vez mais estes padrões são flexíveis e existem ferramentas para contextos de bases de dados não relacionais. No caso de uso para a aplicação da Luban, foram usadas 2 ferramentas muito similares:

- Prisma (MySQL)
- Mongoose (MongoDB)

As razões que levaram a escolher estas bibliotecas são um misto entre experiência passada e simplesmente tentar escolher a ferramenta que melhor se adequa ao trabalho a realizar, neste caso, mongoose, por exemplo, é pensado e otimizado para um contexto de MongoDB daí a escolha ser obvia. Prisma, por outro lado, é mais flexível, mas o seu intuito é pensado para contextos de bases de dados relacionais onde se enquadra a nossa escolha tecnológica de MySQL. Olhando para algumas das diferenças entre ambas, Prisma está dependente da criação de uma *schema* usando uma linguagem própria. Esta *schema* tem similaridades com um modelo relacional da base de dados e é depois usada para gerar as conexões às tabelas que foram definidas, na Figura 4.10 – Exemplo da criação de um modelo na *schema* de Prisma, podemos ver um excerto da definição de uma tabela(*model*) na schema.



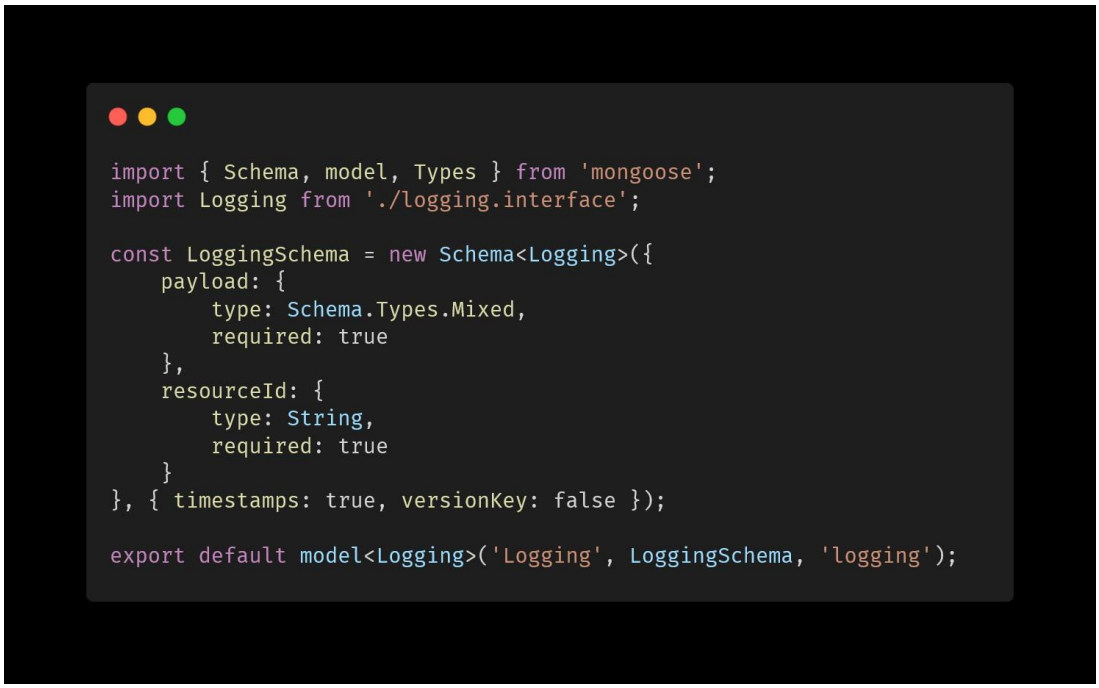
```
generator client {
    provider = "prisma-client-js"
}

datasource db {
    provider     = "mysql"
    url          = env("DATABASE_URL")
    relationMode = "prisma"
}

model Process {
    id          String            @id @default(cuid())
    name        String
    description String?           @db.Text
    createdAt   DateTime          @default(now())
    updatedAt   DateTime          @updatedAt
    resources   ProcessResource[]
    users       ProcessUser[]
    accessToken String?           @db.Text
}
```

Figura 4.10 – Exemplo da criação de um modelo na *schema* de Prisma

Outra das principais razões pelas quais se adota Prisma é pela forma nativa como TypeScript está incorporado na sua utilização. Dois conceitos que andam de mãos dadas quando se utiliza Prisma em contexto de TypeScript são os tipos genéricos e inferência, graças a estes Prisma consegue fazer uso dos LSP's (*Language Server Protocols*) presentes nos editores de código e providenciar sugestões uteis que partem daquilo que é definido na *schema* inicial. É relevante mencionar que grande parte daquilo que o Prisma faz é alcançado com geração de código. Mongoose, por outro lado, utiliza uma abordagem mais tradicional através da escrita de código, os modelos criados são definidos em TypeScript, e fazem parte da estrutura geral do projeto.



```
import { Schema, model, Types } from 'mongoose';
import Logging from './logging.interface';

const LoggingSchema = new Schema<Logging>({
    payload: {
        type: Schema.Types.Mixed,
        required: true
    },
    resourceId: {
        type: String,
        required: true
    }
}, { timestamps: true, versionKey: false });

export default model<Logging>('Logging', LoggingSchema, 'logging');
```

Figura 4.11 – Exemplo da criação de um modelo em Mongoose

Ao utilizar ambas as opções mencionadas, temos acesso a funções que abstraem a construção de *queries* às bases de dados, exemplos de utilização podem ser observados na figura abaixo.

```
//Mongoose
return await logging.findById(id);

//Prisma
return await prisma.process.findUnique({ where: { id } });
```

Figura 4.12 – Exemplo de utilização dos ORM's

4.3. Fase de Ajustes Finais

Por fim, entramos na fase final do protótipo, esta fase acabou por ser a mais trabalhosa, não em termos de complexidade, mas em termos de tempo, o tipo de trabalho que se realizou foi numa ótica de “ajustes” e acabamentos e acabou por ser relativamente extenso. Dito isto, após ter sido finalizado o trabalho na camada de dados e termos as bases montadas na interface em ReactJS estava no momento de se migrar e consolidar o trabalho na *fullstack framework* mencionada anteriormente, NextJS. Esta *framework* oferece uma componente nativa de servidor que está inerentemente integrado com a componente de renderização das páginas web em ReactJS, graças a isto temos acesso a funcionalidades como:

- SSR (*Server-Side Rendering*)
- CSR (*Client-Side Rendering*)
- ISR (*Incremental Static Regeneration*)
- SSG (*Static Site Generation*)

Estes conceitos são essenciais na web moderna e fazem parte de um arsenal de ferramentas que NextJS oferece, em conjunto com diversas otimizações. Um dos pontos positivos é que muitas destas noções são opcionais e permitem ter uma utilização por página, sendo possível escolher a que melhor se adapta ao caso de uso da página em questão. Por exemplo, SSR é muito bom para efeitos de SEO, no entanto, visto esta aplicação ser utilizada num contexto privado, não é relevante de todo, mas algo como CSR tem um foco muito grande na interatividade das páginas e adapta-se melhor para algo como a Luban.

Após ter sido feita a migração dos projetos independentes da fase anterior, obtemos a seguinte *stack* tecnológica final:

- NextJS
- TailwindCSS
- MySQL/MongoDB
- TRPC

Tendo em consideração toda a camada de comunicação criada anteriormente com *sockets* e modbus, a única tecnologia introduzida neste processo foi TRPC (*TypeScript Remote Procedure Call*). Uma biblioteca ideal para o contexto de TypeScript e que ajuda a criar uma definição da API, sendo a ponte perfeita entre o cliente e o servidor do ponto de vista de desenvolvimento, reduzindo a chance de bugs e aumentado a produtividade.

Feitas as introduções tecnológicas, o primeiro passo foi o de introduzir conceitos básicos, mas necessários para uma aplicação deste género, começando pela implementação de um sistema de autenticação, sistema este que foi incorporado na base de dados MySQL existente permitindo que a Luban seja dona do seu modelo de autenticação e tudo o que lhe é inerente como gestão de sessões, permissões através de *roles* e processos de login, recuperação de password, etc. Na Figura 4.13 – Página para Login no protótipo da Luban, é possível observar a página de login criada para este efeito. Foi também criado um simples sistema de notificações, acessível após autenticação, no

canto superior direito das páginas e que reage após certas ações, com prospeção de expansão.

