Sistema de recolha de dados meteorológicos

Paulo José Capa Azevedo Meneses

Nº 17611 – Regime Pós-laboral

Orientação

Joaquim Silva

Ano letivo 2020/2021

Licenciatura em Engenharia de Sistemas Informáticos

Escola Superior de Tecnologia

Instituto Politécnico do Cávado e do Ave

Identificação do Aluno

Paulo José Capa Azevedo Meneses

Aluno número 17611, regime pós-laboral

Licenciatura em Engenharia de Sistemas Informáticos

Orientação

Joaquim Silva

Professor

Informação sobre o Estágio

Phosphorland

Rua do Conhecimento nº10, 4730-575

Orientação de Ricardo Ferreira

RESUMO

Este projeto teve como objetivo o desenvolvimento de um sistema de recolha de dados meteorológicos através de estações meteorológicas com o objetivo da sua disponibilização pela internet aos utilizadores, para auxiliar nas decisões da produção agrícola. Na fase inicial foi efetuada uma pesquisa sobre as estações meteorológicas, as suas formas de comunicação de dados e os sistemas semelhantes. Na fase seguinte, foram definidos os requisitos e projetada a arquitetura do sistema. Partindo da arquitetura do sistema, foi desenvolvido um modelo para a base dados. Este projeto resultou no desenvolvimento de uma base de dados e de uma *Application Programming Interface,* desenvolvida em .Net Core, para realizar a recolha e armazenamento na base de dados e a posterior disponibilização dos dados num website desenvolvido na *framework* Bootstrap.

ABSTRACT

This project aimed to develop a system for collecting meteorological data through meteorological stations with the objective of making it available over the Internet to users, to assist them in the decisions of agricultural production. In the initial phase, it was carried out a search about weather stations, their forms of data communication and similar systems implementations. In the next phase, the requirements were defined, and the system architecture was designed. Based on the system architecture, a model was developed for the database. Therefore, this project resulted in the development of a database and an Application Programming Interface, implemented in .Net Core, to perform the data collection and storage in the database, and the subsequent availability of the data in a website developed using the Bootstrap framework.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

Glossário

Framework – Uma *framework* permite um nível mais elevado de abstração no desenvolvimento.

Open Source – *Open-source* é um modelo de desenvolvimento que promove a distribuição gratuita, utilização e modificação do código fonte.

Siglas e Acrónimos

API – Application Programming Interface (Interface de Programação de Aplicações)

CDN – Content Delivery Network (Rede de distribuição de conteúdo).

CSV – Comma-separated values é um formato de texto regulamentado pelo RFC 4180, em que os valores de uma tabela são separados por vírgulas.

FTP – File Transfer Protocol (Protocolo de Transferência de Ficheiros)

HTTP – HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto)

IOT – Internet of Things (Internet das coisas)

MVC – Model-view-controller (Modelo-Visão-Controlo) é um padrão de arquitetura de software, composto por 3 camadas, o modelo, a visualização e o controlador.

SI – Sistema de informação.

1. Introdução

A agricultura é uma atividade com milhares de anos, que atualmente está a ficar cada vez mais interconectada com sistemas de informação e de gestão, usando esses sistemas para aumentar o rendimento das plantações. Esta agricultura é um novo conceito chamado agricultura de precisão.

A agricultura de precisão pode ser definida como a aplicação de tecnologias e princípios para gerir a variabilidade espacial e temporal associada a todos os aspetos da produção agrícola, com o objetivo de melhorar o desempenho das culturas e a qualidade ambiental (Francis & Peter, 1999).

Há um grande crescimento de sistemas de *Internet of Things* para auxiliar na gestão agrónoma. Com a IoT emergem novas formas de olhar para agricultura e novas soluções para problemas existentes na mesma, tais como sistemas para gestão de rega inteligente, gestão de gado, entre outros, abrindo novas possibilidades nesta área.

O sistema informático deste projeto tenciona auxiliar na gestão e apoio à tomada de decisões. Para isso é necessário criar um sistema de recolha de grandes quantidades de dados através de sistemas remotos, como estações meteorológicas autónomas para a leitura dos dados meteorológicos no terreno e em tempo real (ver Figura 1).

Figura 1: Estação meteorológica automática.

Estes sistemas remotos enviarão os dados, através da internet, para um servidor que armazena esses dados numa base de dados otimizada para dados temporais. Os dados são servidos a uma página web para o utilizador poder visualizar, ajudando na gestão e decisões a tomar.

1.1. Objetivos

Pretende-se o desenvolvimento de um sistema de recolha de dados meteorológicos de estações meteorológicas. Este sistema deve ser constituído por uma base dados, uma *Application Programming Interface* (API) para receber e inserir os valores registados pelos sensores da estação na base dados e disponibilização de dados relacionados com meteorologia agronómica. Por fim uma página *web* para que o utilizador possa ver e analisar os dados.

1.2. Contexto

Este projeto foi proposto pela empresa Phosphorland lda, a qual oferece ao mercado um software de gestão agrícola que abrange todo o setor agroalimentar. Este software é usado pelos agricultores para realizarem a gestão das suas explorações agrícolas. Com o recurso cada vez maior dos dados meteorológicos para ajudar e definir as operações de campo, tornou-se essencial o software disponibilizar

estes dados de forma prática para que os agricultores possam tomar as melhores decisões.

Desta forma, este projeto vem ajudar a facultar aos agricultores a possibilidade de integrar as suas estações meteorológicas para poderem ter estes dados em tempo real e poderem alimentar o controlo e eficiência das suas explorações.

1.3. Estrutura do documento

Este documento está dividido em cinco capítulos. No capítulo 2 - Estudo e definição dos requisitos – são descritos os aspetos importantes do estudo inicial efetuado, e é realizada a definição dos requisitos funcionais e não-funcionais.

Depois, no capítulo 3 – Desenvolvimento - é apresentada a arquitetura do sistema e base de dados, incluindo as principais etapas do desenvolvimento.

