

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 realizou um sistema de recolha de dados de estações meteorológicas para uma base de dados, com o objetivo da disponibilização pela internet desses dados a utilizadores, para auxiliar decisões na produção agrícola. Este projeto envolveu o desenvolvimento de uma base de dados para o armazenamento dos dados e uma API em .Net Core para realizar a recolha e armazenamento na base de dados, e a posterior disponibilização dos dados num website desenvolvido em Bootstrap.

[Resumo do trabalho realizado. Deve ser sucinto, e cobrir todo o relatório: uma introdução ao problema que se pretendeu resolver, um pequeno resumo da abordagem realizada, e algumas conclusões do trabalho atingido.

Poderão ser criados vários parágrafos, até para que cada um corresponda às três fases de introdução, desenvolvimento e conclusão.

Não é relevante colocar no resumo o local de estágio ou a referência ao curso. Essa informação já consta da capa.]

ÍNDICE

[1. Introdução 6](#_Toc73526093)

[1.1. Objetivos 7](#_Toc73526094)

[1.2. Contexto 7](#_Toc73526095)

[1.3. Estrutura do documento 7](#_Toc73526096)

[2. Estudo e definição dos requisitos 8](#_Toc73526097)

[2.1. O que são dados meteorológicos 8](#_Toc73526098)

[2.2. Soluções existentes 8](#_Toc73526099)

[2.3. Requisitos funcionais 9](#_Toc73526100)

[2.4. Requisitos técnicos e restrições a considerar 9](#_Toc73526101)

[3. Desenvolvimento 10](#_Toc73526102)

[3.1. Arquitetura 10](#_Toc73526103)

[3.2. Principais etapas de desenvolvimento 11](#_Toc73526104)

[3.2.1. Criação da base de dados 11](#_Toc73526105)

[3.2.2. Estrutura da API 12](#_Toc73526106)

[3.2.3. Exemplo de um endpoint da API 15](#_Toc73526107)

[3.2.4. Página web 16](#_Toc73526108)

[4. Análise dos resultados 17](#_Toc73526109)

[4.1. Endpoints disponíveis na API 17](#_Toc73526110)

[4.2. Página web 18](#_Toc73526111)

[4.3. Principais dificuldades 22](#_Toc73526112)

[5. Conclusão 22](#_Toc73526113)

ÍNDICE DE FIGURAS

[Figura 1: Estação meteorológica automática. 7](#_Toc477446365)

[Figura 2: Arquitetura do sistema. 14](#_Toc477446365)

[Figura 3: Diagrama da base dados. 3](#_Toc477446365)

[Figura 4: Endpoints para o retorno de dados. 3](#_Toc477446365)

[Figura 5: Endpoints para a receção de dados. 3](#_Toc477446365)

[Figura 6: Propriedades da estação. 3](#_Toc477446365)

[Figura 7: Dados atuais da estação. 3](#_Toc477446365)

[Figura 8: Quadro de sumário de dados. 3](#_Toc477446365)

[Figura 9: Gráfico de temperatura e ponto de orvalho. 3](#_Toc477446365)

[Figura 10: Rosa dos ventos. 3](#_Toc477446365)

[Figura 11: Tabela de dados. 3](#_Toc477446365)

Glossário

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

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

Siglas e Acrónimos

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

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

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

**IOT** – Internet of Things (Internet das coisas)

**SI** – Sistema de informação.

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

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

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

# Introdução

A agricultura é uma atividade com milhares de anos, que atualmente está a cada vez a ficar 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.

*Precision agriculture is the application of technologies and principles to manage spatial and temporal variability associated with all aspects of agricultural production for the purpose of improving crop performance and environmental quality. (Francis J.Pierce, Peter Nowak, 1999).*

Há um grande crescimento de sistemas de IOT, para auxiliar na gestão agrónoma, com a *Internet of Things,* 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, abrindo novas possibilidades nesta área.

O sistema informático deste projeto tenciona auxiliar na gestão e apoio á 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 no terreno (ver figura 1).