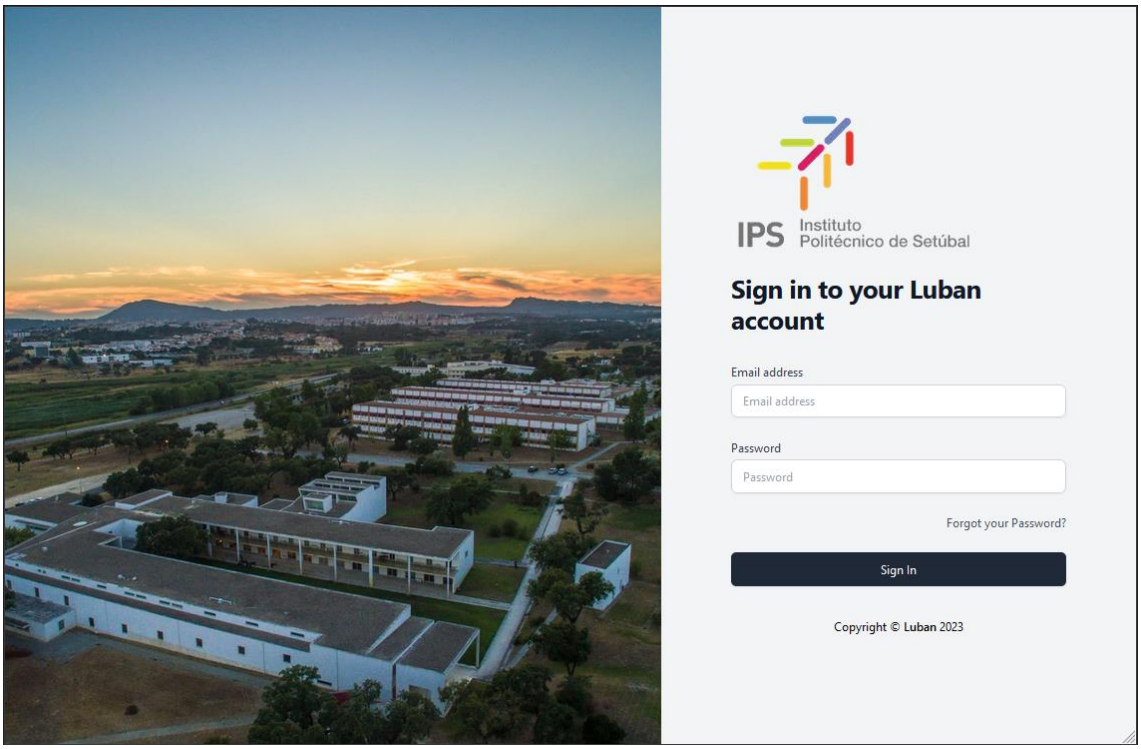


Figura 4.13 – Página para Login no protótipo da Luban

Para suportar toda a componente de autenticação e gestão de utilizadores foi criada uma interface para ajudar a gerir os utilizadores, nomeadamente na criação de novos, ativação ou gestão de permissões. Esta página está acessível apenas para administradores.

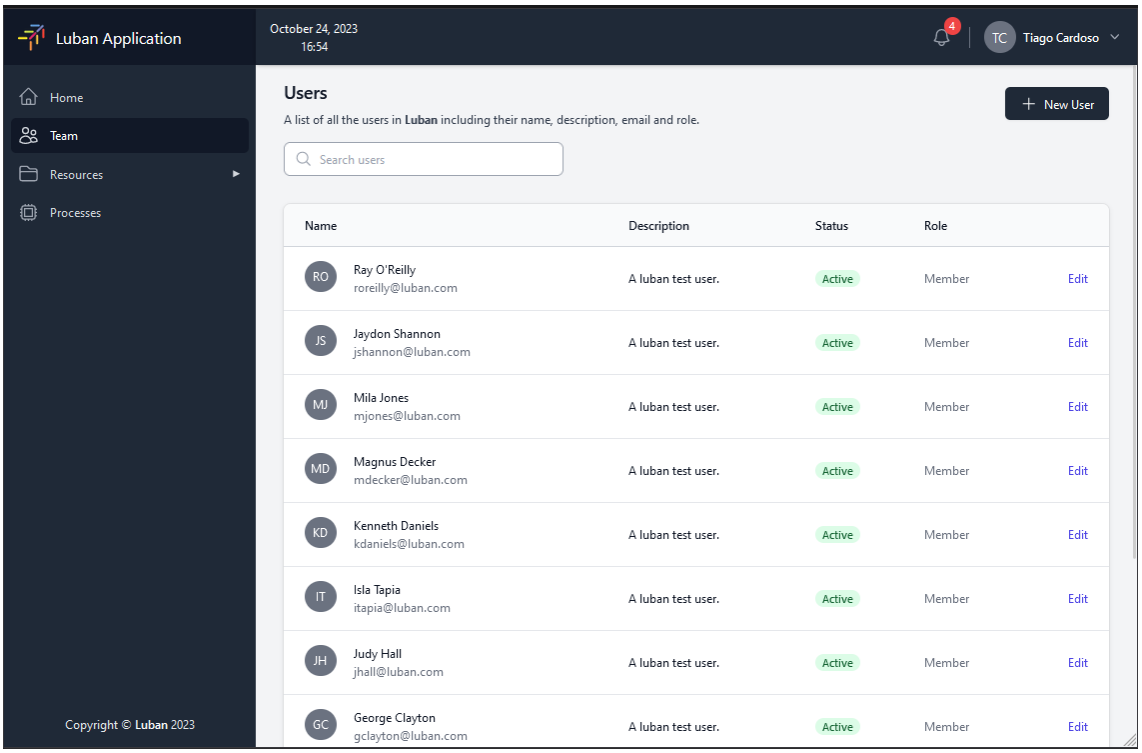


Figura 4.14 – Página para gestão de utilizadores da Luban

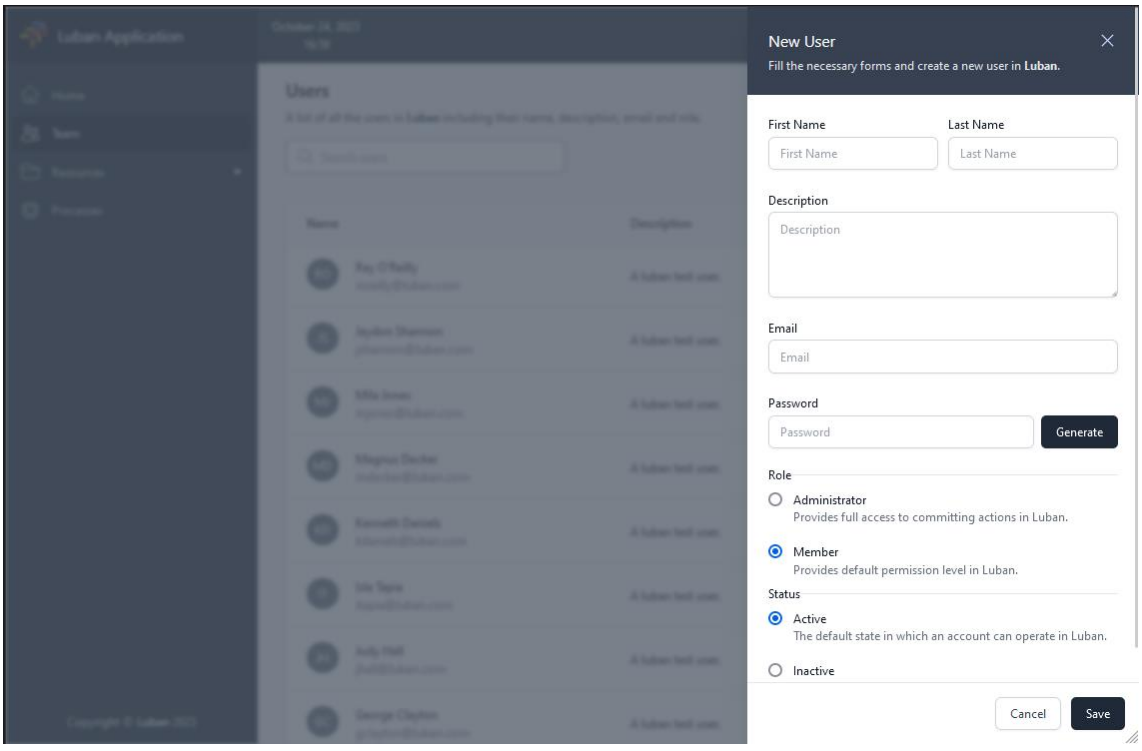


Figura 4.15 – Página para criar um utilizador no sistema

Após um utilizador ser criado, deve ser enviado um email com os dados para se autenticar pela primeira vez. As ações CRUD associadas aos processos da Luban também receberam uma atualização visual seguindo o tema presente nas recentes imagens.