Seguidamente no capítulo 4 - Análise de resultados - são exibidos os resultados do desenvolvimento, tais como a API e a página web, e as principais dificuldades no desenvolvimento.

Por último, o capítulo 5 - Conclusão - encerra o projeto com as principais conclusões e melhorias futuras ao projeto

2. Estudo e definição dos requisitos

No estudo inicial do projeto foi necessário perceber como as estações meteorológicas automáticas comunicam, o tipo de dados que recolhem e perceber se existem soluções deste tipo no mercado.

2.1. O que são dados meteorológicos

Os dados meteorológicos são dados que representam o estado do tempo de alguma localização geográfica. Os dados são lidos através de sensores presentes nas estações meteorológicas. A recolha dos dados é efetuada por estações meteorológicas automáticas que estão equipadas com sensores para medir vários parâmetros que podem variar dependendo da estação. Os tipos de dados normalmente são todos valores numéricos representando alguma medida, tais como: temperatura (Cº), humidade relativa (%), radiação solar (KJ/m^2), intensidade do vento (Km/h), direção do vento, precipitação (mm), pressão (HPA), temperatura do solo (Cº) a 0,5 m, a 1,5 m e a 3 m, e a humidade (%) do solo a 0,5 m, a 1,5 m e a 3 m.

2.2. Soluções existentes

No início do projeto, realizou-se uma pesquisa por soluções deste tipo no mercado, tendo se chegado à conclusão que existem muito poucas soluções e que estas são algo limitadas no seu funcionamento. Percebeu-se que as estações meteorológicas para o envio de dados remotamente podem utilizar vários tipos de redes como a rede Global System for Mobile Comunications (GSM) e Long Range Wide Area Network (LoRaWAN).

As estações meteorológicas encontradas, pertencem a várias empresas da área, como, a Pessl Instruments, que oferece uma variedade de estações meteorológicas e sensores, mas não é possível receber os dados diretamente dessas estações sem passar

pela *cloud* deles. Existe ainda a Davis Instruments e a RainWise que funciona como a Pessl Instruments. Depois ainda existe Ambient Weather, que as estações só oferecem comunicação através de uma rede wi-fi local.

Quanto a outras estações meteorológicas existe pouca documentação sobre como é que se efetuam as ligações às mesmas através das redes mencionadas. A maior parte das estações meteorológicas não são modulares.

As redes de comunicação que estas estações usam, em quase todas e a única com maior viabilidade para Portugal é a rede GSM, pois tem uma boa cobertura em todo o país, ao contrário da rede LoRaWAN que é relativamente recente e possui pouca cobertura até à data. A LoRaWAN, é uma rede aberta que utiliza frequências de rádio para comunicar e permite a comunicação a longo alcance e de baixo consumo energético.

Existem ainda alguns serviços que apresentam dados online como o *Weather Underground* (https://www.wunderground.com/), que é uma comunidade global de pessoas que ligam as suas estações meteorológicas pessoais a esta rede, e podem aceder remotamente a todos os dados.

Como não foi possível encontrar uma estação meteorológica que cumprisse os requisitos necessários, tomou-se a decisão de implementar um novo sistema que permita enviar os dados meteorológicos através de um *endpoint* na API, para serem inseridos na base de dados e apresentados ao utilizador.

2.3. Requisitos funcionais

Após uma pesquisa de soluções existentes no mercado e da solução idealizada chegou-se aos seguintes requisitos, apresentados na Tabela 1.

Ref.	Grupo	Requisito	Descrição
RF01	Gráficos	Gráfico de temperatura	Gráfico com dados da temperatura e ponto de orvalho ao longo do tempo
RF02		Gráfico de humidade	Gráfico com dados da humidade ao longo do tempo
RF03		Gráfico de pressão	Gráfico com dados da pressão ao longo do tempo
RF04		Gráfico de precipitação	Gráfico com dados da precipitação

RF05		Gráfico de vento	Gráfico com dados da intensidade do vento ao longo do tempo ao longo do tempo
RF06		Rosa dos ventos	Rosa dos ventos com a direção e intensidade do vento
RF07	Dados	Dados sumariados	Sumário de dados num determinado período de tempo, com o mínimo, máximo e média
RF08		Dados atuais	Consultar a última observação de uma estação
RF09	Inserção de dados	Entrada de dados	Disponibilizar ponto para entrada de dados das estações automáticas para a base dados
RF10		Adicionar estações	Disponibilizar ponto para entrada de novas estações na base dados para a recolha de dados
RF11	Dados brutos	Tabela de dados	Visualizar uma tabela com os dados em bruto de uma estação
RF12	Exportação	Exportar dados	Exportar dados para o formato CSV.

Tabela 1: Tabela de requisitos.

2.4. Requisitos não-funcionais

Foram definidos vários requisitos técnicos no início do projeto: o sistema deverá operar em Linux, deverá existir uma página web que mostre os dados desenvolvido com a tecnologia Bootstrap e, devido a poder haver a ingestão de uma grande quantidade de dados, estes deverão ser armazenados numa base de dados de séries temporais. O Bootstrap é uma *framework* para o desenvolvimento web de *front-end* em código-fonte aberto direcionada a criar sites e aplicações responsivos, utilizando modelos de design para tipografia, formulários, botões, navegação, entre outros.

As opções para a base de dados de séries temporais foram a InfluxDB e a TimescaleDB, que oferecem performance melhorada para este tipo de dados. A InfluxDB é uma plataforma para o desenvolvimento de IoT e é construída para gerir grandes volumes de dados temporais de sensores, aplicações e infraestruturas. A TimescaleDB é uma base dados relacional *open-source* para dados de série temporal, oferecendo uma capacidade superior em relação à velocidade das consultas do que PostgreSQL, na qual é baseada. Além disso, tem maior capacidade de ingestão de dados e permite ser utilizada em qualquer tipo de ambiente. Uma comparação entre ambas as opções é apresentada na Tabela 2.

