Faculdade de Ciências Exatas e Engenharia

Licenciatura em Engenharia Informática

Relatório de Estágio

Discentes:

2081620 Toni Garcês

Orientador:

Prof. Lucas Pereira

junho,

2023

Conteúdo

Lista de Figuras

Lista de Códigos

1 Introdução

No ponto de vista de aperfeiçoar competências de programação, obter mais experiência no mundo do trabalho e de estabelecer uma conexão entre os vários conteúdos lecionados ao longo do curso de Engenharia Informática da Universidade da Madeira, o estágio profissional foi uma oportunidade única para este fim.

Uma vez que a **Agência Regional para o Desenvolvimento da Investigação, Tecnologia e Inovação (ARDITI)** é uma empresa com experiência e renome ao nível da engenharia presente na Ilha da Madeira e devido a estar inserido na mesma como investigador, esta foi a escolha óbvia para o estágio curricular.

O principal objetivo deste relatório é indicar as tarefas realizadas no campo do estágio realizado na ARDITI, particularmente no projeto **NexIK**, com a orientação do Professor Doutor Lucas Pereira. A responsabilidade principal definiu-se como o desenvolvimento e atualização contínua de uma interface para a visualização e monitoramento do uso de energia em cozinhas.

De acordo com Quintal et. al, a área de monitorização de eletricidade apresenta uma grande lacuna entre o monitoramento personalizado académico e opções comerciais proprietárias [9]. Posto isto, o projeto NexIK (anteriormente designado por EnnerSpectrum) pretende criar uma opção de monitoramento mais dinâmica e acessível ao consumidor.

1.1 Objetivos

- Atualizar as páginas de visualização do consumo de energia nas cozinhas:

  A solução existente do projeto contava com algumas interfaces já implementadas que necessitaram apenas de algumas alterações para acomodar os objetivos do projeto e funcionarem corretamente.

- Integrar o sistema de tracking Matomo:

  O software Matomo proporciona uma maneira de efetuar o tracking personalizado dos comportamentos do utilizador, influenciando através destes dados futuras decisões no projeto.

- Desenvolver uma página com dicas para o consumo de energia:

Com suporte em documentos baseados em dados reais, foi desenvolvida uma nova interface dedicada a fornecer dicas para reduzir o consumo de energia.

1.2 Organização do Documento

O presente relatório está estruturado em quatro capítulos, a *Introdução* descreve um ponto de vista geral do estágio e as metas esperadas no decorrer do mesmo.

Em seguida, a *Contextualização* irá elucidar o leitor sobre a visão, o âmbito e como funciona o projeto NexIK, a solução existente para o problema apresentado e, também, uma revisão de tecnologias que serão utilizadas para melhorar as interfaces existentes e cumprir as tarefas requeridas.

O terceiro capítulo *Solução Desenvolvida* concentra-se na solução, começando por expor ao leitor quais as alterações efetuadas nas interfaces de consumo, seguidamente apresentando a nova interface de dicas, terminando com o trabalho de configuração do Docker e, consequentemente, do Matomo.

Por fim, a *Conclusão* trata-se de uma reflexão sobre todo o trabalho desenvolvido, que opções diferentes poderiam ser tomadas em relação ao desenvolvimento e perspetivas para o trabalho futuro.

2 Contextualização

Este projeto constrói sobre a investigação em curso de dois laboratórios da **Fundação para a Ciência e Tecnologia (FCT)**, especificamente o **LARSyS** e o **INESC-ID**, ambos dedicados a sistemas de energia sustentável e investigação da gestão de energia [6].

Estudos indicam que, na União Europeia, 30% da energia consumida em cozinhas industriais é utilizada por estabelecimentos puramente comerciais, como restaurantes [5]. Apesar deste impacto, o papel das cozinhas industriais (IKs) não é muito explorado na investigação por sistemas de energia sustentável.

2.1 Projeto NexIK

O projeto NexIK irá debater este tópico, trazendo uma nova metodologia gerir os dados das interações entre a Água, Energia e Alimentos nas cozinhas.

Em primeiro lugar, o projeto NexIK procurou entender as necessidades dos utilizadores interessados no projeto, baseando-se numa abordagem centrada no utilizador assegurando que o produto final estará de acordo com as expetativas dos envolvidos. Em seguida, foram distribuídos vários sensores de eletricidade em cozinhas especificas com o intuito de coletar dados reais por um longo período de tempo [9].

As informações recolhidas serão úteis para desenvolver e melhorar a metodologia concebida para a avaliação do impacto das cozinhas industriais no consumo de água e eletricidade. Adicionalmente, os resultados deste projeto serão também empregues para favorecer a participação de projetos académicos, industriais, nacionais e internacionais na validação deste tipo de modelos em outros países com condições diferentes.

O projeto está dividido em vários objetivos, e tais objetivos estão englobados em cinco tarefas principais [6]:

1. **Pesquisa Centrada No Utilizador:** esta tarefa tem como objetivo tomar conhecimento sobre as perceções dos principais stakeholders em relação às repercussões do uso de energia em cozinhas industriais e a oportunidades de melhorar a eficiência energética.

   Logo, os objetivos introduzidos anteriormente, estão inseridos nesta tarefa principal. A utilização do Matomo irá proporcionar uma maneira de avaliar o comportamento

dos utilizadores perante o sistema, da mesma forma, a nova interface com dicas para redução do consumo de energia poderá surtir efeitos nos utilizadores provocando com que sejam mais conscientes quanto ao uso de energia.

2. **Monitoramento de Recursos:** a tarefa dedica-se a todas as atividades relacionadas com a implementação e gestão dos dados recolhidos das IKs selecionadas com base na tarefa 1. Os dados monitorados serão exclusivamente o consumo de eletricidade e água.

   O primeiro objetivo discutido na introdução, atualizar as páginas do consumo de energia nas cozinhas, encontra-se nesta tarefa. A atualização de tais páginas contribui para facilitar a visualização de informação relevante para o utilizador, em um caso particular, foi necessário efetuar modificações mais profundas para apresentar nova informação para além da que já se encontrava exposta.

