



**Gonçalo José
Almeida Dias**

Relatório de estágio



**Gonçalo José
Almeida Dias**

Relatório de estágio

Relatório de Estágio apresentado à Universidade de Aveiro para obtenção do da Licenciatura em Tecnologia de Informação, realizado sob a orientação do Prof. Ivan Miguel Serrano Pires, da Escola Superior de Tecnologia e Gestão de Águeda, Universidade de Aveiro.

**agradecimentos /
acknowledgements**

Agradeço a todos os colaboradores da Wiseware Solutions pela colaboração e hospitalidade. Agradeço ao Prof. Doutor Ivan Pires pela orientação. Um agradecimento especial ao orientador Gustavo Corrente pela disponibilidade, apoio na resolução de problemas e abertura a novas ideias.

palavras-chave

Tecnologias, software, equipa, tarefas, conhecimentos.

resumo

O presente relatório comprehende todas as atividades realizadas na empresa WISEWARE Solutions durante o estágio de 466 horas de duração. Durante o estágio foram definidos certos objetivos: adquirir competências acerca do funcionamento da entidade, integração com as tecnologias utilizadas, desenvolvimento de software e elaboração do relatório final de estágio. Estes objetivos permitiram aplicar os conhecimentos adquiridos no decorrer do curso.

A primeira parte do relatório apresenta uma descrição sucinta da empresa juntamente com uma breve descrição das tarefas nela realizadas. Em seguida são descritas as atividades planeadas, para a realização do estágio. Numa terceira fase descreve-se detalhadamente cada uma das atividades desenvolvidas, as dificuldades sentidas na sua realização e as competências adquiridas com as suas realizações. Para finalizar é feita uma análise das experiências vivenciadas no estágio e as expectativas para o futuro com base nos conhecimentos obtidos.

keywords

Technology, software, team, tasks, knowledge.

abstract

The present report outlines all the activities carried out at the company WISEWARE Solutions during the 466 hours of the internship. During the internship, specific objectives were defined: to acquire new skills related to the functioning of the organization, integration with the technologies used, software development and elaboration of internship report. These objectives made it possible to apply the knowledge acquired throughout the course.

The first part of this report presents a concise description of the company including a brief description of the activities carried out there. Next, the activities planned for the internship are described. In the third part, each activity is described in detail, including the difficulties encountered during their execution and the skills acquired through their completion. In the end, an analysis is carried out of the experiences gained during the internship and of future expectations, based on the knowledge acquired.

**acknowledgement of use of
AI tools****Recognition of the use of generative Artificial Intelligence
technologies and tools, software and other support tools.**

I acknowledge the use of [insert AI system(s) and link] to [specific use of generative artificial intelligence or other tasks]. I acknowledge the use of [software, codes or platforms] to [specific use software, codes or platforms or to other tasks].

Example 1: I acknowledge the use of ChatGPT 3.5 (Open AI, <https://chat.openai.com>) to summarise the initial notes and to proofread the final draft and the use of Office365 (Microsoft, <https://www.office.com>) for text writing and productivity.

Example 2: No content generated by AI technologies has been used in this Thesis.

I acknowledge the use of GPT-4-turbo (Open AI, <https://chat.openai.com>) for text revision and assisting with streamlining information searches.

Conteúdo

Conteúdo	i
Lista de Figuras	iii
Lista de Tabelas	iv
Glossário	v
1 Introdução	1
2 Entidade de acolhimento	2
3 Plano de trabalho	3
4 Trabalho realizado	4
4.1 Análise de bibliotecas	4
4.1.1 APIs georreferenciação	4
4.1.2 APIs de visualização gráfica	7
4.2 Interface gráfica	9
4.2.1 Svelte 5	9
4.2.2 Tecnologias fundamentais	11
4.2.3 Componente Mapa	12
4.2.4 Página <i>reset password</i>	15
4.2.5 Página de dashboard	16
4.2.6 Página de status	17
4.2.7 Página de gráficos	18
4.2.8 Página de erro	19
4.2.9 Rotas sem inicio de sessão	20
4.3 Servidor da aplicação	21
4.3.1 O que é o <i>backend</i> ?	22
4.3.2 Rota para obter número de pedidos por dispositivo	23

4.3.3	Sistema de envio de emails	24
4.3.4	Dados do <i>acc</i> e de temperatura no sample	25
4.3.5	Rotas para aceder a dados sem <i>login</i>	26
4.3.6	Correção de um <i>bug</i> que duplicava dados	27
5	Analise Resultados	28
6	Conclusões	29
A	Apendices	30

Listas de Figuras

2.1	WiseWare Solutions	2
4.1	OpenLayers	5
4.2	Google maps	5
4.3	Mapbox	6
4.4	Gráfico d3js	7
4.5	Gráfico chartJs	8
4.6	Gráfico Apache Echarts	8
4.7	Diferenças <i>Svelte</i> 4 e 5	9
4.8	Exemplo <i>props svelte</i>	10
4.9	<i>\$derived & \$effect</i>	10
4.10	Exemplo de interface criada com <i>tailwind</i>	11
4.11	Interface com componentes <i>flowbite</i>	11
4.12	Camadas	12
4.13	Atualizar pontos selecionados no mapa	12
4.14	Exemplo de marcadores	13
4.15	<i>Map utils</i>	13
4.16	Desenhar polígonos	14
4.17	<i>Flow</i> de edição de polígonos	14
4.18	Páginas <i>reset-pass</i>	15
4.19	Páginas <i>reset-pass</i> alterar palavra passe	15
4.20	Redirecionar para o login	16
A.1	Plano trabalho	31

Lista de Tabelas

Glossário

R&D Research and development
SMT Surface-mount technology
UI User interface
IT Instituto de Telecomunicações

UA Universidade de Aveiro
UM Universidade do Minho
DOM Document Object Model

CAPÍTULO

1

Introdução

O presente relatório pretende descrever as atividades desenvolvidas durante o Estágio inserido no 3º ano da Licenciatura em Tecnologias da Informação, lecionado na Escola Superior de Tecnologia e Gestão de Águeda e coordenado pelo Prof. Doutor Ivan Pires.