	TimescaleDB	InfluxDB
Fundação da base de dados	PostgreSQL	Proprietária
Licença	Fonte disponível, permissiva e livre	Licença MIT
Linguagem de consulta	SQL	FluxQL
Modelo de dados	Relacional	NoSQL
Performance de consulta	Até 7000x mais rápido que o InfluxDB, especialmente à medida que a cardinalidade aumenta	Competitiva quando cardinalidade da métrica é pequena

Tabela 2: TimescaleDB vs InfluxDB

A TimescaleDB foi escolhida, por apresentar melhor performance em consultas simples do que o InfluxDB, e apresentar outras características desejadas para este projeto. Nas estações meteorológicas podem ser utilizadas diferentes unidades para as medições e diferentes sensores, por exemplo, umas estações meteorológicas podem ter sensores de temperatura do solo e outras não. A tabela de requisitos não funcionais é exibida em baixo na Tabela 3.

Ref.	Requisito	Descrição
RNF01	Compatível com Linux	Os servidores são máquinas Linux
RNF02	Bootstrap	*Framework open-source* já utilizada no resto organização
RNF03	TimeScaleDB	Grande quantidade de dados de séries temporais

Tabela 3: Requisitos não-funcionais.

3. Desenvolvimento

O desenvolvimento inicia-se por idealizar a estrutura do sistema e quais as ferramentas necessárias para execução do projeto. É com base nestas decisões que se definem as etapas de desenvolvimento.

3.1. Arquitetura

Para este Sistema de Informação é necessária uma API para fazer todas as transações de receção e disponibilização de dados, uma base dados que seja boa para dados temporais e uma página web para a visualização desses dados.

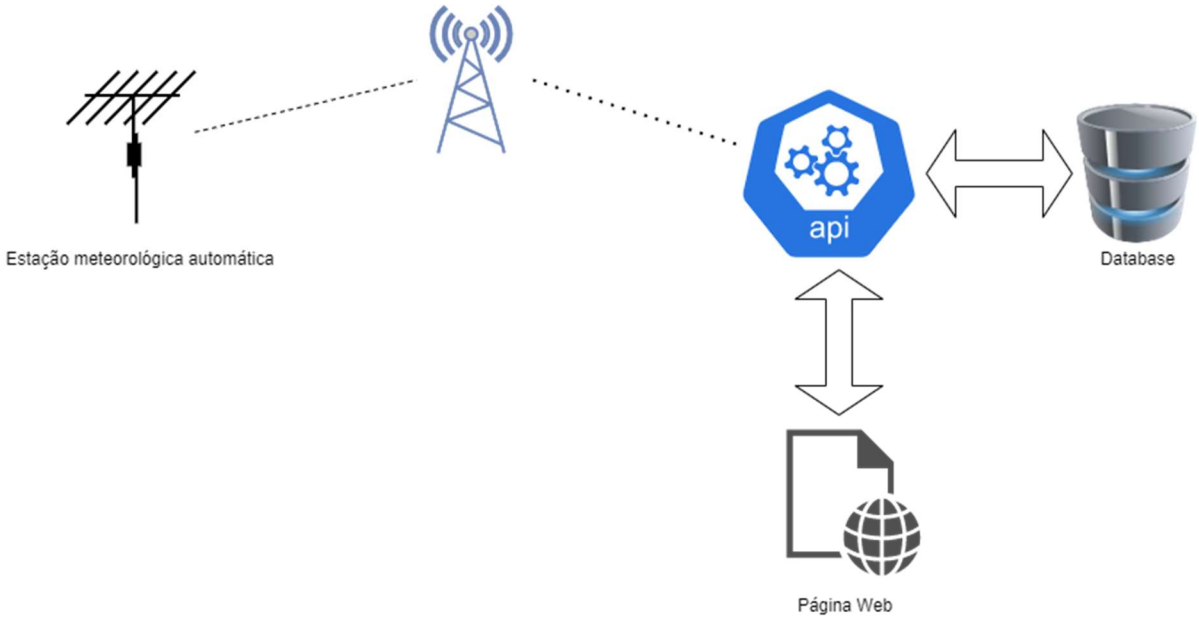


Figura 2: Arquitetura do sistema.

O motor deste projeto é uma API desenvolvida com a *framework* .NET Core 5.0, pois é muito fácil e rápido o desenvolvimento deste tipo de aplicações. É compatível com Linux, cumpre os requisitos não-funcionais determinados para este

projeto e permite a criação de serviços para serem acedidos através da internet. Esta *framework* utiliza a linguagem C# para o seu desenvolvimento.

A tecnologia utilizada para construção da página web foi limitada à *framework* Bootstrap, à linguagem JavaScript e à JQuery. O JQuery é uma biblioteca de funções de JavaScript que auxilia no desenvolvimento da página que oferece uma maior compatibilidade com todos os navegadores. JavaScript é a linguagem que serve de motor à página web.

3.2. Principais etapas de desenvolvimento

As principais etapas de desenvolvimento podem ser divididas em três partes: 1) a criação da base de dados; 2) o desenvolvimento da API; 3) e a construção da página web.

3.2.1. Criação da base de dados

Partindo dos requisitos e restrições, as tabelas necessárias para este projeto são a tabela *station*, que vai guardar as características de cada estação e a tabela *stationrecord* para as observações que cada estação efetuar. O diagrama de base de dados criado na ferramenta Visual Paradigm está apesentado na Figura 3.