3. **Modelagem do Nexus Baseada em Dados:** esta responsabilidade dirige-se ao desenvolvimento e avaliação de métodos baseados em dados para dar um modelo ao Nexus Água-Energia-Alimentos, isto é, a metodologia modelada fornecerá uma abordagem completa para avaliar o impacto das IKs no consumo de eletricidade e água. Este modelo será feito com base em informações, medições e dados coletados nas tarefas 1 e 2.

4. **Flexibilidade do Lado da Demanda e Eficiência Energética:** engloba todas as atividades relativas à exploração de possibilidades para melhorar a eficiência energética das IKs, garantindo flexibilidade aos operadores do sistema. Esta tarefa explora igualmente o papel dos Sistemas de Energia Renovável e dispositivos de armazenamento em IKs.

5. **Abstração e Scale-Up:** concentra-se em impulsionar os resultados das tarefas anteriores para elaborar um modelo abstrato de cozinha que pode ser usado para generalizar os resultados do projeto NexIK em diferentes tipos de IKs. Do mesmo modo, também visa encontrar oportunidades de financiamento adicionais.

O projeto conta com a colaboração de três parceiros, o **Laboratório Associado de Robótica e Sistemas de Engenharia (LARSyS)** fundado em 2001, o **Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento**

em Lisboa (INESC-ID), é uma instituição de investigação sem fins lucrativos, com financiamento da **Fundação para a Ciência e Tecnologia (FCT)** [6].

2.2 Solução Existente

O sistema existente, denominado *EnnerSpectrum*, é uma solução baseada em software para realizar a monitorização de energia num determinado estabelecimento, este fornece serviços para a visualização e administração dos dados pelo consumidor final. Para o funcionamento e integração suave deste sistema é requerida a instalação de um dispositivo físico na rede de energia local para efetuar a recolha e envio dos dados, denominado gateway (GT).

2.2.1 EnnerSpectrum — Back-End

Com intuito a garantir a maior flexibilidade e facilidade de integração com outros projetos o Back-End está organizado em cinco conceitos principais [9]:

1. **Source Schemas:** definem um tipo especifico de dados que o sistema tem a capacidade de registar.

2. **Source:** é a implementação concreta do schema correspondente. Cada source é responsável por converter as informações que recolhe no tipo apropriado de dados, efetuando a sua gravação em memória.

3. **Producer:** uma entidade que produz dados para um ou mais sources específicos. Em várias instalações, um producer poderá conter múltiplos dispositivos que têm acesso ao sistema gerando diferentes tipos de informação.

4. **Device:** cada dispositivo é diferenciado do seu producer, este tem o seu próprio token e grupo para ser possível visualizar os seus dados individualmente.

5. **Device Groups:** outros grupos poderão ser criados para visualizar os dados de forma agregada ou ter uma visão global. Os grupos são reservados a dispositivos que partilham o mesmo tipo de instalação, isto é, apenas podem ser combinados dois dispositivos que estejam a efetuar medições iguais como o consumo ou a produção.

O gateway é o componente principal, este é composto por um microcomputador Raspberry Pi 3B+, um Juice4Halt para proteger o equipamento de possíveis perdas de energia. O uso de um Raspberry Pi, permitiu tirar vantagem do seu desempenho tendo em conta o tamanho reduzido, para cada gateway foi concedido um cartão de memória de 32 GB funcionando como uma base de dados local [9].

Figura 1: Gateway (GT)

O gateway guarda os dados referentes à produção e consumo de energia localmente na sua base de dados. Cada entrada na base de dados é composta pela: data, potência ativa, reativa e aparente, frequência, fator de potência, corrente e tensão. Os dados são recolhidos a cada segundo, ao passar um minuto a base de dados é atualizada, caso existam erros por mais de um minuto o sistema alerta sobre esse problema. Identicamente, a cada 60 segundos o gateway calcula a média de cada campo de dados e envia as mesmas para o Back-End, como precaução diariamente é efetuado um backup com os dados originais em formato CSV no Google Drive, após o backup os dados são removidos da memória local [9].

Em situações particulares, o local selecionado para a recolha de dados já possuía equipamento de monitorização. Logo, o gateway foi usado simplesmente para interagir com tal equipamento recolhendo os dados relevantes sem interferir com o seu funcionamento normal [9].

2.2.2 EnnerSpectrum — Front-End

O Front-End está dividido em duas componentes, uma página visualização dos dados para o utilizador final e uma ferramenta de administração.

No caso da primeira, a informação da energia está representada utilizando métricas como kWh ou Euros, ou gramas de CO_2. Nesta secção o utilizador poderá fazer pesquisas de gráficos que representem a informação do consumo e produção de determinada data, sendo possível organizar os dados por hora, dia, semana ou mês expondo um gráfico informativo para cada tipo de seleção. Da mesma forma, o utilizador também dispõe da opção de comparar os dados relativos a duas datas diferentes [9].

