Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Tiago Emanuel da Costa Santana

Tecnologia MERN em projetos de IoT no LINE.IPT

Relatório de Estágio

Orientado por:

Professor Doutor Manuel Fernando Martins de Barros, Instituto Politécnico de Tomar

Pedro Saraiva, Laboratório de Inovação Empresarial (LINE.IPT), Abrantes

Júri:

Prof. Doutor Ricardo Campos, Instituto Politécnico de Tomar

Prof. Doutora Ana Cristina Lopes, Instituto Politécnico de Tomar

Relatório de Estágio apresentada ao Instituto Politécnico de Tomar para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática – Internet das Coisas

“Not all who wander are lost”

— J.R.R. Tolkien

AGRADECIMENTOS

A finalizar esta etapa académica, quero deixar um apreço e agradecimento às pessoas que me ajudaram a chegar à meta.

Em primeiro lugar, um agradecimento aos meus colegas e amigos Nuno Cardoso e Nelson Amaral, pelo companheirismo e entreajuda ao longo desta etapa.

Um agradecimento a todos os docentes pelos conhecimentos transmitidos ao longo destes dois anos.

Um agradecimento a toda minha família e amigos pelo apoio ao longo desta jornada.

Por último, e mais importante, um agradecimento ao meu ídolo e pai, Vítor Manuel Ribeiro Santana. O seu apoio em todas as fases da minha vida académica e pessoal tornaram possível começar e terminar este capítulo.

RESUMO

O presente documento tem como objetivo descrever o trabalho, as tecnologias e os projetos desenvolvidos no âmbito do Estágio Curricular, integrado no curso de mestrado em Engenharia Informática – Internet das Coisas da Escola Superior de Tecnologia de Tomar do Instituto Politécnico de Tomar, que foi realizado no Laboratório de Inovação Empresarial (LINE.IPT), situado no Tecnopolo do Vale do Tejo (TAGUSVALLEY), em Alferrarede – Abrantes.

Durante o estágio, participei em diferentes projetos de I&D aplicados onde me foram apresentados diferentes desafios e no quais pude testar e pôr em prova os conhecimentos adquiridos ao longo do meu percurso académico. Porém, o contacto com um ambiente real de trabalho e a necessidade de ultrapassar constantes desafios relacionados com o desenvolvimento de novos projetos, como o projeto VESTAS que teve três diferentes versões, obrigou ao estudo e à exploração de novos conceitos e abordagens inovadoras.

O projeto VESTAS tinha como objetivo a determinação do desgaste das rodas dentadas nas torres eólicas de um cliente. O protótipo inicial recorria a câmaras de alta resolução apontadas à face frontal dos dentes, em que os conhecimentos obtidos em Inteligência Artificial se revelaram importantes no tratamento de imagem e me ajudaram a chegar a conclusões importantes para determinar o passo seguinte no desenvolvimento do projeto. Com os resultados obtidos, optou-se por utilizar um laser de medição, ligado a um PLC, apontado à face frontal dos dentes. Para comunicar com o PLC através do computador e recolher dados, horas de pesquisa apresentou-me o protocolo FINS e solucionar através da metodologia MERN.

O projeto 3iBioeconomia consistia na monitorização um sistema de zonas húmidas construídas com tratamento por ultravioleta, recorrendo a um sistema de sensorização para verificar a qualidade da água tanto à entrada como à saída do tanque de tratamento. Os sensores são ligados a um microcontrolador ESP32 que enviam os dados por MQTT a um computador central, e que posteriormente, os encaminha para uma base de dados localizada no servidor virtual do LINE. A apresentação e tratamento dos dados é feita numa página online programada em React.

Durante a realização do estágio foi proposto a criação do novo site da TAGUSVALLEY. Neste projeto foi-me atribuída a supervisão e guia do projeto elaborado com recurso às tecnologias lecionadas no curso de Informática e Tecnologias Multimédia (ITM) da Escola Superior de Tecnologia de Abrantes, assim como, o apoio e orientação de um estagiário afeto à execução desta tarefa.

Os constantes desafios técnicos, de conceção do design e da arquitetura associados aos vários projetos, obrigou à tomada de decisões que envolveram inúmeras horas de pesquisa e estudo. A procura da melhor maneira de resolver um problema, de escolher a melhor abordagem e de organizar tarefas e desenvolver o código, foram as tarefas mais desafiantes e que levo comigo no meu futuro profissional.

Palavras-chave: IOT, monitorização industrial, inteligência artificial, FINS, MERN

ABSTRACT

This document aims to describe the work, technologies and projects developed within the scope of the Curricular Internship, integrated in the Master's course in Computer Engineering - Internet of Things at the Higher School of Technology of the Instituto Politécnico de Tomar, which was carried out at the Business Innovation Laboratory (LINE.IPT), located in the Tecnopolo do Vale do Tejo (TAGUSVALLEY), in Alferrarede - Abrantes.

During the internship, I took part in different applied R&D projects which enabled me to embrace new challenges and methodologies and to test the knowledge acquired throughout my academic career. However, the contact with a real work environment and the need to overcome constant challenges related to the development of new projects, such as the VESTAS project, which had three different versions, forced the study and exploration of new concepts and innovative approaches.

The VESTAS project aimed to determine the wear of the sprockets in a customer's wind towers. The initial prototype used high-resolution cameras aimed at the front of the teeth, in which the knowledge obtained in Artificial Intelligence proved to be important in the treatment of images and helped me to reach important conclusions to determine the next step in the development of the project. With the results obtained, it was decided to use a measurement laser, connected to a PLC, aimed at the front face of the teeth. To communicate with the PLC through the computer and collect data, hours of research introduced me to the FINS protocol and to solve it using the MERN methodology.

The 3iBioeconomia project consisted of monitoring a system of wetlands built with ultraviolet treatment, using a sensor system to check the quality of the water before entering the treatment tank, as well as when leaving it. The sensors are connected to an ESP32 microcontroller that send the data by MQTT to a central computer, which later forwards to a database located on the LINE virtual server. The presentation and processing of data is done on an online page programmed in React.

During the internship I also participated in the planning and development the new TAGUSVALLEY website. In this activity, I was responsible by the supervision and guidancee of the project, elaborated using the technologies taught in the IT and Multimedia Technologies (ITM) course at the Abrantes Higher School of Tecnhology, as well as, the support and guidance of the intern assigned to the execution of this project.

The constant technical, design and architecture challenges associated with the various projects, forced the decision-making process that involved countless hours of research and study. The search for the best way to solve a problem, choose the best approach and organize tasks and develop the code, were the most challenging tasks that I take with me in my professional future

Keywords: IOT, industrial monitoring, artificial intelligence, FINS, MERN

Índice

Índice de Figuras

Acrónimos

API	Application Programming Interface
CRUD	Create, read, update and delete
CSS	Cascading Style Sheets
FINS	Factory Interface Network Service
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
NUC	Next Unit of Computing
PHP	Personal Home Page
PLC	Programmable Logic Controller
REST	Representational State Transfer
SQL	Structured Query Language

1. Introdução

O estágio curricular é uma excelente forma de consolidar e complementar o conhecimento adquirido nas várias unidades curriculares que compõem o plano de estudos de um curso e nos preparar para o mercado profissional. Dá-nos a experiência de estar e viver em ambiente corporativo e de nos habituar à rotina que o mercado de trabalho espera encontrar num profissional. Dentro das qualidades desejadas num bom profissional, para além da sua competência técnica, está a pontualidade, assiduidade, cumprimento de prazos de entrega e trabalho em equipa. Sendo um colaborador contratado da entidade onde o estágio decorreu, estas qualidades foram e são, a base da responsabilidade e ética no estágio/trabalho.

1.1. Objetivos

Neste relatório pretende-se apresentar e descrever o trabalho e as tecnologias aplicadas pelo candidato ao longo do estágio, nomeadamente a metodologia MERN (iniciais de MongoDB, ExpressJS, ReactJS, NodeJS), assim como os resultados práticos obtidos no desenvolvimento de vários projetos.

Os desafios apresentados ao longo do estágio foram solucionados recorrendo à programação de microcontroladores para recolha de dados de sensores; programação de um dispositivo central com o objetivo de receber os dados dos microcontroladores e guardá-los numa base de dados; ao desenvolvimento de relatórios e *dashboards* dinâmicos que permita ao cliente ou à organização um acesso simples aos dados e acompanhar o desempenho e otimizar o processo de decisão.

A utilização de uma estrutura similar nos vários projetos envolvidos, motivou a adoção de uma metodologia robusta para que as linguagens de programação, ao longo das várias partes da arquitetura, fossem compatíveis e fiáveis. Desta forma, decidiu-se adotar o MERN por usar apenas uma linguagem tanto no lado do cliente como do lado do servidor, o JavaScript, facilitando a integração das várias partes do projeto. A transferência de dados é feita por JSON que, sendo inicialmente derivado do JavaScript e hoje estandardizado com um formato de troca de dados, é simples, rápida, e é a base da criação

do formato BSON (Binary JSON) utilizado pelo MongoDB que permite guardar e mostrar dados em JSON.

1.2. Organização do documento

Em termos de organização, o presente relatório começa por fazer no Capítulo 2 uma pequena descrição da instituição de acolhimento, seguido da apresentação da tecnologia aplicada nos projetos envolvidos no Capitulo 3, e nos capítulos seguintes apresenta o trabalho desenvolvido dividindo-o nos vários projetos constituintes, ou seja, cada capítulo descreve o envolvimento do estagiário num determinado projeto ou atividade, nomeadamente, o Capítulo 4 é dedicado ao projeto "VESTAS" e o Capítulo 5 ao "3iBioeconomia".

Por fim, será apresentado um último capítulo onde estão descritas as várias intervenções realizadas noutros projetos.

Como algumas das informações contidas neste relatório dizem respeito a projetos ainda em curso ou em início de desenvolvimento, as mesmas são consideradas confidenciais, não podendo ser divulgadas na totalidade.

De referir que, em virtude de os projetos serem bastante abrangentes em termos de áreas de especialidade estes não são unicamente fruto do trabalho do autor do relatório, sendo desenvolvidos em conjunto por equipas multidisciplinares de acordo com as funções de cada uma.

2. Local de Estágio

O estágio decorreu no Laboratório de Inovação Industrial e Empresarial (LINE), no TAGUSVALLEY – Tecnopolo do Vale do Tejo - Abrantes.

2.1. TAGUSVALLEY

O TAGUSVALLEY - Tecnopolo do Vale do Tejo é um Parque de Ciência e Tecnologia, localizado nas antigas instalações da União Fabril do Azoto da CUF, em Alferrarede, no concelho de Abrantes, desde 7 de novembro de 2003. O Parque, espaço com 15 hectares, é propriedade do Município de Abrantes, e a sua gestão cabe à Associação sem fins lucrativos, como o mesmo nome. A estratégia do TAGUSVALLEY - Tecnopolo do Vale do Tejo assenta nos sectores das Tecnologias da Informação e Comunicação, Energia, Metalomecânica e Agroalimentar, áreas onde se procura identificar as oportunidades e as sinergias junto dos atores regionais, com vista à criação de um sistema potenciador de inovação e de empreendedorismo, a par de uma política de atração e estímulo à fixação de recursos humanos qualificados.

Figura 1 - Edifício INOV.POINT

O TAGUSVALLEY divide-se em três departamentos:

O INOV.POINT é uma infraestrutura de incubação e desenvolvimento de empresas vocacionada para o acolhimento e apoio ao arranque de iniciativas empresariais inovadoras e tecnológicas;

O INOV.LINEA, Centro de Transferência de Tecnologia Alimentar é uma estrutura de apoio à inovação, focado na aplicação de novas tecnologias, desenvolvimento de novos produtos e técnicas inovadoras no processamento e conservação de alimentos;

E por fim, o LINE.IPT é um centro de investigação vocacionado para as empresas, sendo o seu principal objetivo o desenvolvimento de novos produtos, tecnologias e processos, visando sempre a melhoria dos produtos/processos já existentes.

2.2. LINE.IPT

O LINE.IPT – Laboratório de Inovação Industrial e Empresarial, surge da parceria entre o Instituto Politécnico de Tomar, a Câmara Municipal de Abrantes, a TAGUSVALLEY e a Nersant.

Tem como objetivos a incorporação de tecnologia e inovação pelas empresas, o desenvolvimento de competências nas áreas de Engenharia e Desenvolvimento de Produtos, a criação de empresas de base tecnológicas e a promoção de redes de cooperação científica e tecnológica entre empresas e instituições de I&DT regionais, nacionais e internacionais. [1]

Especificamente, os serviços prestados pelo LINE são o desenvolvimento e conceção de placas de circuito impresso, desenvolvimento de circuitos e sistemas eletrónicos à medida, desenvolvimento de hardware e software dedicados, manutenção e upgrade de placas eletrónicas, quinagem CNC, corte laser, impressão 3D, melhoria e adaptação de processos industriais e desenvolvimento de soluções tecnológicas à medida.

O LINE divide-se em duas partes, a oficina e o laboratório. A oficina é onde se encontram as máquinas para os trabalhos manuais, já no laboratório situam-se a área de trabalho do engenheiro mecânico, engenheiro informático, dois estagiários de mecânica, e dois estagiários de informática.

Figura 2 - Oficina LINE.IPT

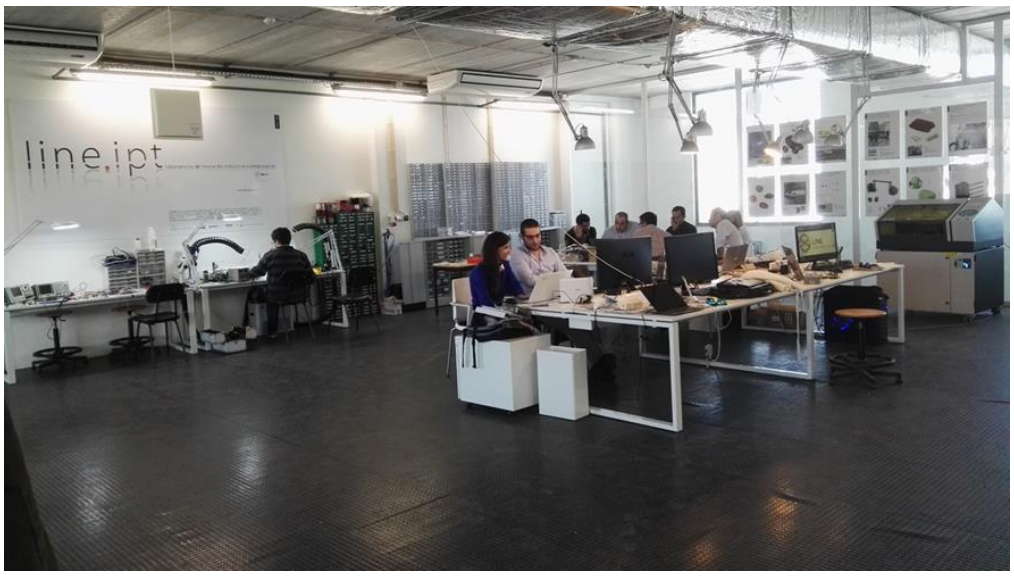

Figura 3 - Laboratório de pesquisa e desenvolvimento do LINE.IPT

2.3. Certificações

A TAGUSVALLEY possui as seguintes certificações, ordenadas por ordem cronológica, da mais antiga para a mais recente:

- 2011 – EU BIC Business Innovation Center (EBN);
- 2015 – Vales de Inovação
- 2015 – Vales de Empreendorismo;
- 2015 – Vales I&D
- 2016 – Vales de Incubação

- 2017 – Vales de Indústria 4.0

Ser detentor destas certificações constitui uma mais valia pois, desta forma as empresas poderão ter acesso a uma comparticipação nos projetos que pretendem desenvolver, mediante candidatura prévia. Desta forma, a TAGUSVALLEY, mais concretamente o LINE.IPT, beneficiaram de um aumento de clientes/projetos.

Além das certificações enunciadas anteriormente, a TAGUSVALLEY assegura também a Presidência TECPARQUES; a Comissão Executiva RIERC (Rede de Incubadoras da Região Centro) e o Conselho Fiscal BIC'S (Rede Nacional do Business Innovation Center).

3. Tecnologias

3.1. Technology Stack

Uma pilha (Stack) de tecnologia (Technology) é uma combinação de produtos de software e linguagens de programação usadas para criar um aplicativo da Web ou móvel. Os aplicativos têm dois componentes de software: lado do cliente e lado do servidor, também conhecidos como front-end e back-end.