A realização do estágio tem com objetivo colocar em prática todos os conhecimentos, teóricos e práticos adquiridos e desenvolvidos nas unidades curriculares. O estágio decorreu na Wiseware Solutions, Gafanha da Encarnação, entre 11 de Fevereiro e 23 de Maio de 2025 sobre a orientação de Gustavo Corrente.

Este estágio teve como principais objetivos a obtenção de experiência em contexto de trabalho, adquirir novos conhecimentos e desenvolver competências práticas e teóricas a nível profissional.

O presente relatório encontra-se dividido em quatro partes. Na primeira parte, é apresentada uma breve descrição da entidade de acolhimento. No segundo capítulo, é feito um comentário à cerca do plano de trabalho. A terceira parte contempla uma descrição detalhada de todo o trabalho realizado ao longo do estágio, no âmbito do projeto *aniposture*, nomeadamente ao nível da interface gráfica e do *backend*. Por fim é realizada uma análise com base nos conhecimentos adquiridos durante as 466H de estágio.

Para concluir o relatório são apresentadas considerações finais, uma reflexão sobre as competências adquiridas e um balanço geral do estágio.

CAPÍTULO 2

Entidade de acolhimento

A Wiseware Solutions é uma empresa de R&D localizada na Zona Industrial da Mota, Gafanha da Encarnação, com a missão de desenvolver soluções inovadoras e de alta qualidade para empresas.

A Wiseware é responsável por produzir soluções em áreas como robótica, micro eletrônica, inteligência artificial, comunicações sem fio e muito mais. Aplicaram os seus conhecimentos em projetos de saúde e bem-estar, telemetria, controlo de qualidade, inspeção e automação de sistemas.



Figura 2.1: Wiseware Solutions

A Wiseware realiza vários serviços como a criação de PCB, montagem de SMT, inspeção de SMT, desenho técnico, cortes 2D/3D a laser, fresagem, etc.

Apresentam um conjunto de parceiros como a *Universidade de Aveiro (UA)*, o *Instituto de Telecomunicações (IT)*, a escolha de engenharia da *Universidade do Minho (UM)*, a *inovaria*, entre outros. Como seus clientes estão nomes como a *Adidas*, *Digital Logic*, *Efcom*, grupo *iPesa*, entre outros.

CAPÍTULO 3

Plano de trabalho

Antes do estágio começar, foi definido um plano de trabalho que estabelecia os principais objetivos a atingir no decorrer do mesmo, Figura A.1. Apesar de constituir a base para a realização do estágio, é natural que, no final, possam existir algumas variações entre o trabalho realizado e o trabalho inicialmente planeado.

No plano de trabalho foram definidos vários objetivos, bem como o número de horas previstas para a sua concretização. Estes objetivos foram organizados por áreas de trabalho distintas.

1. O primeiro objetivo correspondeu à fase de *Acolhimento e introdução ao desenvolvimento*. Esta etapa teve como principal finalidade a minha introdução com as tecnologias e ferramentas utilizadas pela empresa no processo de desenvolvimento.
2. O segundo objetivo foi o *Levantamento de requisitos*. Nesta fase, a missão consistiu em adquirir um conhecimento aprofundado sobre o funcionamento do projeto *aniposture*, incluindo os seus objetivos, estrutura, tecnologias envolvidas, entre outros aspectos.
3. O terceiro objetivo foi a *Escolher a API de Mapas*. Esta etapa teve como finalidade a realização de um estudo e comparação de funcionalidades, preços e relação custo-benefício das diferentes *APIs* de georreferenciação.
4. O quarto objetivo correspondeu ao *Desenvolvimento da UI*. Esta fase visava a implementação da interface gráfica integrando a *API* de georreferenciação previamente selecionada.
5. O quinto objetivo consistiu o *Desenvolvimento de serviços*. Este objetivo procurava o desenvolvimento de serviços RESTful e a implementação de endpoints úteis para o funcionamento do backend da aplicação.
6. O sexto objetivo foi *Testes funcionais*. Esta etapa teve como principal finalidade a realização de testes funcionais de forma a verificar o bom funcionamento da aplicação.
7. O setimo e último objetivo foi o *Relatório*. Ainda durante o período de estágio, foram alocadas algumas horas para a redação e finalização do presente relatório.

CAPÍTULO 4

Trabalho realizado

Neste capítulo irei abordar todo o trabalho desenvolvido durante o período de estágio, organizado em três grandes secções. Na primeira secção abordarei o trabalho realizado durante a análise de APIs e bibliotecas, incluindo o porque da escolha final de cada uma. A segunda secção visa comentar o trabalho realizado na interface gráfica do projeto aniposture. A terceira secção comenta o trabalho desenvolvido no respetivo backend.

4.1 ANÁLISE DE BIBLIOTECAS

Durante este estágio, foram-me atribuídas tarefas de análise e pesquisa, com o objetivo de identificar e selecionar as ferramentas mais adequadas que cumprissem os requisitos necessários para o desenvolvimento da aplicação. Foram, assim, realizadas duas grandes análises: a primeira sobre as APIs de georreferenciação e a segunda nas APIs de visualização gráfica.

4.1.1 APIs georreferenciação

Esta foi uma das primeiras tarefas que foram-me atribuídas durante o estágio. Antes de começar as comparações, foram definidos alguns requisitos fundamentais, nomeadamente a necessidade de *tiles* em satélite, marcadores, bom desempenho com uma elevada quantidade de pontos e desenho de polígonos. Depois de levantados os requisitos, iniciou-se a análise comparativa entre três APIs de georreferenciação o mapbox, openlayers e o google maps.

Apesar de partilharem o mesmo objetivo, criação de mapas interativos, as três bibliotecas são consideravelmente diferentes nas suas implementações.

OpenLayers

Openlayers é o biblioteca free e Open Source que permite a criação de mapas interativos. Disponibiliza recursos como *raster tiles*, camadas vetoriais e marcadores. Por ser *Open Source*, contem vários *addons* criados pela comunidade que permite alargar as suas funcionalidades. Contudo, apresenta uma linha de aprendizado um pouco mais complexa do que outras alternativas e, por padrão, não contém nenhum *tile* em satélite.