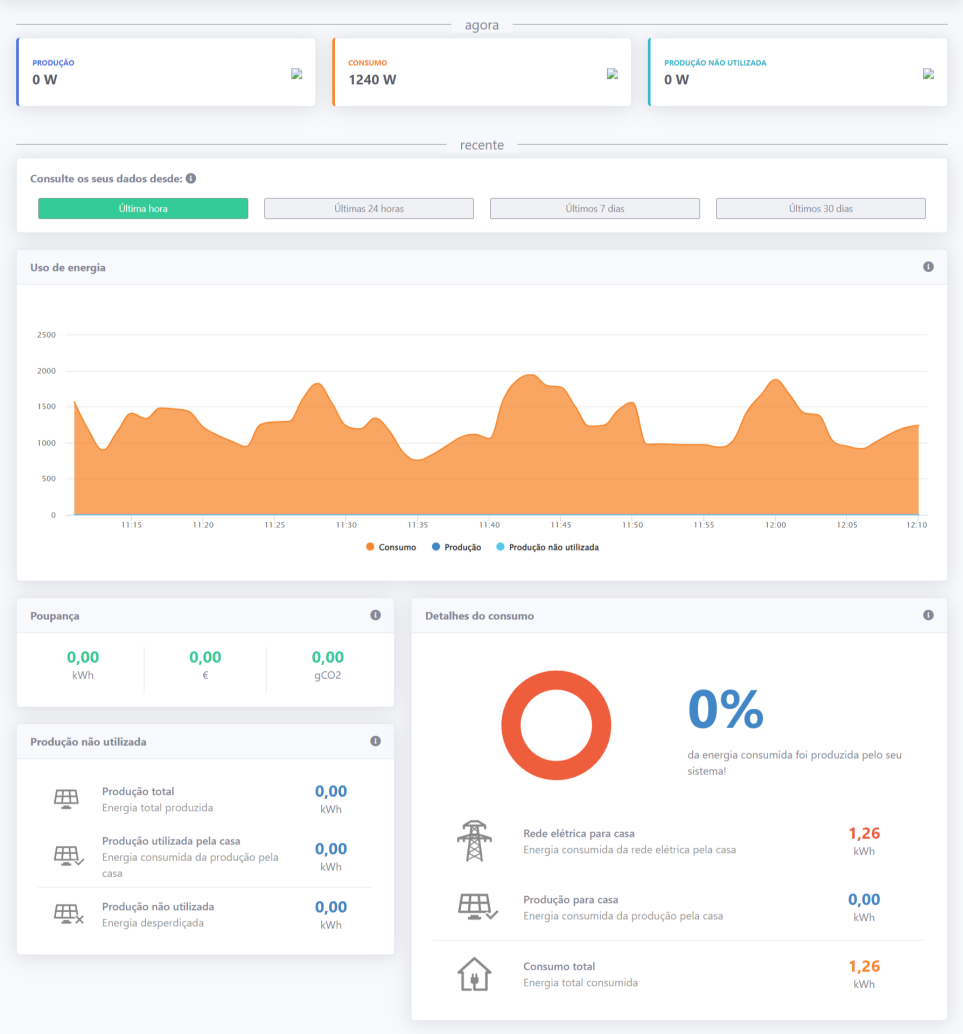

Figura 2: Interface do Consumidor - Página Home (Versão Inicial)

A segunda componente é a zona em que a equipa de investigação faz a gestão e monitorização do equipamento instalado nos diferentes locais. Este serviço permite criar,

remover, editar ou visualizar os dados de producers. Igualmente, esta interface permite a criação ou remoção de dispositivos associados a um producer e a administração total dos grupos. Além disso, é possível gerir os sources e definir quais os dispositivos que lhes fornecem dados. Outras funcionalidades relevantes também estão implementadas como a gestão de backups, a exportação de dados em formato CSV fornecendo uma visão geral do equipamento instalado [9].

Figura 3: Interface do Administrador - Página Inicial

2.3 Revisão de Tecnologias

A tecnologia é definida como qualquer modificação do mundo natural para atender às necessidades ou desejos humanos. Tecnologias, portanto, são produtos resultantes da aplicação de processos de design de engenharia. [8]. As tecnologias listadas abaixo serviram para atender as necessidades do projeto, possibilitando o desenvolvimento de uma solução que vai de acordo com o propósito do projeto NexIK.

2.3.1 Tecnologias de Desenvolvimento

- Vue JS, Node JS, MongoDB:

  Vue é uma framework de JavaScript progressiva para a construção de interfaces de utilizador. Este permite que a criação de aplicações web interativas, através de componentes reativos. O Vue oferece uma abordagem intuitiva e flexível para a construção de interfaces, facilitando a manutenção e atualização da aplicação. Além disso, este oferece alto desempenho devido ao seu sistema de renderização eficiente e implementação do DOM virtual [10].

  Node é um ambiente de execução que permite que a execução de código JavaScript fora de um navegador web. É construído sobre o motor de JavaScript V8 do Chrome e fornece um modelo de I/O baseado em eventos. Este permite a criação de aplicações web escaláveis e de alto desempenho, sendo adequado para a construção de aplicativos e APIs do lado do servidor. A vantagem principal está na capacidade de lidar com um grande número de conexões concorrentes de forma eficiente [7].

  MongoDB é uma base de dados NoSQL que oferece uma abordagem flexível e escalável para armazenar e recuperar dados. Utiliza um modelo orientado a documentos, armazenando dados em um formato JSON chamado BSON. Este oferece alta disponibilidade e escalabilidade, permitindo a distribuição de dados em vários servidores facilmente [4].

2.3.2 Tecnologias de Deployment

- Docker:

  Docker é uma plataforma de "contentores virtuais"que permite embrulhar e distribuir aplicações em ambientes isolados. Estes "contentores"fornecem uma maneira

consistente e confiável de executar aplicativos em diferentes sistemas operacionais e ambientes de implantação [1].

O **Docker Compose** é uma ferramenta que permite definir e gerir aplicativos multi-contentor em um único ficheiro YAML. Ele simplifica o processo de organização dos dockers, permitindo que se definam facilmente as dependências, se configurem variáveis de ambiente, redes e volumes compartilhados [2].

2.3.3 Software de Terceiros

- Matomo:

  Matomo é um software de análise que permite rastrear, analisar e relatar dados de tráfego em um site especifico. Este proporciona o controlo total sobre os dados, pois a ferramenta pode ser implementada em servidores privados, garantindo privacidade e segurança das informações. Adicionalmente, este oferece opções de personalização e a capacidade de rastrear dados de várias plataformas, fornecendo *insights* valiosos sobre o comportamento do utilizador [3].

3 Solução Desenvolvida

A solução elaborada englobou diversos aspetos do projeto. Inicialmente passando pela atualização das interfaces existentes com intuito de mostrarem apenas informações relevantes ao utilizador e melhorar o seu aspeto visual. Posteriormente, desenvolveu-se duas componentes totalmente novas. A primeira consta num botão de ajuda para guiar o utilizador na utilização do site, caso seja necessário. No caso da segunda, trata-se de uma página com dicas para reduzir o consumo de energia e um pequeno simulador para verificar o impacto de algumas dicas.

3.1 Interfaces de Consumo

- **Página Home:**

  A página referida, na versão inicial, está composta por três secções principais. A primeira parte contém os dados atuais que indicam em cartões qual é o consumo, a produção e produção não utilizada no momento. Abaixo encontra-se a secção onde o utilizador pode selecionar um período de tempo para visualizar nos gráficos respetivos as informações acerca da energia. Por fim, a última divisão diz respeito aos detalhes do consumo, por outras palavras, qual a poupança e gasto de energia do sistema **(Visualizar figura 2)**.