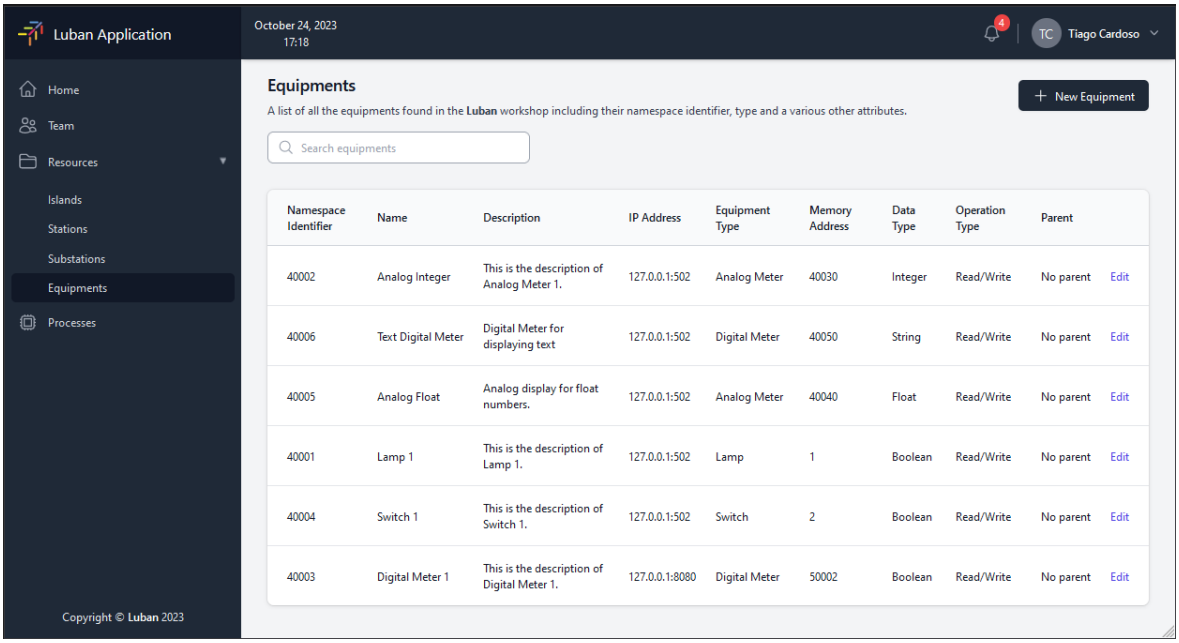


Figura 4.16 – Página para listar os recursos do tipo equipamentos

Como é possível observar na Figura 4.16 – Página para listar os recursos do tipo equipamentos, o processo de listagem e criação de recursos segue um tema visual muito semelhante ao observado para os utilizadores, sendo uma boa forma de reutilizar componentes em ReactJS para diferentes contextos. Por fim, as páginas dedicadas à interação com os equipamentos através de modbus também sofreram alterações visuais para o contexto dos testes realizados.

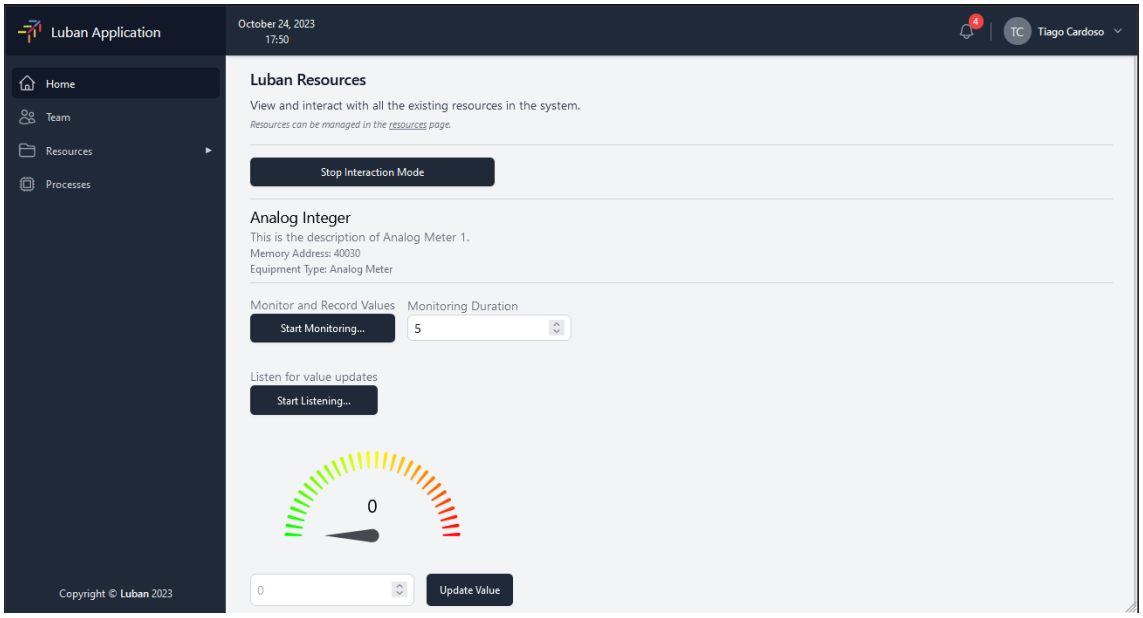


Figura 4.17 – Página para simulação da interação com um equipamento por modbus

O último requisito implementado para o protótipo foi o de criar um mecanismo que através de uma API no servidor, permite interagir com os processos em modbus sem utilizar necessariamente a interface visual observada na Figura 4.17 – Página para simulação da interação com um equipamento por modbus. Este sistema parte do princípio de que o acesso é controlado, como tal, todos os pedidos feitos à API devem conter no seu cabeçalho alguma chave de acesso ou algo semelhante que valide a autenticidade de um pedido para além de garantir que existam as permissões necessárias. Para efeitos de teste, foi criada uma interface que permite gerar um *token* de acesso, este permite escolher quais os equipamentos que podem ser acedidos e com que permissões. Para efeitos de registo e controlo, o *token* tem associado qual o utilizador que o gerou. A API que interage com o modbus acaba por ser a mesma utilizada pela interface visual, simplesmente foi adaptada para aceitar pedidos externos, validando os mesmos e garantindo que toda a informação necessária para o pedido é incluída no pedido.

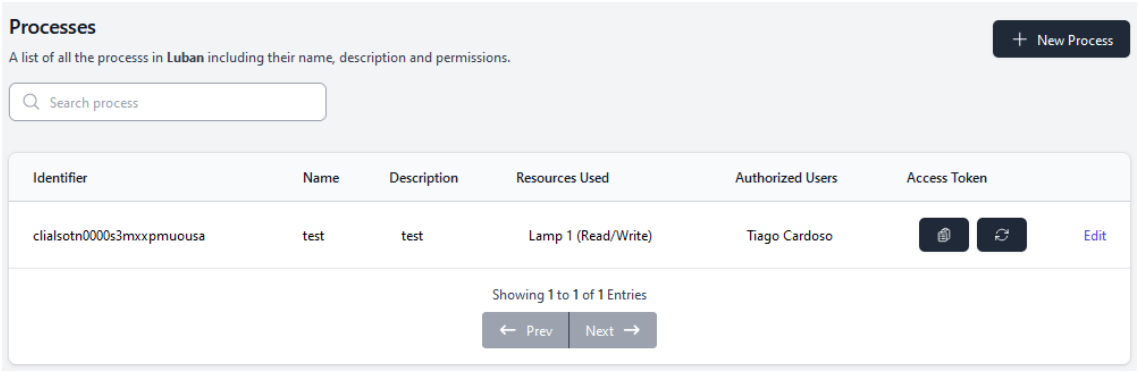


Figura 4.18 – Listagem dos processos para acesso externo

Na Figura 4.18 – Listagem dos processos para acesso externo, podemos observar dois botões na coluna "*Acess Token*", estes permitem copiar o *token* que é gerado ou gerar um novo caso o antigo fique comprometido. Estas chaves são baseadas em JWT (*JSON Web Tokens*), considerado um padrão standard para criar dados com assinatura criptográfica.

Por fim, visto estamos novamente num contexto sem interface visual, a utilização do Tesla Scada como suporte visual foi novamente usado para este caso de uso.

4.3.1. Arquitetura

Após todas as consolidações tecnológicas, o diagrama abaixo reflete a infraestrutura final definida para o protótipo.