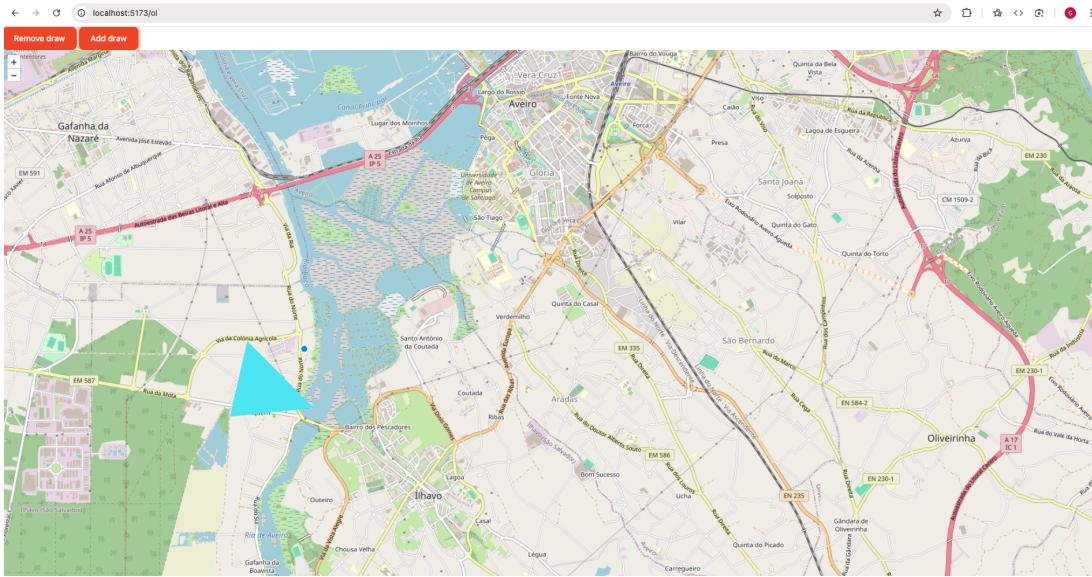


Figura 4.1: OpenLayers

Foram ainda realizados alguns testes com opções de desenho vetorial. No entanto, o que levou o *OpenLayers* não ser o escolhido foi a sua, ainda, baixa integração com *WebGL*, a documentação menos completa em comparação com outras alternativas e a ausência de suporte nativo para mapa com satélite.

Google Maps

O *google maps* é, muito provavelmente, o sistema de mapas mais utilizado e atualizado a nível mundial. Por essa razão, foi considerada a sua utilização, tendo sido analisada a forma como é implementado e as suas funcionalidades. Inicialmente, esta solução revelou-se a mais promissora, uma vez que permite adicionar camadas vetoriais, marcadores, vista em satélite e possui um bom desempenho.

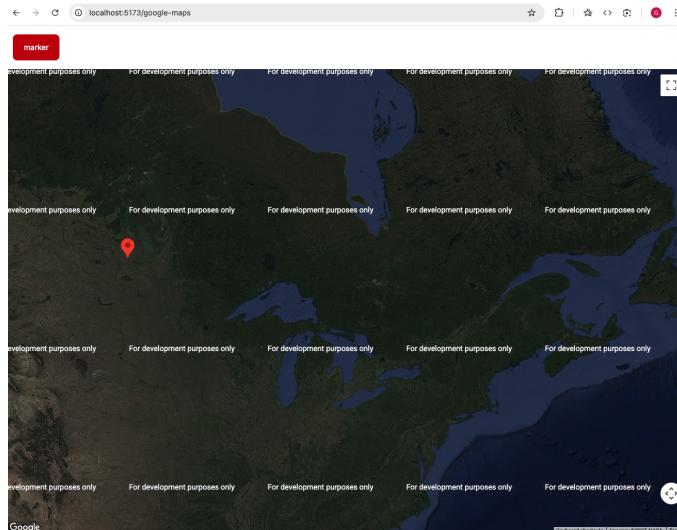


Figura 4.2: Google Maps

Ao contrário do *Openlayers*, 4.1.1, esta é uma solução paga, que apresenta, algumas vantagens como mapas satélite mais atualizados e integração com os serviços da *Google*. Contudo, o maior problema é o seu preço, consideravelmente elevado se formos comprar com outras alternativas. A *Google* permite a utilização da *Maps JavaScript API* de forma gratuita até 10.000 pedidos mensais. A partir dessa quantidade começam a ser aplicadas cobranças. Devido a essa limitação o *google maps* também não foi a solução escolhida.

Mapbox

Chegamos, por fim, à opção escolhida: o *Mapbox*. Esta biblioteca revelou-se a melhor entre os dois mundos, implementação inicial simples, utiliza *WebGL* por padrão e disponibiliza vários tipos de *tiles*, incluindo satélite.

Embora seja uma solução paga, apresenta modelos mais em conta do que a opção da *google*, a versão gratuita, contem um total de 50.000 pedidos por mês. Esta mesma quantidade de pedidos no *google maps* teria um custo de cerca de 280€ mensais. Para atingir um custo parecido no *mapbox*, seria necessário utilizar pelo menos 100.000 pedidos mensais. Para além do fator preço/económico, esta solução, compe todos os requisitos definidos para a nossa aplicação, juntamente de uma documentação bem estruturada.