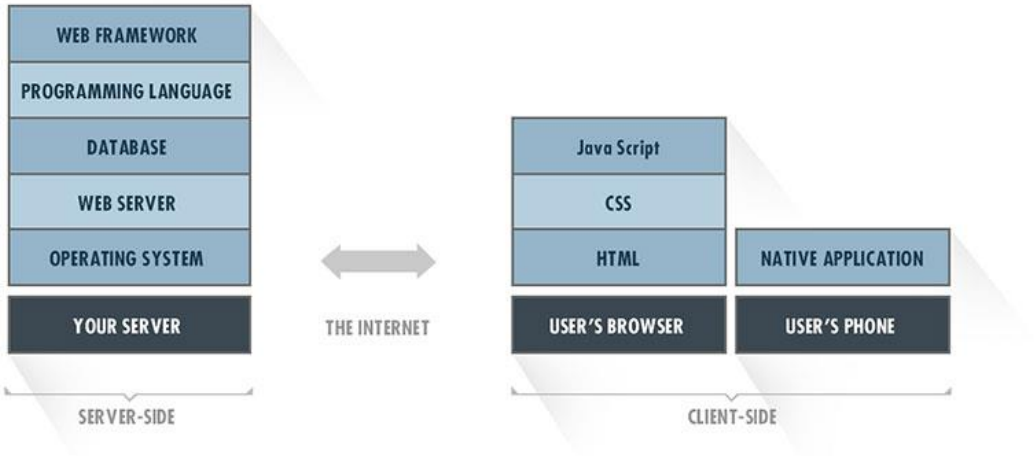

Figura 4 - Pilha de tecnologias

Cada camada do aplicativo baseia-se nos recursos da camada abaixo, criando uma pilha [2]. A figura 4 mostra os principais componentes de uma típica pilha de tecnologia, mas pode haver outros componentes de suporte incluídos.

Relativamente ao desenvolvimento Web, que é onde se enquadra os desafios apresentados no LINE, ficam aqui três exemplos amplamente usados atualmente:

- MERN (MongoDB, ExpressJS, ReactJS, NodeJS)
- LAMP (Linux, Apache, MySQL, PHP)
- MEAN (MongoDB, ExpressJS, AngularJS, NodeJS)

3.2. MERN Stack

MERN Stack é uma estrutura de desenvolvimento web que consiste em MongoDB, ExpressJS, ReactJS e NodeJS como seus componentes de trabalho. O termo MERN nasce das iniciais das quatro tecnologias que o constitui que, de forma sucinta, representam:

- MongoDB – base de dados de documentos;
- ExpressJS – *framework* web de NodeJS;
- ReactJS – *framework* de construção de *user interface* em JavaScript;
- NodeJS – JavaScript *runtime environment*. É executado do lado da máquina e não do *browser*.

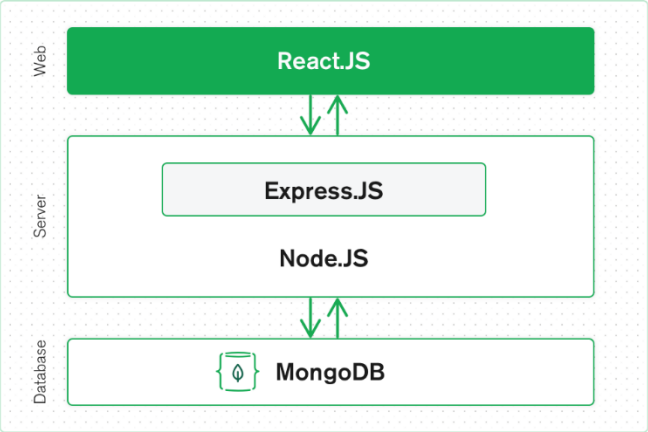

Figura 5 - MERN Stack

MERN Stack permite uma fácil construção de uma arquitetura de três camadas (front-end, back-end, base de dados) usando apenas JavaScript e JSON [3]. Para entender a interação entre camadas e as suas sinergias, abaixo fica uma explicação mais profunda de cada uma das quatro tecnologias.

3.2.1. MongoDB

MongoDB é uma base de dados NoSQL (não relacional) aberta (open source) orientado a documentos, suportando várias formas e grandes volumes de dados. Em vez de usar

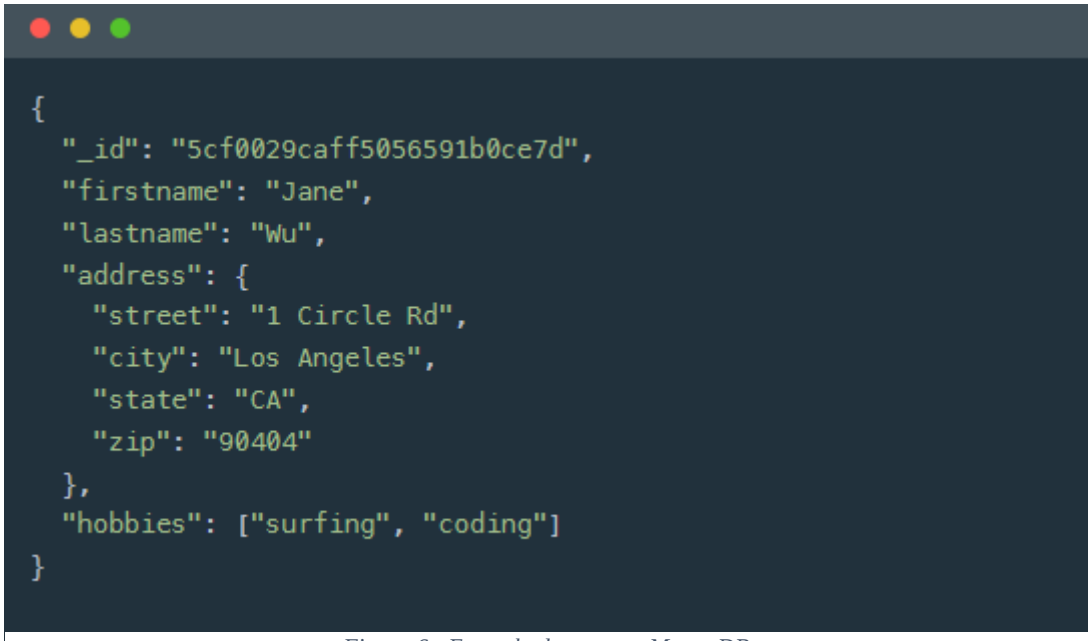

Figura 6 - Exemplo documento MongoDB

tabelas e linhas como as bases de dados SQL (relacional), utiliza coleções e documentos [4]. Os documentos consistem em pares de valores-chave, que são a unidade básica de dados no MongoDB, o que significa que ele armazena dados em documentos semelhantes a JSON.

Suporta matrizes e objetos aninhados como valores, e permite esquemas flexíveis e dinâmicos [5]. As coleções contêm conjuntos de documentos, que são equivalentes a tabelas em base de dados relacionais e, seguindo a mesma linha, cada base de dados pode ter várias coleções/tabelas.

Mas o que torna MongoDB diferente das bases de dados relacionais? Pode-se comparar diretamente e mapear as terminologias variadas dos dois sistemas: a tabela de uma base dados SQL é uma coleção em MongoDB, a coluna é um campo, a linha é um documento e o JOIN é um documento incorporado. A tabela a seguir mostra como uma base de dados NoSQL como o MongoDB difere de uma base de dados relacional [6].

MongoDB	**Base dados Relacional (SQL)**
Base de dados orientado a documentos e não relacional	Base de dados relacional

Baseado em documento	Baseado em linha
Baseado em campo	Baseado em coluna
Baseada em coleções e par chave-valor	Baseado em tabela
Fornece cliente JavaScript para consulta	Não fornece JavaScript para consulta
Relativamente fácil de configurar	Comparativamente, não é tão fácil de configurar
Não afetado por SQL injection	Bastante vulnerável a SQL injection
Possui esquema dinâmico e ideal para armazenamento hierárquico de dados	Tem esquema predefinido e não é bom para armazenamento hierárquico de dados
100 vezes mais rápido e escalável horizontalmente através de sharding	Ao aumentar a RAM, o dimensionamento vertical pode acontecer

Figura 7 - Tabela diferenças entre MongoDB e SQL

Ao analisar a tabela, há três pontos a favor de MongoDB que se tornam relevantes: rapidez e escalabidade; segurança; e o fornecimento de cliente JavaScript para consultas. É através deste fornecimento que NodeJS interliga facilmente com MongoDB e haja uma boa conectividade entre ambos.

Assim, as vantagens de usar MongoDB deve-se a:

a) Base de dados flexível: MongoDB sendo uma base dados em esquema, podemos ter qualquer tipo de dados nem um documento separado, dando flexibilidade e liberadade para armazenar diferentes tipos;
b) Sharding: Podemos armazenar grandes dados distribuindo-os para vários servidores conectados ao aplicativo. Se um servidor não puder manipular dados tão grandes, não

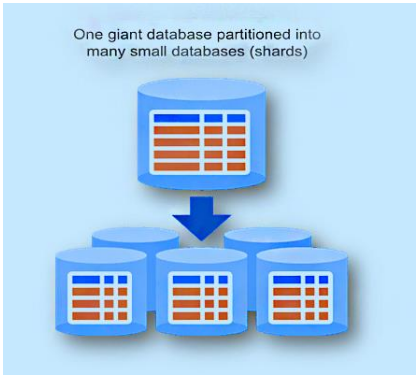

Figura 8 - Ilustração de Sharding

haverá condição de falha. O termo que podemos usar aqui é "sharding automático".
c) Alta velocidade: O MongoDB é uma base de dados orientada a documentos. É fácil aceder documentos por indexação. Portanto, fornece uma resposta rápida à consulta. A velocidade do MongoDB é 100 vezes mais rápida que uma base de dados relacional;

d) Alta disponibilidade: O MongoDB possui recursos como replicação e gridFS. Esses recursos ajudam a aumentar a disponibilidade de dados no MongoDB. Portanto, o desempenho é muito alto;
e) Escalabilidade: Uma grande vantagem do MongoDB é uma base dados escalável horizontalmente. Quando é necessário manipular grandes dados, pode-se distribuí-los por várias máquinas;
f) Configuração fácil do ambiente: É mais fácil configurar o MongoDB do que base de dados relacionais. Também fornece cliente JavaScript para consultas.

Desvantagens de MongoDB:

a) JOIN não suportado: O MongoDB não suporta JOINs como uma base de dados relacional. No entanto, é possível usar a funcionalidade de JOIN adicionando-a codificando-a manualmente. Mas isso pode retardar a execução e afetar o desempenho;
b) Alto uso de memória: O MongoDB armazena nomes de chaves para cada pares de valores. Além disso, devido à falta de funcionalidade de JOINs, há redundância de dados. Isso resulta no aumento do uso desnecessário de memória;
c) Tamanho de dados limitado: Pode-se ter um tamanho de documento não superior a 16 MB.

Coberto as principais vantagens e desvantagens do uso de MongoDB [7], fica entendido o seu uso em MERN STACK.

3.2.2. ExpressJS

O Express.js, ou simplesmente o Express, é uma estrutura de aplicativo da Web para o Node.js, lançada como software livre e de código aberto sob a licença MIT. Foi projetado para criar aplicativos da Web e APIs e é chamado de estrutura de servidor padrão para o Node.js [8]. Para perceber a sua funcionalidade, usar-se-á a analogia de um restaurante recorrendo ao texto de Kevin Kononenko [9]. Nesta analogia, dividida em quatro passos,

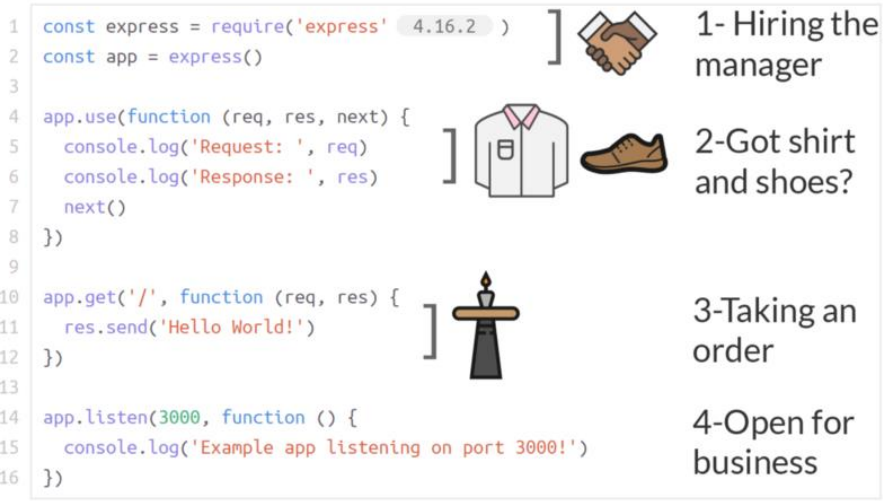

Figura 9 - Analogia ExpressJS como gestor restaurante

o proprietário de um restaurante procura contratar um gerente – a pessoa que cria todos os processos e gere o local para que funcione sem problemas e que os clientes saiam felizes.

Primeiro passo: Contratação do gerente.

Sendo o proprietário do restaurante, precisa de contratar um especialista para executar as operações diárias do restaurante. Para administrar o restaurante de forma eficiente e seguro, precisa de alguém para manter a equipe a trabalhar com a máxima eficiência. Express é o novo gerente.

Segundo passo: Tomar decisões no restaurante (middleware).

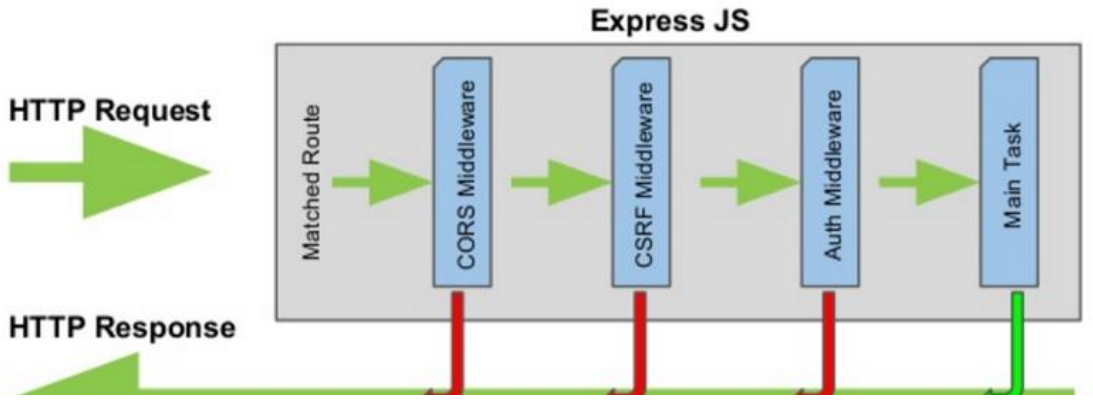

Figura 10 - Ilustração de middleware ExpressJS

Seguindo a linha de analogia, há rotinas comuns que acontecem em restaurantes, como por exemplo: assentar novos clientes; receber pedidos e comida; apresentar a conta no final da refeição. Para cada um destes exemplos, há uma serie de verificações a ser feitas antes de executar. No exemplo de sentar clientes, há mesas vagas? E pedidos de comida, há em stock o que está a ser pedido? Se for clientes especiais, têm cartão VIP (como conta no site, e digitaram as credenciais corretamente)?

É aqui que entra o conceito de middleware. As funções de middleware permite executar qualquer solicitação recebida e modificá-la antes de enviar a resposta. Analisando a figura 11 abaixo, exemplifica o gerenciamento que o Express sobre se os clientes têm t-shirt e sapatos vestidos.

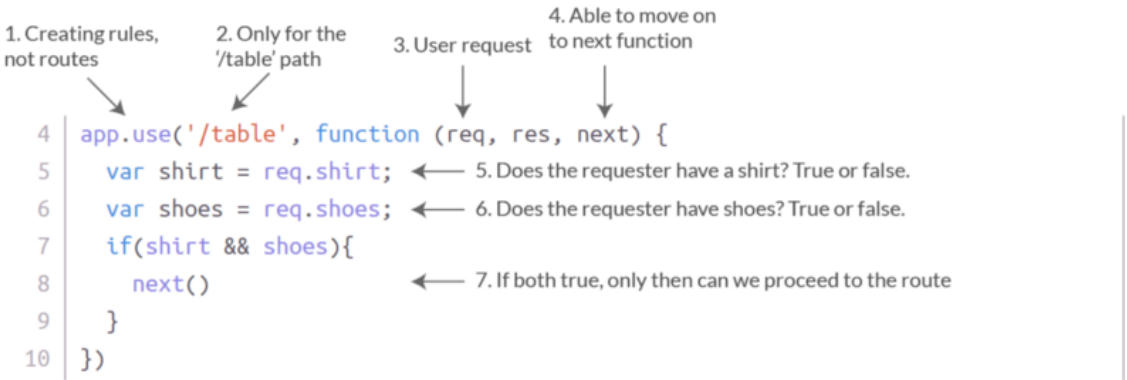

Figura 11 - Exemplo código ExpressJS

Terceiro passo: execução comum de rotinas (routing).

Seguindo o exemplo de assentar, e já sabendo como validar se é possível sentar ou não, fica por explicar como se leva os clientes à mesa e sentá-los. É aqui que entra as rotas, que permite criar ações específicas com base no caminho. As opções são GET, POST,

PUT e DELETE (REST API). Para sentar, digamos 2 pessoas, é necessário fazer pedido GET para verificar existência de mesa para dois, caso exista a resposta é o número da mesa.