Uma imagem com relva, céu, exterior, árvore

Descrição gerada automaticamente

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.

## Objetivos

Os objetivos deste projeto é a recolha, armazenamento e disponibilização de dados relacionados com meteorologia agronómica. Para isso é preciso receber os dados, vindos de um sistema remoto, armazená-los e servi-los pela web.

## Contexto

Este projeto foi proposto pela empresa Phosphorland, a qual oferece software de gestão agrícola, este projeto faz

## Estrutura do documento

1. Introdução, ao problema, objetivos e contexto.
2. Análise ao problema
3. Modelação e requisitos
4. Desenvolvimento
5. Conclu

# Estudo e definição dos requisitos

## 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 estações meteorológicas têm. A recolha dos dados é efetuada por estações meteorológicas automáticas que estão equipadas com vários sensores para medir vários parâmetros, esses parâmetros podem variar dependendo da estação. Os tipos de dados normalmente são todos valores numéricos representando alguma medida, tais como, temperatura, humidade relativa, radiação solar, intensidade do vento, direção do vento, precipitação, pressão, temperatura do solo a 0.5 m, a 1.5m e a 3m e a humidade do solo a 0.5 m, a 1.5m e a 3m.

## Soluções existentes

No início do projeto, iniciou-se uma pesquisa por soluções deste tipo no mercado e como elas operavam, mas na realidade poupas soluções existem e são algo limitadas no seu funcionamento. Percebeu-se que as estações meteorológicas para o envio de dados remotamente, podem utilizar várias redes como a rede GSM, LoRaWAN e Sigfox.

Quanto às estações meteorológicas existe pouca documentação sobre como é que se efetuam as ligações às mesmas através das redes mencionadas, maior parte das estações meteorológicas são pouco modulares. As redes de comunicação que estas estação usam a única com maior viabilidade para Portugal será a rede GSM pois tem uma boa cobertura em todo o país, ao contrário das redes LoRaWAN e Sigfox que são relativamente recentes e possuem pouca cobertura ainda.

Como não foi possível encontrar uma estação meteorológica que cumprisse os requisitos necessários, continuou-se com a concepção do sistema, partindo do pressuposto que os dados de algum modo serão enviados através de um endpoint na API, para serem inseridos na base de dados.

## Requisitos funcionais

Após uma pesquisa de soluções existentes no mercado, e da solução idealizada chegou-se aos seguintes requisitos:

* Gráfico com dados da temperatura e ponto de orvalho
* Gráfico com dados da humidade
* Gráfico com dados da pressão
* Gráfico com dados da precipitação
* Gráfico com dados da intensidade do vento
* Gráfico com dados da temperatura
* Rosa dos ventos com a direção e intensidade do vento
* Sumário de dados num determinado período de tempo, com o mínimo, máximo e média
* Consultar a última observação de uma estação
* Ponto para entrada de dados das estações
* Ponto para entrada de novas estações
* Visualizar uma tabela com os dados em bruto de uma estação
* Exportar dados para o formato CSV.

## Requisitos técnicos e restrições a considerar

Os requisitos técnicos requerem que o sistema opere em Linux, página que mostre os dados seja em BootStrap e devido a poder haver uma grande quantidade de ingestão de dados e os dados serem de ordem temporal a base de dados deve acomodar essas necessidades.

Nas estações meteorológicas elas podem ter diferentes unidades para as medições e diferentes sensores, por exemplo umas estações meteorológicas podem ter sensores de temperatura do solo e outras não.

# Desenvolvimento

## Arquitetura

Para este SI é necessário 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.