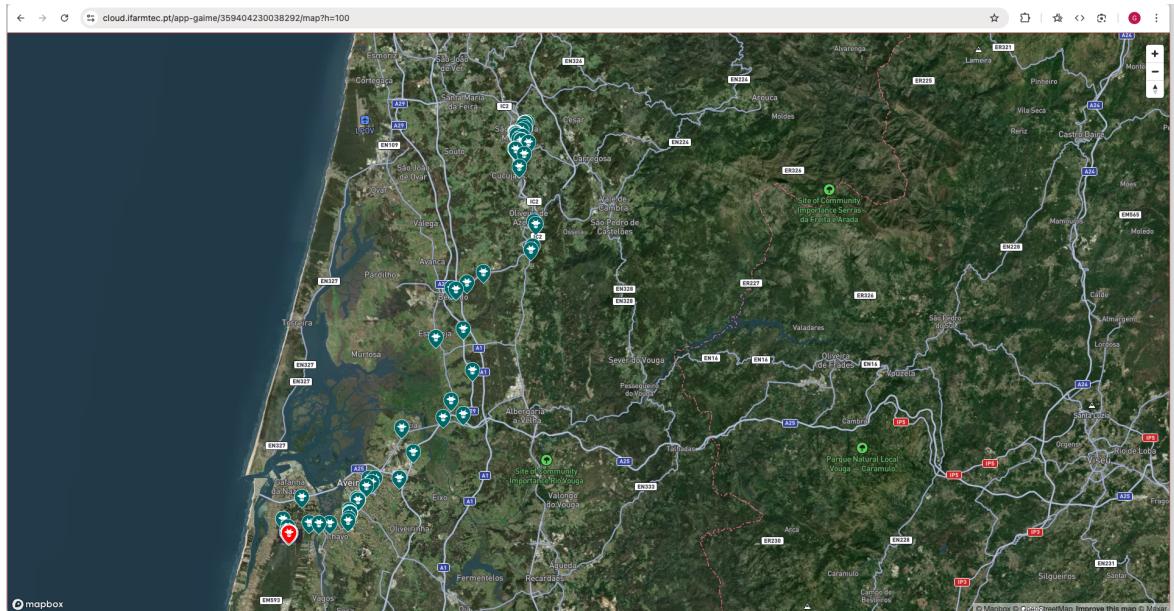


Figura 4.3: Mapbox

Tendo em conta todos os requisitos levantados, o *mapbox* cumpre-os de forma exímia. Graças à sua grande comunidade, o *mapbox*, dispõe vários plugins que permitem expandir significativamente as suas funcionalidades, o que acrescenta ainda mais valor à sua escolha como opção para o projeto.

4.1.2 APIs de visualização gráfica

Após análise e avaliação das bibliotecas de georreferenciação, realizei uma pequena pesquisa e comparação na área dos gráficos.

D3js

Esta foi a biblioteca identificada pelo orientador, como uma possível opção a implementar. O *d3js* não é uma biblioteca de gráficos tradicional, uma vez que possui o conceito de "gráficos". Quando visualizamos os dados com o *d3js*, estamos a compor uma variedade de objetos primitivos, como linhas, círculos, retângulos, entre outros.

Apesar de apresentar um desempenho elevado e permitir um elevado nível de interatividade, a sua utilização é consideravelmente mais complexa que outras alternativas.



Figura 4.4: Exemplo de gráfico criado com o *d3js*

ChartJs

O *chartJs* é uma das bibliotecas mais conhecidas e amplamente utilizadas no mundo *JavaScript* para representação de dados. À data atual, conta com mais de **66 mil** estrelas no *github* e é regularmente atualizada pela comunidade e pelos seus contribuidores.

Ao contrário do *d3js*, 4.1.2, o *chartJs* é consideravelmente mais simples de utilizar. Trata-se de uma biblioteca que incorpora o conceito de "gráfico", permite a visualização da informações em 8 estilos diferentes, sendo ainda responsivo.

No entanto, essa mesma simplicidade pode ser vista como uma limitação: os gráficos, por padrão, oferecem poucos funcionalidades além da visualização básica dos informações.

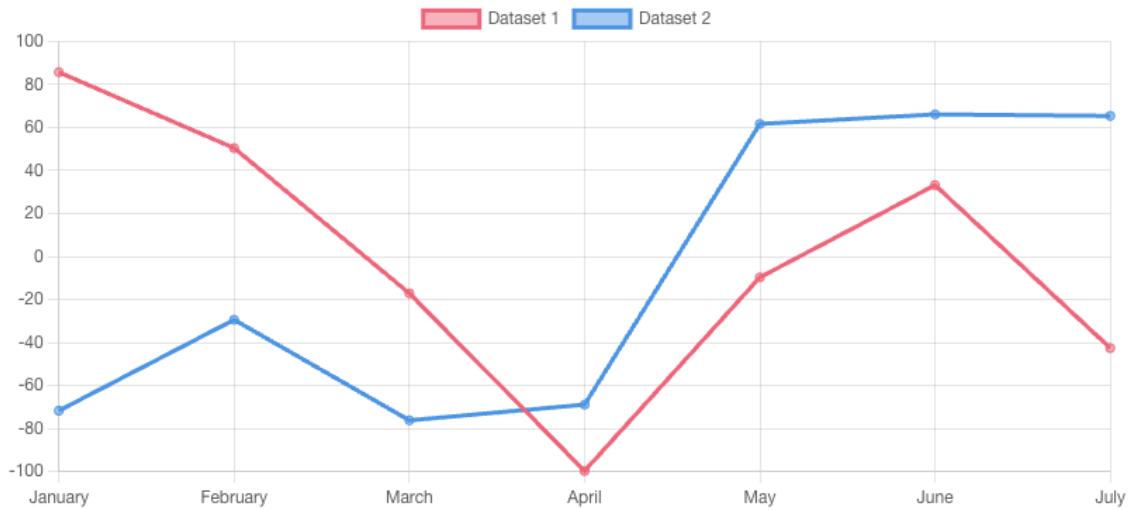


Figura 4.5: Exemplo de gráfico criado com o chartJs

Echarts

Apache Echarts é uma biblioteca gratuita, desenvolvida no âmbito da fundação *Apache*, que apresenta características similares ao chartJs. Suporta mais de 20 tipos diferentes de gráficos, é responsivo e permite a renderização de até 10 milhões de dados em tempo real.



Figura 4.6: Exemplo de gráfico criado com Echarts

O *Apache Echarts* foi a solução escolhida para integrar no projeto, não apenas por apresentar um maior número de opções de visualização de dados, mas também pelas suas funcionalidades nativas e pela elevada performance a carregar grandes volumes de dados.

4.2 INTERFACE GRÁFICA

Nesta fase, será abordado com detalhes todo o trabalho realizado no *frontend/interface* gráfica da aplicação, incluindo as dificuldades encontradas, conhecimentos adquiridos, e as tarefas realizadas durante o desenvolvimento.