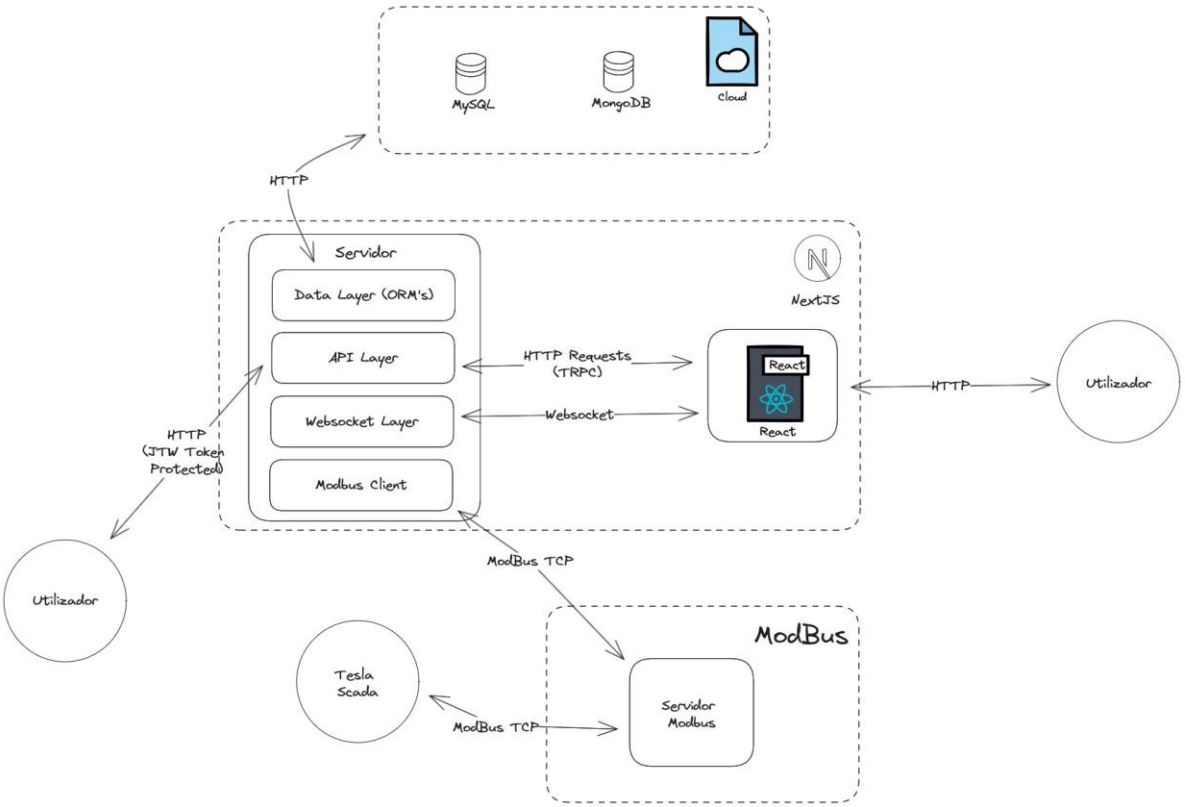
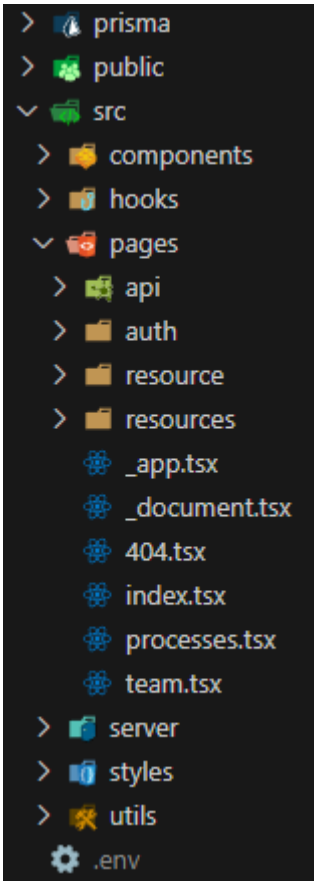


Figura 4.19 – Diagrama de Arquitetura para a Fase de Ajustes Finais

4.3.2. Tecnologias

Para esta fase final, do ponto de vista tecnológico, a maior introdução foi o NextJS. Esta framework, por muitos considerada de meta-framework, para ReactJS, oferece praticamente todas as características necessárias para construir aplicações web modernas. ReactJS está pensado para construir interfaces web, tem acesso a um ecossistema enorme de pacotes e bibliotecas externas que vêm complementar alguns défices, no entanto, NextJS montado em cima de ReactJS colmata nativamente muitos dos problemas existentes no desenvolvimento de aplicações webs mais complexas, especialmente graças a sua componente de servidor. Para além disso, tem acesso ao ecossistema previamente existente para ReactJS, mas com a capacidade de expandir em cima desse para oferecer formas mais eficientes de criar aplicações para vários contextos. Algumas das funcionalidades adicionadas nativamente foram:

- Componente de Servidor (mencionado previamente e que oferece toda uma forma de renderizar páginas que não era possível anteriormente);
- *Routing* Dinâmico (baseado no sistema de ficheiros do projeto);
- Preparado para SEO (Search Engine Optimization) devido à componente de servidor em comparação às tradicionais SPA's (Single Page Applications) onde esta funcionalidade era bastante limitada;
- Otimização de Imagens e Fonts;
- Otimizações no momento de fazer pedidos para obter dados, especialmente se a API usada for a existente no NextJS;
- Entre muitas outras características que melhoram a qualidade de vida durante o desenvolvimento de aplicações;



Nesta figura, para além de estar presente a organização estrutural de um projeto em NextJS, podemos observar como tirar partido da funcionalidade de *Routing* dinâmico. A pasta *"pages"* é utilizada pela framework para interpretar quais os pedidos válidos que são feitos no browser, como exemplo, ao ser feito um pedido ao endereço *"localhost:300/team"* o ficheiro *"team.tsx"* é utilizado e os seus conteúdos são mostrados na página. Qualquer pedido realizado que não se enquadre com nenhum dos ficheiros existentes é automático considerado de 404 – *Not Found*. Na imagem é possível observar a opção de customizar o que é mostrado na página 404, se não for definida nenhuma página para tal é mostrada uma pré-definida. Páginas como *_app* e *_document* são páginas reservadas para a framework, bem como a página *"index.tsx"* que acaba por ser a página introdutória da nossa aplicação.

Figura 4.20 – Estrutura de ficheiros do projeto em NextJS

Capítulo 5

Proposta de Arquitetura de Referência para a Luban

Com base nas experiências realizadas, principalmente na componente de protótipo, e nas aprendizagens obtidas após diversos momentos de investigação, no capítulo dos fundamentos teóricos e tecnológicos, para os vários conceitos arquiteturais chegamos ao momento de sugerir uma arquitetura de alto nível que se enquadra com o contexto da oficina Luban e que esteja preparada para responder às necessidades atuais e possivelmente futuras da mesma. Esta arquitetura deve ter uma estrutura modular que possa ser facilmente adaptada com o passar do tempo, e para isso deve ser montada sob um misto de conceitos arquiteturais bem estabelecidos e sólidos, inspirados em boas práticas definidas pelas arquiteturas de referência existentes em ambientes similares. A sua investigação prévia tem como principal objetivo inspirar a definição, sempre que possível, de muitos dos princípios estabelecidos neste trabalho.

Dito isto, o propósito desta arquitetura é o de conseguir responder aos diversos problemas identificados na Luban, e que foram relatados ao longo deste trabalho, deve ser montada de forma a suportar futuras implementações técnicas com base em vários conceitos de engenharia de software utilizados para criar infraestruturas neste âmbito tecnológico. O objetivo é que possa conter fundamentos centrados em muitos dos bons padrões arquiteturais mais conhecidos, que continuam a manter-se relevantes para responder aos desafios da atualidade, não esquecendo que, o objetivo desta arquitetura é ser flexível o suficiente para encapsular todos os problemas atuais da Luban, mas também estar preparada para crescer e suportar novos conceitos.

5.1. Contexto Geral

Antes de se avançar para diagramas ou desenhos lógicos que representam uma arquitetura, é necessário contextualizar o nível de abstração que o diagrama pretende representar e para isso é necessário olhar para os níveis de abstração que fazem sentido para o nosso problema.

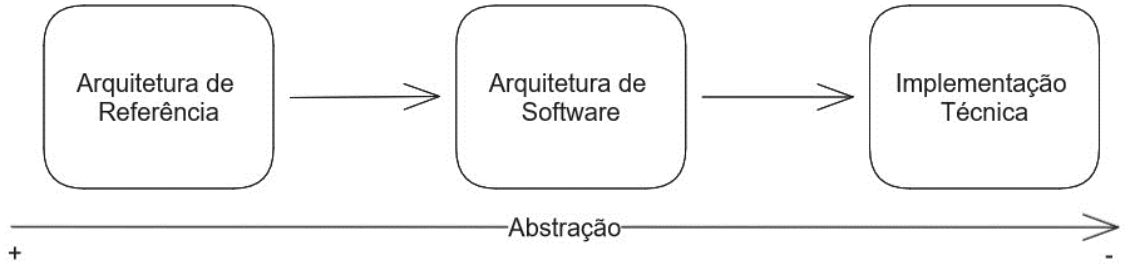


Figura 5.1 – Diagrama para demonstrar os níveis de abstração

Observando a Figura 5.1 – Diagrama para demonstrar os níveis de abstração, conseguimos representar o diagrama da arquitetura sugerida consoante o seu nível de abstração, separando-o em 3 níveis, de mais abstrato para menos abstrato (concreto):

- Arquitetura de Referência
- Arquitetura Software (para implementar os diversos elementos da arquitetura de referência)
- Implementação Técnica

Com base nisto, o objetivo deste capítulo passa por apresentar uma arquitetura lógica, focar-se nos diferentes elementos que a compõem, suportando-os com fundamentos teóricos e dando alguns exemplos de possíveis arquiteturas de software a utilizar dependendo do contexto em que o elemento em questão se encontra. As sugestões dadas são apresentadas com base nos problemas previamente identificados, tentando, sempre que possível, correlacionar as aprendizagens observadas nas arquiteturas de referência exploradas no capítulo dos fundamentos teóricos e tecnológicos, com conhecimentos de arquiteturas de software para responder as todos os problemas identificados na Luban.