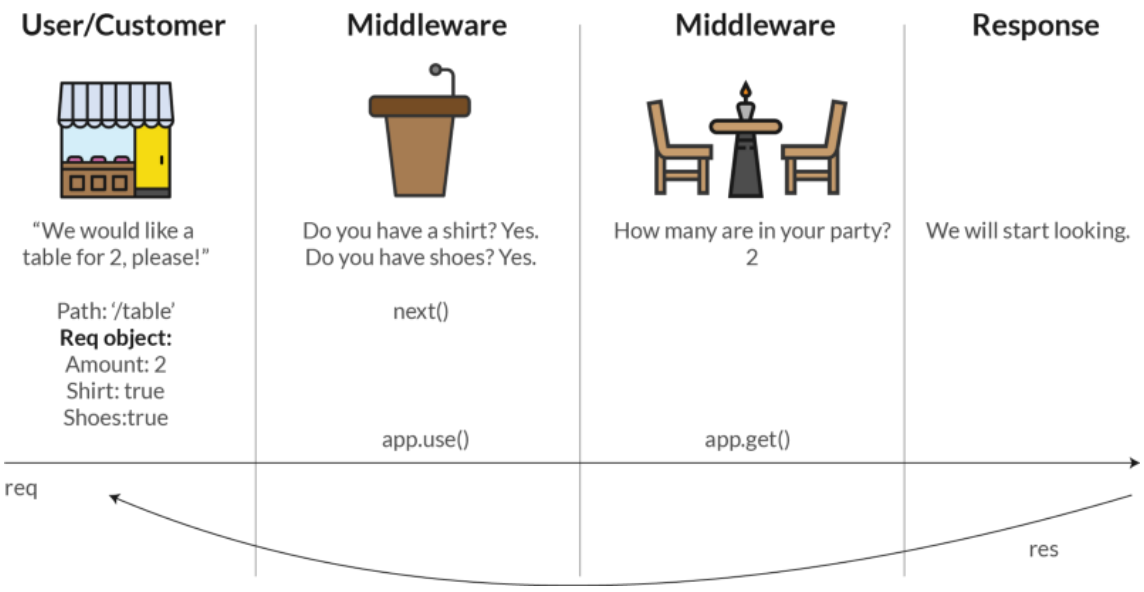

Figura 12 - Várias etapas de processamento ExpressJS

Quarto passo: abertura para negócios (portas).

Estando estruturado o funcionamento do restaurante, é necessário determinar o endereço do local. O servidor possui portas, assim como os restaurantes, e como o servidor pode gerir vários restaurantes ao mesmo tempo é necessário informar em que porta o restaurante está aberto.

```
16 app.listen(3000, function () {
17   console.log('Restaurant at Port 3000 open for business.')
18 })
```

Figura 13 - Exemplo código execução ExpressJS

Abrindo o navegador de internet na porta indicada no servidor (neste caso, https://localhost:3000/), irá ter ao restaurante. Mas o Express gere o restaurante, mas neste momento encontra-se sem conteúdo (páginas web para fazer routing e o utilizador navegar no browser). É necessário criar essas páginas, e é aqui que entra o ReactJs.

Em termos de prós e contras, sendo uma framework de NodeJS, estão diretamente ligados aos da linguagem NodeJS, pelo que serão apresentados mais abaixo.

3.2.3. ReactJS

React (também conhecido por React.js ou ReactJS) é uma biblioteca JavaScript de código aberto usada para criar *user interfaces* especificamente para aplicativos de página única e é usado para manipular a camada de visualização de aplicativos da Web e móveis. O React também nos permite criar componentes reutilizáveis de *user interface*. Foi criado por Jordan Walke, um engenheiro de software que trabalha no Facebook, e foi implantado pela primeira vez no feed de notícias do Facebook em 2011 e no Instagram.com em 2012 [10].

A sua função primária foi destinada para resolver o problema de estado atual de um aplicativo/página web. Se visitar uma página da Web, ela começará em um certo "estado" inicial. Mas se clicar em uma opção que abre um menu, o aplicativo está em um novo estado quando o menu é aberto. Se agora clicar no botão "like", o polegar fica azul e aumenta o número de gostos, coloca assim um novo estado para o aplicativo. Quando escalável a várias opções que alteram o estado, torna-se complicado acompanhar o verdadeiro estado do aplicativo. React facilita o gerenciamento de estado, armazenando uma verdade central e deixando a verdade chegar aos componentes que compõem o aplicativo, em oposição ao ato de malabarismo de dizer explicitamente a cada componente como agir. Eles são configurados para serem um reflexo do estado. Depois de clicar no botão “like”, é impossível para o contador não saber quantos gostos eles devem contar, desde que cada componente seja renderizado com base no mesmo estado [11].

Para centralizar a verdade do estado, React é organizado por hierarquias de componentes onde um conjunto de valores imutáveis é passado para o representante de componentes como propriedades em suas tags HTML. O componente não pode modificar diretamente nenhuma propriedade, mas pode transmitir uma função de retorno de chamada com a

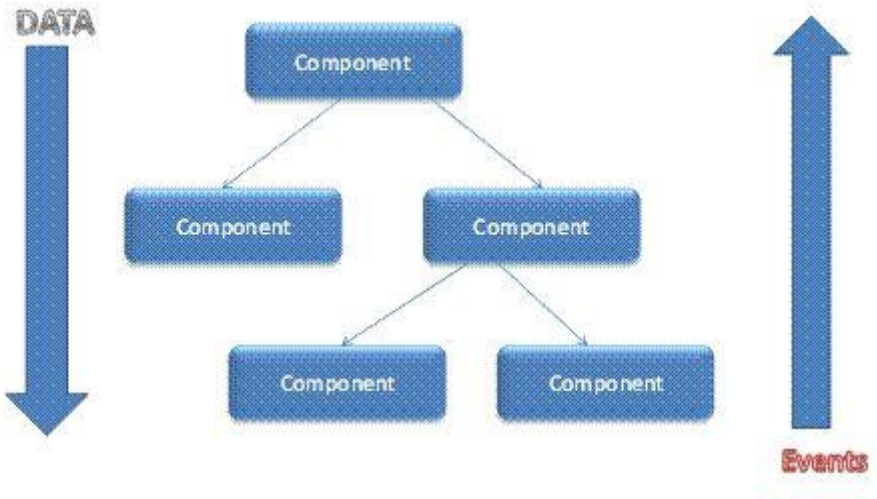

Figura 14 - Ilustração data e eventos em React

ajuda da qual podemos fazer modificações. Esse processo completo é conhecido como "propriedades fluem para baixo; ações fluem para cima" [10].

Uma boa analogia é comparar componentes React a peças de Lego. Um conjunto de peças criam a aplicação e pode usar a mesma tipo peça durante a sua construção. Exemplo de um componente React na figura 14 mostra a criação de um componente com o nome "MyComponent". Esse componente pode ser incorporado noutro, como é verificável no exemplo na figura 15. Assim, é possível compor um *user interface* mais complexa e útil e permite reutilizar componentes dentro de si, incentivamente a reutilização de código e

```
import React from 'react';

class MyComponent extends React.Component {
  render () {
    return <div> This is a component </div>
  }
}
```

Figura 15 - Exemplo criação componente React

fácil manutenção da base de código [12].

Sabendo a estrutura de um componente React, fica mais simples explicar o conceito

```
class MyComponent extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      name: "Manoj"
    };
    this.changeName = this.changeName.bind(this);
  }


  changeName () {
    this.setState({
      name: "Your Name"
    });
  }


  render () {
    return <div onClick={this.changeName}> My name is
{this.state.name} </div>
  }


}
```

Figura 17 - Incorporação de um componente React

"Props" e "State". Como é visível na figura 16, React usa o método "render()" que corre sempre que o "State" ou "Props" atualizam. Este processo garante que sempre que há

```
import React from 'react';


class MyComponent extends React.Component {
  render () {
    return <div> This is a component </div>
  }
}


class MyOtherComponent extends React.Component {
  render () {
    return (
      <div>
        <MyComponent />
      </div>
    )
  }
}
```

Figura 16 - Componente React com estados

alterações nos dados, a exibição é automaticamente atualizada. Então o "State", ou estado,

é único em cada componente, mas este estado pode ser passado por outro componente

```
class MyHeading extends React.Component {
  render () {
    return <div>
      <h1>{this.props.heading}</h1>
      <p>{this.props.subtitle}</p>
    </div>
  }
}

// Now I can use this component like this
<MyHeading heading="Whoo! this is awesome" subtitle="And this is a
subtitle" />
<MyHeading heading="Whoo! this is More awesome" subtitle="And this is
second subtitle" />
```

Figura 18 - Componente React com passagem de estados através de props

acima na hierarquia através do “Props”, seguindo a linha de raciocino da figura 13. Na figura 16 é visualizável a instanciação da variável de estado “name”, que é renderizada e que pode ser alterada ao clicar na *div*. O exemplo na figura 17 mostra o uso the “Props”, que é passado uma string mas poderia ser uma variável de estado, mostrando assim como a verdade de estado se mantém ao longo de todos os componentes centralizando os estados num componente e passando estes para os componentes que incorpora.

As vantagens do uso de ReactJS deve-se a:

a) Fácil de aprender e usar: Vem com uma boa oferta de documentação, tutoriais e recursos de treino. Qualquer developer que possua experiência em JavaScript pode entender e começar a criar aplicativos da Web com facilidade usando o React em alguns dias.
b) Criar aplicativos dinâmicos da Web torna-se mais fácil: Criar um aplicativo da web dinâmico especificamente com sequências de caracteres HTML foi complicado porque requer uma codificação complexa, mas o React JS resolveu esse problema e o tornou mais fácil. Ele fornece menos codificação e oferece mais funcionalidade.
c) Componentes reutilizáveis: Um aplicativo da web ReactJS é composto de vários componentes e cada componente possui sua própria lógica e controles. Esses componentes são responsáveis pela saída de um pequeno pedaço reutilizável de código HTML que pode ser reutilizado sempre que precisar. O código reutilizável ajuda a tornar aplicativos mais fáceis de desenvolver e manter.

d) Melhoria de desempenho: O ReactJS melhora o desempenho devido ao DOM virtual. O DOM[1] é uma API de plataforma cruzada e programação que lida com HTML, XML ou XHTML. A maioria dos developers enfrentou o problema quando o DOM foi atualizado, o que diminuiu o desempenho do aplicativo. O ReactJS resolveu esse problema introduzindo o DOM virtual. O React Virtual DOM existe inteiramente na memória e é uma representação do DOM do navegador da web. Devido a isso, quando escrevemos um componente React, não escrevemos diretamente no DOM. Em vez disso, escrevemos componentes virtuais que React transforma no DOM, levando a um desempenho mais suave e rápido. [13]

Desvantagens do uso de ReactJS:

a) O alto ritmo de desenvolvimento: O alto ritmo de desenvolvimento tem vantagens e desvantagens. Em caso de desvantagem, como o ambiente muda continuamente tão rápido, alguns developers naturalmente não se sentem confortáveis com um ritmo de desenvolvimento tão alto, preferindo usar tecnologias de desenvolvimento da Web mais maduras;
b) Falta de convenções: Como o ReactJS é uma biblioteca relativamente jovem, desenvolvida muito ativamente pelo Facebook e por colaboradores de todo o mundo, existem muito poucas normas, e as que existem vêm do Facebook, como estrutura ou estado de componentes, adereços e contexto distinções. Por enquanto, a documentação do ReactJS é a fonte mais autorizada de melhores práticas e normas do ReactJS;
c) Curva de aprendizagem acentuada: Para uma biblioteca relativamente jovem, o ReactJS é bastante grande e sua documentação está longe de ser excelente. De facto, deixa muito a desejar, especialmente quando se trata de novas bibliotecas como Redux e Reflux. Pode-se encontrar muitos recursos do React JS na internet, mas não há nada como ter acesso a documentação oficial abrangente. [14]

3.2.4. NodeJS

[1] Document Object Model (DOM) é uma interface multiplataforma e independente de idioma que trata um documento XML ou HTML como uma estrutura em árvore, em que cada nó é um objeto que representa uma parte do documento [29].

O Node.js é um JavaScript *runtime*[2] de código aberto, *cross-platform*, que executa código JavaScript fora de um navegador da web. O Node.js permite que os developers usem JavaScript para escrever ferramentas de linha de comando e para scripts no lado do servidor - executando scripts no lado do servidor para produzir conteúdo dinâmico da página da web antes que a página seja enviada ao navegador da web do utilizador [15]. Sendo um JavaScript *runtime* assíncrono orientado a eventos, o Node.js foi projetado para criar aplicativos de rede escaláveis. Muitas conexões podem ser tratadas em simultâneo e em cada conexão, o retorno de chamada é acionado, mas se não houver trabalho a ser feito, o Node.js entrará em suspensão [16]. Pode ser usado para criar diferentes tipos de aplicativos, como aplicativo de linha de comando, aplicativo Web, aplicativo de chat em tempo real, servidor REST API etc. No entanto, é usado principalmente para criar programas de rede como servidores Web, semelhantes ao PHP, Java ou ASP.NET [17].

Os benefícios de NodeJS deve-se a:

a) Pilha de tecnologia robusta - O JavaScript provou ser um líder indiscutível entre as linguagens de programação mais populares. Por sua vez, o NodeJS se tornou um nome independente no setor. Com um total de 368.985.988 downloads e mais de 750 novos colaboradores, conforme declarado no relatório Node-by-numbers 2018, o projeto

[2] Javascript runtime é um local onde o código Javascript é executado. É como um container ou um ambiente em que nosso código JS é executado.

parece estar mais forte do que nunca;

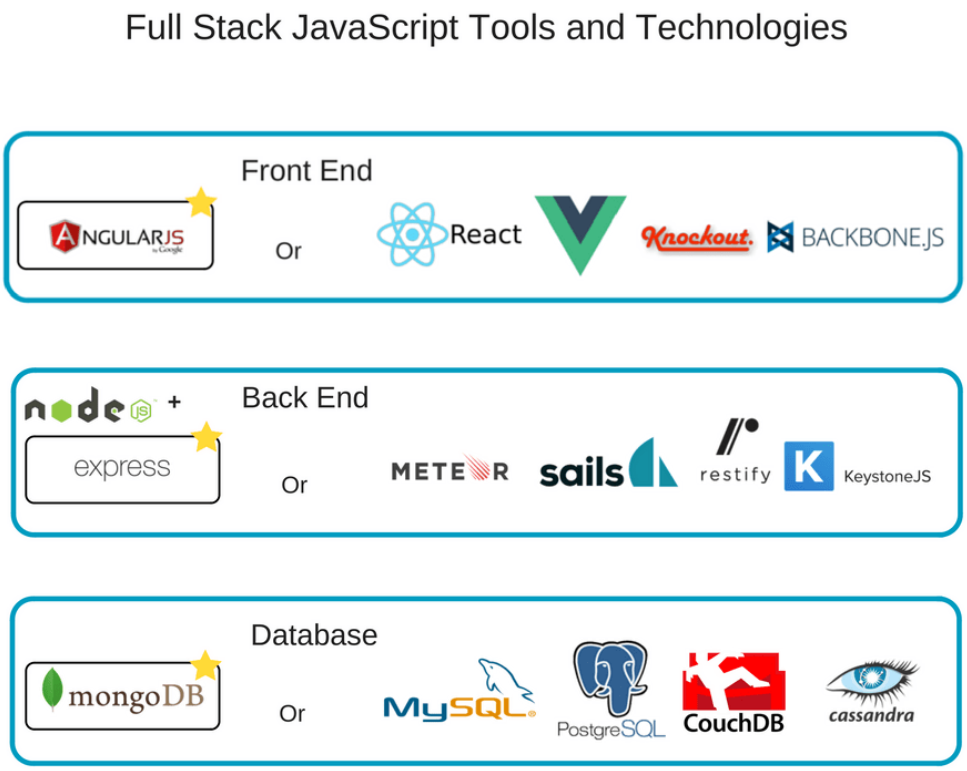

Figura 19 - Tecnologias e ferramentas de Full Stack JavaScript

b) Modelo de processamento rápido e baseado em eventos: O mecanismo usado na implementação do Node.js foi originalmente desenvolvido para o navegador Chrome. Escrito em C ++, o V8 do Chrome é usado para compilar funções escritas em JavaScript em código de máquina e faz o trabalho a uma velocidade impressionante. E no não bloqueio de entrada / saída e a manipulação de solicitações assíncronas tornaram o Node.js capaz de processar solicitações sem atrasos;

c) Tecnologia escalável para micros serviços: Por ser uma ferramenta tecnológica leve, usar o Node.js para arquitetura de micros serviços é uma ótima opção. Consequentemente, dividindo a lógica do aplicativo em módulos menores, micros

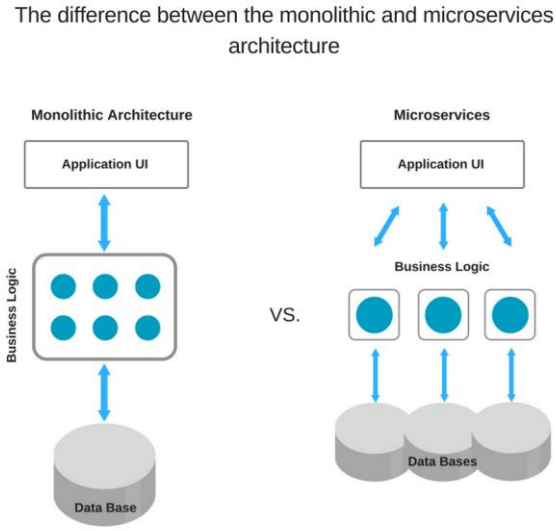

Figura 20 - Ilustração de diferença entre uma arquitetura monolítica e de micros serviços

serviços, em vez de criar um único núcleo monolítico grande, é permitido uma maior flexibilidade e lança as bases para um crescimento adicional. Como resultado, é muito mais fácil adicionar mais micros serviços sobre os existentes do que integrar recursos adicionais à funcionalidade básica do aplicativo.

d) Ecossistema Rico: "npm", um gerenciador de pacotes padrão do NodeJS, tem cerca de 836.000 bibliotecas disponíveis de momento e mais de 10.000 novas sendo publicadas a cada semana, tornando o ecossistema NodeJS bastante rico. As mesmas estatísticas[3] apontam que 97% dos aplicativos modernos da Web consistem em módulos "npm", uma prova da sua indiscutível popularidade entre developers.

e) Suporte JSON contínuo: Embora outras tecnologias de back-end como PHP e Ruby on Rails possam usar o formato JSON para comunicação, o NodeJS faz isso sem converter entre modelos binários e usa JavaScript. Isso é especialmente útil quando é necessário criar APIs RESTful para suporte de base de dados NoSQL, como MongoDB usado no MERN.