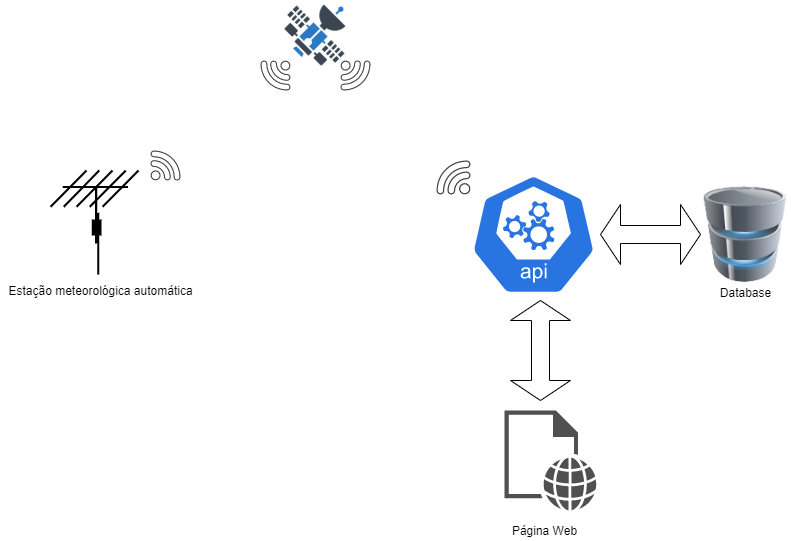


Figura 2: Arquitetura do sistema

A base dados escolhida foi uma extensão do PostgreSQL, chamada TimeScale. O TimeScale é uma base dados relacional *open source* para dados de serie temporal, oferecendo uma capacidade superior em relação á velocidade das queries do que PostgreSQL, maior capacidade de ingestão de dados e permite ser utilizada em qualquer tipo de ambiente.

O motor deste projeto é uma API em desenvolvida com a framework .NET Core 5.0, pelo facto de ser uma tecnologia familiar, consegue operar em Linux 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 a página web é limitada a ser construída com a framework BootStrap, JavaScript e JQuery. A framework BootStrap de código-fonte aberto que possibilita o desenvolvimento de páginas responsivas. O JQuery é uma libraria de funções de JavaScript que auxilia no desenvolvimento da página sendo mais simples e oferece uma maior compatibilidade com os navegadores.

## Principais etapas de desenvolvimento

### Criação da base de dados

Partindo dos requisitos e restrições, as tabelas necessárias para este projeto são a tabela station para 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 é o seguinte:

Uma imagem com mesa

Descrição gerada automaticamente

Figura 3: Diagrama de base de dados

Para a criação da base de dados, é necessário o PostgreSQL 13 e o TimeScaleDB, depois a base dados foi criada com o nome fourmeteo e adicionadas duas extensões, a extensão do TimeScaleDB e o uuid ossp que é um modulo do PostgreSQL que permite a utilização de identificadores únicos como campos de uma tabela, que serão o id das estações.

A 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 pedaços mais pequenos para aumentar performance de query e insert.

### Estrutura da API

A framework .Net Core têm ferramentas para gestão de projetos e as suas dependências, utilizando essas ferramentas para criar um projeto novo do tipo MVC. O projeto criado pela ferramenta tem 3 pastas principais, a pasta Views foi apagada, pois a parte de visualização será em BootStrap. Foi adicionado 3 pacotes para acelerar o processo e a facilidade de desenvolvimento. Os pacotes adicionados foram o EntityFramework.Design, que permite a geração do contexto de base de dados e os modelos, o EntityFramework.PostgreSQL, para efetuar ligações a bases de dados PostgeSQL e a gestão da base de dados através do EntityFramework, e o Swashbuckle.AspNetCore que têm o SwaggerUI, usado para gerar automaticamente documentação e possibilita a iteração com a API sem ter nenhuma interface desenvolvida ou logica implementada.

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

Na camada de controle encontram-se dois controladores, um deles é só para a receção dos dados e a outra disponibilização dos dados.

No ficheiro appsettings.json encontra-se a configuração da ligação á base de dados, que é demonstrada pelo seguinte código.

  "ConnectionStrings": {

    "4meteodb": "Host=192.168.1.79;Port=5432;Username=meteoapi;Password=\*\*\*\*\*\*\*;Database=fourmeteo"

  },

O ficheiro Startup.cs inicia todos os serviços no início de execução do programa. Aqui está a inicialização dos controladores, a ligação á base de dados, a definição da política CORS e o serviço do SwaggerUI.