5.2. Diagrama da Arquitetura de Referência

A figura seguinte, descreve o diagrama da arquitetura sugerida. Como foi mencionado previamente, este foi criado de forma mais abstrata por se tratar de uma representação lógica num nível mais alto de abstração, pretende-se que a ideia a transmitir seja clara e facilmente interpretada. O objetivo não é o de influenciar decisões técnicas, mas sim o de apresentar uma estrutura que consiga corresponder um conjunto de necessidades especificas independentemente das decisões arquiteturais, sejam de alto nível ou mais concretas, como por exemplo tecnologias específicas.

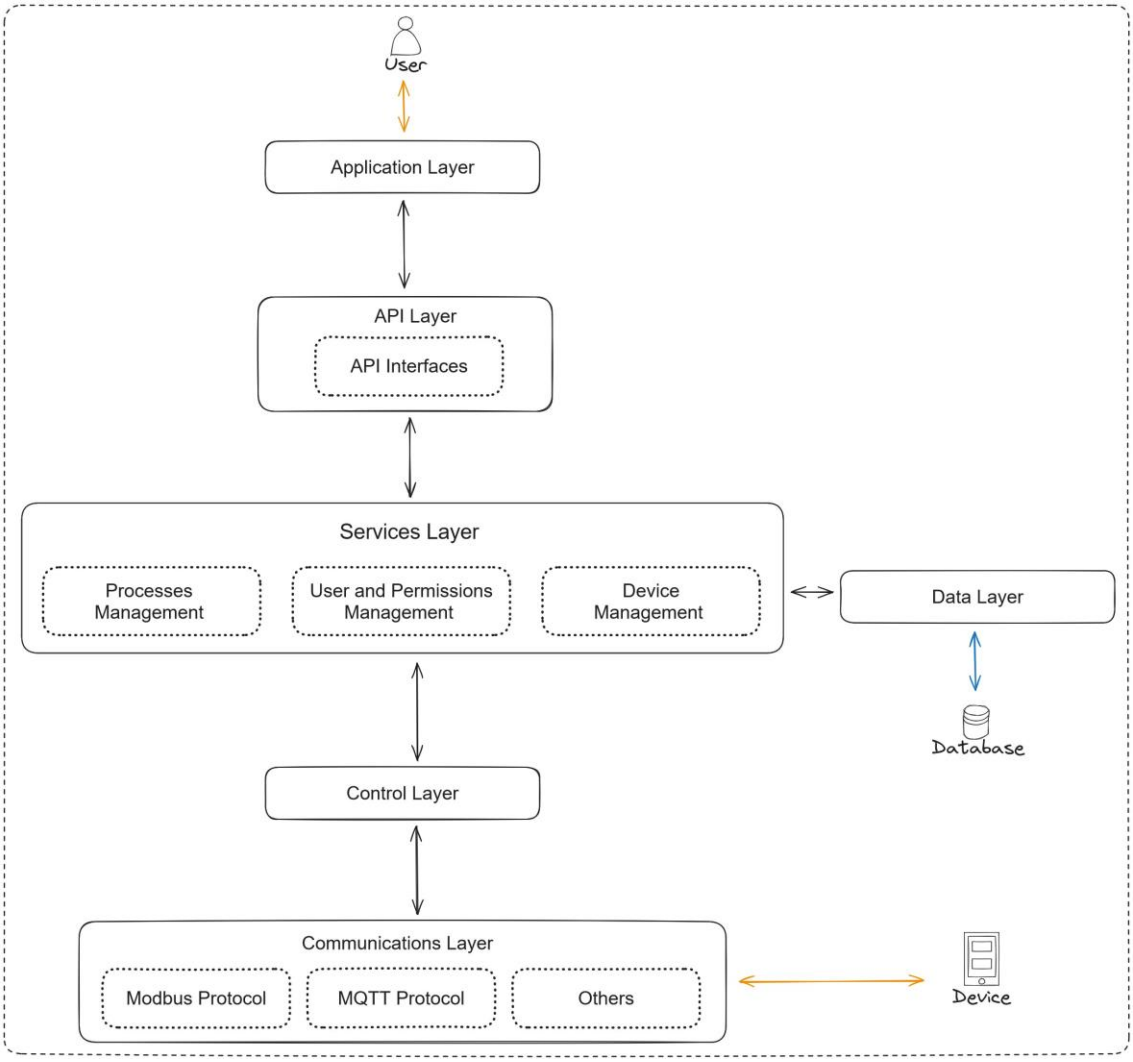


Figura 5.2 – Diagrama para a Arquitetura final sugerida

5.3. Estrutura Lógica

Do ponto de vista lógico, foram definidos alguns componentes de alto nível que são responsáveis por abstrair diferentes comportamentos específicos, desde o instante em que inputs são realizados pelo utilizador ao momento em que algo é guardado numa base de dados e a todo o processo de comunicação com os diferentes equipamentos (PLC's).

Os componentes definidos enquadram-se numa ótica de camadas de forma a simplificar a comunicação entre as diferentes partes lógicas, por norma cada camada apenas deve comunicar com a respetiva camada acima ou abaixo. Um ponto importante a realçar é que esta estrutura por camadas não deve ser confundida com uma arquitetura de software por camadas, apesar das semelhanças, a nossa arquitetura representa um nível abstrato mais elevado e oferece flexibilidade nos momentos mais concretos.

Dito isto, um dos principais objetivos passa por tentar seguir, sempre que possível e caso faça sentido, algumas das boas práticas inerentes à utilização de uma arquitetura por camadas. Para além da comunicação, conseguimos também abstrair a complexidade do sistema em diferentes etapas ajudando também o processo de desenvolvimento, é mais fácil desenvolver o sistema por fases de acordo com as camadas, partindo do princípio que as suas responsabilidades estão bem definidas e isoladas, não menosprezando o possível impacto que esta distribuição de responsabilidades trará no momento de corrigir possíveis bugs no sistema. Por último, é expectável que as camadas possam ser substituídas por outras, se assim fizer sentido, e que não seja comprometido o normal funcionamento do sistema.

Um ponto relevante nesta arquitetura é que, ao serem definidas camadas lógicas, podemos assumir que cada uma terá uma implementação independente e que, obviamente, se adeque melhor ao contexto da mesma, podendo utilizar diversas arquiteturas de software para definir as diferentes partes, comunicando depois entre si.

Com base nisto, e analisando a Figura 5.2 – Diagrama para a Arquitetura final sugerida, conseguimos identificar 6 camadas que compõem a arquitetura final, estas são:

- *"Application Layer"*, interpretada como camada de aplicação, é sob esta camada que recai a responsabilidade de abstrair as diferentes formas de interação com o sistema, tradicionalmente feita através de uma interface de utilizador, num contexto web ou não, ou até mesmo em ambos, existe flexibilidade para tal desde que a comunicação com a camada abaixo assim o permita. Caso o mecanismo de interação seja através de uma interface web, é neste contexto que é sugerido explorar as tecnologias de *frontend* atuais e perceber o que melhor se adapta às necessidades da implementação especifica, naturalmente dependendo das circunstâncias dos requisitos desejados. Esta camada acaba por ser o meio de interação com uma interface de serviços para aceder ao sistema, seja este acedido por meio de uma interface visual ou não.
- *"API Layer"*, funciona como elemento de ligação entre os diferentes serviços na camada abaixo, e as diferentes interações realizadas através da camada de aplicação. É conhecedora das diversas interfaces no sistema e trata de fazer o garantir a

interoperabilidade do sistema através das diferentes interfaces, para além disso, no contexto da Luban é relevante constatar que a utilização de websockets é um ponto importante, como tal, esta camada também é uma boa localização para as definições dos diversos eventos no sistema. Tradicionalmente o acesso a uma web API é realizado sobre a forma de uma arquitetura REST, porém neste âmbito há oportunidade para inovar e procurar alternativas se assim fizer sentido, cada vez mais surge como alternativa a framework *gRPC*, é uma arquitetura montada sob o protocolo de comunicação RPC (Remote Procedure Call) que faz uso de alguns conceitos da web mais modernos e eficientes.

- *"Services Layer"*, camada responsável por semanticamente organizar um conjunto de serviços agrupados em 3 categorias:
    - "*Processes Management*", para organizar a lógica de comunicação com os equipamentos, monitorização dos mesmos e tudo o que constitui um processo no contexto da Luban.
    - *"User and Permissions Management",* como o nome indica, estamos a tratar de agrupar os serviços responsáveis pela autenticação, gestão de utilizadores e controlo de acessos através de um sistema de permissões.
    - *"Device Managament",* agrupar todas as entidades semânticas da Luban (Ilhas, Estações, Subestações, Equipamentos) em serviços isolados que tratam de gerir a sua documentação, manuais e possível distribuição física de forma independente.