Desvantagens do NodeJS:

[3] https://blog.npmjs.org/post/180868064080/this-year-in-javascript-2018-in-review-and-npms

a) Constrangimento de desempenho com tarefas pesadas de computação: A maior desvantagem do NodeJS até agora é a incapacidade de processar tarefas ligadas ao CPU. NodeJS processa JavaScript, que é um thread único. Um modelo de entrada / saída sem bloqueio significa que o NodeJS responde à chamada do cliente para iniciar um pedido e processa a tarefa durante o tempo em que dispara o *callback*, que é quando a tarefa está pronta. Sendo assíncrono, o Node executa o código JS num único thread em uma base de evento, a isso se chama loop de eventos. O problema ocorre quando o NodeJS recebe uma tarefa vinculada à CPU: sempre que uma solicitação pesada chega ao loop de eventos, o NodeJS define toda a CPU disponível para processá-la primeiro e, em seguida, responde a outras solicitações na fila. Isso resulta em processamento lento e atraso geral no loop de eventos, motivo pelo qual o Node.js não é recomendado para computação pesada;
b) Problema de *callback*: Devido à sua natureza assíncrona, o NodeJS depende muito de *callbacks*, as funções executadas após a conclusão de cada tarefa na fila. Manter várias tarefas na fila em segundo plano, cada uma com seu *callback*, pode resultar no chamado inferno de *callback*, que afeta diretamente a qualidade do código. [18]

3.3. Protocolo FINS

O protocolo de comunicação FINS (Factory Interface Network Service) é o protocolo de rede usado pelos PLCs (Programmable Logic Controller) da Omron. Foi desenvolvido pela Omron para fornecer uma forma consistente de comunicação entre PLCs e computadores em várias redes [19]. O serviço de comunicações FINS permite o controle operações, como leitura ou escrita de dados da área de memória do PLC. Os dados são enviados e recebidos como pacotes UDP em uma rede Ethernet [20]. O número de porta FINS (valor padrão: 9600) definido na configuração do sistema juntamento com o IP do PLC é usado para comunicações FINS com o PC.

No uso deste protocolo, recorreu-se à biblioteca de node-omron-fins [21] de NodeJS, não só por ser a linguagem usada na implementação do projeto em questão como pela forma simples de usar. A figura 21 demonstra que apenas é necessário indicar a porta e IP do PLC e ter uma função de escuta da resposta do PLC, e que esta é recebida após efetuado

```
var fins = require('omron-fins');

// Connecting to remote FINS client on port 9600 with default tim
var client = fins.FinsClient(9600,'127.0.0.1');

// Setting up our error listener
client.on('error',function(error) {
  console.log("Error: ", error);
});

// Setting up the response listener
// Showing properties of a response
client.on('reply',function(msg) {
        console.log("Reply from: ", msg.remotehost);
        console.log("Transaction SID: ", msg.sid)
    console.log("Replying to issued command of: ", msg.command);
    console.log("Response code of: ", msg.code);
    console.log("Data returned: ", msg.values);
});


//Read 10 registers starting at DM register 00000
client.read('D00000',10);
```

Figura 21 - Protocolo FINS em NodeJS

o pedido de leitura do endereço de memória pretendido.

4. Projeto VESTAS

Neste e nos capítulos que se seguem, vai-se apresentar o trabalho desenvolvido ao longo do estágio nos vários projetos em que o estagiário esteve envolvido. Este capítulo começa por descrever o envolvimento no projeto "VESTAS".

Com este projeto pretende-se desenvolver uma solução para quantificação do

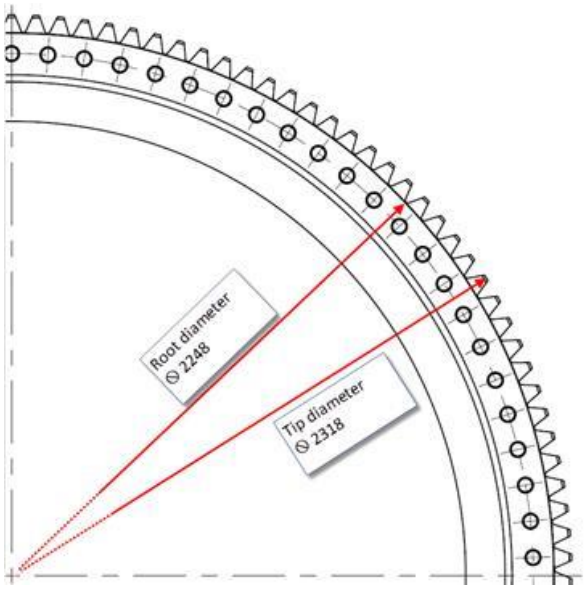

Figura 22 - Yaw Ring torres eólicas VESTAS onde assentam os dentes a medir

desgaste de dentes de rodas dentadas. Neste caso, pretende-se determinar o *yaw ring* das torres eólicas V90 da fabricante Vestas, o cliente para o qual está a ser desenvolvido este projeto.

Uma das especificações do projeto é definida pela necessidade de verificar a

espessura do dente mesmo quando este se encontra com um desgaste elevado.

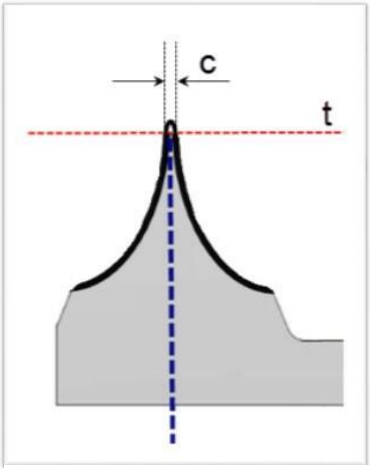

Figura 23 - Ilustração da medida a calcular num dente

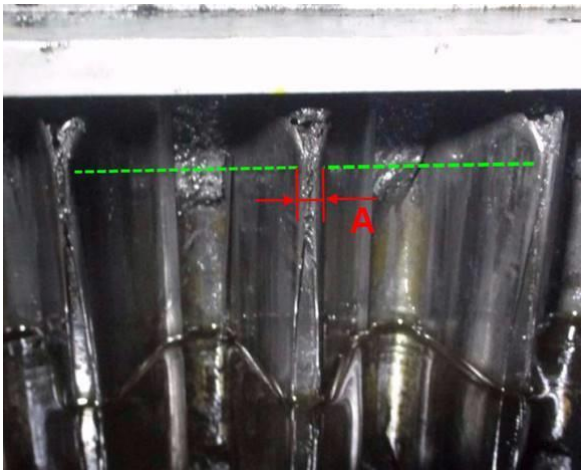

Figura 24 - Medida a calcular num dente real

Nestes casos o dente deixa de ter forma retangulares, ficando com as arestas arredondadas e o excesso de material cresce para a frente, o que pode resultar em alguns erros de medida. As figuras 23 e 24 são imagens para melhor perceber esta situação.

A necessidade de ter um sistema automático de quantificação de desgaste é importante para o cliente para minimizar o tempo de paragem da torre quando se procede à inspeção efetuada de forma manual, assim como para eliminar erros humanos.

4.1. Abordagem inicial

A equipe de projeto e desenvolvimento escalonada para este projeto estabeleceu, como abordagem inicial, uma solução baseada em visão artificial, recorrendo a câmaras de alta resolução e ao tratamento de imagem.

Tendo em vista garantir o correto funcionamento do sistema mesmo em casos extemos como os apresentados na figura 23, utilizam-se duas câmaras e uma matriz de linhas laser. A utilização das duas câmaras justifica-se com a necessidade de garantir que são visíveis

Figura 25 - Esquema da primeira versão do protótipo

as duas faces laterais de cada dente em análise, ao contrário do que é visível na figura 23, e efetuar a correta medição do topo do dente. A matriz de linhas laser ao incidir no topo do dente cria uma reta e no seu vale, pelo desnível em profundidade, a reta torna-se curva. A identificação na imagem da reta no topo do dente indica o seu comprimento através do número de pixéis.

Quando integrei este projeto, a equipa tinha já iniciado o trabalho, adotando solução baseada no processamento de imagem e visão artificial. O processamento de imagem era suportado no software da Halcon, que contém bibliotecas desenvolvidas especificamente para as suas câmaras de vídeo assim como era utilizado o próprio IDE da empresa. A utilização deste software fechado e de custos elevados, torna a ajuda e o acesso a documentos e exemplos escassa. Os serviços de apoio técnico consistiam quase exclusivamente no contacto telefónico da empresa. Tendo em conta este facto, e a dificuldade em alterar o seu complexo código desenvolvido em C++, optou-se por refazer o trabalho anteriormente desenvolvido, mas agora utilizando a plataforma de visão por computador OpenCV [22] e linguagem Python. Esta decisão permitiu recorrer a vários exemplos e documentação online e chegar a várias conclusões.

Primeiramente é efetuada a calibração das câmaras a fim de retificar a sua posição e saber o valor correspondente em mm de cada pixel. A função de calibração (figura 26) de ambas as câmaras são similares, mudando apenas a pasta das imagens para a sua calibração capturadas anteriormente pelas mesmas. Foi impressa numa peça de encaixe nos dentes com um tabuleiro de xadrez usado para calibrar, visível na figura 29. Estando encaixado no vale dos dentes permite que o tabuleiro esteja à mesma altura que o topo dos dentes. Algoritmos de OpenCV e Numpy fazem o trabalho de calibração através de múltiplas interações nas imagens e guarda o resultado em ficheiro para ser utilizado posteriori.

```python
def camera_calibration_left():
    #========================================
    # Camera calibration
    #========================================

    #Define size of chessboard target.

    chessboard_size = (9,6)

    #Define arrays to save detected points
    obj_points_l = [] #3D points in real world space
    img_points_l = [] #3D points in image plane

    #Prepare grid and points to display

    objp = np.zeros((np.prod(chessboard_size),3),dtype=np.float32)


    objp[:,:2] = np.mgrid[0:chessboard_size[0], 0:chessboard_size[1]].T.reshape(-1,2)

    #read images

    calibration_paths = glob.glob('./images_cal_left/*')

    #Iterate over images to find intrinsic matrix
    for image_path in tqdm(calibration_paths):

        #Load image
        image = cv2.imread(image_path)
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        print("Image loaded, Analizying...")
        #find chessboard corners
        #ret,corners = cv2.findCirclesGrid(gray_image, chessboard_size, None, flags = cv2.CALIB_CB_SYMMETRIC_GRID)
        ret, corners = cv2.findChessboardCorners(gray_image, chessboard_size,None)

        if ret == True:
            print("Chessboard detected!")
            print(image_path)
            #define criteria for subpixel accuracy
            criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
            #refine corner location (to subpixel accuracy) based on criteria.
            cv2.cornerSubPix(gray_image, corners, (9,6), (-1,-1), criteria)
            obj_points_l.append(objp)
            img_points_l.append(corners)

    path='/home/santana/Desktop/Vestas/images_cal_left/'
    img = cv2.imread(os.path.join(path, 'opencv_frame_left2.tif'))
    gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(obj_points_l, img_points_l, gray_image.shape[:2], None, None)

    """Save camera calibration parameters in json file."""
    data = {
        "camera_matrix": mtx.tolist(),
        "dist_coeff": dist.tolist(),
        }
    with open('cal_file_left', "w") as file:
        json.dump(data, file)
```

Figura 26 - Código que efetua a calibração das câmaras utilizadas no protótipo

De seguida, recorrendo já ao ficheiro de calibração, é efetuada a junção das duas imagens das câmaras, resultando numa imagem (figura 30) correspondente à área abrangida pelos dois dispositivos. Para esse fim foi usado a biblioteca Python “imutils” de processamento

```python
def detectAndDescribe(self, image):
    # convert the image to grayscale
    #gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # check to see if we are using OpenCV 3.X
    if self.isv3:
        # detect and extract features from the image
        descriptor = cv2.xfeatures2d.SIFT_create()
        (kps, features) = descriptor.detectAndCompute(image, None)

    # otherwise, we are using OpenCV 2.4.X
    else:
        # detect keypoints in the image
        detector = cv2.FeatureDetector_create("SIFT")
        kps = detector.detect(image)

        # extract features from the image
        extractor = cv2.DescriptorExtractor_create("SIFT")
        (kps, features) = extractor.compute(image, kps)

    # convert the keypoints from KeyPoint objects to NumPy
    # arrays
    kps = np.float32([kp.pt for kp in kps])

    # return a tuple of keypoints and features
    return (kps, features)
```

Figura 27 - Código que detecta pontos em comum nas imagens a concatenar

de imagem, que recorre ao algoritmo RANSAC para estimar a matriz homográfica através de vetores em comum.

```python
def drawMatches(self, imageA, imageB, kpsA, kpsB, matches, status):
    # initialize the output visualization image
    (hA, wA) = imageA.shape[:2]
    (hB, wB) = imageB.shape[:2]
    vis = np.zeros((max(hA, hB), wA + wB, 3), dtype="uint8")
    vis[0:hA, 0:wA] = imageA
    vis[0:hB, wA:] = imageB

    # loop over the matches
    for ((trainIdx, queryIdx), s) in zip(matches, status):
        # only process the match if the keypoint was successfully
        # matched
        if s == 1:
            # draw the match
            ptA = (int(kpsA[queryIdx][0]), int(kpsA[queryIdx][1]))
            ptB = (int(kpsB[trainIdx][0]) + wA, int(kpsB[trainIdx][1]))
            cv2.line(vis, ptA, ptB, (0, 255, 0), 1)

    # return the visualization
    return vis
```

Figura 28 - Código que desenha os pontos em comum nas duas imagens

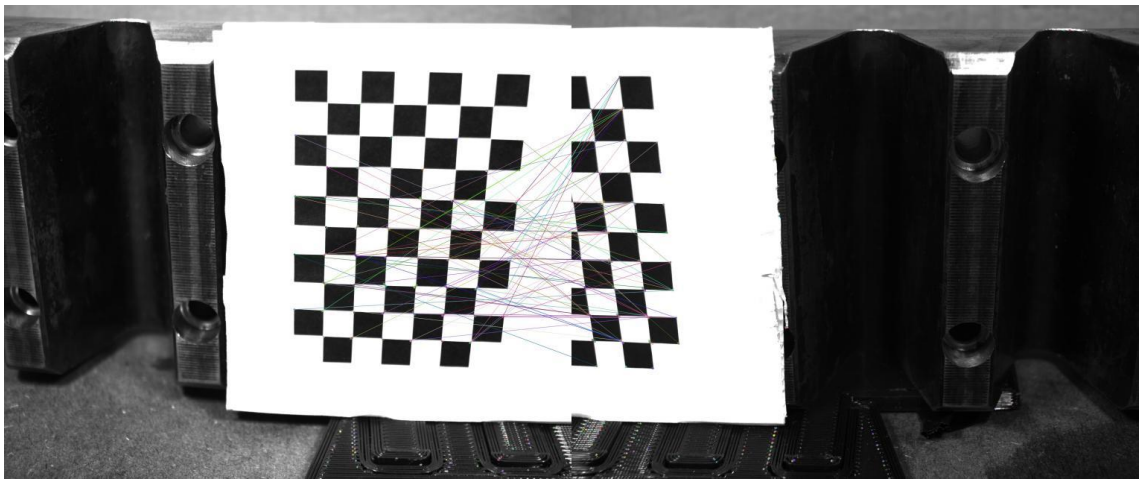

Figura 29 - Ilustração de uma imagem concatenada

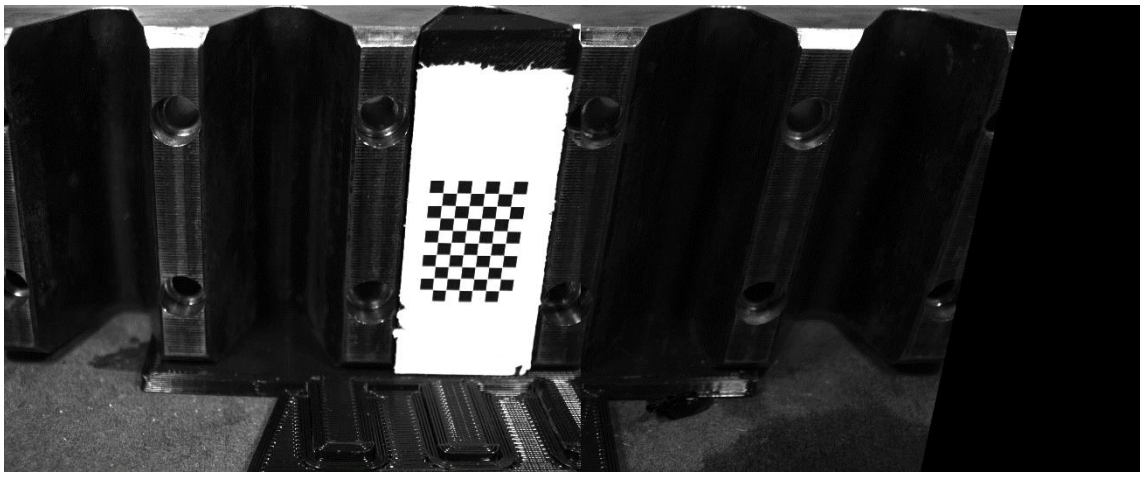

Figura 30 - Ilustração dos pontos em comum das duas imagens

Para detetar as retas, que corresponde ao laser, foram testados vários algoritmos. Recorreu-se ao algoritmo Canny [23], que deteta contornos na imagem, por ser o que