Na função ConfigureServices, a política CORS é definida através do excerto de código.

         services.AddCors(options =>

        {

            options.AddDefaultPolicy(

                              builder =>

                              {

                                 builder.AllowAnyOrigin();

                              });

        });

A inicialização dos controladores é efetuada por este código.

services.AddControllers();

A configuração do contexto de base de dados

            services.AddDbContext<*fourmeteoContext*>(options =>

            options.UseNpgsql(Configuration.GetConnectionString("4meteodb")));

Depois o Swagger é configurado pelo seguinte codigo, aonde se configura a versão, o titulo, a descrição e os termos de serviço e todos os comentários em formato XML no código são compilados num ficheiro só 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().Name}.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 identificados 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; buildtransitive</IncludeAssets>

      <PrivateAssets>all</PrivateAssets>

    </PackageReference>

    <PackageReference Include="Swashbuckle.AspNetCore" Version="5.6.3" />

    <PackageReference Include="Npgsql.EntityFrameworkCore.PostgreSQL" Version="5.0.2" />

  </ItemGroup>

</Project>

### 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.

/// <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, int mode)

Depois é feita logica toda dos pedido e retorno dos dados.

return Ok(data);

### Página web

A página web mostra no primeiro cartão a localização e nome da estação que é a fonte dos dados apresentados abaixo, juntamente com um mapa embutido com a localização geográfica da estação, este mapa é disponibilizado pela libraria OpenLayers que permite embutir mapas dinâmicos em páginas com marcadores.

O segundo cartão mostra a última observação da estação.

O terceiro cartão, deixa escolher o modo que pode ser diário, semanal ou mensal e uma data através de um calendário de escolha de dadas. Se o modo for diário, semanal ou mensal mostra os dados 1 dia, 7 dias ou 30 dias para trás do dia selecionado, respetivamente. Esses dados selecionados, é apresentado um sumário com os mínimos, máximos e medias e depois os gráficos com a variação ao longo do tempo desses valores. Os gráficos são obtidos através de uma libraria de desenho de gráficos chamada Charts.js, a livraria é importada através do CDN. No fim existe uma rosa dos ventos que expõe a direção e intensidade do vento durante o período selecionado, esta rosa dos ventos é desenhada pela libraria Plotly, a livraria é importada através do CDN.

No fim o último cartão permite escolher todos os dados entre duas datas dessa estação e visualizá-los numa tabela ou podem ser copiados e exportados para CSV, Excel ou PDF.

# Análise dos resultados

## Endpoints disponíveis na API

Os endpoint resultantes podem ser visualizados e testados na página gerada pelo Swagger. Eles podem ser separados em duas partes os de retorno de dados (ver figura 4) e os de receção de dados (ver figura 5).

Uma imagem com texto

Descrição gerada automaticamente

Figura 4: Endpoints de retorno de dados.

Figura 5: Uma imagem com texto

Descrição gerada automaticamenteEndpoints para receção de dados.

## Página web

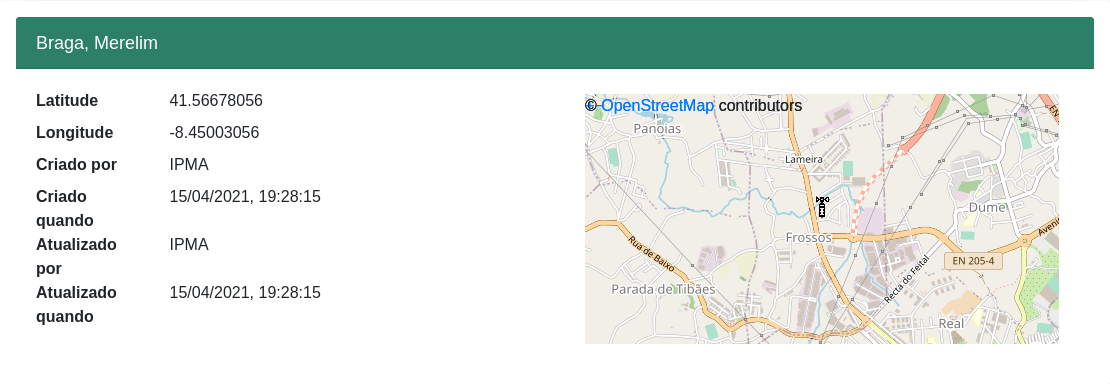
A apresentação da estação é mostrada no primeiro cartão aonde estão os dados da estação, como o nome, latitude, longitude, criador, data de criação, atualizado por e atualizando quando, e ainda um mapa com a sua localização.