  A primeira mudança a efetuar está relacionada com os cartões, neste caso, só deverão aparecer os que contem dados importantes para o utilizador. Irão aparecer apenas os cartões que tenham valores maiores que zero, adicionalmente, foi adicionado um novo cartão que representa o consumo total do estabelecimento para o período selecionado.

```
1  <!-- Production Card -->
2  <div v-if="data.hasOwnProperty('production') && data.production > 0"
   ↪  class="mb-4 col-12" :class="classObject">
```

Código 1: Código de verificação da Produção

O código acima foi utilizado para qualquer cartão e faz uma pequena verificação deduzindo se a produção não é null e, em seguida, garante que o seu valor é maior

que zero. Este código funciona devido à infraestrutura do Vue JS que permite utilizar o comando *v-if*, caso a condição se verifique o conteúdo dentro do div (ou outra tag) será renderizado, caso contrário o elemento não irá aparecer na página.

```
1    total_consumption: function () {
2      let consumptionData = this.$store.getters.data_consumption[0];
3      if (consumptionData != null){
4        let value = friendlyFloatingPoint(consumptionData.Batt_House +
         ↪  consumptionData.PV_House + consumptionData.Grid_House)
5        if(value != "NaN") {
6          return value;
7        } else return "-";
8      } else return "-";
9    }
```

Código 2: Função para obter o consumo total

Deve ser destacado que, em qualquer projeto Vue, há a possibilidade de definir um *store* que funciona como um repositório de todas as informações que serão utilizadas por qualquer componente, este também é responsável por efetuar os pedidos ao Back-End. Posto isto, para definir o cartão do consumo total, foi necessário efetuar um pedido ao store para receber os dados do consumo. Seguidamente, é feita a soma de todos os consumos e verifica-se se é realmente um número, nesse caso o valor será apresentado no cartão, por outro lado caso não seja um número ou trate-se de um objeto null o cartão irá apresentar apenas um traço.

Por fim, a última alteração a ser efetuada relaciona-se com a produção pelos painéis solares e a poupança, ambas devem só aparecer quando existir algum valor acima de zero. Dado que não existe outra forma de detetar se um estabelecimento possuí ou não painéis solares, decidiu-se fazer a verificação do valor apenas. A verificação funciona de forma análoga à função anterior, por outras palavras, é feito um pedido à store para obter os dados sobre a poupança e gasto de energia, se os valores forem superiores a zero as informações serão apresentadas.

- **Página Histórico:**

  A página de Histórico que permite ao consumidor observar os dados para uma data específica e para um período de tempo específico também foi alvo de pequenas alterações. Particularmente, a secção de gastos, poupança e detalhes do consumo foi tratada da mesma forma que na página anterior, ou seja, só irão ser apresentados dados se existirem valores significativos.

  Nesta página implementou-se os gráficos relativos ao período de tempo selecionado, por exemplo, se for selecionada a opção de "semana"irá aparecer mais um gráfico que expõe o consumo dividido pelos dias da semana. Esta funcionalidade está presente na página principal, logo assumiu-se como uma mais valia também estar presente no histórico. Outra modificação adicionada foi a pré-definição de uma data (o dia atual), desta forma quando o utilizador entra no histórico já visualiza como a interface funciona e torna-se mais intuitivo onde clicar para efetuar outra pesquisa.

```
1    <div class="col-12 col-sm-12 col-md-12 col-lg-12 col-xl-9">
2      <WeekChart v-if="range=='week'"></WeekChart>
3    </div>
4    <div class="col-12 col-sm-12 col-md-12 col-lg-12 col-xl-9">
5      <MonthChart v-if="range=='month'"></MonthChart>
6    </div>
```

Código 3: Adição de gráficos

Primeiro, dentro da componente do histórico foram importadas duas outras componentes de gráficos, o *WeekChart* e o *MonthChart*. Em seguida, estes só serão representados caso o período selecionado seja o correspondente, reutilizando o comando *v-if* mencionado anteriormente.

- **Página Comparação:**

  A interface de comparação não sofreu muitas alterações para além do que foi mencionado anteriormente, isto é, foi removida definitivamente a parte de detalhes do consumo e pré-definidas duas datas para a comparação. A título de exemplo, quando utilizador entrar na secção de comparação, as datas selecionadas serão o dia atual e o mesmo dia da semana anterior.

```
1    // Today's Date
2    $(`#datetime-picker-${num}`).data("DateTimePicker").date(new
     ↪  Date());
3
4    // 7 days ago Date
5    $(`#datetime-picker-${num}`).data("DateTimePicker").date(new
     ↪  Date(now.getFullYear(), now.getMonth(), now.getDate() - 7));
```

Código 4: Código para definir os dias

Com o auxílio de *JQuery* selecionamos os dois campos para introduzir as datas do calendário, cada campo será representado pelo num (1 ou 2), o primeiro é definido como a data atual e o segundo será definido como o mesmo dia na semana passada, para isso basta remover 7 dias à data.

- **Página Aparelhos:**

  A interface dos aparelhos é composta por duas partes, a primeira contém um pequeno agregado dos dispositivos e apresenta o consumo de energia mais recente em Watts, a segunda apresenta uma lista dos aparelhos, o seu consumo total naquele período de tempo e a percentagem do total que ocupam. Ao clicar em algum aparelho da lista será aberto um gráfico com os dados do seu consumo ao longo do tempo.

  O campo de seleção estava limitado, logo teve de ser alterado permitindo selecionar os restaurantes e carregar os dados dos dispositivos adjacentes.

```
1    producers_with_egauge: function () {
2      const producers = this.$store.getters.producers;
3      let producers_with_egauge = producers.filter((p) => {
4        //eGauge id = "5be5b8dd750bf6f205ac70c1" @T
5        let device = p.devices.find(
6          (d) => d.sourceId == "5be5b8dd750bf6f205ac70c1"
7        );
8        if (device) return p;
9      });
10     return producers_with_egauge;
11   }
```

Código 5: Função filtro de producers

Ao carregar esta página, a store do projeto irá fazer o pedido dos producers correspondentes, neste caso são apenas os restaurantes. Após verificar na interface de administrador que todos os restaurantes possuem dispositivos com o source *eGauge* decidi fazer o filtro com base nesse aspeto. A função busca todos os producers e, em seguida, filtra-os deixando somente aqueles que têm pelo menos um dispositivo que envie dados para o source eGauge.