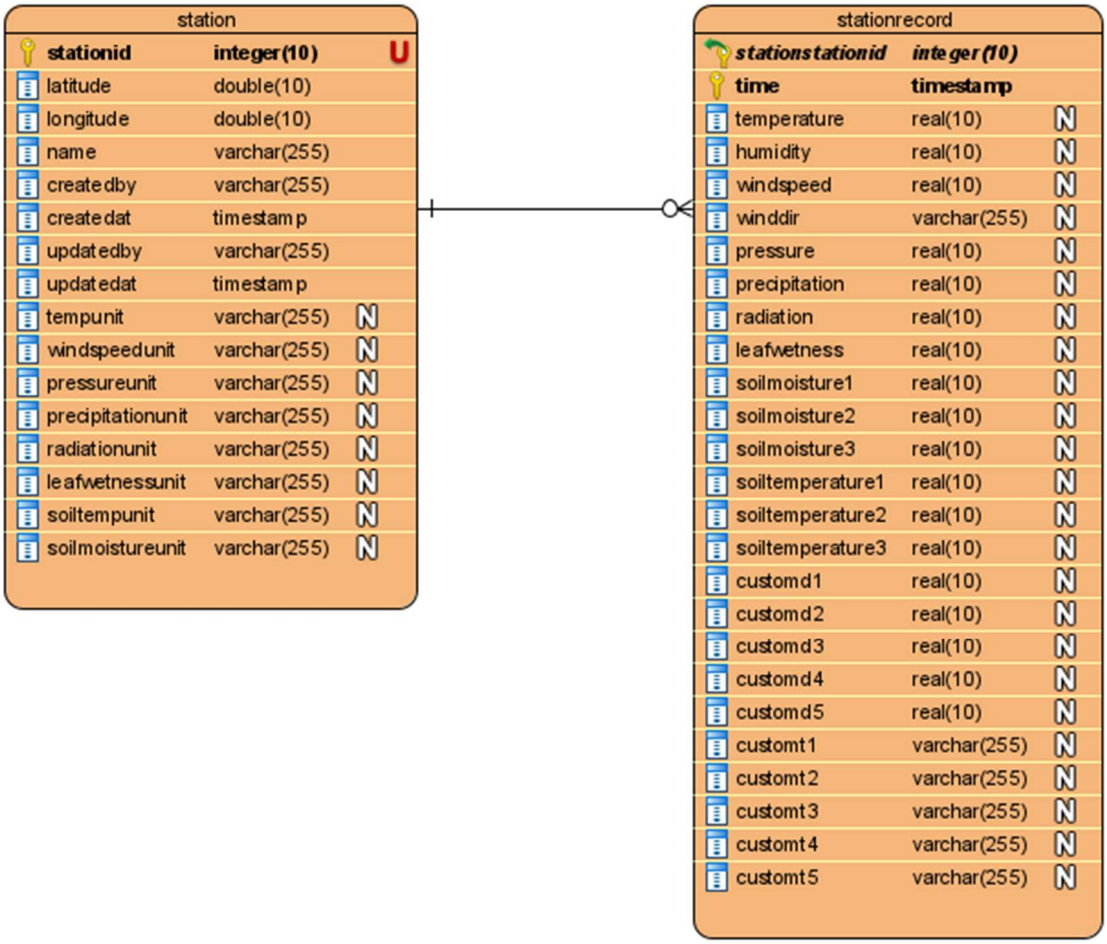


Figura 3: Diagrama de base de dados.

Para a criação da base de dados, foi necessário instalar o PostgreSQL 13, que é um motor de base dados relacional, sobre a qual se instalou o extensão TimeScaleDB. A base dados foi criada com o nome *fourmeteo* e adicionadas duas extensões, a extensão do TimeScaleDB e o módulo do PostgreSQL uuid-ossp. Este módulo permite a utilização de identificadores únicos como campos de uma tabela que serão os identificadores das estações. Os identificadores são constituídos por 128-bits e são gerados automaticamente pelo motor de base dados ao ser inserido um novo registo nessa tabela.

Para tirar partido das vantagens do TimescaleDB é necessário criar uma *hyper-table* da tabela stationrecord que contém dados temporais. Ao criar a *hyper-table*, a tabela irá ser divida em partes mais pequenas para aumentar a performance de consulta e inserção. Estes pedaços da tabela serão mapeados através de uma tabela de metadados. Tudo isto é gerido automaticamente pela extensão.

Os comandos para adicionar as extensões à base de dados são:

```
CREATE EXTENSION IF NOT EXISTS timescaledb;
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

A criação da *hypertable* é a seguinte:

```
SELECT create_hypertable('stationrecord', 'time');
```

A tabela station tem uma chave primária (stationid) que é o identificador da estação e faz uso do uuid. A tabela stationrecord tem uma chave primária composta pelo identificador da estação (stationid) e o tempo (time).

3.2.2. Estrutura da API

A *framework* .Net Core tem ferramentas para gestão de projetos e as suas dependências, utilizando essas ferramentas para criar um projeto novo do tipo MVC.

Foram adicionados três pacotes para acelerar o processo e a facilidade de desenvolvimento: (1) o EntityFramework.Design, que permite a geração do contexto de base de dados e os modelos; (2) o EntityFramework.PostgreSQL, para efetuar ligações a bases de dados PostgeSQL; e (3) a gestão da base de dados através do EntityFramework e do Swashbuckle.AspNetCore. Este último pacote inclui o SwaggerUI que é usado para gerar automaticamente a documentação e possibilitar a interação com a API sem ter nenhuma interface desenvolvida ou lógica implementada.

Na camada modelo encontram-se os modelos que representam as tabelas da base de dados e ainda os modelos incluídos nos pedidos ou que são retornados como resultados de pedidos ao controlador.

Na camada de controlo encontram-se dois controladores, um deles é só para a receção dos dados e o outro para a disponibilização dos dados. No ficheiro appsettings.json encontra-se a configuração da ligação à base de dados, que é demonstrada pelo seguinte código.

```
"ConnectionStrings": {
"4meteodb": "Host=192.168.1.79;
Port=5432;Username=meteoapi;Password=*******;Database=fourmeteo"
},
```

O campo “4meteodb” é o nome desta configuração. O “Host” é o endereço IP da base de dados. A “Port” é a porta de acesso. O “Username” é o nome de utilizador que irá ser utilizado para aceder à base dados e a sua palavra-passe é guardada no campo “Password”. “Database” é o nome da base dados a ser a acedida.

Os modelos e o contexto da base dados pode ser facilmente gerado pela ferramenta do EntityFramework.Design, usando o seguinte comando:

```
Dotnet ef dbcontext scaffold
“Host=localhost;Database=fourmeteo;Username=postgres;Password=password”
Npgsql.EntityFrameworkCore.PostgreSQL -o Models --schema public -f
```

Depois, o Swagger é configurado pelo seguinte código, onde se configura a versão, o título, a descrição e os termos de serviço e todos os comentários em formato *Extensible Markup Language* (XML) no código são compilados num ficheiro único para alimentar a documentação da API pelo Swagger.