Figura 6: Propriedades da estação.

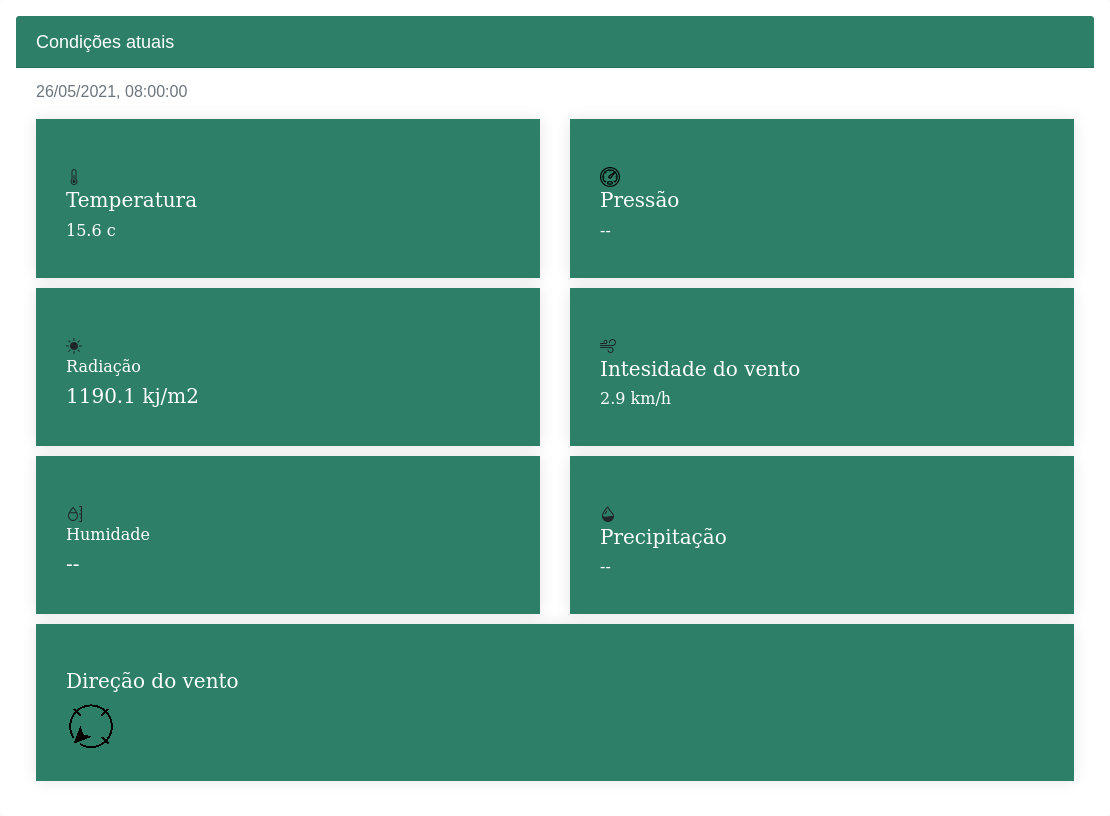
No segundo cartão está os últimos dados recebidos da estação, tais como temperatura, pressão, radiação, intensidade do vento, humidade, precipitação e direção do vento