4.2.1 Svelte 5

Durante a fase de desenvolvimento, esta foi das primeiras tarefas que foram-me atribuídas. No último trimestre de 2024, a equipa responsável pelo *svelte* lançou uma nova versão, que alterou conceitos fundamentais da *framework*. Fui, então, encaminhado a aprender esta nova versão, testar as novas funcionalidades, compreender as principais alterações e analisar o processo de atualização de um projeto desenvolvido na versão 4 para a versão 5 do *svelte*.

O *svelte* 5 introduziu um novo conceito denominado *runes* (runas), que consistem num mecanismo para declarar estado de reatividade. As runas são símbolos que influenciam o compilador do *svelte*, permitem ao programador definir, de forma explícita, quais objetos são reativos e quais não são, Figura 4.7b.

Nas versões anteriores, o *svelte* tornava todos os objetos reativos por padrão, isto dificultava a percepção de quais objetos podiam, efetivamente, provocar alterações na interface gráfica.

Estas mudanças representam uma alteração significativa na forma como criavamos código em *svelte*. A nova abordagem, removeu a sintaxe do *svelte* 4, Figura 4.7a, que, muitas das vezes, era confusa, já que a mesma expressão podia realizar ações diferentes. Com a introdução da nova sintaxe, cada uma das ações realizas é bem definida, o que torna o código mais previsível e fácil de manter.

```
<script>
  let count = 0;

  $: doubled = count * 2
  $: document.title = `count is ${count}`
</script>
```

(a) Exemplo *Svelte* 4

```
<script>
  let count = $state(0)

  let doubled = $derived(count * 2)

  $effect(() => console.log(`count is ${count}`))
</script>
```

(b) Exemplo *Svelte* 5

Figura 4.7: Diferenças *Svelte* 4 e 5

Uma das alterações introduzidas nesta nova versão do *svelte* foi a forma como passamos elementos para dentro dos componentes.

Na versão anterior, utilizava-se a palavra-chave *export*, o que permitia declarar propriedades em qualquer parte do componente. Essa flexibilidade, embora útil, podia tornar o código menos organizado e mais difícil de compreender. Na nova versão, todos os elementos recebidos pelo componente são definidos num único local, o que promove uma estrutura mais clara e facilita a leitura e manutenção do código. Ver Figura 4.8.

```
// Svelte 4
export let prop1;
export let prop2;

// Svelte 5
let { prop1, prop2 = $bindable() } = $props();
```

Figura 4.8: Exemplo *props svelte*

Depois de familiarizar-me com os novos conceitos introduzidos no *svelte 5*, iniciei a modificação e criação novos componentes nesta versão. Os dois conceitos mais importantes, e os que exigiram mais tempo a compreender foram o *\$derived* e o *\$effect*.

O *\$derived* é uma runa utilizada para definir uma variável derivada. Ou seja, cria-se uma nova variável cujo o valor é automaticamente atualizado com base numa expressão, sempre que ocorre alguma alteração noutra variável, Figura 4.7b.

Já o *\$effect* é uma função que é executada sempre que um determinado estado é atualizado. Esta runa pode ser utilizada, por exemplo, para chamar biblioteca externas, desenhar elementos num canvas ou realizar pedidos à rede, Figura 4.7b.

```
let hasStats = $derived.by(() => {
  const rs = sampleList.filter((sample) => sample.details?.stats);
  if (rs.length > 0) return true;
  return false;
});
```

(a) Exemplo *\$derived*

```
$effect(() => {
  if (derivedPoints && derivedPoints.length > 0 && mapLoaded) {
    resetPointsLayers();
    updateMapLayers();
    boundingZoom();
  }
});
```

(b) Exemplo *\$effect*

Figura 4.9: *\$derived* & *\$effect*

Na Figura 4.9, podemos ver um exemplo um pouco mais complexo da utilização das runas *\$derived* e *\$effect*.

No caso do *\$derived*, a variável *hasStats* é atualizada sempre que ocorrem alterações no array *sampleList*. Já o *\$effect* executa todas as funções chamadas no seu interior sempre que o mapa está carregado e a variável *derivedPoints* é alterada.

4.2.2 Tecnologias fundamentais

Nesta secção irei comentar um pouco sobre as outras tecnologias que existem no projeto *aniposture*.

Tailwind

O *tailwind* é uma *framework* css que permite construir interfaces diretamente no *HTML*, através da utilização de classes pré definidas. Esta abordagem promove rapidez e consistência no design, o que reduz necessidade de escrever *CSS* personalizado.

Esta foi a minha primeira experiência com *tailwind*. Encontrei algumas dificuldades nas primeiras semanas, sobretudo por ainda não compreender totalmente o seu conceito nem as suas classes pré definidas. No entanto, após ultrapassar esta fase de adaptação, consegui desenvolver interfaces de forma mais rápida e prática.

```
<div class="flex flex-col items-center justify-center gap-2 m-4 mb-2">
  <div class="text-sm text-gray-700 dark:text-gray-400">
    Showing
    <span class="font-semibold text-gray-900 dark:text-white">{filteredItems.length}</span>
    of
    <span class="font-semibold text-gray-900 dark:text-white">{devicesLists.size}</span>
    Devices. Updated @
    <span class="font-semibold text-gray-900 dark:text-white">
      {UTILS.timestampToDateStr(data.requestDate)}
    </span>
  </div>
</div>
```

Figura 4.10: Exemplo de interface criada com *tailwind*

Flowbite svelte

O *flowbite* é uma biblioteca de componentes baseada em *tailwind*, que tem como objetivo acelerar e simplificar a criação de interfaces. Esta fornece um vasto conjunto de componentes prontos a usar, como tabelas, botões, modais, entre outros. A utilização destes componentes contribui para uma interface mais uniforme e coerente.

Filters				
<input checked="" type="checkbox"/> Points with GPS	<input checked="" type="checkbox"/> Points without GPS	<input checked="" type="checkbox"/> BootUp		
Points				
DATE	FIX TIME	NO FIX COUNT	TYPE	DETAILS
31/05/2025 15:19:38	0	196		
Info				
Fix time in ms	Device battery	Device status		
0	5			
Network data				
CELL ID	OPERATOR	RSRP	TAC	
313100	26806	-96	48710	

Figura 4.11: Interface com componentes *flowbite*

4.2.3 Componente Mapa

Um dos trabalhos mais complexos nesta fase da interface foi a implementação do mapa. Trata-se de um elemento com características especiais, que exige funcionalidades como o carregamento de pontos, executar ações ao detetar cliques, carregar *layouts*, atualização automática de novas informações, desenho de elementos, entre outros.