```
services.AddSwaggerGen(c =>
    {
       c.SwaggerDoc("v1", new OpenApiInfo
         {
           Version = "v1",
                Title = "4Meteo Api",
                Description = "4Meteo API",
                TermsOfService = new Uri("https://example.com/terms"),
              });
              var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Nam
e}.xml";
              var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
              c.IncludeXmlComments(xmlPath);
          });
```

O API.csproj é um ficheiro de gestão do projeto do .Net Core no qual são identificadas quais as dependências do projeto e outras configurações.

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <GenerateDocumentationFile>true</GenerateDocumentationFile>
    <NoWarn>$(NoWarn);1591</NoWarn>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
```

```
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version
="5.0.4">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildt
ransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="Swashbuckle.AspNetCore" Version="5.6.3" />
    <PackageReference Include="Npgsql.EntityFrameworkCore.PostgreSQL" Versio
n="5.0.2" />
  </ItemGroup>
</Project>
```

3.2.3. Exemplo de um *endpoint* da API

Os comentários no início da função do *endpoint* são usados pelo Swagger para a documentação da funcionalidade do *endpoint*. Estes comentários são feitos em XML. Isto permite definir uma pequena descrição para o *endpoint*, definir os tipos, os parâmetros de entrada e as suas descrições e ainda uma descrição para o retorno.

```
/// <summary>
/// Returns a array with a time and a corresponding temperature
/// </summary>
/// <param name="uuid">Station uuid</param>
/// <param name="from">End data for the data</param>
/// <param name="mode">Mode 1 = days, 2 = weeks, 3 = months </param>
/// <returns></returns>
```

O tipo de pedido HTTP, rota, nome e parâmetros da função são definidos da seguinte forma.

```
[HttpGet]
[Route("[action]/{uuid}_{from}_{mode}")]
public async Task<IActionResult> GetTempRecords(Guid uuid, DateTime from, in
t mode)
```

É importante definir bem os tipos dos parâmetros de entrada, porque o *endpoint* desta forma não irá aceitar pedidos com tipos diferentes, aumentando a segurança. Os métodos podem ainda ser assíncronos e retornam o resultado da tarefa com os respetivos erros e campos do HTTP.

Depois é feita lógica toda do pedido, que inclui aceder à base de dados, fazer a consulta e esperar pela resposta, sendo, depois disso, os dados retornados

O retorno é feito através do IActionResult, que é apropriado quando vários tipos de retorno são possíveis numa ação. Os tipos de retorno representam vários códigos de estado HTTP. No exemplo abaixo, é retornada a variável “data” dentro do resultado da ação, que suporta qualquer tipo de dados. Neste caso é retornado um Ok no protocolo HTTP, que é o código 200.

```
return Ok(data);
```

3.2.4. Página web

A página web é constituída essencialmente por 3 ficheiros. O index.html que contém o HTML todo da página, fazendo o uso da *framework* Bootstrap. O main.js possui o JavaScript e JQuery da página, que faz os pedidos ao servidor. O ficheiro style.css tem os estilos da página.

4. Análise dos resultados

Os resultados obtidos deste projeto são a base dados, que contém duas tabelas, a API que contém os *endpoints*, e a página web para a visualização dos dados.

4.1. *Endpoints* disponíveis na API

Os *endpoint* resultantes podem ser visualizados e testados na página gerada pelo Swagger. Eles podem ser separados em duas partes: (1) os *endpoints* de retorno de dados - são todos os que disponibilizam os dados recolhidos das estações (ver Figura 4); e (2) os *endpoints* de receção de dados - os que servem para receber dados das estações e inseri-los no sistema (ver Figura 5).