Efetua-se a mesma verificação para os dispositivos, ou seja, só é possível selecionar dispositivos, de um dado restaurante, que enviem dados para o source eGauge, pois este source representa a eletricidade.

Outro objetivo para esta interface foi visualizar no gráfico, ambas as informações de consumo do dispositivo como o total de todos os aparelhos, facilitando a perceção do impacto de um dispositivo.

```
1    if (total_group_id) {
2        this.dispatch('request_single_data', { range, producerId,
         ↪  groupId: total_group_id, datetime: datetimes, multi_request:
         ↪  true })
3        .then(response => {
4              commit('data_loading', true)
5              commit('store_data_total_group', response.data)
6        })
7    }
```

Código 6: Adição de dados do grupo total à Store

No store do projeto foi criado um novo campo que diz respeito aos dados do grupo total, constituído por todos os dispositivos. Para obter as informações dos vários aparelhos e do total, é efetuado um pedido à store chamando a função *request_multi_data*. Nesta função adicionou-se o código acima fazendo com que seja efetuada outra busca de dados em paralelo para o total que depois será guardado através da função *store_data_total_group*.

Os dados para cada aparelho são guardados em um objeto, esse objeto está composto por quatro partes: a informação acerca do consumo atual, do consumo ao longo do período selecionado para preencher o gráfico, o grupo e o respetivo dispositivo.

Seguidamente, com os dados do total e de cada dispositivo acessíveis através da store, a componente do gráfico foi ligeiramente modificada para acomodar estas mudanças. Esta alteração é exclusiva da página dos aparelhos e não interfere com os restantes gráficos de outras páginas do site.

```
1        //Consumption data for the selected device
2        if (this.consumption_data && this.consumption_data.length)
3          this.chart.series[1].setData(this.consumption_data.map(r =>
           ↪  [moment.tz(r.datetime,timezone).valueOf(), r.House]),
           ↪  false)
4        else this.chart.series[1].setData([], false) // Case if there's
         ↪  no data
5
6        //In the backgroup show the data relative to the whole
         ↪  restaurant
7        if (this.total_data && this.total_data.length)
         ↪  this.chart.series[0].setData(this.total_data.map(r =>
         ↪  [moment.tz(r.datetime,timezone).valueOf(), r.House ]),
         ↪  false)
```

Código 7: Alteração do gráfico

Verifica-se primeiro se existem dados para o dispositivo selecionado, nessa situação mapeamos os mesmos no gráfico na série 1 por ordem do timestamp. O mesmo acontece para os dados do total, porém agora a série escolhida trata-se da 0, desta forma garantimos que o total fica no background do gráfico.

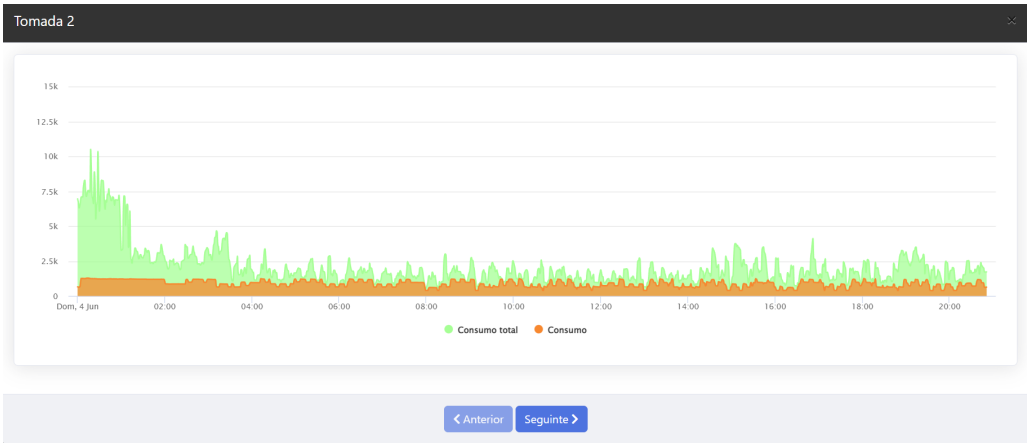

Figura 4: Gráfico Página Aparelhos

Por último, a secção do consumo atual dos aparelhos por vezes não funcionava e apresentava o valor como 0, após alguns testes foi possível constatar que deve-se a um atraso no envio dos dados para o Back-End de mais ou menos 3 minutos. Com vista a contornar este problema, ao fazer o pedido à API do consumo atual, em vez de utilizar um o minuto anterior é realizado um pedido de toda a hora anterior e, consequentemente, é feito um *shift* na lista até ao último minuto com um valor significativo.

```
1       const last_data = await
        ↪  axios.get(`/api/device/${device.deviceId}/samples/hour`,
        ↪  {params: { producerId } }).then(response => response.data)
2       while (last_data.length > 1) last_data.shift()
```

Código 8: Alteração do pedido do consumo atual

- **Componente de Ajuda:**

  Esta componente pode ser entendida como mensagens de ajuda que irão explicar as partes que compõem uma página específica. Para facilitar a sua implementação, foi decidido utilizar o plug-in vue denominado *Vue-Tour*, esta extensão permite criar um tutorial/guia listando os passos que devem ser mostrados em cada página num array.

  Primeiramente, foi criada uma nova componente com um ícone de um ponto de exclamação (?), se o utilizador clicar neste, o guia irá começar seguindo os passos para a página em que se encontra.

```
1   {
2       target: "#v-tour-step-2",
3       header: {
4           title: "Tutorial - Aparelhos",
5       },
6       content:
7           "Esta secção mostra o consumo atual de cada aparelho em
            ↪  Watts (W).",
8       params: {
9           enableScrolling: false,
10      },
11  }
```

Código 9: Estrutura etapa do tutorial

O código acima representa um passo no guia da página dos aparelhos. O target indica o id da componente que a mensagem será apresentada ao lado, o header simplesmente define o título da dica, em seguida o content é a mensagem concreta e, por fim, os parâmetros são outras opções que poderão ser adicionadas à tour, neste caso o scrolling foi desativado e para passar ao próximo passo é necessário clicar no botão seguinte.