Um dos pontos mais importantes durante o desenvolvimento deste componente foi garantir que os dados fossem carregados e atualizados automaticamente. Este aspeto revelou-se particularmente importante devido à arquitetura (ref arquitetura do sistema) *multi tenant* do projeto. Ao escolher um novo *tenant*, todas as informações tinham de ser limpas e substituídas por novas. Para atingir este objetivo, foram utilizadas várias ferramentas do *svelte 5*, Subseção 4.2.1.

Layers (Camadas)

As *Layers* (ou camadas) são componentes principais em qualquer sistema de mapas. O *mapbox* permite a adição de diversas *camadas* ao mapa, como elementos vetoriais, símbolos, imagens *raster*, modelos 3D, entre outros. Estas camadas permitem representar diferentes tipos de dados geográficos e visuais, sendo fundamentais para a construção de mapas interativos.

Na criação deste componente, foram adicionadas duas camadas para cada tipo de animal existente (ref arquitetura do sistema). Uma das camadas é responsável por mostrar todos os elementos, enquanto a outra exibe apenas os elementos selecionados.

```
helper.addLayer({
  id: 'sheep-layer',
  type: 'symbol',
  source: 'sheep-source',
  layout: {
    'icon-image': 'sheep',
    'icon-ignore-placement': true
  }
});
```

(a) Camada de elementos

```
helper.addLayer({
  id: 'sheep-selected',
  type: 'symbol',
  source: 'sheep-source',
  layout: {
    'icon-image': 'sheep-selected',
    'icon-ignore-placement': true
  },
  filter: ['in', 'id', '']
});
```

(b) Camadas de elementos selecionados

Figura 4.12: Camadas

Como ilustrado na Figura 4.12a, na camada correspondente a todos os elementos é carregada a informação através da fonte *sheep-source*, sendo os elementos posteriormente carregados no mapa. Já a camada *sheep-selected* apenas apresenta os elementos quando o filtro correspondente está ativado. Essa seleção ocorre através do evento *on click*, que identifica o ponto naquela coordenada. Após seleção, o *svelte* atualiza automaticamente os pontos no mapa. Figura 4.13.

```
$effect(() => {
  if (selectFeature && mapLoaded) {
    if (selectFeature.properties && selectFeature.geometry) {
      map.setFilter('points-selected', ['in', 'id', selectFeature.properties.id]);
      map.setFilter('sheep-selected', ['in', 'id', selectFeature.properties.id]);
      map.setFilter('goat-selected', ['in', 'id', selectFeature.properties.id]);
      map.setFilter('cow-selected', ['in', 'id', selectFeature.properties.id]);
    }
  }
});
```

(a) Selecionar filtros

```
map.on('click', (e) => {
  if (!disableClicking) {
    const selectedFeatures = map.queryRenderedFeatures([e.point.x, e.point.y], {
      layers: ['no-type-layer', 'goat-layer', 'cow-layer', 'sheep-layer']
    });
    if (selectedFeatures.length > 1 || selectedFeatures.length === 0) return;
    selectedFeature = selectedFeatures[0];
  }
});
```

(b) *Map onClick*

Figura 4.13: Atualizar pontos selecionados no mapa

Marcadores

Antes da utilização das camadas vetoriais, descritas na 4.2.3, foram realizados alguns testes com os marcadores nativos do mapbox.

Um marcador é uma representação visual de uma coordenada específica no mapa. Estes marcadores são elementos *html* inseridos no *DOM* da página, Figura 4.14. Devido ao baixo desempenho, esta abordagem revela-se inviável quando se pretende representar mais do que algumas dezenas de pontos. Por esta razão, decidi abandonar esta solução e adotar as camadas vetoriais para a visualização de grandes volumes de dados.

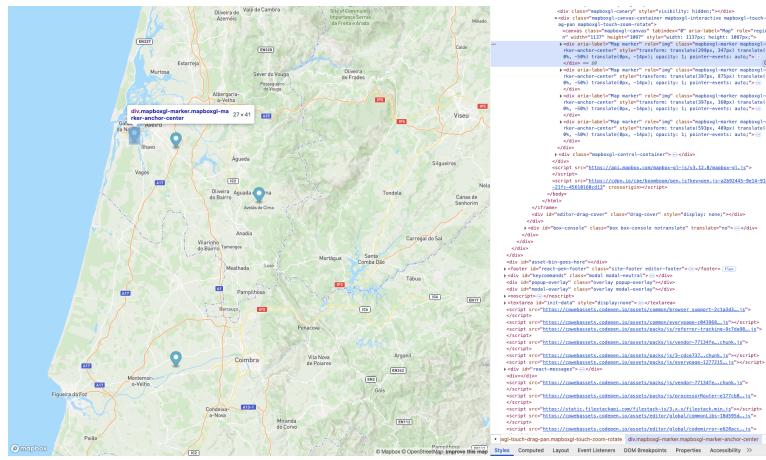


Figura 4.14: Exemplo de marcadores

Map Utils

Para facilitar a criação do mapa e simplificar o desenvolvimento de novos sistemas, foram desenvolvidas algumas classes e interfaces que abstraem e organizam as funcionalidades necessárias. Algumas dessas estruturas podem ser observadas na Figura 4.15.

```
export interface GpsMapProps {
  style: MapStyle;
  center: LngLatLike;
  zoom: number;
  points?: GeoPoint[];
  markerOnClick?: () => void;
  drawFeatures?: FeatureCollection;
  editMode?: boolean;
  selectFeature?: Feature | null;
  divClass?: string;
  drawButtons?: boolean;
  heatMapButton?: boolean;
  disableClicking?: boolean;
  mapLoaded?: boolean;
}

export interface MarkerStyle { ... }

/** Possible map styles.
 */
export enum MapStyles {
  standard = 'standard',
  standard_satellite = 'standard-satellite',
  streets_v12 = 'streets-v12',
  outdoors_v12 = 'outdoors-v12',
  light_v11 = 'light-v11',
  dark_v11 = 'dark-v11',
  satellite_v12 = 'satellite-satellite-v12',
  satellite_streets_v12 = 'satellite-streets-v12',
  navigation_day_v1 = 'navigation-day-v1',
  navigation_night_v1 = 'navigation-night-v1'
}
```