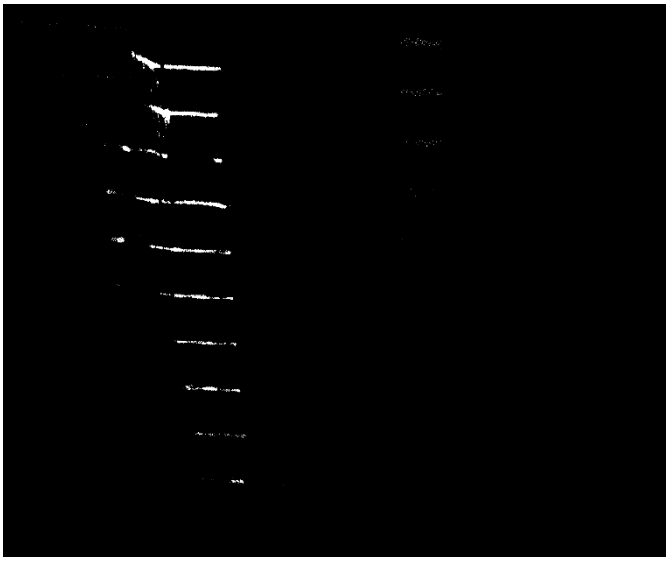

Figura 31 - Imagem com contraste pintado após uso do algoritmo Canny

demonstrou melhor deteção do contorno do laser no topo do dente. O seu uso necessita de primeiro tornar a imagem preto e branco, com valores de contraste definidos pelo utilizador.

Aqui surgiram dificuldades porque a luminosidade afeta a imagem e o seu contraste, pelo que reproduzir exatamente o mesmo ambiente com o mesmo nível de luz é difícil.

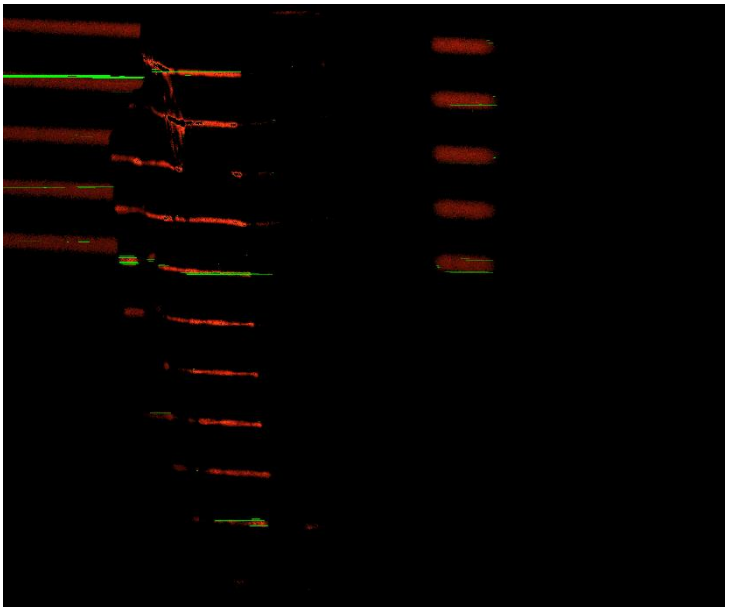

Figura 32 - Retas detetadas e desenhadas a verde através do algoritmo Hough Line

No fim recorre-se ao algoritmo Hough Line [24] para detetar retas dentro dos contornos detetados pelo algoritmo Canny. Como o algoritmo deteta várias retas de pixéis dentro de cada feixe de laser, e são projetadas dez linhas de feixes de laser, agrupou-se os dados de

```python
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 30, minLineLength = 20, maxLineGap=80)
edges = cv2.Canny(img_erosion, 50, 150)
lines2 = cv2.HoughLinesP(edges, 1, np.pi/180, 30, minLineLength = 20, maxLineGap=80)

l=[]
if lines is not None:
    for line in lines:
        #get values of each line, start and end
        x1, y1, x2, y2 = line[0]
        myradians = math.atan2(y2-y1, x2-x1)
        mydegrees = math.degrees(myradians)
        print(x2)
        #if mydegrees<20 and mydegrees>-20:
            #if x1>1800 and y1>300:
        cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 1)
        l.append(line[0])

data=pd.DataFrame(l)
#print(data)
X=np.array(l)
km = KMeans(n_clusters=10, random_state=0).fit(data)
cluster_map = data
#cluster_map['data_index'] = data.index.values
cluster_map['cluster'] = km.labels_

for i in range(10):
    data=cluster_map[cluster_map.cluster == i]
    #print(data)
    #print(data[0])
    x1=data[0].min()
    x11=int(data[0].mean())
    #print(data[2])
    x2=data[2].max()
    x22=int(data[2].mean())
    #print(x2-x1)
    lengthP = x2-x1
    length = (x2-x1) *0.006
    print("tamanho em cm", length)
    print("tamanho em pixeis", lengthP)
    y1=int(data[1].mean())
    print(y1)
    y2=int(data[3].mean())
    print(y2)
    cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
    cv2.line(img, (x11, y1), (x22, y2), (255, 0, 0), 2)
```

Figura 33 - Algoritmo Hough Line e algoritmo Kmeans para agrupar retas detetadas

retas detetadas através do algoritmo de Kmeans [25] e, dentro dos grupos formados, usar o valor mínimo e máximo da coordenada y do pixel para ter o valor da reta mais elevado. Também se fez uma média de cada grupo para obter o valor médio das retas detetadas, sendo inferior ao método anterior porque os contornos detetados pelo algoritmo Canny não são retos nas laterais dos feixes de laser, como visível na figura 32.

Dado as questões levantadas durante os testes que colocam a veracidade dos dados, o

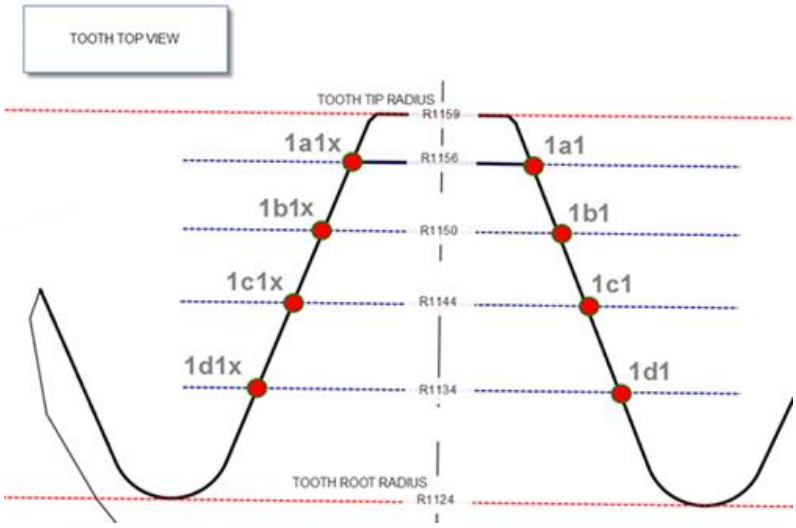

Figura 34 - Desgaste a calcular em cada dente

cliente alterou os dados que necessita durante a medição para poder ter precisão na análise. Ao invés de saber o desgaste no topo do dente, pretende-se saber o desgaste nas laterais do dente, a diferentes alturas, a fim de saber as zonas onde o dente desgasta mais facilmente e fazer análise do desgaste.

4.2. Segunda Abordagem

Figura 35 - Esquema da segunda abordagem

A equipa de desenvolvimento, nesta segunda abordagem, optou por determinar o desgaste nas laterias do dente, a diferentes alturas e para a sua realização foi utilizado um laser de medição apontado à face frontal dos dentes do *yaw ring*.

A distância obtida pelo laser, ao longo dos doze minutos e meio que demora o *yaw ring* a dar uma volta completa em modo de reparação, dará o perfil do dente ao longo do tempo. Sendo a relação tempo e distância a solução para determinar o desgaste dos dentes, calculou-se que é necessário obter uma medida do laser a cada décima de segundo para corresponder a 1mm. O laser de distância foi ligado a um PLC, que por sua vez este comunica com uma consola que indica os valores do laser e onde é possível efetuar o zero ao laser. Este zero é feito no vale do dente (local onde não tem desgaste) para que os valores positivos apresentados sejam a altura do dente.

É ligado um computador por ethernet ao PLC para ler as medidas apresentadas pelo laser.

```
const express = require('express');
const mongoose = require('mongoose')
  , Admin=mongoose.mongo.Admin;
const bodyParser = require('body-parser');
const fins = require('omron-fins-es6');

// IMPORT MODELS
require('./models/Measures');
require('./models/Teeths');
require('./models/Detrition');

const app = express();
const server = require('http').createServer(app)
const io = require('socket.io')(server)
```

Figura 36 - Bibliotecas NodeJS usadas

Sendo um PLC da Omron, o protocolo de comunicação é o FINS.

Recorreu-se a várias bibliotecas de NodeJS, visível na figura 36, e a de omron-fins [26] foi usada para a interligação, em que o seu funcionamento consiste em efetuar um pedido

```
// Connecting to remote FINS client on port 9600 with default timeout value.
var client = new fins.FinsClient(9600, '192.168.250.1');

// Setting up our error listener
client.on('error', function (error) {
  console.log("Error: ", error);
});

// Setting up the response listener
// Showing properties of a response
client.on('reply', function (msg) {
  if(msg.values){
  console.log("Data returned: ", msg.values);
  socket.emit('measures', msg.values);
  measures.push(msg.values[0]);
  all_measures.push(msg.values[0]);
  timers();
  }
  else{
    console.log('SID: ', msg.sid);
    console.log('Command Code: ', msg.command);
    console.log('Response Code: ', msg.response);
    console.log('Remote Host: ', msg.remotehost);
    socket.emit('status', msg.response);
  }
});
```

Figura 37 - Ligação ao PLC e função de escuta do protocolo FINS

de leitura ao um determinado byte do PLC e, através de uma função de escuta representada na figura 37, recebe o valor desse byte.

Como indicado anteriormente, durante a medição, são efetuados pedidos a cada décima de segundo. O início da medição é iniciado através de um pedido no dashboard criado, em React e visível na figura 38, na página de medição. A interligação entre o dashboard

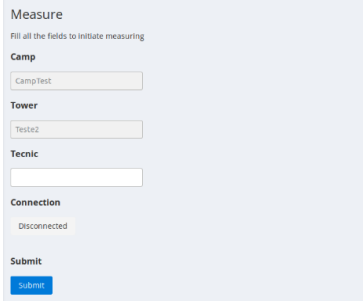

Figura 38 - Parte do dashboard em que o utilizador inicia medição

e o servidor NodeJS que interliga com o PLC é feita através de Socket.io [27] representado na figura 39.

```
socket.on('startMeasurement', function (msg) {
  console.log(msg);
  console.log(msg[2]);
  console.log(msg[1]);
  console.log(msg[0]);
  camp = msg[0];
  tower = msg[1];
  tecnic = msg[2];
  level = "1";
  if (started == false) {
    started = true;
    console.log('startMeasurement');
    socket.emit('startMeasurements', {
      status: 'Started',
      error: ''
    });
    let interval = 0010;

    //Call function dataread every one second
    tid = setInterval(measuring, interval);

    function measuring() {
      client.read('W00034', 1,function (){});
      time=time-interval;
    }

  } else {
    socket.emit('startMeasurements', {
      status: 'Measure already started',
      error: ''
    });
  }
});
```

Figura 39 - Código para iniciar medição

Quando terminada a medição, os valores da medição são guardados numa base de dados MongoDB, assim como a cada dente. Foi efetuada a medição de um dente novo para

```
function timers() {
  i++;
  if (i == 525) {
    teethOver();
    j++;
    i = 1;
    measures = [];
  }
  if (j == 143) {
    abortTimer();
    j = 0;
    i = 1;
    all_measures = [];
  }
}


function teethOver() { // to be called when you want to save data from each teeth
  socket.emit('startMeasurements', {
    status: 'Over teeth ' + String(dente),
    error: ''
  });
  insertData(camp, tower, tecnic, level, dente, measures);
}

function abortTimer() { // to be called when you want to stop the timer and save all data
  clearInterval(tid);
  socket.emit('startMeasurements', {
    status: 'Measure over',
    error: ''
  });
  insertAllData(camp, tower, tecnic, level, 'all', all_measures);
  started = false;
  medida1=0;
  medida2=0;
  medida3=0;
  medida4=0;
  desgaste=[];
  medidasTeste=[];
```

Figura 40 - Temporizadores para guardar dados de cada dente e da medição completa.

determinar o número total de pontos que corresponde a cada dente, e com esse cálculo criar um temporizador para guardar dados periodicamente (figura 40).

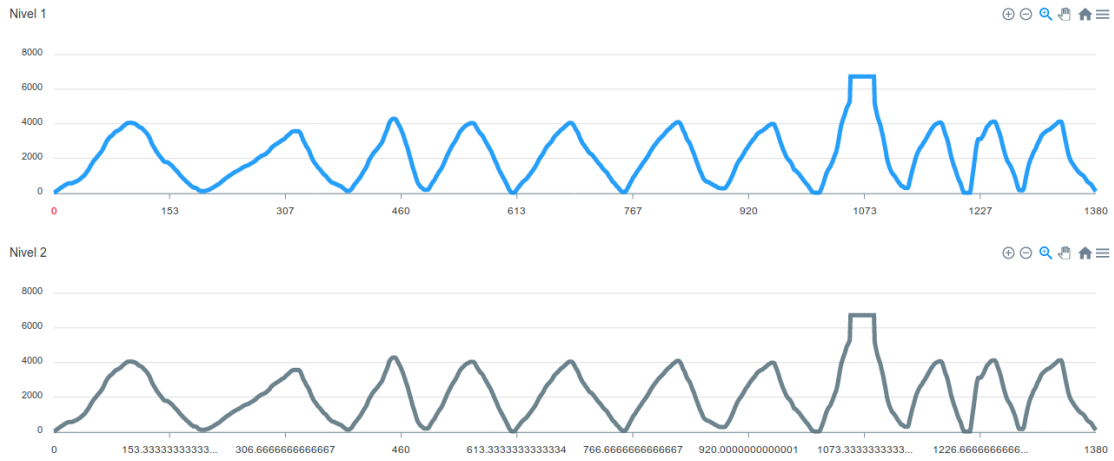

Figura 41 - Ilustração gráfica dos dados guardados

Os dados são acedidos pelo *dashboard* para apresentar os valores de medição graficamente, figura 41, assim como em formato tabular (figura 42).