Do ponto de vista de implementação, estamos obviamente a olhar para uma arquitetura de microserviços, onde cada serviço deve ter uma responsabilidade concreta e isolada. Esta tipologia permite um desenvolvimento e respetiva operação de forma independente para cada serviço, proporcionando agilidade no processo e facilitando futuras alterações. Relevante para uma arquitetura deste género é a necessidade de existir um "orquestrador de serviços", é importante que existe uma entidade central que possa ditar o fluxo das mensagens no contexto dos serviços.

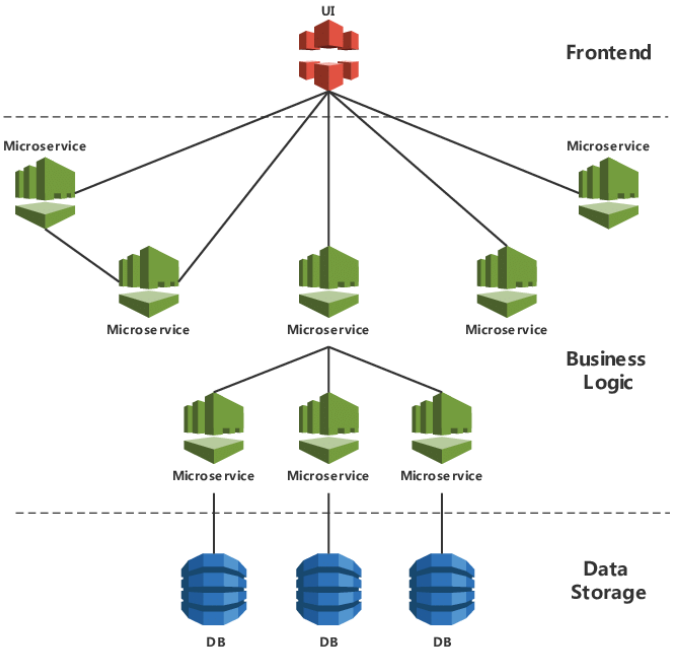


Figura 5.3 – Exemplo de uma arquitetura em Microserviços

- "*Data Layer*", camada de dados, objetivamente estamos perante a camada responsável por controlar como o acesso aos dados é feito e gerido. Para o contexto da Luban, de acordo com o tipo de dados, estruturados ou não estruturados, existe flexibilidade para escolher qual o motor de bases de dados a utilizar, tendo como base as experiências realizadas no protótipo é perfeitamente aceitável ter bases de dados relacionais e não relacionais na mesma infraestrutura. A camada de serviços é a única que pode interagir com a camada de dados de forma direta, o contexto de cada serviço dita o tipo de dados que são necessários e é com base nisso que depois é decidida qual a base de dados a utilizar. Porém, nesta camada existem alguns padrões recomendados seguir para garantir que existe um acesso lógico aos dados, tradicionalmente podemos encapsular esta lógica através do padrão "Repository", funcionado como camada de tradução com uma ou mais linguagens especificas de bases de dados, em muitos casos também faz sentido emparelhar este conceito com um ORM (Object-Relational Mapping), um padrão muito comum para abstrair a interação com uma base de dados. Desta forma, esta camada pode disponibilizar para o exterior repositórios que detêm todas as funções necessárias para interagir com uma base de dados especifica.

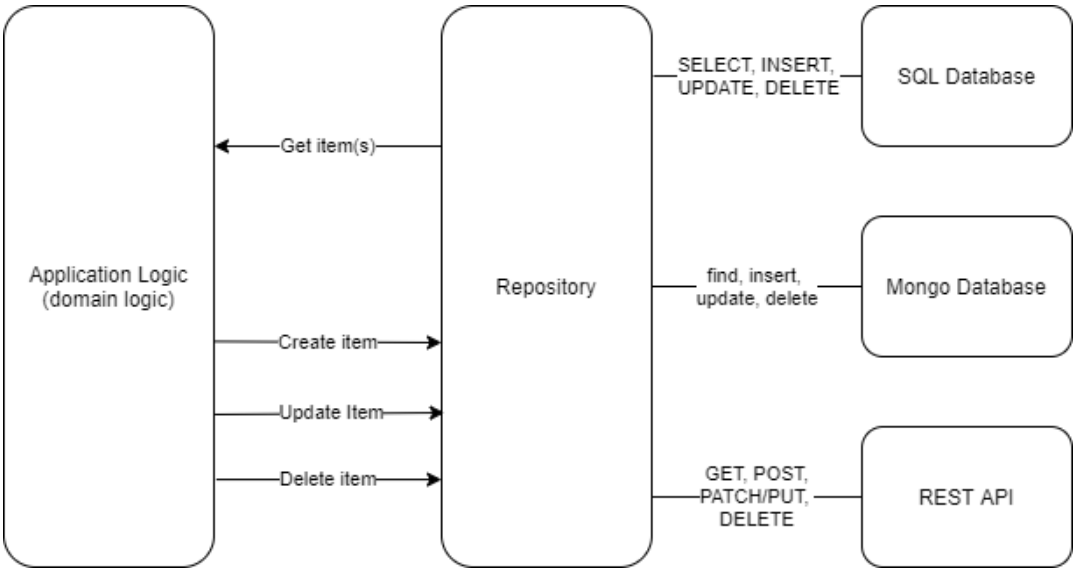


Figura 5.4 – Exemplo de uso do padrão "Repository"

- *"Control Layer"*, camada responsável por definir o controlo dos processos e a forma como fluem dentro da arquitetura. Trata de gerir a aquisição de dados ao comunicar com serviços necessários que por sua vez interagem com a camada de dados. Esta informação é utilizada na troca de dados com os equipamentos, através da camada de comunicação. É relevante mencionar que toda a informação deve ocorrer em tempo real e o sistema tem de ser capaz de suportar este fluxo de informação, principalmente no contacto com os equipamentos, é importante que em momentos de monitorização, leitura ou escrita nos equipamentos, que toda esta informação flua naturalmente. Para tal, temos arquiteturas de alto nível eficientes para este caso de uso. Com base no que foi falado recentemente, esta arquitetura tem um foco muito grande em serviços, principalmente na cama de serviços, como tal, a utilização de arquitetura orientada a serviços é uma realidade aceitável. Foi previamente comentado na camada de serviços a necessidade de ter um elemento nesta infraestrutura que tenha a responsabilidade de "orquestrador de serviços", é nesta camada de controlo que estamos perante essa realidade. Como alternativa, podemos também olhar para uma arquitetura mais moderna baseada em eventos, "*Event-Driven Architecture"* (EDA), nesta tipologia tudo o que ocorre no sistema deve ser considerado um evento e as diferentes partes reagem consoante os eventos que necessitem. Neste caso é necessário que exista um elemento central que funcione como "*Message Broker*", este tem a função de ser o elemento facilitador na comunicação por eventos.

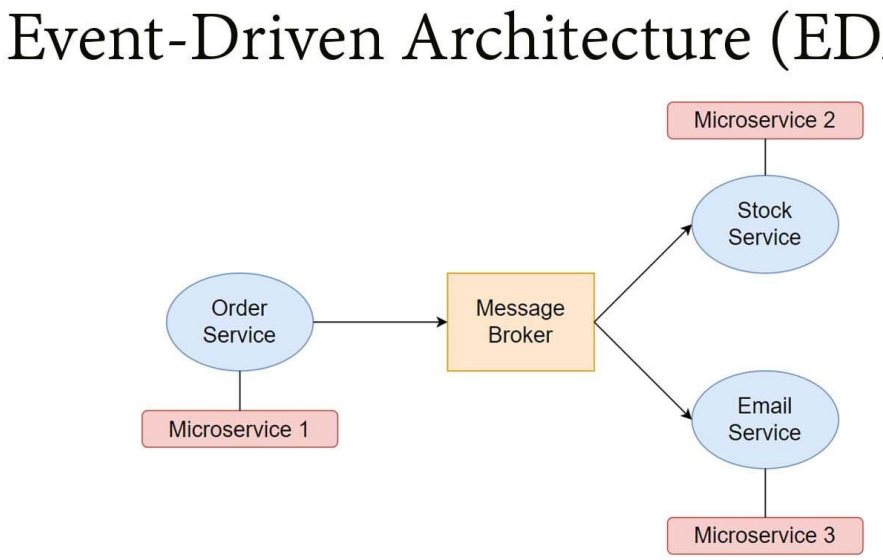


Figura 5.5 – Exemplo de EDA com um “Message Broker”

- *“Communications Layer”*, estamos perante a camada que trata de fazer a ponte entre o sistema e os processos com os seus respetivos equipamentos na Luban. Esta camada é responsável por conter os diversos adaptadores para as várias formas de comunicação que possam vir a ser necessárias na infraestrutura. Tendo como ponto de partida os trabalhos realizados no protótipo, esta camada será ideal para conter toda a lógica necessária para realizar e receber pedidos por modbus, abstraindo para o exterior os detalhes de implementação associados à comunicação especifica. Este tipo de implementação permite que as formas de comunicação sejam complementadas com conceitos novos ou totalmente substituídas por outras tecnologias e o sistema no seu todo funcione da mesma forma. Do ponto de vista de implementação existem algumas opções que podem ser tomadas, tudo depende dos requisitos, no caso da Luban é sabido que temos vários tipos de comunicação, possivelmente muitos deles nem faz sentido que a comunicação seja por modbus, é nesta ótica que faz sentido ter várias formas de comunicação disponíveis para colmatar os requisitos necessários. No diagrama de infraestrutura demonstrou-se com a utilização de MQTT, que acabou por ser outra forma de comunicação explorada para a Luban e é nestes casos que podemos olhar para algo como o padrão “Strategy” para servir de inspiração, é possível abstrair e simplificar a implementação de diferentes “estratégias” de comunicação para os vários equipamentos, com a particularidade de ser facilmente expansível caso existam novas formas de comunicar com os dispositivos.