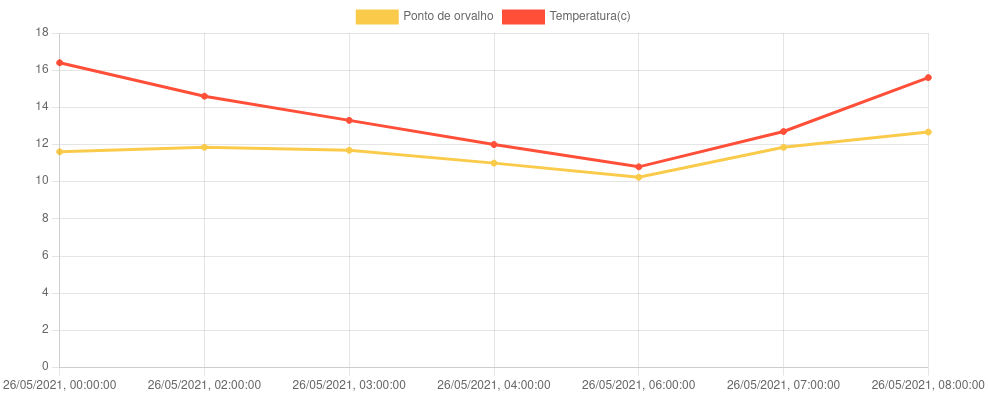
Figura 7: Dados atuais da estação.

Uma imagem com mesa

Descrição gerada automaticamenteO terceiro cartão mostra um sumário de dados baseado na data selecionada, a data selecionada por defeito é o dia atual, este sumário contém os máximos, mínimos e medias observados no período selecionado.

Figura 8: Quadro de sumário de dados.

Mais abaixo encontram-se gráficos com os dados ao longo do tempo. Os gráficos aparecem conforme existam dados ou não.