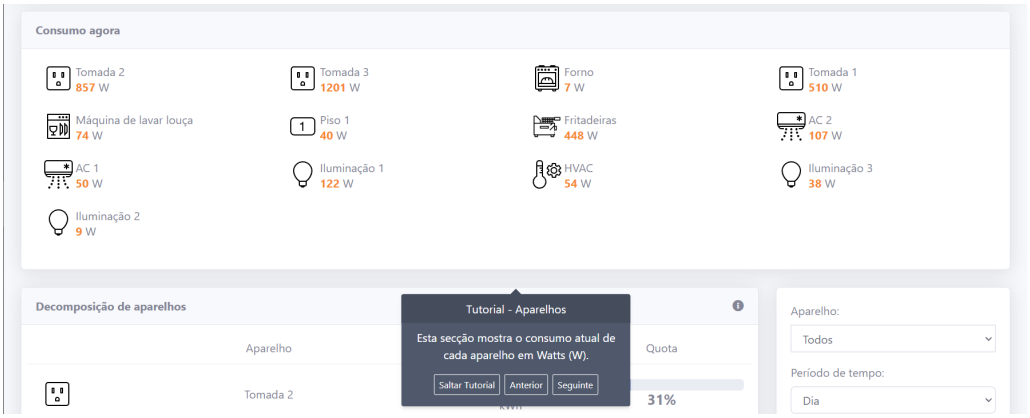

Figura 5: Tutorial Página de Aparelhos - Passo 2

Esta imagem reflete o código explicado anteriormente, a secção do Consumo Agora é o target para o passo que está ligado em baixo. Saliente-se que, se o scroll estivesse ativado o utilizador poderia entrar em confusão, pois a página não iria para baixo, mas o próximo passo do tutorial iria ser apresentado, logo deduziu-se ser uma melhor escolha deixar essa opção desativada.

3.2 Página de Dicas

A página de dicas para o consumo de energia foi uma ideia que surgiu ao longo do estágio. Esta é constituída por duas grandes partes, a primeira concerne apenas a uma lista de dicas com a opção de pesquisa no topo, a outra parte tem um cariz mais complexo pois é uma calculadora para testar o impacto que algumas dicas realmente possuirão no uso de energia para um estabelecimento.

Toda a informação presente nesta secção foi recolhida e sintetizada por uma colega, que está inserida no projeto NexIK, através de documentos relacionados a investigações nesta área. Adicionalmente, os cálculos efetuados na calculadora tem como base a média de três restaurantes que foram acompanhados pelo projeto, desta forma garantimos que os cálculos não possuem muita discrepância do real.

Inicialmente, as dicas foram todas inseridas em uma lista de objetos, cada dica era composta por um id, uma imagem, um resumo e uma descrição. Seguidamente, foi criada a estrutura em html de um div com os respetivos elementos, consequentemente, utilizando o comando *v-for* foi possível criar a lista de dicas muito rapidamente seguindo o mesmo padrão.

```
1   searchFunction() {
2           var filter, tipsWrapper, tips, tipTitle, i, textValue;
3           filter = this.mySearch.toUpperCase();
4           tipsWrapper = document.getElementById("tipsWrapper");
5           tips = tipsWrapper.getElementsByClassName("tipContainer");
6           for (i = 0; i < tips.length; i++) {
7               tipTitle = tips[i].getElementsByClassName("tipTitle")[0];
8               textValue = tipTitle.textContent || tipTitle.innerText;
9               if (textValue.toUpperCase().indexOf(filter) > -1) {
10                  tips[i].style.display = "";
11              } else {
12                  tips[i].style.display = "none";
13              }
14          }
15      }
```

Código 10: Função pesquisa de dicas

A função de pesquisa primeiro retira o texto que foi escrito na caixa de pesquisa através do filter, seguidamente, cria uma lista com as dicas buscando todas que se encontrem dentro do wrapper. Depois percorre a lista iterativamente comparando o valor do filtro com o texto que está no título de cada dica, caso não sejam minimamente relacionados o estilo dessa dica é alterado para que não seja apresentada na página. Um aspeto a melhorar seria efetuar a comparação não só com o título, mas também com a descrição de cada dica, proporcionando uma pesquisa mais abrangente ao utilizador.

Mudando agora o foco para a calculadora de dicas, tal como referido anteriormente, a base teórica deduziu-se de três restaurantes, cada um tinha os seus eletrodomésticos distribuídos por 5 categorias: **Cozinhar** são aparelhos como o forno e fritadeira, **Refrigeração** como os frigoríficos ou congeladores, **Limpeza** como as máquinas de lavar louça, **HVAC** com o ar condicionado e **Outros** que representam as lâmpadas, exaustores, entre outros que não se encaixam concretamente nas outras categorias. Utilizando estes dados foi possível concluir qual a percentagem de energia que cada categoria representa do total, do mesmo modo, também foi calculada a média de aparelhos para cada categoria.

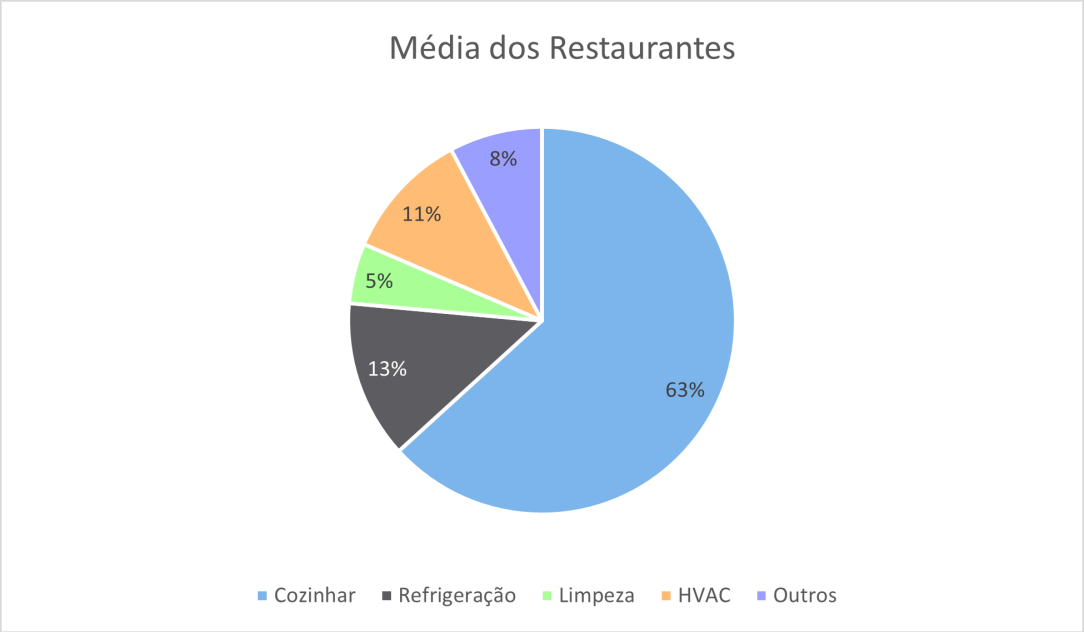

Figura 6: Média dos Restaurantes

Posto isto, a calculadora irá trabalhar sobre um restaurante fictício que contem as categorias e distribuição de energia do gráfico acima. Adicionalmente, para a categoria

