Universidade do Minho
Escola de Engenharia
Departamento de Informática

Paulo Filipe Moreira Barros

Smart Irrigation System
Otimização do sistema de rega em espaços verdes

February 2022

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Paulo Filipe Moreira Barros

Smart Irrigation System
Otimização do sistema de rega em espaços verdes

Master dissertation
Master's in Informatics Engineering

Dissertation supervised by
Professor Doutor Paulo Manuel Martins de Carvalho
Professora Doutora Maria Solange Pires Ferreira Rito Lima

February 2022

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

LICENÇA CONCEDIDA AOS UTILIZADORES DESTE TRABALHO:

CC BY

https://creativecommons.org/licenses/by/4.0/

AGRADECIMENTOS

Este trabalho só foi possível graças às pessoas que sempre estiveram ao meu lado ao longo de todo o processo de desenvolvimento desta dissertação.

Em primeiro lugar gostaria de agradecer aos meus orientadores, o Professor Doutor Paulo Manuel Martins de Carvalho e a Professora Doutora Maria Solange Pires Ferreira Rito Lima que foram incansáveis ao longo de todo este processo, sempre estiveram disponíveis para me orientar e ajudar em tudo o que precisei, contribuindo sempre com criticas muito construtivas permitindo assim o sucesso no desenvolvimento deste trabalho.

Em segundo lugar gostaria de agradecer ao Simão Gonçalves, não só pela ajuda no processo da escrita da dissertação mas também por me ter motivado ao longo destes meses.

Em terceiro lugar gostaria de agradecer a doutora Magda Lopes por todas as chamadas telefónicas com o intuito de perceber em que ponto estava o processo de escrita desta dissertação, colocando um pouco de pressão de forma a que estivesse sempre dentro dos prazos.

Finalmente, gostaria de agradecer a minha família por todo o apoio, suporte, por sempre terem acreditado em mim e por terem sido sempre uma fonte de inspiração de esforço e dedicação.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho

ABSTRACT

An optimization of irrigation systems and better management of green spaces is essential nowadays, as one of our main resources, water, is often wasted and the soil does not contain the necessary nutrients, which can lead to death of vegetation. This work presents a solution where, using a set of public APIs through which the environment data is collected, it is possible to intelligently and autonomously activate or deactivate the irrigation system, taking into account a group of previously defined metrics. The system is also prepared to receive real data from sensors implemented in the field. A web application is also developed so that these data are presented in a clear and intuitive way, in order to support decision-making by the owner of a particular land. Finally, a machine learning algorithm was created that, based on the history of rain occurrence, tries to predict the occurrence of precipitation for a particular day, thus contributing for a more efficient solution.

KEYWORDS APIs, Web Application, Irrigation System, Water, Machine Learning

RESUMO

Uma otimização dos sistemas de rega e uma melhor gestão dos espaços verdes é imprescindível nos dias de hoje, pois um dos nossos principais recursos, a água, é muitas vezes desperdiçado e o solo acaba por não conter os nutrientes necessários o que pode levar à morte da vegetação. Neste trabalho é apresentada uma solução onde, usando um conjunto de APIs públicas através das quais é feita a recolha de dados do ambiente, é possível ativar ou desativar de forma inteligente e autónoma o sistema de rega tendo em consideração um grupo de métricas previamente definidas. O sistema está ainda preparado para receber dados reais de sensores implementados no terreno. É também desenvolvida uma aplicação web para que estes dados sejam apresentados de uma forma clara e intuitiva com o objetivo de suportar a tomada de decisão por parte do proprietário de um determinado terreno. Por fim foi criado um algoritmo de machine learning que, tendo como base o histórico de ocorrência de chuva tenta prever a ocorrência de precipitação para um determinado dia, por forma a tornar o sistema desenvolvido mais eficiente.