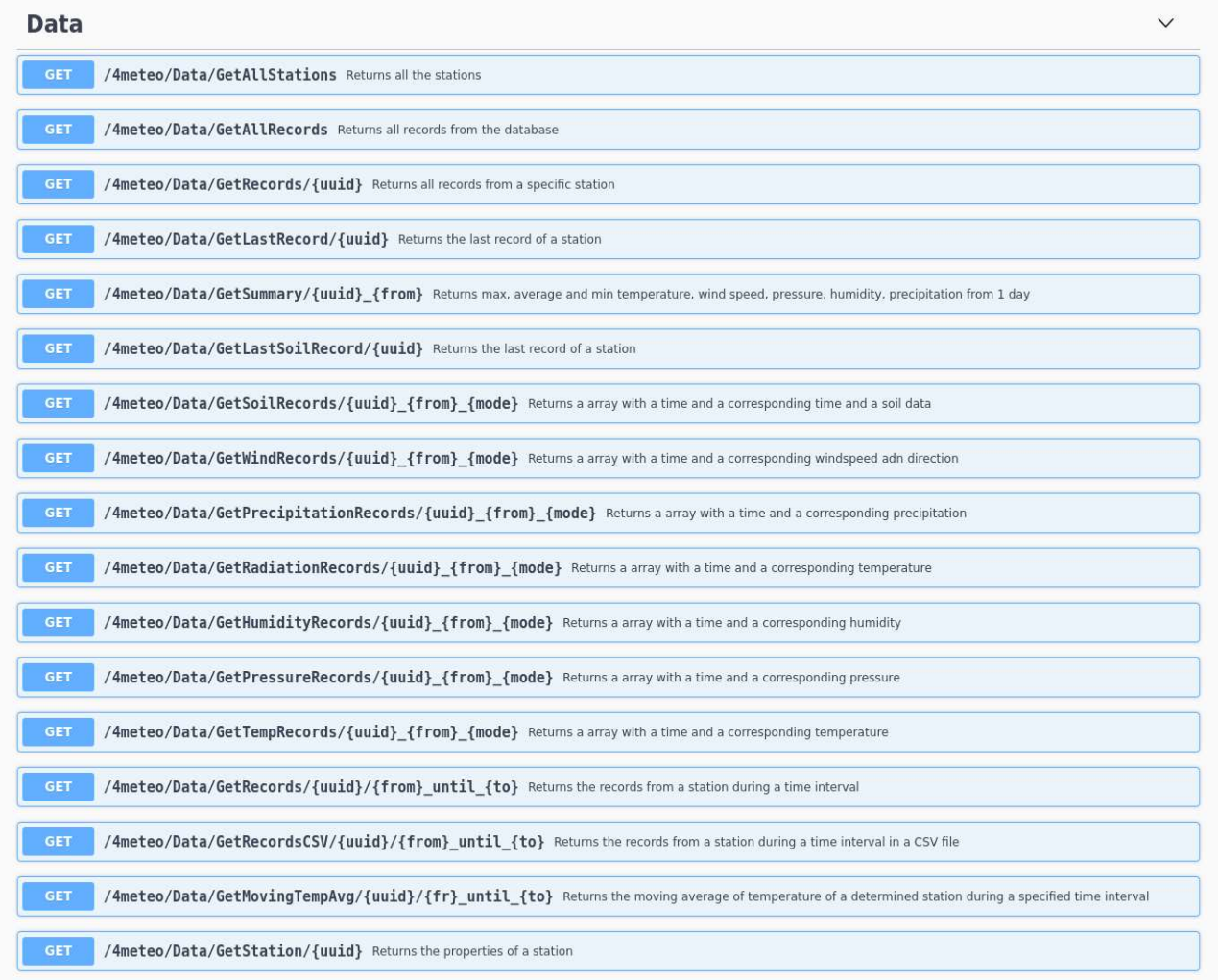


Figura 4: Endpoints de retorno de dados.

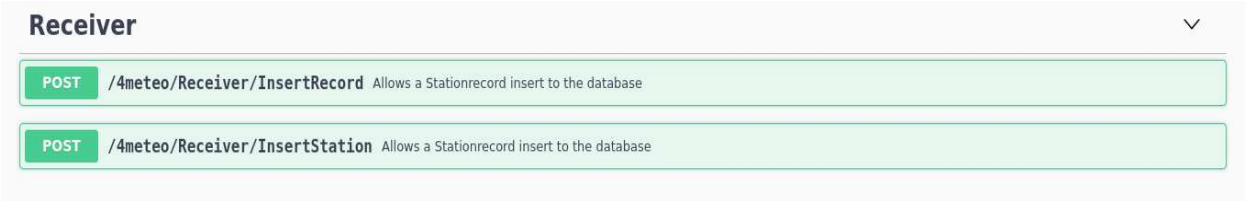


Figura 5: Endpoints de receção de dados

4.2. Página web

A apresentação da estação é mostrada no primeiro cartão onde estão os dados da estação e um mapa com a sua localização, conforme se apresenta na Figura 6.

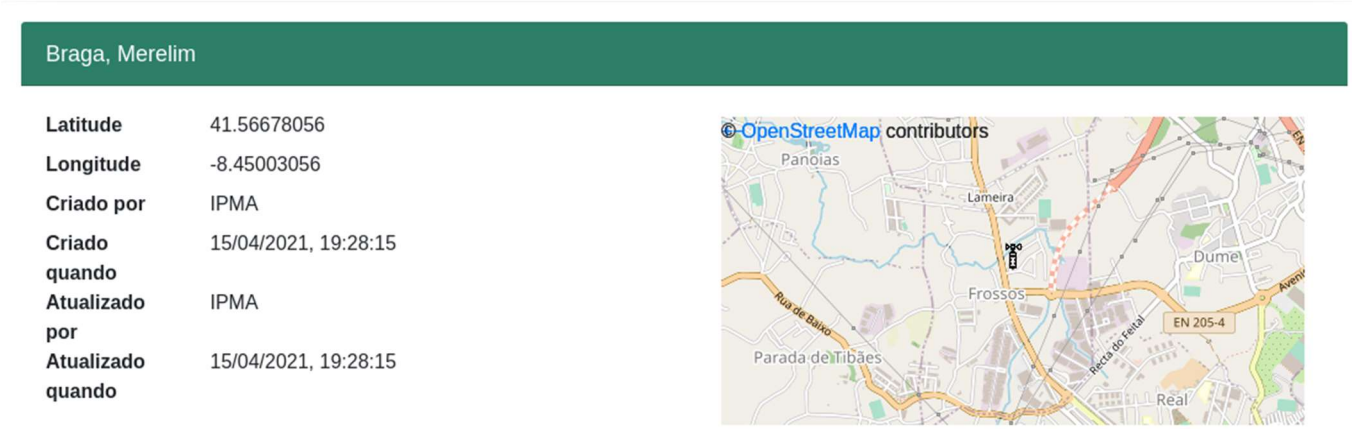
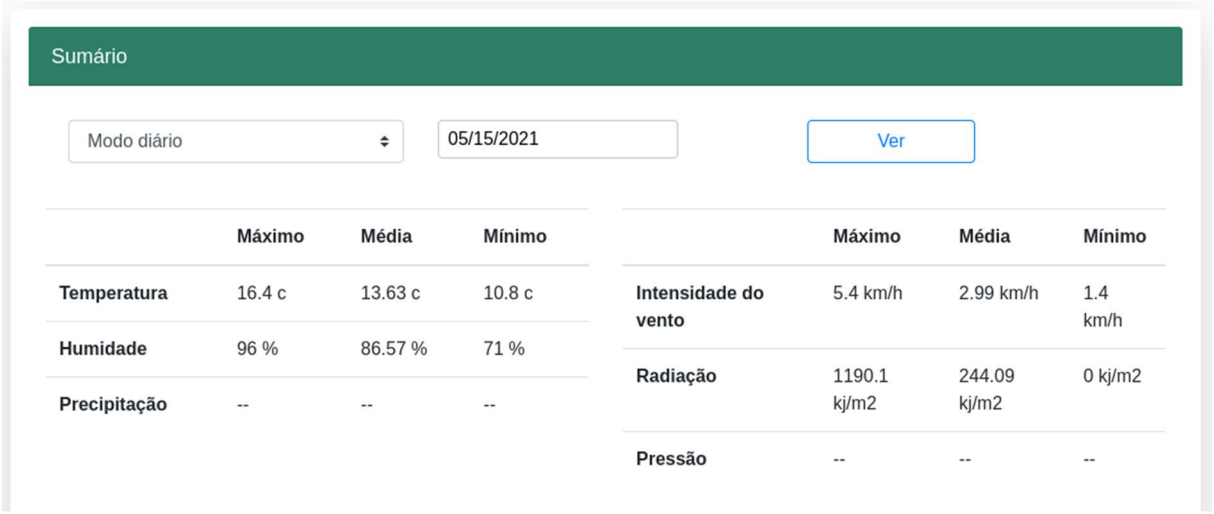