(a) *Map Interfaces*

```
export class MapHelper {
  public constructor(private readonly map: Map) {}

  /**
   * Adds a marker to specified coordinates.
   * Also receives a callback function to execute when the marker is 'clicked'.
   */
  public addMarker(options: AddMarkerOptions): mapboxgl.Marker {
    const marker = new mapboxgl.Marker(options.style)
      .setLngLat(options.lngLat)
      .addTo(this.map)
      .setPopup(options.popup);

    if (options.callback) {
      marker.getElement().addEventListener('click', options.callback);
    }

    return marker;
  }

  /**
   * Add a source to map.
   * @param source
   */
  public addSource(layerId: string, source: SourceSpecification) {
    this.map.addSource(layerId, source);
  }

  /**
   * updateSources
   */
  public updateSources(layerId: string, featureCollection: FeatureCollection) {
    const source = this.map.getSource(layerId) as GeoJSONSource | undefined;
    if (!source) return;
    source.setData(featureCollection);
  }

  /**
   * Adding new layer. 'We need to add a new source before adding a layer.'
   * @param layer layer configuration.
   */
  public addLayer(layer: Layer) {
    this.map.addLayer(layer);
  }
}
```

(b) *Map classes*

Figura 4.15: *Map utils*

Desenho de polígonos

Uma das funcionalidades mais interessantes que desenvolvi foi a implementação de um sistema de desenho e edição de polígonos sobre o mapa.

Esta funcionalidade foi implementada em cima de um *plugin* criado pela comunidade, *mapbox-gl-draw*, que permite adicionar funcionalidades de desenho diretamente sobre o mapa. Um dos elementos que deram mais esforço na implementação desta funcionalidade foi a componente de edição.

Este permite ao utilizador remover, mover, aumentar e diminuir polígonos desenhados. Foram igualmente realizadas alterações no *plugin* de desenho, de modo a adaptar toda a sua interface, cores e opções, garantindo a consistência visual e funcional com o restante da aplicação, Figura 4.16.



Figura 4.16: Desenhar polígonos

O sistema funciona conforme representado na Figura 4.17. Esta abordagem revelou-se a mais prática e eficiente para desenvolver a funcionalidade de desenho e edição de polígonos. Além disso, ao estruturar o sistema desta forma, torna-se possível, no futuro, carregar polígonos a partir de uma base de dados, editá-los diretamente no mapa e, posteriormente, guardar as alterações realizadas.

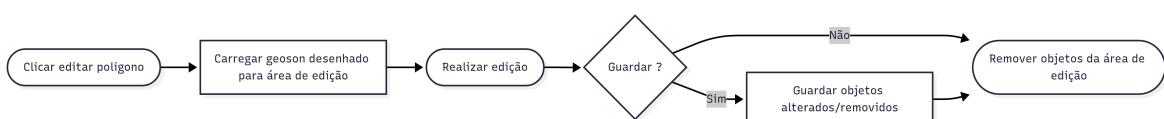


Figura 4.17: Flow de edição de polígonos

4.2.4 Página *reset password*

Uma das tarefas que realizei durante este estágio foi a criação das páginas e do sistema para recuperar palavras passe. No que diz respeito à interface foram, foi criada uma nova rota que permite ao utilizador indicar o seu email para receber um *link* para recuperar a palavra passe e outra página para alterar a mesma.

Todo este sistema foi criado na mesma rota, */reset-pass*, ao entrar nesta página somos redirecionados para uma tela que pede ao utilizador para inserir o endereço de *email* conectado à conta, ao inserirmos o email e clicarmos no botão *Send password reset email*, é-nos enviado um email, se esse email for enviado então vai aparecer uma mensagem de sucesso.

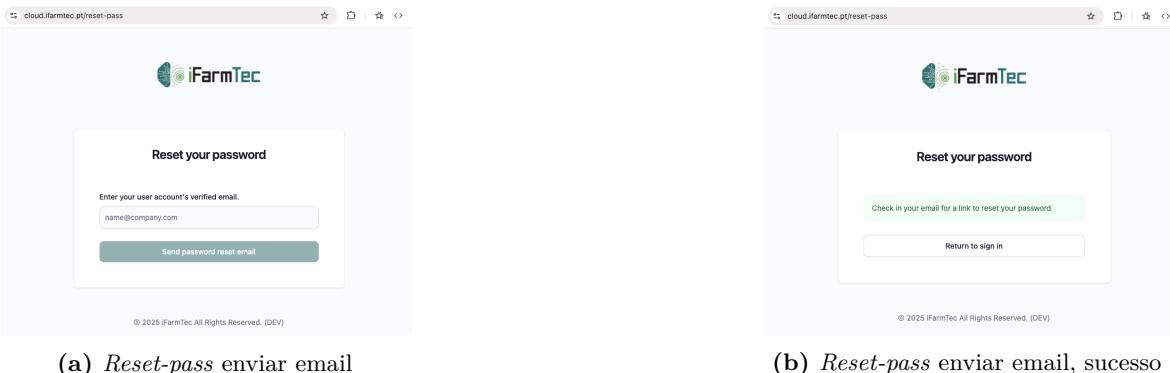


Figura 4.18: Páginas *reset-pass*

Após criação desta tela, criei a outra página numa sub-rota, */reset-pass/[token]*. É nesta página que o utilizador pode alterar a sua palavra passe. Precisa de confirmar o seu email, e inserir a nova palavra passe.