PALAVRAS-CHAVE APIs, Aplicação Web, Sistema de Rega, Água, Machine Learning

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE LISTAGENS

1

INTRODUÇÃO

1.1 CONTEXTO

A tecnologia pode ser definida como um conjunto de técnicas que permitem a aplicação prática do conhecimento e da realização de tarefas passíveis de serem de automatizadas. [Martyusheva O. 2014] Nos dias de hoje podemos constatar que um grande volume de tecnologia é criado com o intuito de ajudar as pessoas no dia a dia, tornando as tarefas rotineiras mais acessíveis e menos trabalhosas. Esta melhoria na qualidade de vida fez com que grande parte da população das áreas rurais se mudasse para as áreas urbanas. Essa migração cria nas cidades uma necessidade de evolução que, por sua vez, leva as pessoas a usarem mais tecnologia para as ajudar em áreas como segurança e sustentabilidade.

Surge assim o conceito de Smart Cities, designação dada a uma cidade que incorpora tecnologias de informação e comunicação para melhorar a qualidade e o desempenho dos serviços urbanos, como energia e transportes, com o objetivo de minimizar o consumo de recursos, desperdícios e custos globais. O principal objetivo de uma cidade inteligente é melhorar a qualidade de vida do cidadão por meio de tecnologia inteligente. [Techopedia 2015]

As Smart Cities têm um outro objetivo: solucionar os problemas que ocorrem nas grandes cidades devido à grande utilização dos serviços prestados, gerando dois tipos de problemas a ter em mente: Eficiência de Transporte e Ecologia. [Moffitt S. 2018]

1.2 PROBLEMA

Neste projeto é abordado um problema de gestão ecológica, principalmente irrigação em locais públicos ou locais agrícolas. [Unesco 2015] Segundo a Associação Empresarial de Portugal (AEP), "As atividades de abastecimento de água às populações e de saneamento de águas residuais urbanas constituem serviços de interesse geral, que visam a prossecução do interesse público, essenciais ao bem-estar dos cidadãos (...)" Diario Oficial República Portuguesa [2009]. Há muito desperdício de água produzido pelos sistemas de irrigação existentes hoje em dia. Ocasionalmente, esses sistemas funcionam em condições aquém das preferidas, ou seja, quando o solo está muito quente, muito húmido ou quando a temperatura é demasiado alta ou excessivamente baixa, o que pode provocar a morte da vegetação. Para além disso, com o passar dos anos a tendência é que as infraestruturas de fornecimento de água se tornem cada vez mais antigas e deterioradas, havendo por isso uma

redução da eficácia no que concerne ao fornecimento de água à população geral, o que resulta num aumento do desperdício de água e, por consequência, uma diminuição de receitas e um aumento dos problemas de escassez de água, [Martyusheva O. 2014] Assim, é necessário um melhor controlo desses sistemas, uma vez que a água é um recurso natural essencial para a existência dos seres vivos em geral e para o ser humano em particular. Segundo o relatório sobre o desenvolvimento dos recursos hídricos no mundo apresentado pela Organização das Nações Unidas (ONU) no 8º fórum Mundial da água, até 2050, 5 biliões de pessoas não terão acesso a água potável, o que corresponde praticamente a metade da população mundial. Assim sendo, à medida que se esgota este recurso, a tendência é que o seu valor aumente, aumentando assim a possibilidade da existência de conflitos e tensão em torno deste setor, potenciando ainda a discrepância no que toca ao acesso a este recurso entre os países mais desenvolvidos e os menos desenvolvidos. [Bigas H. 2011] Neste sentido, e tendo em conta a importância deste recurso, urge a investigação de exploração de técnicas que permitem um aumento da eficácia no que toca a utilização de água.

1.3 OBJETIVOS

Face ao problema mencionado, na presente dissertação é proposta uma solução onde, recorrendo a um conjunto de APIs públicas, é feita a recolha de dados essenciais para a tomada de decisão de (des)ativação do sistema de rega de forma inteligente, implementando um protótipo denominado Smart Irrigation System (SIS). Este sistema irá permitir que a ativação do sistema de rega seja feita tendo em conta um conjunto de parâmetros que se considerem ideais, tendo por base a recolha de dados feita por meio de estações meteorológicas ou de sensores reais. Após essa recolha de dados, os mesmos são tratados antes de serem armazenados na base de dados do sistema, de forma a haver uma normalização dos dados.