Connect Open Save Export Format Options Fields Fullscreen

	1	2	3	4	5	6	7	8	9	10	11	12
1	_ID	CAMPO	TORRE	DATA.YEAR	DATA.MONTH	DATA.DAY	DENTE	NIVEL	MEDIDA1	MEDIDA2	MEDIDA3	MEDIDA4
2							30	21	1 554	2 070	2 610	2 970
3	5db6baff07de971893eb1eb6	campoTeste	torreTeste	2019	October	10	1	1	176	230	290	330
4	5db6d1e107de971893eb1eb7	campoTeste	torreTeste	2019	October	10	2	1	176	230	290	330
5	5db6d1e807de971893eb1eb8	campoTeste	torreTeste	2019	October	10	3	1	176	230	290	330
6	5db6d1ee07de971893eb1eb9	campoTeste	torreTeste	2019	October	10	4	1	176	230	290	330
7	5db6d1f407de971893eb1eba	campoTeste	torreTeste	2019	October	10	4	2	176	230	290	330
8	5db6d1f807de971893eb1ebb	campoTeste	torreTeste	2019	October	10	4	3	176	230	290	330
9	5db6d1fe07de971893eb1ebc	campoTeste	torreTeste	2019	October	10	4	4	176	230	290	330
10	5db6d32b07de971893eb1ebd	campoTeste	torreTeste	2019	October	10	4	4	166	230	290	330
11	5db6d33407de971893eb1ebe	campoTeste	torreTeste	2019	October	10	4	4	156	230	290	330
12												
13												

Figura 42 - Ilustração tabular dos dados guardados

4.3. Abordagem Final

Foi levantado pelo cliente questões sobre a relação tempo/distância usada na segunda abordagem, devido ao facto de a velocidade não ser constante, provocadas pelas oscilações produzidas pelo vento. Assim foi necessário adicionar um componente que indique a velocidade de rotação a fim de saber ao certo cada momento de sua rotação.

A abordagem final consiste em adicionar um encoder ao interior da torre dado que esta é fixa, e a *nacelle* (parte de cima da torre eólica que gira para se posicionar a favor do vento) ao girar em modo de manutenção o encoder capta a rotação desta. Através da rotação, calculando o número de voltas que o encoder dá sobre si para fazer uma volta completa na torre, consegue-se obter a posição em que o laser se encontra a cada momento e assim calcular o desgaste dos dentes.

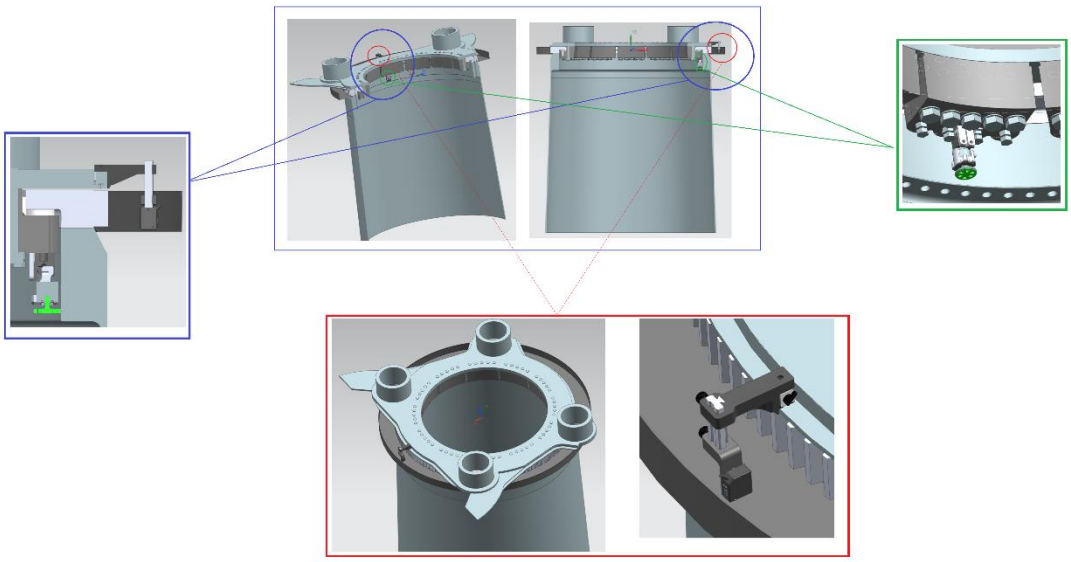

Figura 43 - Esquema da abordagem final

No término do estágio, esta abordagem encontrava-se na fase de implementação do encoder e alteração da lógica do servidor NodeJS, mas a lógica segue o mesmo princípio de leitura do laser que é ler o byte correspondente ao encoder no PLC.

5. Projeto 3iBioeconomia

No projeto 3iBioeconomia foi também aplicado a tecnologia MERN. Este projeto, financiado pelo Programa Portugal 2020 e pela União Europeia, através do FEDER, aborda o tema da utilização eficiente de recursos e sustentabilidade, através da inovação e consiste na produção de um sistema de zonas húmidas construídas com tratamento por ultravioleta, um sistema de sensorização de linhas de produção e a criação de conteúdos para os guias técnicos. O sistema é baseado num tanque de tratamento de água responsável por fazer a monitorização da água à entrada e à saída do tanque.

5.1. Arquitetura

Para atingir os objetivos esperados, são obtidos dados de sensores num pequeno tanque de água antes de esta entrar no tanque principal e recolhidos novos dados após a de água passar pelo tanque principal. Optou-se por fazer a comunicação dos sensores (ligados a ESP32) para um dispositivo central, com a designação de NUC (dispositivo da Intel[4]),

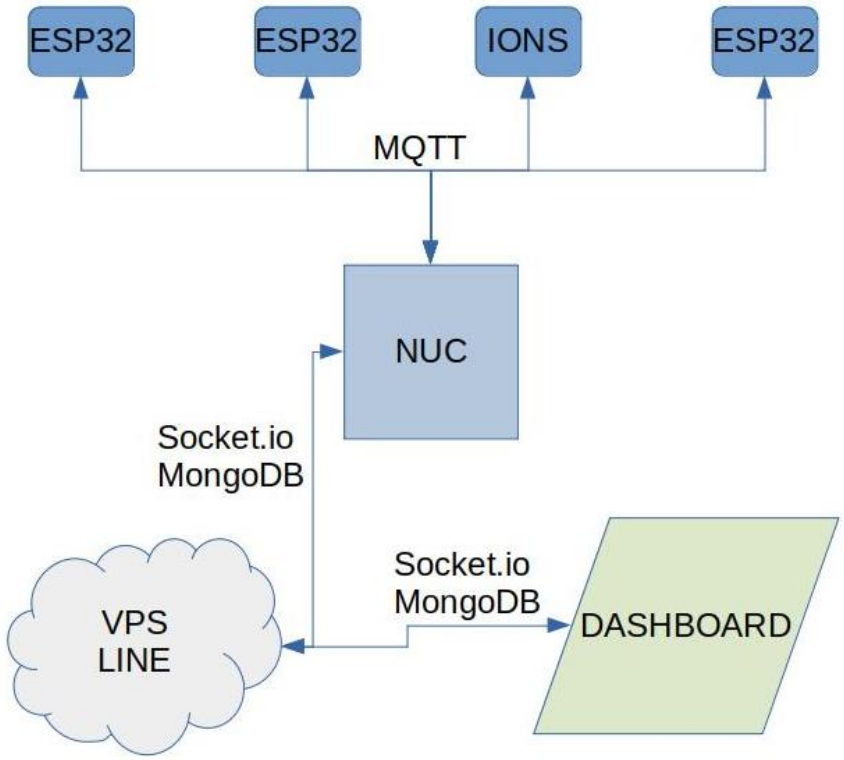

Figura 44 - Arquitetura informática 3iBioeconomia

[4] https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html

através do protocolo de comunicação MQTT. No sensor de teste de iões na água a aquisição da informação dos sensores é feito através de um pedido *http*, por este ser uma arquitetura fechada e disponibilizar uma API para leitura de dados.

5.2. Back-end on-site

Centralizada a informação obtida pelos sensores no dispositivo NUC, configurou-se aí

```
require('colors');
var mosca = require('mosca');
var MongoClient = require('mongodb').MongoClient;
//var urlMongo = "mongodb://localhost:27017/";
var urlMongo = "mongodb:[redacted]2:27017/";
var express = require('express');
var request = require('request');
var app = express();
var serverhttp = require('http').Server(app);
var port = 8086;
var socket=io.connect('http[redacted].232', {path: '/[redacted]/socket.io'});

//intervalo que o pedido mqtt é feito
var interval = 40000;

//contadores para controlo de comunicação dos esp32
var entradaCtl=1;
var geralCtl=1;
var saidaCtl=1;

```

Figura 45 - Bibliotecas e variáveis usadas no código NodeJs

um script em NodeJS, figura 45, com a função de efetuar as seguintes funcionalidades:

- MQTT broker: o servidor de MQTT que os ESP32 comunicam para enviar os dados obtidos pelos sensores;

- Guardar dados em MongoDB: conectar com a base de dados remota no servidor do LINE e guardar aí os dados obtidos quando recebe os dados via MQTT (figura 46);

```
// fired when a message is received
server.on('published', function(packet, client) {
  console.log('Published', packet.topic.toString());
  console.log('Published', packet.payload.toString());
  if(packet.topic.toString()== "nodeEntrada/Data"){
    entradaCtl=1;
    var payload = JSON.parse(packet.payload);
    payload.timestamp = Math.floor(Date.now() / 1000);
     MongoClient.connect(urlMongo, function(err, db) {
            if (err) throw err;
            var dbo = db.db("margaridos");
            dbo.collection("nodeEntrada").insertMany([payload], function(err, result1) {
            console.log("DONE INSERT ENTRADA payload");
            db.close();
          });
        });
    temppayload=JSON.parse(packet.payload.toString('utf-8'));
    mosca_browser_message['topic']=packet.topic+'/nodeEntrada';
    mosca_browser_message['payload'] = temppayload.data;
    //console.log(name + ": " + temppayload.data[name]);
    io.emit('data', mosca_browser_message);
  }
```

Figura 46 - Código que recebe dados via MQTT

- Cliente Socket.io: comunicar por socket.io com o servidor remoto na VPS (Virtual Private Server) do LINE, com a finalidade de poder enviar dados *realtime* e poder fazer alterações na periodicidade de obtenção de dados;

```
function alertControl(){
  entradaCtl++
  geralCtl++
  saidaCtl++
  //waterCtl++
  if(entradaCtl==60){
    console.log("No signal form entrada 60sec");
  }
  else if(entradaCtl==120){
    console.log("No signal form entrada 120sec");
  }
  else if(geralaCtl==0){
    console.log("No signal form geral 60sec");
  }
  else if(saidaCtl==120){
    console.log("No signal form saida 60sec");
  }
}

function loopCtl(){
  alertControl();
  setTimeout(loopCtl,1000);
}

loopCtl();
```

Figura 47 - Temporizadores de controlo das comunicações MQTT

- Controlos na verificação de conexão MQTT: contadores que reiniciam sempre que recebem uma publicação MQTT de um ESP32, a fim haver poder enviar alertas para quando estes deixam de comunicar (figura 47);

- Loop de pedido de dados (figura 48): os dados são obtidos através de pedidos feitos no dispositivo central NUC, com uma determinada periodicidade. No caso dos ESP32, estes estão subscritos a um tópico que, ao receber uma publicação, enviam os dados por MQTT. Já no sensor de iões, é feito um pedido HTTP à API. Ficando a periodicidade do lado do NUC, obtém-se mais controlo no sistema e torna possível alterar a periodicidade sem ter de alterar código nos ESP32.

```
function publish(){
  if(clientMqtt.connected==true){
    request('http://192.168.0.133/api/last/sample', {json:true},(err,res,body) =>{
      if(err){return console.log(err);}
      console.log(body.test);
      var payload = body.test;
      payload.timestamp = Math.floor(Date.now() / 1000);
      MongoClient.connect(urlMongo, function(err, db) {
        if (err) throw err;
        var dbo = db.db("margaridos");
        dbo.collection("ioes").insertMany([payload], function(err, result1) {
          console.log("DONE INSERT ioes payload");
          db.close();
        });
      });
    });
    clientMqtt.publish('sensors');
  }
}

function loop(){
  publish();
  setTimeout(loop,interval);
}

loop();
```

Figura 48 - Função em loop que efetua o pedido de dados dos sensores

5.3. Back-end online

Na VPS do line é onde se aplica a metologia MERN a 100%. O script NodeJS recorre à

```
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");
const passport = require("passport");
const path = require('path')
const router = require('./routes/margaridos-router')
const app = express()
const http = require('http')
const socketIo = require('socket.io')


// Bodyparser middleware
app.use(
  bodyParser.urlencoded({
    extended: false
  })
);
app.use(bodyParser.json());

// DB Config
const db = require("./config/keys").mongoURI;

// Connect to MongoDB
mongoose
  .connect(
    db,
    { useNewUrlParser: true, dbName: 'margaridos' }
  )
  .then(() => console.log("MongoDB successfully connected"))
  .catch(err => console.log(err));

// Routes
app.use('/api', router)

// Passport middleware
app.use(passport.initialize());
```

Figura 49 - Parte do código NodeJS que alimenta o dashboard

framework ExpressJS para gerir os pedidos de front-end (Dashboard) com o uso do middleware "bodyparser" para lidar com os pedidos HTTP POST, e também o middleware "passport" para fins de autenticação (figura 49).

Aqui é usado a biblioteca "mongoose" ao invés de "mongodb" para comunicar com a base de dados, por esta ter mais opções como a de criar um esquema do documento que

se deseja inserir na base de dados. Este esquema serve de verificação antes de guardar na

```
app.use(express.static(path.join(__dirname,'build')));


app.get('/*', (req, res)=>
  {  res.sendFile(path.join(__dirname, 'build', 'index.html'));
})

const server = http.createServer(app);
const io = socketIo(server);

io.on("connection", (socket) => {
  console.log("New client connected");

  socket.on("entrada", (msg) =>{
    //console.log(msg);
  });

  socket.on("updateIntervalData", (msg) => {
    if(typeof msg == "number"){
      socket.emit("setIntervalData", msg);
    }
  });

  socket.on("infoIntervalData", () => {
    socket.emit("getIntervalData");
  })

  socket.on("intervalData", (msg) => {
    socket.emit("answerIntervalData",msg);
  })

  socket.on("disconnect", () => {
    console.log("Client disconnected");
  });
});
```

Figura 50 - Funções de Express e Socket.io em uso

base de dados, dando assim mais segurança e robustez ao sistema.

É configurado o servidor de Socket.io, visível na figura 50, para comunicar com o NUC e com o Dashboard. Uma da funcionalidade é o utilizador poder alterar a periodicidade do pedido de dados dos sensores, em que o pedido é feito por socket.io e o servidor ao receber esse pedido, encaminha para o NUC.

Por fim, executa-se o ficheiro que fica a escutar numa porta do servidor (figura 51) e, ao ligar, o Express encaminha para o ficheiro “index.html” na pasta “build”, pasta esta que foi criada pelo React.

```
const port = process.env.PORT || 5000;
server.listen(port, () => console.log(`Server up and running on port ${port} !`));
```

Figura 51 - Código que coloca o servidor NodeJS online

5.4. Front-end

O frontend desenvolvido em React, sendo baseada em componentes, apresenta a sua estrutura desenhada em duas partes:

1. Componente principal (*App*, figura 53), nível máximo na hierarquia ilustrada na figura 52, composto por um roteador que encaminha entre 3 componentes: o de login, de registo e, caso esteja autenticado, encaminho para dashboard, com o

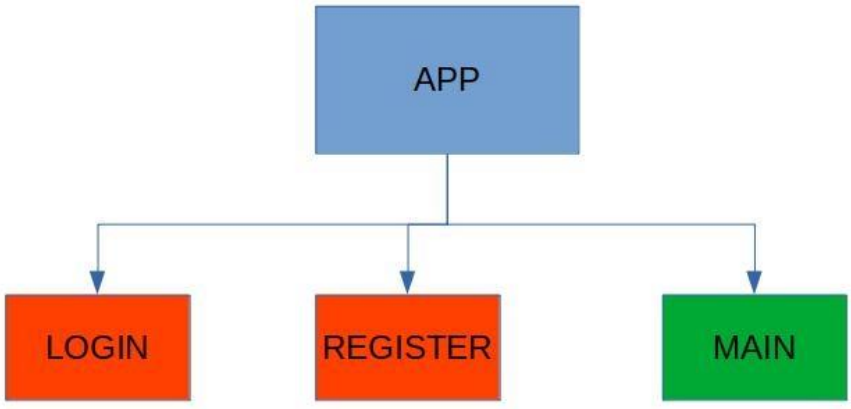

Figura 52 - Estrutura dos componentes React no topo da hierarquia

```
function AuthLayout() {
  return (
    <div>
      <Route path="/auth/register" exact component={Register} />
      <Route path="/auth/login" exact component={Login} />
      <Redirect from="/auth" to="/auth/login" exact />
      <Route />
    </div>
  );
}

function Layouts() {
  return (
    <Switch>
        <Route path="/auth" component={AuthLayout} />
        <PrivateRoute path="/app" component={Main} />
        <Redirect from="/" to="/auth" exact />
        <Route component={NotFound} />
        <Route />
    </Switch>
  );
}

class App extends Component {


  render() {
    return (
      <Provider store={store}>
        <Router basename={process.env.PUBLIC_URL} >
          <Layouts />
        </Router>
      </Provider>
    );
  }
```

Figura 53 - Parte do código React do componente App

nome de componente *Main*. Em suma, controla a vista do utilizador quando não está autenticado.