Figura 9: Gráfico de temperatura e ponto de orvalho

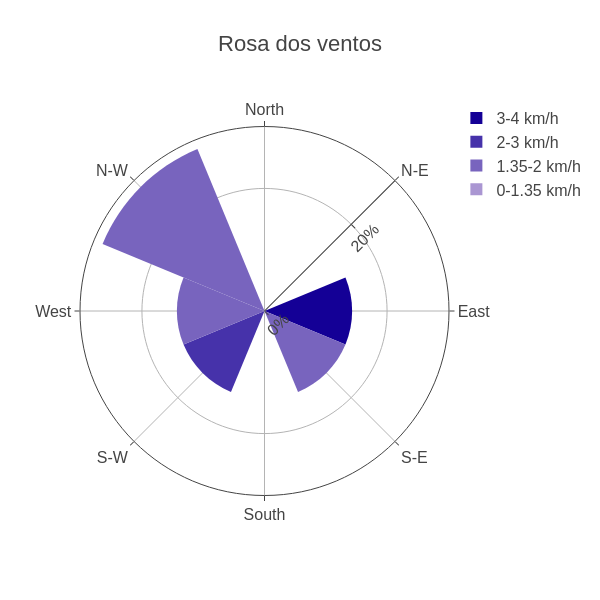
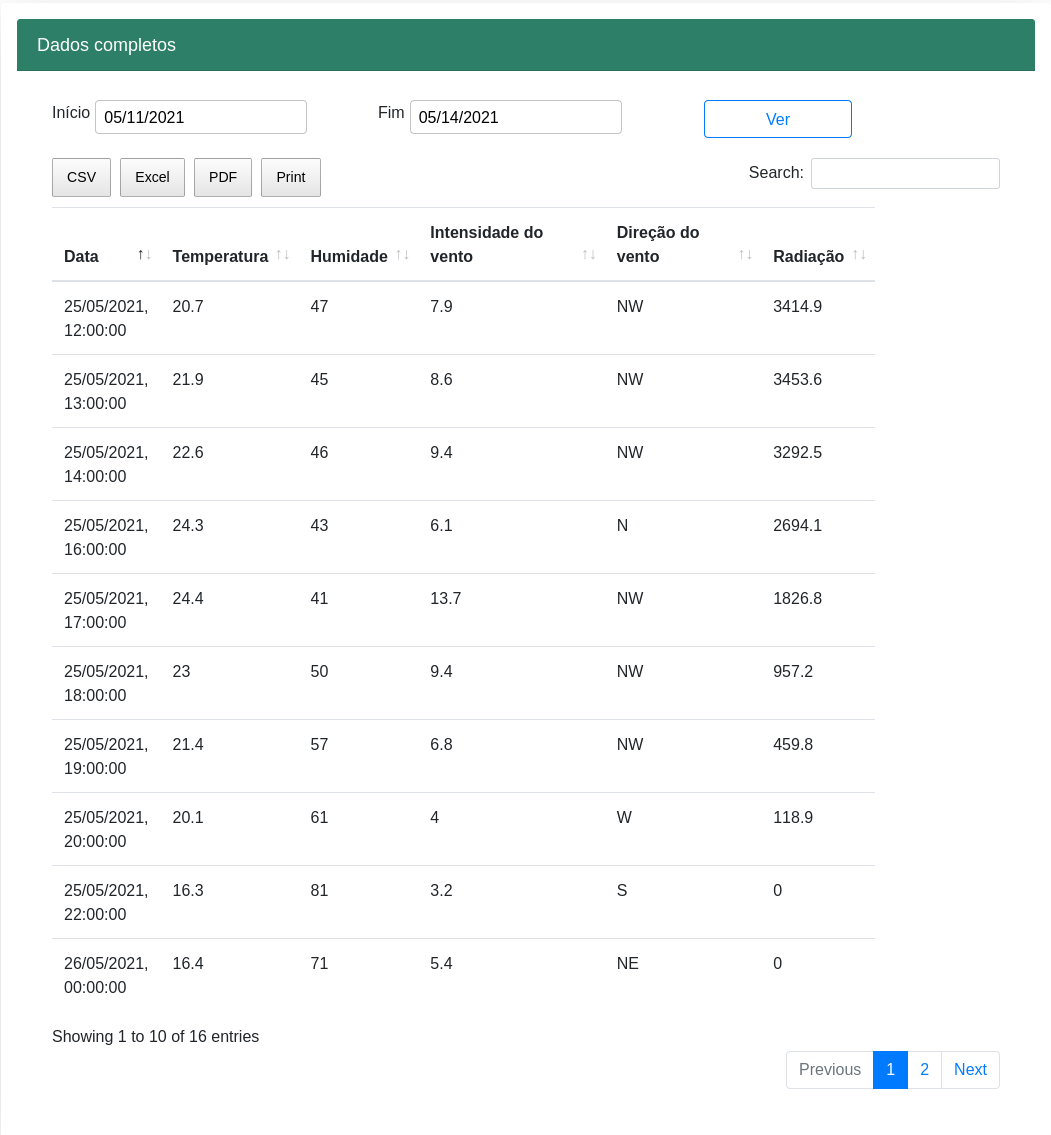
Depois no fim do cartão existe uma rosa dos ventos que mostra velocidades e direções do vento.

Figura 10: Rosa dos ventos.

No fim existe um cartão com uma tabela de dados, dados esses que podem ser escolhidos através da seleção de duas datas, esta tabela pode ser exportada para CSV, Excel, PDF ou imprimida.

Figura 11: Tabela de dados.

## Principais dificuldades

A pesquisa inicial apresentou alguns problemas por não existir grande documentação sobre os processos de recolha de dados e o funcionamento das estações meteorológicas automáticas, e o desenvolvimento da página web por falta de conhecimentos em programação web.

# Conclusão

O mercado tecnológico da agricultura tem muito potencial para a criação de todo o tipo de sistemas de informação, nas suas diferentes áreas. No desenvolvimento web a utilização de livrarias agiliza e acelera o processo de desenvolvimento, no entanto as dependências podem trazer outros problemas.

Este projeto é uma boa base para adicionar futuras funcionalidades e atualização das existentes.

Bibliografia

(Francis J.Pierce, Peter Nowak, 1999). *Aspects of Precision Agriculture.*[[1]](https://www.sciencedirect.com/science/article/pii/S0065211308605131)