de Cozinhar a média de aparelhos foi 5, para a Refrigeração foi novamente 5, para a Limpeza deu-se 2 e, por fim, para HVAC apenas 1.

A calculadora funciona da seguinte forma, o utilizador escolhe que tipo de dados deseja introduzir entre euros (€) ou energia (kWh), também pode especificar o preço por quilowatt (€/kWh). Seguidamente, no lado direito existe uma lista de dicas com caixas de seleção, basta selecionar uma das dicas que a calculadora efetua todo o processo e devolve ao utilizador quanto seria a poupança quer em euros, como em quilowatt. Concluindo, a calculadora também expõe um PieChart que apresenta o consumo de energia divido pelas categorias, quando é aplicada uma dica este gráfico atualiza de acordo permitindo que o utilizador visualize melhor o efeito que cada dica possui.

```
1   calculateSavingsPerCategory() {
2       const result = this.selectedTips.map(tip => {
3           let saving = 0
4           if (tip.value) {
5               this.resultEnergyPerCategory.forEach(category => {
6                   if (tip.category == category.name) {
7                       saving = ( category.amount /
                        ↪  tip.avgAppliancesInCategory ) * (tip.savings /
                        ↪  100)
8                       }
9                   })
10              return {name: tip.category, amount: saving}
11          } else {
12              return {name: tip.category, amount: 0}
13          }
14      })
15      this.resultSavingsPerCategory = result
16  }
```

Código 11: Função de calcular poupança

A função acima faz o mapeamento das dicas em outro array que será construído pelo nome da categoria e a poupança adjacente. Desta forma, as dicas são todas percorridas, as

que forem selecionadas são verificadas e ultrapassam a linha 4. Seguidamente, percorre-se o array que contem a energia de cada categoria em busca da categoria que corresponde à dica para calcular a poupança. O cálculo da poupança é o seguinte, divide-se o total da categoria pelo número médio de aparelhos obtendo um valor representativo de um aparelho, depois este valor é multiplicado pela percentagem de poupança associada à dica.

Segue um exemplo, digamos que a categoria HVAC (Ar Condicionado) representa apenas 100kWh do total, com em média 1 aparelho e a dica deduz que poderá ser poupado até 50% do consumo. O cálculo efetuado será o seguinte:

$$\frac{100kWh}{1} \times 50\% = 50kWh$$

Logo a poupança para a categoria será 50 kWh, as outras dicas sofrem o mesmo processo.

Por fim, o gráfico recebe os dados depois de terem sido tratados pela calculadora e apresenta-os ao utilizador.

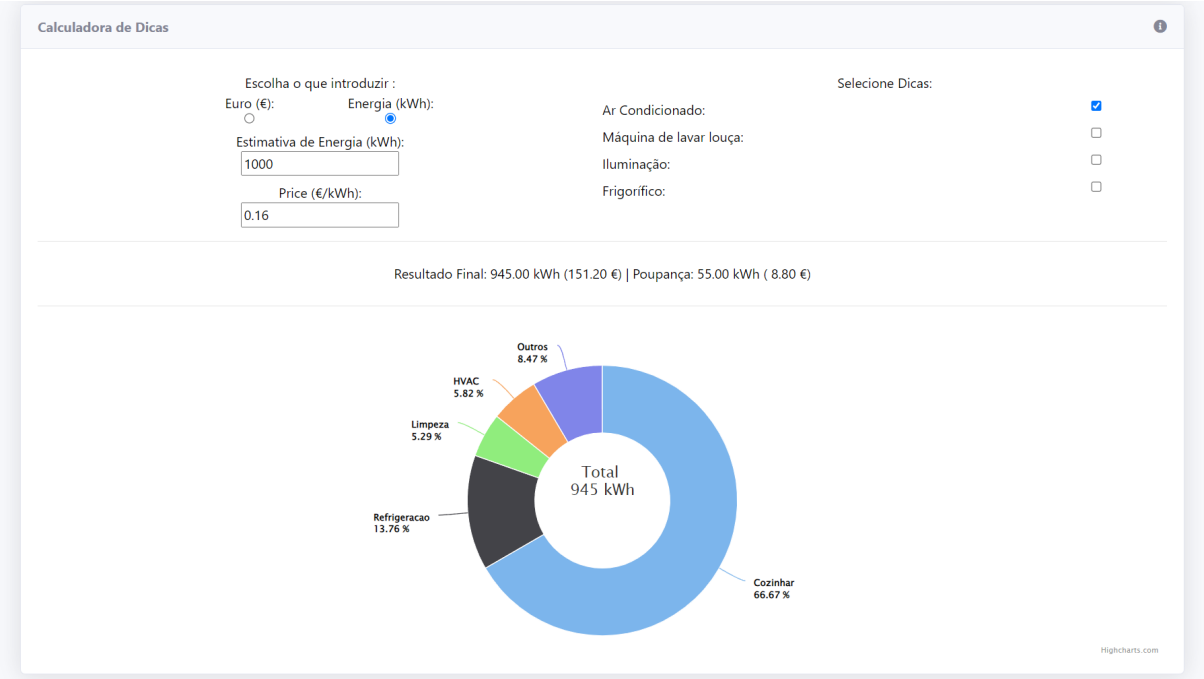

Figura 7: Calculadora de Dicas

Destaque-se que foi criado um disclaimer da calculadora para relembrar o utilizador de que os cálculos são hipotéticos e podem desviar da sua situação atual, o disclaimer situa-se no canto superior direito (i).