2. Componente *Main* (figura 55) composto pelos componentes *Header* e *Footer*, que consiste no cabeçalho e rodapé do site comum em todas as páginas, e um roteador que muda o componente renderizado no centro da página (figura 54). É a vista

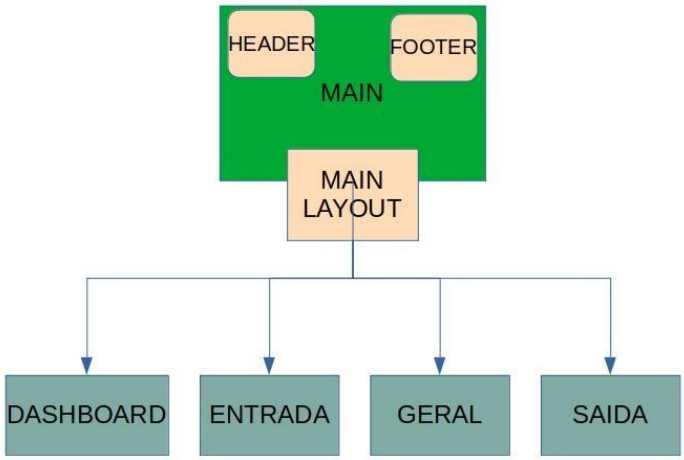

Figura 54 - Estrutura dos componentes React na segunda camada da hierarquia

que o utilizador tem quando se encontra autenticado.

```
return (
    <div className={wrapperClass} onClick={this.onWrapperClick}>
        <AppTopbar onToggleMenu={this.onToggleMenu}/>
        <div ref={(el) => this.sidebar = el} className={sidebarClassName} onClick={this.onSidebarClick}>
            <div className="layout-profile">
                <div>
                    <img src={Avatar} alt="" />
                </div>
                    <span className="profileName">{user.name.split(" ")[0]}</span>
            </div>
            <MainMenu model={this.menu} onMenuItemClick={this.onMenuItemClick} />
            <div className="layout-logo logo">
                <img alt="Logo" src={Logo} style={{ width: "100%"}} />
            </div>
        </div>

        <div className="layout-main">
        <HashRouter>
          <ScrollToTop>
          <Switch>
                <Route exact path="/" component={Dashboard} />
                <Route exact path="/entrada" component={Entrada} />
                <Route exact path="/geral"component={Geral} />
                <Route exact path="/saida" component={Saida} />
                <Route component={NotFound} />
            </Switch>
          </ScrollToTop>
        </HashRouter>
        </div>

        <AppFooter />

        <div className="layout-mask"></div>
    </div>
)
```

Figura 55 - Parte do código do componente Main

O resultado é um dashboard de página única (figura 56), mudando a vista da "div" principal consoante o que o utilizador deseja ver, como por exemplo os dados do tanque de entrada representado na figura 57.

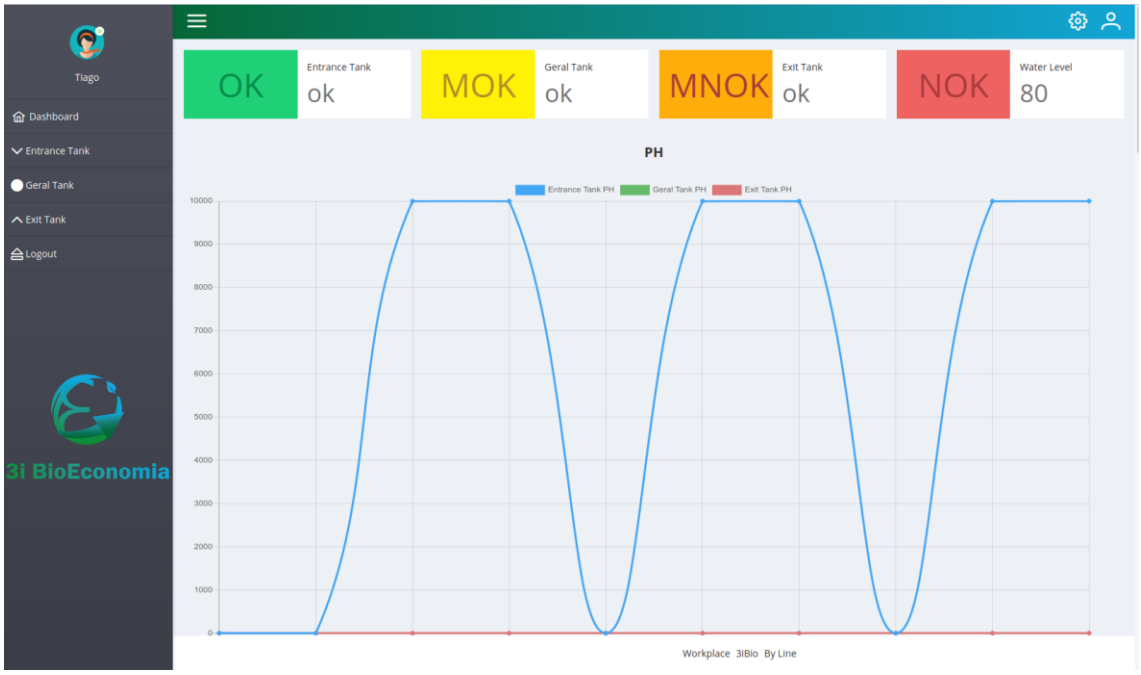

Figura 56 - Página online desenvolvida em React, Dashboard

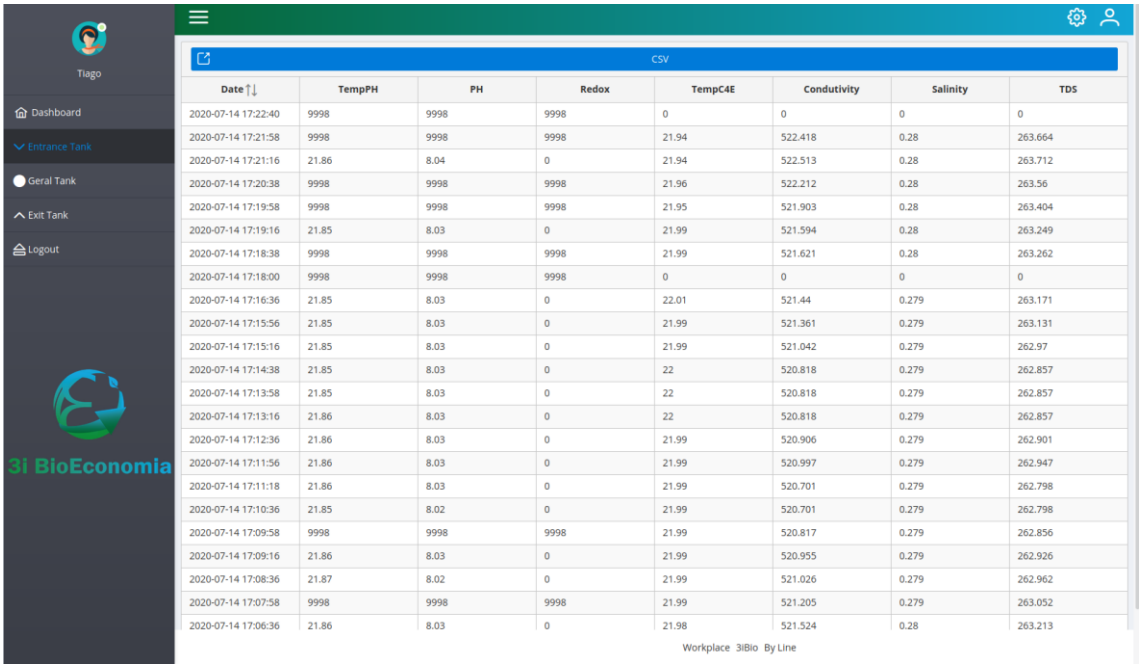

Figura 57 - Página online desenvolvida em React, mostrando os dados em formato tabular

6. Site TAGUSVALLEY

Entre setembro de 2019 e fevereiro de 2020, o LINE recebeu um estagiário de licenciatura em Informática e Tecnologias Multimédia (ITM) da Escola Superior Tecnologia de Abrantes. A sua orientação ficou a meu cargo e o seu trabalho era projetar e desenvolver o novo site da TAGUSVALLEY (figura 58). O site foi desenvolvido com recurso às linguagens de programação PHP, HTML, CSS e JS, por ser as lecionadas no curso ITM. Partiu-se de um design fornecido por uma entidade especializada, externa, ao TAGUSVALLEY, mas cuja arquitetura e desenvolvimento foi desenhada pelo LINE (https://tagusvalley.pt/index.php).

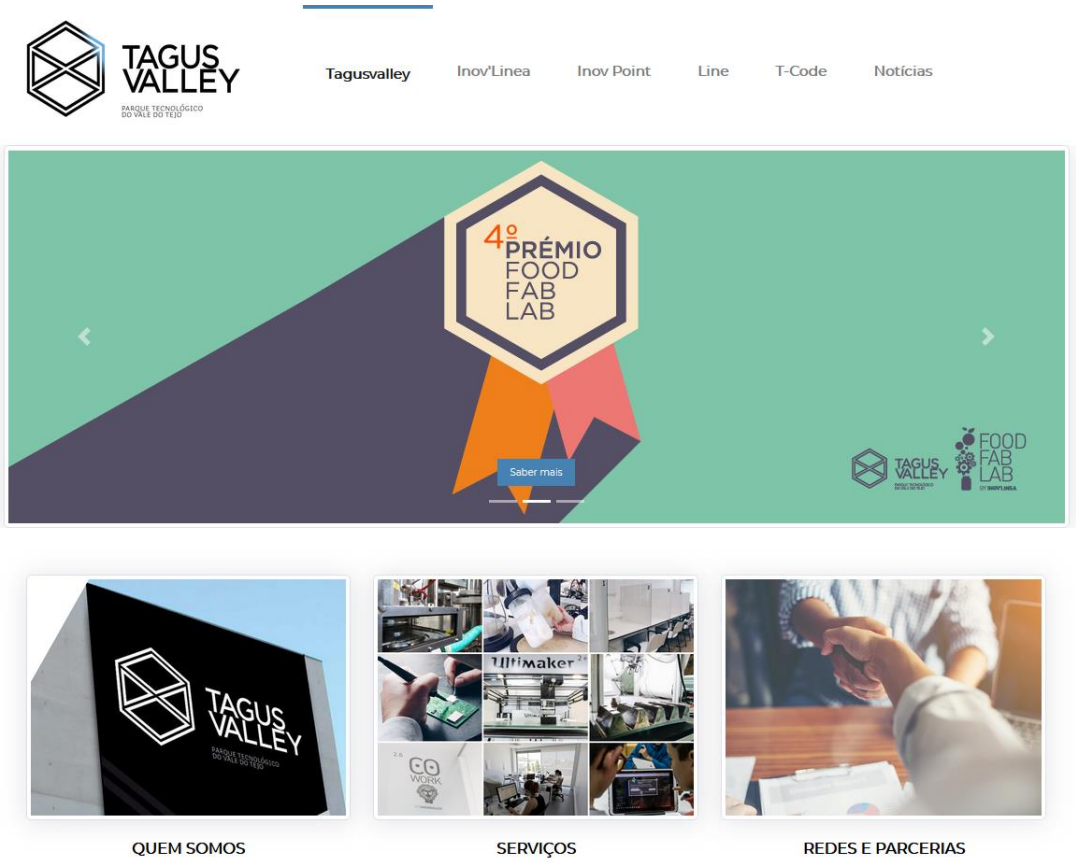

Figura 58 - Frontend site TAGUSVALLEY

A sua estrutura, figura 59, foi implementada da seguinte forma:

- Páginas para os visitantes verem toda a informação da TAGUSVALLEY, seguindo o design dado;

- Autenticação para aceder ao backoffice;

- Páginas de backoffice para inserção e edição de notícias, projetos e alteração de estatísticas visíveis aos visitantes;

- Auto geração de uma página HTML com a notícia inserida;

- Uma API para interligar com os modelos de base de dados.

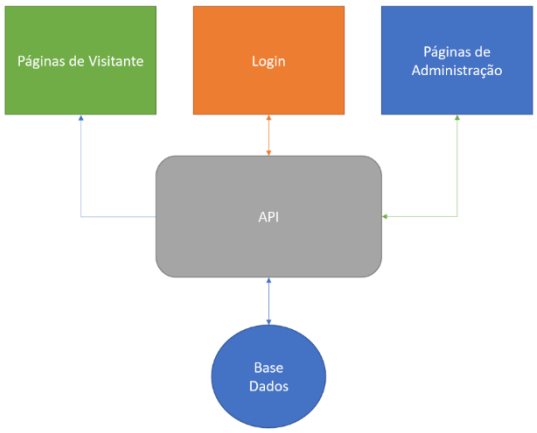

Figura 59 - Arquitetura site TAGUSVALLEY

Muito do trabalho foi efetuado pelo estagiário, pelo que será aqui abordado apenas as partes do site onde a minha intervenção foi necessária.

```php
<?php
//declaração da classe noticia
class news {
    protected $connection;
    protected $tabela="tg_noticia";
    protected $primarykey="IdNoticia";

    public function __construct() {
        $this->connection=mysqli_connect("[redacted]");
        mysqli_set_charset($this->connection,'utf8');
    }
    public function __destruct() {
        mysqli_close($this->connection);
    }
```

Figura 60 - Declaração da classe noticia

- Criação de classes (figura 60 e 61): As operações CRUD nas tabelas de notícias e projetos passam pela classe respetiva, havendo assim encapsulamento e especialização em cada classe.

```php
public function update($id,$idUser, $idTags, $colunas=array()) {
    if (!$id>0) return "A chave é obrigatória";
    if (sizeof($colunas)==0) return "Deve atualizar pelo menos uma coluna";
    $sql="UPDATE ".$this->tabela." SET ";
    //ciclo para iterar as colunas a atualizar
    $colunas2=array();
    foreach($colunas as $coluna=>$valor) {
        $colunas2[]="$coluna='$valor'";
    }

    $sql1="SELECT IdUtilizador FROM tg_utilizador WHERE Utilizador ='". $idUser . "'";
    $results = mysqli_query($this->connection, $sql1);
    if (mysqli_num_rows($results) > 0) {
        while($result=mysqli_fetch_array($results)) {
            $valor=$result['IdUtilizador'];
            array_push($colunas2,"utilizador_IdUtilizador=$valor");
        }
        $sql.=implode(", ", $colunas2);  //transformar o array numa string, usando um separador
        //$sql=substr($sql,0,strlen($sql)-1);
        //filtro colunas a atualizar
        $sql.=" WHERE ".$this->primarykey."=".$id;
        //echo $sql;
        mysqli_query($this->connection, $sql) or die(mysqli_error($this->connection));
        if($idTags>0){
            $sqlDel="DELETE FROM tg_noticia_has_tg_tag WHERE noticia_IdNoticia=".$id;
            $delete=mysqli_query($this->connection, $sqlDel) or die(mysqli_error($this->connection));
            for ($i = 0; $i <sizeof($idTags); $i++) {
                $sql2="INSERT INTO tg_noticia_has_tg_tag(noticia_IdNoticia, tag_IdTag) VALUES (".$id.",".$idTags[$i].")";
                $insert2=mysqli_query($this->connection, $sql2) or die(mysqli_error($this->connection));
            }
        }
        else{
            return "tags invalid";
        }
        return TRUE;
    }
```

Figura 61 - Função de atualizar notícia dentro da classe notícia

- Controlador de login (figura 62): Controlo e validação nos dados inseridos, onde a resposta desse controlo é enviado por JSON para o lado do cliente.

```php
<?php

include "../conf/conn.php";

session_start();

$errors = array();
$data = array();

$password = mysqli_real_escape_string($conn, $_POST['password']);
$email=mysqli_real_escape_string($conn, $_POST['username']);

if (empty($email)) {
    array_push($errors, "Insira um email");
}
if (empty($password)) {
    array_push($errors, "Insira uma password");
}

if (!empty($errors)) {
    $data['success'] = false;
    $data['errors'] = $errors;
}else{
    $password_query = "SELECT Utilizador, password FROM tg_utilizador WHERE email='$email'";
    $result_password = mysqli_query($conn, $password_query);
    $password_good = mysqli_fetch_assoc($result_password);

    $password = password_verify( $password, $password_good["password"]);
    $username = $password_good["Utilizador"];

    if ($password == true) {
        $_SESSION['email'] = $email;
        $_SESSION['username'] = $username;
        $_SESSION['connected'] = true;
        $_SESSION['success'] = "Está logado!";
    } else {
        array_push(  $errors, "Email ou Password errados");
        $data['success'] = false;
        $data['errors'] = $errors;
    }
}

    echo json_encode($data);

?>
```