Figura 6: Propriedades da estação.

No segundo cartão estão os últimos dados recebidos da estação, conforme se apresenta na Figura 7.

Figura 7: Dados atuais da estação.

O terceiro cartão mostra um sumário de dados baseado na data selecionada, a qual por definição corresponde ao dia atual. Este sumário contém os máximos, mínimos e médias observados no período selecionado (ver Figura 8).



	Máximo	Média	Mínimo
Temperatura	16.4 c	13.63 c	10.8 c
Humidade	96 %	86.57 %	71 %
Precipitação	--	--	--

	Máximo	Média	Mínimo
Intensidade do vento	5.4 km/h	2.99 km/h	1.4 km/h
Radiação	1190.1 kj/m2	244.09 kj/m2	0 kj/m2
Pressão	--	--	--

Figura 8: Quadro de sumário de dados.

Mais abaixo encontram-se gráficos com os dados ao longo do tempo. Os gráficos aparecem conforme existam, ou não, dados (ver Figura 9).

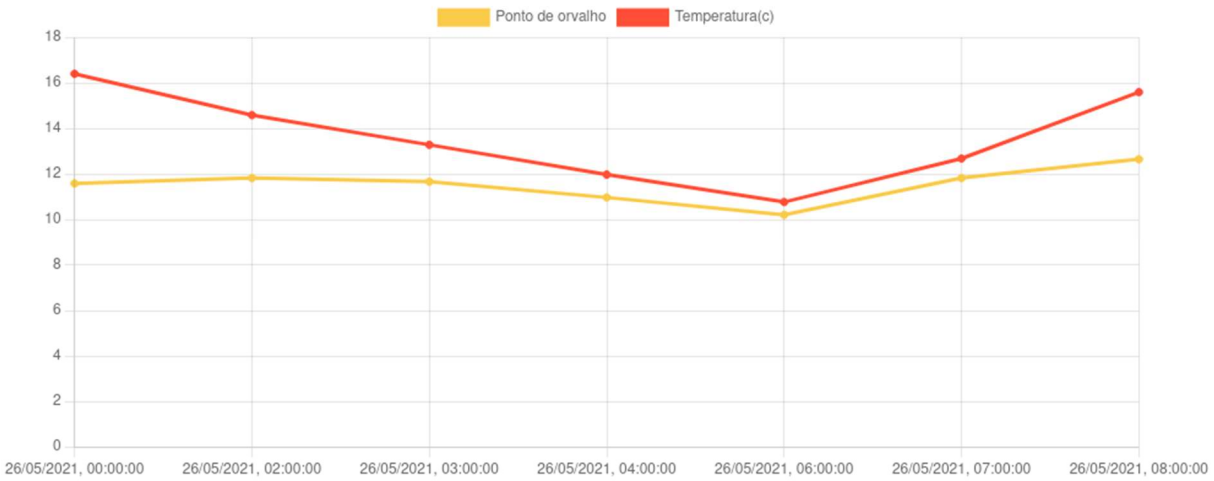


Figura 9: Gráfico de temperatura e ponto de orvalho.

E existe uma rosa dos ventos que mostra velocidades e direções do vento (ver Figura 10).

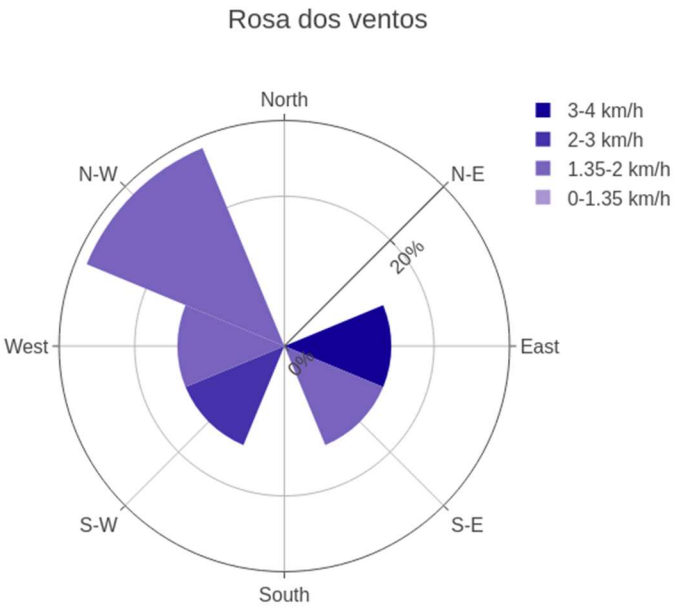


Figura 10: Rosa dos ventos.

No final, é apresentado um cartão com uma tabela de dados, os quais podem ser escolhidos através da seleção de duas datas. Esta tabela pode ser exportada para CSV, Excel, PDF ou imprimida (Ver Figura 11).