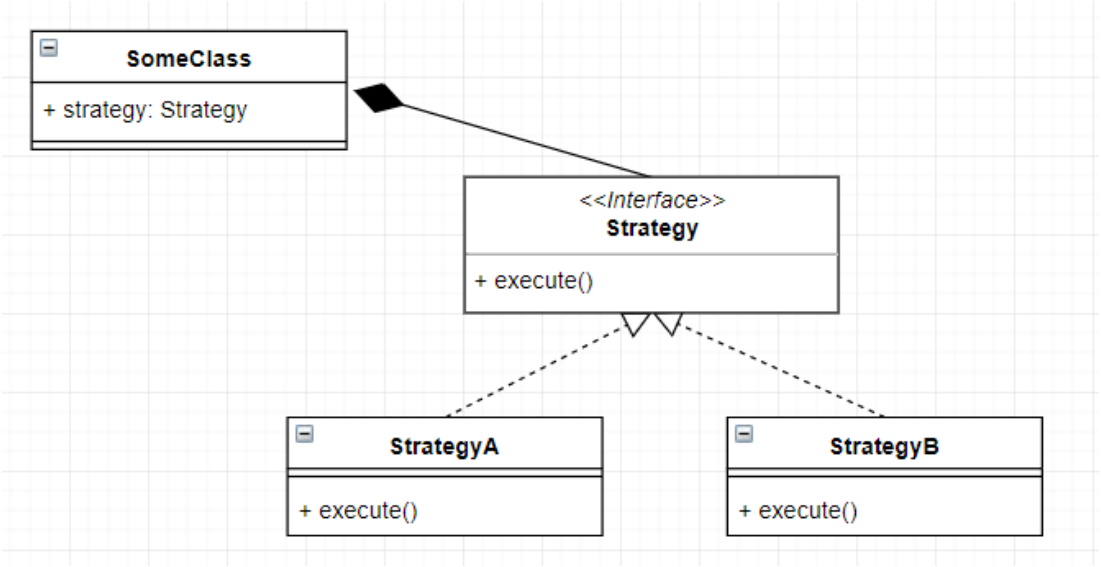


Figura 5.6 – Exemplo demonstrativo do padrão "Strategy"

Resumindo, estamos perante uma arquitetura de referência que foi pensada em 6 camadas independentes, cada uma pensada para resolver uma parte especifica do problema e que se complementam como um todo.

As decisões tomadas para a definição destas camadas focaram-se principalmente em garantir uma arquitetura moderna, inspirada em conceitos aplicados em algumas das arquiteturas de referência mais usadas no contexto da indústria 4.0, IoT e afins, como o IIRA e o RAMI 4.0, e garantir que as soluções para os problemas identificados na Luban estavam abrangidos no contexto da arquitetura final sugerida.

A principal inspiração do RAMI 4.0 retrata-se na organização estrutural da arquitetura, estamos perante uma arquitetura por camadas e vemos algumas semelhanças com diversas das camadas descritas no RAMI. Seria possível arranjar comparações quase diretas das camadas presentes no RAMI para as definidas na arquitetura sugerida para a Luban, mas para efeito de exemplo podemos mencionar as seguintes:

- **Asset**, esta camada no RAMI representa todo um aglomerado de ativos, conseguimos fazer a ponte com os ativos presentes na oficina Luban, sendo estes físicos, documentos ou software. O fluxo da sua informação tem de estar bem presente na forma como a arquitetura é pensada e é aqui que entram elementos como a camada de dados, serviços, API e até mesmo *Application*;
- **Communication**, pelo nome o seu propósito é fácil de interpretar, no contexto da Luban foi necessário ter vários pontos de comunicação, seja entre serviços, utilizadores ou equipamentos físicos. No contexto da RAMI a camada de comunicação tem uma ligação importante com a camada de integração para controlar o fluxo dos pedidos e essa realidade é comparável à relação existente entre a camada de comunicação e a camada de controlo no contexto da arquitetura para a Luban;

O contexto do RAMI vai muito além daquilo que uma arquitetura referência significa para um software ou um problema específico e as suas camadas são muito abstratas para abrangerem diversos contextos, negócios, *stakeholders* e indústrias dentro do paradigma da indústria 4.0, comparando com a arquitetura de referência criada para a Luban e que oferece uma visão sobre o problema, embora abstrata, mas com um contexto em mente mais focado numa realidade diferente.

As inspirações obtidas no IIRA, recaem mais sobre a mentalidade que se deve adotar para o desenvolvimento de uma solução e na forma como o problema foi pensado numa ótica de abstração. Como foi mencionado anteriormente, o IIRA divide o seu foco em dois conceitos, Internet Industrial e Arquitetura de Referência, e é este último ponto que foi mais relevante para este estudo da Luban, para todos os efeitos o objetivo a alcançar passa por ser a criação de uma arquitetura de referência, simples e para um contexto especializado, mas que partilha conceitos e diretrizes praticadas por outras arquiteturas mais complexas, especialmente na vertente da abstração, graças a isso é possível seguir procedimentos, caso façam sentido para o problema em questão, evitando ter de lidar com restrições arbitrárias. Por sua vez, uma arquitetura deste tipo deve estabelecer um quadro comum que possibilita discussões mais detalhadas, é sobre este ponto que as diversas camadas da Figura 5.2 – Diagrama para a Arquitetura final sugerida foram descritas, todos os comentários com teor mais técnico partiram de sugestões e entram neste paradigma de comunicação e discussão de forma que se possa ter uma implementação final mais adequada ao problema em questão.

A Figura 5.7 – Representação da Interoperabilidade das Arquiteturas de Referências, retrata de certa forma o objetivo que se pretende alcançar, criar uma arquitetura que faça uso de conceitos praticados tanto pelo RAMI 4.0 como pela IIRA, permitindo quase esta interoperabilidade entre ambas.

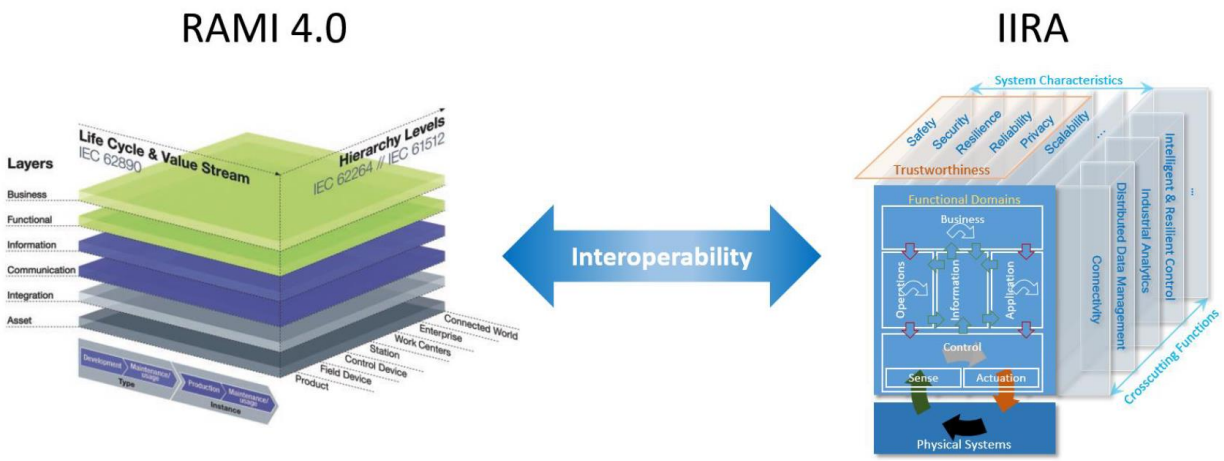


Figura 5.7 – Representação da Interoperabilidade das Arquiteturas de Referências RAMI 4.0 e IIRA

5.4. Protótipo no contexto da Arquitetura de Referência

Para efeitos de síntese final, será importante perceber como o protótipo, mencionado no capítulo 4, influenciou as decisões de um ponto de vista mais técnico sendo que o protótipo acabou por ir a um nível mais concreto com o objetivo de possibilitar a exploração de todo o fluxo da informação.