Figura 62 - Controlador de login

- Controlador de recuperação de password (figura 63): Caso exista o e-mail na base de dados, é gerado uma password aleatória que, depois de atualizado na base de dados, é enviada por e-mail ao utilizador.

```php
<?php

include "../conf/conn.php";
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

/* Exception class. */
require '../conf/Exception.php';

/* The main PHPMailer class. */
require '../conf/PHPMailer.php';

/* SMTP class, needed if you want to use SMTP. */
require '../conf/SMTP.php';

session_start();

$errors = array();
$data = array();

$email = strtolower(mysqli_real_escape_string($conn, $_POST['email']));

$user_check_query = "SELECT * FROM tg_utilizador WHERE email = '$email' LIMIT 1";
$result_register = mysqli_query($conn, $user_check_query);
$user = mysqli_fetch_assoc($result_register);

if ($user) {

    $alphabet = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890';
    $pass = array(); //remember to declare $pass as an array
    $alphaLength = strlen($alphabet) - 1; //put the length -1 in cache
    for ($i = 0; $i < 8; $i++) {
        $n = rand(0, $alphaLength);
        $pass[] = $alphabet[$n];
    }
    $password = implode($pass); //turn the array into a string

    $data["password"] = $password;


    $options = [
        'cost' => 12,
    ];

    $passwordEncrypted = password_hash($password, PASSWORD_BCRYPT, $options);
    $register_query = "UPDATE tg_utilizador SET password = '$passwordEncrypted' WHERE email = '$email'";
    mysqli_query($conn, $register_query);

    $mail = new PHPMailer(TRUE);
```

Figura 63 - Controlador de recuperação de password

- Atualizar o perfil de utilizador (figura 64): Verificação dos dados inseridos pelo utilizador e, caso tenham sido corretamente, há uma verificação na base de dados

```php
if (!empty($errors)) {
    $data['success'] = false;
    $data['errors'] = $errors;
}else{
    $user_check_query = "SELECT * FROM tg_utilizador WHERE email = '$email' LIMIT 1";
    $result_query = mysqli_query($conn, $user_check_query);
    $user = mysqli_fetch_assoc($result_query);

    if ($user) {
        if ($user['email'] === $email && $user['Utilizador'] != $username) {
            array_push($errors, "Email já existe");
        }else{
            $password = password_verify( $password, $user["password"]);

            if($password != true){
                array_push($errors, "A password actual está errada");
            }
        }
    }

    if (!empty($errors)) {
        $data['success'] = false;
        $data['errors'] = $errors;
    }else{
        if(empty($password)){
            $update_query = "UPDATE users SET name = '$name', email ='$email', updated_at = NOW() WHERE Utilizador = '$username' ";
        }else{

            $options = [
                'cost' => 12,
            ];
            $new_password = password_hash($new_password, PASSWORD_BCRYPT, $options);
            $update_query = "UPDATE tg_utilizador SET Utilizador = '$name', email ='$email', password='$new_password', updated_at = NOW() WHERE
                Utilizador = '$username'";
        }
        $data['mysql'] = mysqli_query($conn, $update_query);

        $_SESSION['username'] = $name;
        $_SESSION['email'] = $email;
    }
}


    echo json_encode($data);
```

Figura 64 - Controlador de atualização dos dados de utilizador

a fim de não haver duplicação de email e se password atual foi digitada corretamente.

- API para pedidos de inserção de uma notícia ou editar uma notícia (figura 65): Há uma verificação prévia que todos os elementos necessários para a sua inserção/edição foram enviados por Ajax. Caso se verifique que sim, a imagem referente ao *banner* da notícia é guardada na pasta correspondente e o conteúdo da notícia inserido na base de dados e num gerador de página HTML (figura 66).

```php
else if($_POST["oper"]=="insert"){
    if (isset($_POST["InputAutorMensagem"])
    and isset($_POST["InputTituloMensagem"])
    and isset($_POST["InputResumoMensagem"])
    and isset($_POST["mensagem"])
    and isset($_POST["Imagem"])){
        $news=new news();
        $registo=array();
        $registo["Titulo"]=$_POST["InputTituloMensagem"];
        $registo["Resumo"]=$_POST["InputResumoMensagem"];
        $registo["Mensagem"]=$_POST["mensagem"];
         //$registo["imagem"]=$_POST["data"]["Imagem"];

         $name_temp = $registo["Titulo"];


         $name = strtr( $name_temp, $unwanted_array );

        $image = $_POST["Imagem"];
        $nameImg = $name.".png";
        $path = "assets/img/noticias";
        file_put_contents(
            $path."/".$nameImg,
            base64_decode(
                str_replace('', '', $image)
            )
        );
        $mainImage= $path."/".$nameImg;
        $registo["imagem"]=$mainImage;
        $imgSource="../".$registo["imagem"];
        $date=date("Y-m-d");
        $pathImg="assets/img/noticias";

        $resultado=$news->insert($_POST["InputAutorMensagem"],$_POST["TagId"],$registo,$mainImage);
        include_once "creatHtml.php";
        creatHtml($name,$registo["Titulo"],$pathImg,$nameImg,$registo["Resumo"],$mainImage,$registo["Mensagem"],$date,$resultado);
        $result= json_encode(["sucesso"=>$resultado]);
    }
    else{
        $result= json_encode(["sucesso"=>FALSE]);
    }
```

Figura 65 - Funções executadas quando efetuado um pedido de inserção de noticia à API

Este gerador foi desenhado, em PHP, para criar a página HTML da notícia para poder ser possível de partilhar via redes sociais, nomeadamente Facebook. O Facebook usa meta dados da página para gerar o pré-visualização e o link para

```php
<?php

 function creatHtml($name,$titulo,$path,$nameImg,$resumo,$imgSource,$mensagem,$date){

                    $myfile = fopen("noticias/$name.html", "w") or die("Unable to open file!");
                $txt .= '<!DOCTYPE html>
<html>
    <head><meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">

        <meta name="title" content="TagusValley">
        <meta name="description" content="Parque Tecnologico do Vale do Tejo">
        <meta property="og:type" content="article" />
        <meta property="og:url" content="https://tagusvalley.pt/noticias/' . $name . ".html" . '" />
        <meta property="og:title" content="'. $titulo .'" />
        <meta property="og:image" content="https://tagusvalley.pt/assets/img/noticias/'.$nameImg . '" />
        <meta property="og:description" content="' . $resumo .'" />
        <meta property="fb:app_id" content="475827453052026" />
        <title>TagusValley</title>
        <link rel="icon" type="image/jpeg" sizes="454x454" href="../assets/img/11204472_1632172660369358_2319159749603799843_n.jpg">
        <link rel="stylesheet" href="../assets/bootstrap/css/bootstrap.min.css">
        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Archivo">
        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lora">
        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Montserrat">
        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Raleway">
        <link rel="stylesheet" href="../assets/fonts/fontawesome-all.min.css">
        <link rel="stylesheet" href="../assets/fonts/font-awesome.min.css">
        <link rel="stylesheet" href="../assets/fonts/fontawesome5-overrides.min.css">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/5.0.0/normalize.min.css">
        <link rel="stylesheet" href="../assets/css/styles.css">
        </head>
        <body>
        <header>
```

Figura 66 - Função que cria o ficheiro HTML da notícia dinamicamente

página, pelo que esta informação é estática e cada notícia necessita de a sua própria página.

- Criação e edição de notícias/projetos: para o utilizador inserir os dados referentes à notícia/projeto que deseja, recorreu-se à biblioteca de JS TinyMCE [28] (figura 67) para criar um editor de texto com opção de inserção de imagens, links, e alterar formatação (figura 68).

```html
<script src="../assets/js/tinymce/tinymce.min.js"></script>
<script>
    tinymce.init({
      selector: '#mensagemEdit',
      height: 600,
      plugins: "link lists advlist image fullscreen help media",
      toolbar: 'undo redo | styleselect | bold italic | alignleft aligncenter alignright alignjustify | outdent indent | numlist bullist | link | media | image | responsivefilemanager | fullscreen | help ',
      link_assume_external_targets: true,
      images_upload_url: 'imageUpload.php',
      image_dimensions: false,
      image_class_list: [
          {title: 'Responsive', value: 'img-fluid'}
        ],
      file_picker_types: 'image',
      media_poster: true,
      media_live_embeds: true,
      external_filemanager_path: "../../assets/js/filemanager/",
      filemanager_title: "Responsive Filemanager",
      external_plugins: {
        "responsivefilemanager": "plugins/responsivefilemanager/plugin.min.js",
        "filemanager": "../../js/filemanager/plugin.min.js"
        },
    });
</script>
```

Figura 67 - Script que executa a bilblioteca TinyMCE para criar o editor de texto

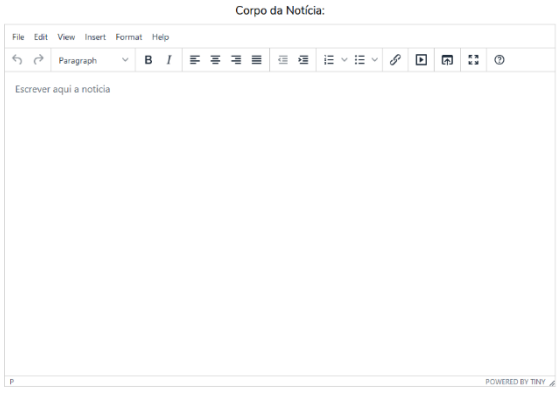

Figura 68 - Resultado visível ao utilizador da biblioteca TinyMCE

7. Conclusão

No âmbito do estágio no LINE.IPT, o aluno estagiário integrou o projeto VESTAS e no projeto 3iBioeconomia e efetuou trabalho de coordenação de estagiários, onde foi implementado o site da TAGUSVALLEY, por serem o que apresentaram os resultados mais relevantes.

O trabalho apresentado neste documento incide maioritariamente no uso da tecnologia MERN, que trouxe uniformidade no desenvolvimento do trabalho/estágio ao usar JavaScript na construção de aplicações MVC.

NodeJS, sendo o motor desta STACK, traz enormes vantagens por ser o ideal para lidar com aplicações de dados (sendo os projetos virados para IOT e Indústria 4.0, é habitual usar sensores e tratar os dados destes), por ter inúmeros pacotes disponíveis (destaca-se o socket.io e MQTT, usados no estágio), e pela forma simples de usar o formato JSON.

A framework Express simplifica a criação de servidor web, facilitando a interação com o front-end, em que a única desvantagem se deve à gestão dos *callbacks*, a mesma de NodeJS.

O uso de React torna o desenvolvimento do front-end mais organizado e compreensível, e a interação com o back-end em NodeJS é natural. Torna-se mais complexo pela juventude da linguagem e estar em constante melhoramento, onde por vezes para a mesma solução há duas formas diferentes de a resolver, mas uma não é compatível com a forma que escrevemos o código até aquele ponto.

MongoDB é a base de dados não relacional mais popular, bastante robusta e bem documentada, que é simples interagir com NodeJS não só com o pacote node MongoDB assim como o pacote node Mongoose.

Ao interagir com PLCs, através do protocolo FINS, foi possível conhecer as suas funcionalidades, utilidades e robustez. Dado que existem PLCs que utilizam o protocolo MQTT para enviar os dados nos bytes para o exterior, tornam-se equipamentos bastante bons para ambientes industriais em certos projetos de IOT.

Relativamente aos projetos descritos, foram atingidos todos os objetivos propostos.

No projeto VESTAS, à data do término do estágio, apenas faltava terminar a implementação do componente que indica a posição de cada dente durante e rotação do *yaw ring*, o encoder, que juntamente com o laser de medição capturam os dados necessários para o cálculo de desgaste dos dentes.

Em relação ao projeto 3iBioeconomia, o sistema de sensorização para recolha de dados da água foi testado com sucesso tanto antes de esta entrar no tanque de tratamento, assim como ao sair do tanque. A implementação no local, à data do término do estágio, está pendente de uma licença pois a água tratada irá para o sistema de rega e esta, através do ciclo da água, entrará nos recursos aquáticos naturais.

Referências

[1] TagusValley, “TagusValley,” TagusValley, 06 11 2019. [Online]. Available: https://tagusvalley.pt/. [Acedido em 10 07 2020].

[2] Silicon Valley Software Group, “How to Choose Your Tech Stack,” Silicon Valley Software Group, [Online]. Available: https://svsg.co/how-to-choose-your-tech-stack/. [Acedido em 10 7 2020].

[3] MongoDB, Inc., “What is MERN Stack? Introduction & Examples | MondoDB,” MongoDB, Inc., [Online]. Available: https://www.mongodb.com/mern-stack. [Acedido em 10 07 2020].

[4] Guru99, “What is MongoDB?,” Guru99, [Online]. Available: https://www.guru99.com/what-is-mongodb.html. [Acedido em 11 07 2020].

[5] MongoDB, Inc., “The most popular database for modern apps | MongoDB,” MongoDB, Inc., [Online]. Available: https://www.mongodb.com/. [Acedido em 11 07 2020].

[6] intellipaat.com, “What is MongoDB ? Introduction, Architecture, Features & Example,” intellipaat.com, [Online]. Available: https://intellipaat.com/blog/what-is-mongodb/. [Acedido em 11 07 2020].

[7] DataFlair, “Advantages od MongoDB | Disadvantages od MongoDB - DataFlair,” DataFlair, 14 09 2018. [Online]. Available: https://data-flair.training/blogs/advantages-of-mongodb/. [Acedido em 11 07 2020].

[8] Wikimedia Foundation, Inc, “Express.js - Wikipedia,” Wikipedia, 22 05 2020. [Online]. Available: https://en.wikipedia.org/wiki/Express.js. [Acedido em 11 07 2020].

[9] K. Kononenko, “Going out to eat and understanding the basic of Express.js,” Kevin Kononenko, 03 11 2017. [Online]. Available: https://www.freecodecamp.org/news/going-out-to-eat-and-understanding-the-basics-of-express-js-f034a029fb66/. [Acedido em 11 07 2020].

[10] Nitin Pandit, “What and Why React.js,” C# Corner, 5 03 2020. [Online]. Available: https://www.c-sharpcorner.com/article/what-and-why-reactjs/.

[11] B. Halpern, “Explain React.js Like I'm Five - DEV,” DEV, 5 03 2018. [Online]. Available: https://dev.to/tiffanywismer/explain-reactjs-like-im-five--2606. [Acedido em 11 07 2020].

[12] M. S. Negi, “React Components Explained. | by Manok Singh Negi | codeburst,” Medium, 27 03 2017. [Online]. Available: https://codeburst.io/react-components-explained-96718311f20b. [Acedido em 11 07 2020].

[13] javaTpoint, “Pros and Cons of ReactJs - javatpoint,” javaTpoint, [Online]. Available: https://www.javatpoint.com/pros-and-cons-of-react. [Acedido em 11 07 2020].

[14] Matt Warcholinski, “What are The Pros And Cons Of React JS? | Blog Brainhub.eu,” Brainhub, [Online]. Available: https://brainhub.eu/blog/pros-and-cons-of-react-js/. [Acedido em 11 07 2020].

[15] Wikipedia, “Node.js - Wikipedia,” Wikimedia Foundation, Inc, 01 07 2020. [Online]. Available: https://en.wikipedia.org/wiki/Node.js. [Acedido em 12 07 2020].

[16] nodejs, “About | Node,js,” OpenJS Foundation, [Online]. Available: https://nodejs.org/en/about/. [Acedido em 12 07 2020].

[17] TutorialsTeacher.com, “What is Node.js?,” [Online]. Available: https://www.tutorialsteacher.com/nodejs/what-is-nodejs. [Acedido em 12 07 2020].

[18] AltexSoft, “Pros and Cons of Node.js Web App Development | AltexSoft,” AltexSoft, 21 10 2019. [Online]. Available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/. [Acedido em 12 07 2020].

[19] Wikimedia Foundation, Inc, “Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Factory_Interface_Network_Service. [Acedido em 18 08 2020].

[20] Universidade do Porto, “https://paginas.fe.up.pt,” [Online]. Available: https://paginas.fe.up.pt/~pfs/recursos/plcs/omron/cs1/eth_manual/sec5.pdf. [Acedido em 25 08 2020].

[21] P. Servello, “https://www.npmjs.com,” [Online]. Available: https://www.npmjs.com/package/omron-fins. [Acedido em 28 08 2020].

[22] OpenCV team , “OpenCV,” [Online]. Available: https://opencv.org/. [Acedido em 10 09 2020].

[23] OpenCV, “OpenCV: Canny Edge Detection,” [Online]. Available: https://docs.opencv.org/master/da/d22/tutorial_py_canny.html. [Acedido em 8 10 2020].

[24] GeeksforGeeks, “Line detection in python with OpenCV,” [Online]. Available: https://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/. [Acedido em 8 10 2020].

[25] scikit-learn developers, “sklearn.cluster.KMeans,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. [Acedido em 9 10 2020].

[26] Steve-Mcl, “ Steve-Mcl /node-omron-fins: An implementation of the Omron FINS library for node js,” [Online]. Available: https://github.com/Steve-Mcl/node-omron-fins#readme. [Acedido em 17 09 2020].

[27] Socket.io. [Online]. Available: https://github.com/socketio/socket.io. [Acedido em 24 09 2020].

[28] Tiny Technologies Inc., “The Most Advanced WYSIWYG HTML Editor | Trusted RIch Text Editor | TinyMCE,” Tiny Technologies Inc., [Online]. Available: https://www.tiny.cloud/. [Acedido em 2 10 2020].

[29] Wikipedia, “Document Object Model - Wikipedia,” Wikimedia Foundation, Inc, 12 06 2020. [Online]. Available: https://en.wikipedia.org/wiki/Document_Object_Model. [Acedido em 11 07 2020].