3.3 Instalação e Configuração Matomo

Para concluir os objetivos de estágio, era necessário efetuar o tracking do tempo que os utilizadores utilizam em cada interface. Assim sendo, decidiu-se utilizar o software Matomo que permite criar métodos personalizados de rastreio e, adicionalmente, fornece uma interface gráfica para gerir e visualizar os dados recolhidos.

Nesta situação foi utilizado o docker para criar um contentor de aplicação exclusivo para o Matomo, desta forma garantimos que o mesmo não irá interferir com outros serviços a correr no servidor. Primeiro, foi definido um ficheiro docker-compose com os serviços necessários, neste caso, o próprio Matomo, uma base de dados SQL e o serviço de Nginx para expor o site à Internet.

```
1   matomo:
2   image: matomo:latest
3   restart: unless-stopped
4   links:
5     - mysql
6   environment:
7     - MATOMO_DATABASE_HOST=mysql
8     - MATOMO_DATABASE_NAME=matomo
9     - MATOMO_DATABASE_USER=matomo
10    - MATOMO_DATABASE_PASSWORD=matomo
11    - MATOMO_DATABASE_PORT=3306
12  # [...]
```

Código 12: Ficheiro Docker - Serviço Matomo

O texto acima escrito em YAML mostra como estão definidos os campos principais do serviço, o primeiro indica qual a imagem base que será utilizada no docker, em seguida, o campo restart faz com que este serviço fique sempre ativo a menos que seja parado através de um comando, por fim, o link e environment indicam a conexão com a base de dados e as credenciais que deverão ser utilizadas durante a instalação do Matomo.

Após a instalação do Matomo, este fornece um código que deverá ser colocado no topo de todas as páginas da nossa aplicação. Um projeto Vue funciona como uma única página

apenas atualizando o seu conteúdo, logo basta apenas colocar o código de inicialização e conexão ao matomo no ficheiro App.vue, considerado como a raiz de todo o projeto.

Por fim, só faltou implementar o código concreto que faz o tracking do tempo utilizado em cada página, porém saliente-se que o Matomo não possui a capacidade de deduzir que uma página sofreu alterações no caso de projetos como Vue. Para contornar este problema, o tracking foi feito na componente do router, isto é, quando o utilizador muda de página o matomo recomeça o tracking com o nome da nova página.

```
1    let pageTitle = to.meta.title || to.name || 'Untitled Page';
2    let userID = store.getters.username;
3    _paq.push(['setDocumentTitle', pageTitle]);
4    _paq.push(['setCustomUrl', to.fullPath]);
5    _paq.push(['trackPageView']);
6    _paq.push(['enableHeartBeatTimer', 30]);
7    _paq.push(['setUserId', userID]);
```

Código 13: Tracking de Páginas

Para começar, deduz-se qual o nome da página em que estamos e qual o utilizador que está a visualizar a página, depois através do comando *_paq.push(Command, optionalData)*, definimos o título da página, o utilizador e o seu url e começa-se a contar o tempo. Destaque-se que o rastreio só funciona enquanto o utilizado estiver com a página em primeiro plano do seu browser, caso contrário o tempo para. Da mesma forma, a linha 6 serve para obter dados mais precisos, a cada 30 segundos é enviado um ping para o Matomo garantindo que o utilizador está, de facto, utilizando o site.

4 Conclusão

A título de conclusão, é possível afirmar que os objetivos de estágio foram cumpridos, apesar de algumas dificuldades que surgiram ao longo deste processo. O principal problema foi entender e adaptar o código já desenvolvido, pois cada programador possui a sua maneira de trabalhar e formas de pensar. Contudo, com dedicação e tempo acabei por aprender as tecnologias utilizadas no desenvolvimento anterior do projeto e a lógica por de trás de todo o sistema, facilitando a implementação das atualizações que foram impostas.

Considero que a escolha das tecnologias vai muito de acordo com o projeto atual, uma vez que Vue possibilita uma grande modularidade ao projeto, sendo possível adicionar, remover ou até editar alguma componente sem necessariamente afetar o restante sistema. Na mesma linha de pensamento, a utilização do Matomo também me agrada pois garante que os dados recolhidos não estão ao alcance de grandes empresas, elevando a privacidade do utilizador.

Como trabalho futuro este projeto poderia ser adaptado a algo que o próprio utilizador crie a sua interface, ou seja, através de uma lista dos componentes (gráficos, lista de aparelhos, seleção de datas, entre outros...) que existem o consumidor escolhe o que pretende ver na sua própria página.

Referências

[1] *Docker*. URL: https://www.docker.com/.

[2] *Docker Compose*. URL: https://docs.docker.com/compose/.

[3] *Matomo*. URL: https://matomo.org/.

[4] *MongoDB*. URL: https://www.mongodb.com/.

[5] S. Mudie et al. «Electricity use in the commercial kitchen». Em: *International Journal of Low-Carbon Technologies* 11.1 (set. de 2013), pp. 66–74. ISSN: 1748-1317. DOI: 10.1093/ijlct/ctt068. eprint: https://academic.oup.com/ijlct/article-pdf/11/1/66/7148325/ctt068.pdf. URL: https://doi.org/10.1093/ijlct/ctt068.

[6] *NexIK*. URL: https://nexik.tecnico.ulisboa.pt/#.

[7] *Node JS*. URL: https://nodejs.org/en.

[8] Edys Quellmalz. «Engineering and Technology: Assessing Understanding of Similarities and Differences Between Them». Em: *Encyclopedia of Science Education*. Ed. por Richard Gunstone. Dordrecht: Springer Netherlands, 2021, pp. 1–5. ISBN: 978-94-007-6165-0. DOI: 10.1007/978-94-007-6165-0_19-2. URL: https://doi.org/10.1007/978-94-007-6165-0_19-2.

[9] Filipe Quintal et al. «Energy Monitoring in the Wild: Platform Development and Lessons Learned from a Real-World Demonstrator». Em: *Energies* 14.18 (2021). ISSN: 1996-1073. DOI: 10.3390/en14185786. URL: https://www.mdpi.com/1996-1073/14/18/5786.

[10] *Vue JS*. URL: https://vuejs.org/.