Olhando para as 6 camadas identificadas na arquitetura de referência sugerida conseguimos enquadrar os diversos elementos do protótipo, observados na Figura 4.19 – Diagrama de Arquitetura para a Fase de Ajustes Finais, nesta estrutura, não esquecendo que o protótipo não representa uma aplicação final completa, mas sim uma base com fundamentos técnicos importantes para o desenvolvimento da arquitetura final, inclusive alguns dos nomes utilizados para definir certos elementos no protótipo já são ilusivos ao que mais tarde é representado na arquitetura.

As camadas de *application* e dados são exemplos claros presentes no protótipo, representados através das interfaces de utilizador criadas com tecnologias frontend e toda a componente de bases de dados desenvolvida sobre os diversos ORM's respetivamente. Na componente de servidor conseguimos encontrar os comportamentos associados às camadas de serviços, API e controlo, se bem que, no protótipo as suas responsabilidades não estão tão bem distribuídas como poderiam estar, mas para todos os efeitos acabou por fazer sentido tendo em conta o contexto técnico da solução concreta utilizada para esse efeito. Por fim, a camada de comunicação não está visualmente representada no protótipo pois para aquele caso de uso a comunicação focou-se muito no uso do protocolo de comunicação Modbus, no entanto, a sua necessidade está omnipresente e demonstra a necessidade ter um elemento com um nível de abstração um pouco maior para que se possa ter flexibilidade nos protocolos de comunicação e não se estar tão preso apenas a um protocolo pois é importante estar preparado para as necessidades futuras e isso é um requisito importante para a arquitetura final.

Capítulo 6

Conclusões

6.1. Síntese

A oficina Luban veio inovar e melhorar a qualidade do ensino no espaço académico, especialmente em toda a componente de automação e controlo. As estações de processo disponibilizadas no seu espaço permitem realizar simulações valiosas para futuros profissionais da área e introduzem muitos destes estudantes a conceitos da indústria 4.0, sendo esta um dos pilares das indústrias modernas, tanto do ponto de vista da inovação como da eficiência de processos físicos, digitais e todos os restantes que complementam o ecossistema, e com isto melhorando receitas destas indústrias.

Porém, como foi falado ao longo do trabalho, a Luban ainda não estava totalmente preparada para dar o salto tecnológico neste contexto, muitos dos problemas identificados acabam por ser obstruções com impacto tecnológico, que requerem recursos significativos para serem investigados e possivelmente melhorados. Foi nesta ótica que a Arquitetura de Referência final foi sugerida, o contexto da Luban abrange diversas frentes e uma revolução tecnológica desta dimensão não pode existir sem primeiro existir suporte do ponto de vista teórico, dito isto, o foco neste trabalho acabou por ser na definição da arquitetura de alto nível que possa vir a suportar uma implementação mais concreta se assim a Luban desejar.

A arquitetura de referência sugerida foi pensada para suportar uma implementação que suporte os problemas atuais da Luban, assim como, possíveis alterações futuras, esta abordagem é um dos pontos mais falados em engenharia de software quando se mencionam sistemas capazes de suportar o passar dos anos, para todos os efeitos o software tende a ter um tempo de vida curto e a inovação de paradigmas, conceitos técnicos e implementações está sempre à espreita.

6.2. Dificuldades

No decorrer do desenvolvimento do trabalho, existiram algumas dificuldades que ficaram evidenciadas em diversos momentos. Uma primeira problemática deu-se devido à componente temporal, foi bastante difícil definir um calendário realista, baseado num diagrama *Gantt*, existiram algumas pausas e atrasos no desenvolvimento do trabalho, por sua vez, este tipo de abordagem também influenciou a metodologia de trabalho escolhida, felizmente, ao longo do trabalho foi possível adaptar e optou-se por aplicar uma metodologia híbrida entre iterativa e incremental e com suporte a protótipos. Um segundo ponto de dificuldade passou pela natureza do trabalho em si devido, estamos a falar de duas realidades que coexistem mas funcionam com base em paradigmas diferentes, o processo inicial demorou algum tempo até arrancar pois foi necessário criar pontes, aprender os vocabulários utilizados no contexto da Luban e perceber como os transitar para uma realidade de engenharia de software, e com isso conseguir interpretar o problema de melhor forma, para tal foi necessário vários momentos de discussão e reuniões entre as diferentes partes envolventes.

Por fim, um momento também relevante mencionar devido à sua dificuldade foi no momento de sugerir uma Arquitetura de Referência com base nas aprendizagens do protótipo, são dois contextos com níveis de abstração distintos. O protótipo foi implementado com um foco muito técnico para se perceber os detalhes técnicos que uma implementação deste nível exigiria e fazer a transição para dois níveis de abstração acima, Figura 5.1 – Diagrama para demonstrar os níveis de abstração, foi desafiante e um ponto de discussão interessante.

6.3. Trabalho Futuro

Apesar do trabalho desenvolvido na presente dissertação, existem sempre pontos por abordar, especialmente devido à questão temporal e por se querer manter o trabalho num âmbito mais focalizado. Dito isto, no caso da Luban, o trabalho futuro vem com naturalidade tendo em conta o que foi falado ao longo deste documento, o foco passou muito pela investigação e consequente criação de uma Arquitetura de Referência que pudesse ser utilizada para acomodar um conjunto de arquiteturas de software que resolvam os problemas atuais da Luban, mencionando novamente a lógica dos níveis de abstração, o trabalho realizado aqui enquadra-se no primeiro nível de abstração, futuramente, caso faça sentido utilizar esta arquitetura como base os desafios passam por olhar para os níveis seguintes, pensar em arquiteturas de software, como as que foram sugeridas no capitulo 5, e nas suas respetivas implementações técnicas para de facto concretizar uma solução final para acomodar todas as necessidades da oficina. Para isso, faz sentido continuar-se a explorar os diferentes paradigmas da Indústria 4.0, olhar para conceitos modernos utilizados em contextos de IoT, implementações de sistemas de monitorização e preparar orquestrar os diferentes elementos lógicos da arquitetura de referência em implementações concretas que comuniquem entre si de forma eficiente. Uma nota também para a possível exploração de normas e regulamentações industriais e exploração de conceitos e boas práticas de *networking* para o contexto isolado da oficina Luban como forma de melhorar e isolar as diversas ligações físicas.

Referências

ANSI/ISA-95. (s.d.). Obtido em 2022, de Wikipedia: https://en.wikipedia.org/wiki/ANSI/ISA-95

AWS. (s.d.). *O que é SOA (arquitetura orientada a serviços)?* Obtido em 2023, de AWS: https://aws.amazon.com/what-is/service-oriented-architecture/

D2.5 – Platform Architectures and Ecosystem Specifications. (2018). *PROPHESY*. Obtido de https://prophesy.eu/outcome/D2.5.pdf

Industrial Internet Consortium. (2018). The Industrial Internet of Things Volume G2: Key System Concerns. *Industrial Internet Consortium*.

Industrial Internet Consortium. (2019). The Industrial Internet of Things Volume G1: Reference Architecture. *Industrial Internet Consortium*. Obtido de https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf

Instituto Politécnio de Setúbal; Cibersur; Data Corner; Ostfalia University of Applied Sciences;. (2020). REINFORCING THE HUMAN CAPITAL AND THE TECHNOLOGICAL CAPACITY OF THE PORTUGUESE LUBAN IN INDUSTRY 4.0. *LUBAN4.0-RI*.

Lin, S.-W., Murphy, B., Clauer, E., Loewen, U., Neubert, R., Bachmann, G., . . . Hankel, M. (2017). Architecture Alignment and Interoperability. *An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper*.

Microsoft. (s.d.). *Estilo de arquitetura de microsserviços*. Obtido em 2023, de Microsoft: https://learn.microsoft.com/pt-pt/azure/architecture/guide/architecture-styles/microservices

Object-relational mapping. (s.d.). Obtido em 2023, de Wikipedia: https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping

Plattform Industrie 4.0. (2018). Platform Industrie 4.0. *RAMI4.0 - a reference framework for digitalisation*. Obtido de https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.pdf?__blob=publicationFile&v=1

Programmable Logic Controller (PLC). (s.d.). Obtido em 2022, de Wikipedia: https://en.wikipedia.org/wiki/Programmable_logic_controller

Soori, M., Arezoo, B., & Dastres, R. (2023). Internet of things for smart factories in industry 4.0. *Internet of Things and Cyber-Physical Systems*. Obtido de https://www.sciencedirect.com/science/article/pii/S2667345223000275

Strategy Pattern. (s.d.). Obtido em 2023, de Wikipedia: https://en.wikipedia.org/wiki/Strategy_pattern

Sundararajan, A., Chavan, A., Saleem, D., & Sarwat, A. (2018). *A Survey of Protocol-Level Challenges and Solutions for Distributed Energy Resource Cyber-Physical Security*.

Supervisory Control and Data Acquisition (SCADA). (s.d.). Obtido em 2022, de Wikipedia: https://en.wikipedia.org/wiki/SCADA

Vercel. (s.d.). *NextJS Documentation*. Obtido em 2023, de NextJS: https://nextjs.org/docs

ZVEI, VDI, VDE. (2015). Reference Architecture Model Industrie 4.0 (RAMI4.0). *Status Report*.