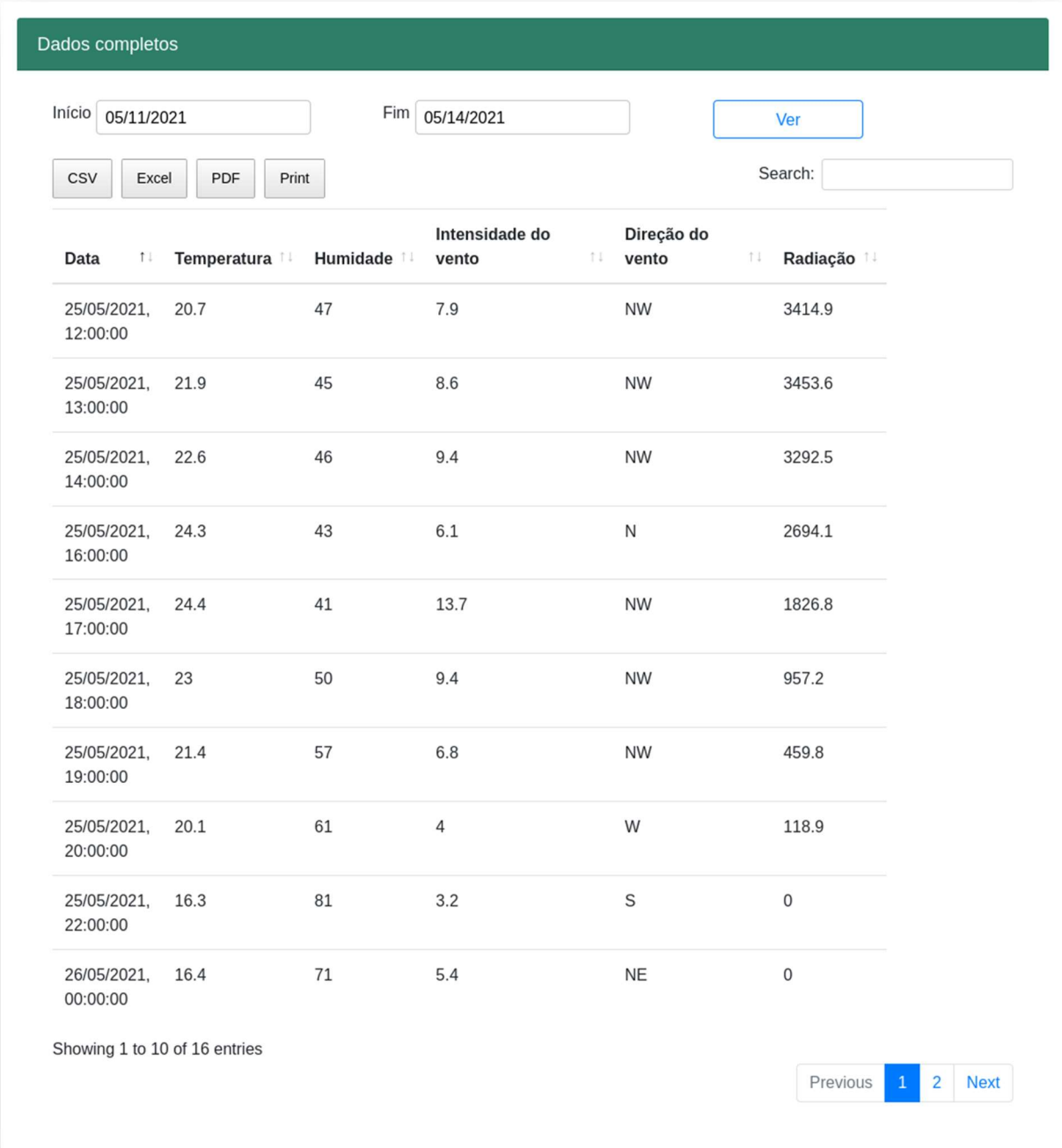


Figura 11: Tabela de dados.

4.3. Principais dificuldades

A primeira dificuldade que surgiu neste projeto foi a pesquisa e estudo, devido à existência de pouca informação sobre sistemas que façam uso das tecnologias envolvidas no projeto. A inexistência de documentação detalhada sobre os processos

de recolha de dados, comunicação e o funcionamento das estações meteorológicas automáticas foi a principal dificuldade.

No desenvolvimento a página web houve alguma dificuldade inicial por não haver experiência académica com as tecnologias utilizadas, mas que foi ultrapassada pela existência de muita e boa documentação sobre a utilização destas.

5. Conclusão

O mercado tecnológico da agricultura tem muito potencial para a criação de todo o tipo de sistemas de informação, nas suas diferentes áreas. Este projeto expandiu largamente os conhecimentos sobre a agricultura, sistemas IOT e dados meteorológicos. Percebeu-se que o desenvolvimento a utilização de bibliotecas agiliza e acelera os processos de desenvolvimento. No entanto as dependências podem trazer outros problemas, como por exemplo, se no futuro s deixarem de ser suportadas ou atualizadas.

As estações meteorológicas existentes são limitadas, pouco flexíveis e encontra-se pouca informação sobre elas. Apesar disto, há alguns sistemas mais flexíveis e acessíveis a emergirem.

Este projeto é uma boa base para adicionar futuras funcionalidades, tais como, adicionar ao sistema utilizadores com autenticação, adicionar outros tipos de dados e permitir a atualização das existentes. Existem ainda aspetos que podem ser revistos e melhorados. Por exemplo, incorporar maior flexibilidade no tipo e quantidade de sensores que as estações possam ter. A interface de visualização pode certamente ser melhorada, sendo importante realizar testes com utilizadores para perceber melhor as suas necessidades.

Bibliografia

Chang, F. a. (June de 2008). Bigtable: A Distributed Storage System for Structured Data. *26*(2), p. 26. doi:10.1145/1365815.1365816

Davcev, D. a. (2018). IoT agriculture system based on LoRaWAN. *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, (pp. 1-4). doi:10.1109/WFCS.2018.8402368

Francis, J., & Peter, N. (1999). *Aspects of Precision Agriculture* (Vol. 67). (D. L. Sparks, Ed.) Academic Press. doi:10.1016/S0065-2113(08)60513-1

Sujatha, G. S. (2018). IOT Based Smart Agriculture System. *2018 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, (pp. 1-4). doi:10.1109/WiSPNET.2018.8538702