Foi possível criar algo deste género graças ao sistema de rotas *[slug]* do *sveltekit*. Esta rota permite criar uma nova página que recebe um parâmetro no *url*, desta forma conseguimos utilizar o mesmo *url* para propósitos diferentes, apenas com a adição de um token. No sucesso da alteração o utilizador é reencaminhado para a página de *login*, Figura 4.19b

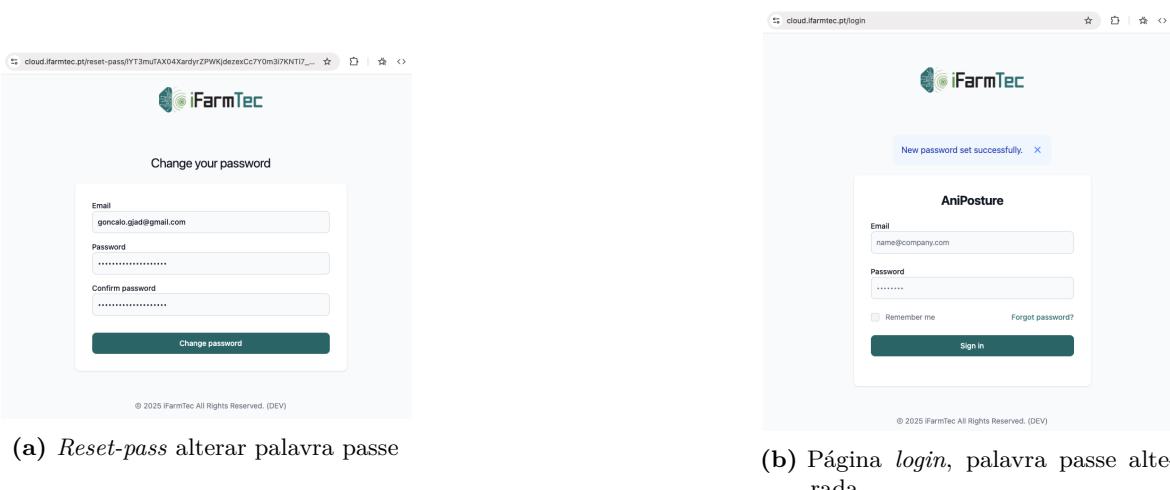


Figura 4.19: Páginas *reset-pass* alterar palavra passe

Para tornar todo este sistema possível e também para permitir que utilizadores logados tenham acesso a estas páginas, foi criado um sistema de redirecionamento que redireciona utilizadores não logados para a página de login, a não ser que eles estejam a tentar modificar a palavra passe, esses páginas também podem ser acedidas por eles.

```
let { children } = $props();
const url_path = page.url.pathname;

setContext('url_path', url_path);
const nonLoginRoutes = ['/login', '/reset-pass', 'app-gaime'];

const filter = nonLoginRoutes.filter(str => url_path.includes(str));

if (sessionService.isAuthenticated() && url_path == '/login') {
    goto('/');
} else if (!sessionService.isAuthenticated() && filter.length === 0) {
    goto('/login');
}
```

Figura 4.20: Redirecionar para o login

Este código é colocado no *layout.svelte* da aplicação, página responsável por criar um interface em comum com várias páginas da aplicação. Ao colocarmos o código aqui conseguimos temos a certeza que sempre que o utilizador tentar entrar numa página que não tem acesso, ou não exista, ele será automaticamente redirecionado para o */login* ou a raiz da aplicação */*.

4.2.5 Página de dashboard

(TODO)

Uma das páginas que mais trabalhei e fui alterando enquanto o estágio decorria, foi a página dashboard.

4.2.6 Página de status

(TODO)

Esta página é responsável por mostrar os dispositivos, o uptime de cada dispositivo por dia.

4.2.7 Página de gráficos

(TODO)

Aqui é mostrado um gráfico com os dados do acelerômetro e da temperatura do dispositivo, em um certo intervalo de tempo.

4.2.8 Página de erro

(TODO)

Esta página foi criada para mostrar códigos de erros que acontecem na aplicação.

4.2.9 Rotas sem inicio de sessão

(TODO)

Foram criadas rotas que permitam aos utilizadores aceder a alguns dados, pontos e gráficos sem iniciar sessão.

4.3 SERVIDOR DA APLICAÇÃO

4.3.1 O que é o *backend* ?

4.3.2 Rota para obter número de pedidos por dispositivo

4.3.3 Sistema de envio de emails

4.3.4 Dados do *acc* e de temperatura no sample

4.3.5 Rotas para aceder a dados sem *login*

4.3.6 Correção de um *bug* que duplicava dados

CAPÍTULO 5

Analise Resultados

CAPÍTULO 6

Conclusões

APÊNDICE A

Apendices

ANEXO
PLANO DE TRABALHO DO ESTÁGIO

IDENTIFICAÇÃO DO ESTÁGIO	
Estudante	Gonçalo José de Almeida Dias
Número Mecanográfico	102344
Unidade Curricular	99142-ESTÁGIO / PROJETO EM TECNOLOGIAS DA INFORMAÇÃO
Curso	Licenciatura em Tecnologias de Informação
Data de início	segunda-feira, 10 de fevereiro de 2025
Data de conclusão	sexta-feira, 13 de junho de 2025
Horário a cumprir	8/horas dia
Local (ou Locais) do Estágio	Zona Industrial da Mota, Rua 12, Lote 51, Freguesia de Gafanha da Encarnação
Orientador da UA-ESTGA	Ivan Miguel Serrano Pires
Orientador da EA	Gustavo Corrente

Resumo do Plano de Trabalho

OBJETIVOS	ATIVIDADES A DESENVOLVER	CALENDARIZAÇÃO PREVISTA (HORAS OU DIAS)
Acolhimento e introdução ao desenvolvimento	Acolhimento Introdução às tecnologias e ferramentas: GIT, GITLAB, JENKINS, Ambiente JAVA e SVELTE	40h
Levantamento de requisitos	Levantamento de requisitos no projeto aniposture	40h
Escolher a API de Mapas	Estudo de funcionalidades entre GoogleMaps e OpenStreetMaps	120h
Desenvolvimento da UI	Desenvolvimento de UI de georreferenciação	120h
Desenvolvimento dos serviços	Desenvolvimento de serviços RESTful	120h
Testes funcionais	Realização de Testes Funcionais	24h
Relatório	Finalização da Escrita do Relatório	2h

Figura A.1: Plano trabalho