Para além da recolha dos dados, é ainda criada uma aplicação web com o objetivo de representar os dados de cada sensor ou estação meteorológica para que seja possível fazer de forma clara e intuitiva, um acompanhamento dos recursos hídricos gastos num determinado terreno, permitindo uma monitorização inteligível.

O principal objetivo desta solução é fornecer uma plataforma de suporte à tomada de decisão de forma a permitir uma diminuição substancial na quantidade de água desperdiçada pelos sistemas de rega, bem como a monitorização dos diferentes parâmetros meteorológicos nos locais presentes no sistema. Desta forma pretende-se que o sistema tenha autonomia suficiente para a tomada de decisão de ativação do sistema de rega de forma automática, mas também permitir que a mesma decisão possa ser tomada de forma manual pelo responsável por um determinado terreno. Pretende-se reunir dados e aumentar o conhecimento relativo ao processo de monitorização dos sistemas de rega .

Um outro objetivo importante prende-se com a possibilidade de deteção de erros nas leituras que poderão ocorrer devido a problemas relacionados com hardware responsável por fazer a recolha dos parâmetros meteorológicos no terreno.

Este protótipo pode ser aplicado ao nível da agricultura bem como ao nível de pequenos jardins domésticos, de forma a permitir uma poupança de água. No caso da agricultura, este protótipo dispõe ainda de um con-

junto de técnicas de *machine learning*, de forma a permitir, com recurso ao histórico de dados meteorológicos recolhidos, prever a possibilidade de existência de chuva no dia seguinte com uma precisão bastante elevada.

1.4 ESTRUTURA DA DISSERTAÇÃO

Para além deste capítulo inicial, este documento é constituído por mais 7 capítulos: Contextualização teórica (Capítulo 2); análise e planeamento do sistema (Capítulo 3); modelo do sistema de rega inteligente (Capítulo 4); conceção do sistema (Capítulo 5); resultados obtidos (Capítulo 6); e finalmente as conclusões e trabalho futuro.

No segundo capítulo é abordada uma breve contextualização teórica, bem como a definição de um conjunto de conceitos que se revelarão de extrema importância para a compreensão do caso de estudo. Neste capítulo é ainda exibido um conjunto de estudos e artigos no âmbito deste projeto e de que modo a solução apresentada neste projeto pode-se distanciar das demais presentes no mercado .

O terceiro capítulo incide sobretudo sobre a análise das funcionalidades do sistema e os comportamentos que devem ser inerentes à solução apresentada.

No capítulo seguinte, é representada a arquitetura da solução de forma genérica, é feita uma caracterização dos módulos e de que forma estes interagem entre si. Este capítulo também é orientado para a aplicação web, desde a modulação até à sua arquitetura. No quinto capítulo são especificadas as ferramentas utilizadas, bem como é a presentada a justificação para a utilização de cada uma das ferramentas. Neste capítulo é ainda apresentada a conceção prática do sistema e este capítulo está dividido em 3 subcapítulos, referentes a cada um dos módulos do sistema (Automatic service, Aplicação web e Algoritmo de Machine Learning).

No Capítulo 6 são apresentados os resultados obtidos e é feita uma breve reflexão sobre os mesmos e, finalmente, no sétimo capítulo é apresentada uma conclusão e trabalho futuro.

2

CONTEXTUALIZAÇÃO TEÓRICA

Para que haja uma melhor compreensão da problemática a que este trabalho se propõe resolver, é fundamental que haja um entendimento prévio de alguns conceitos. Neste capítulo são clarificados alguns tópicos que serão de suma importância para que haja uma melhor compreensão da temática. No final do capitulo são apresentadas um conjunto de soluções já existentes no mercado e de que forma a solução apresentada neste trabalho se pode destacar das demais existentes.

2.1 DEFINIÇÃO DE CONCEITOS

2.1.1 *Smart cities*

De acordo com o United Nations World Urbanization Prospects em 1950, 30% da população mundial vivia em zonas urbanas. Este número tem vindo a crescer de forma muito acentuada e é expectável que em 2050 este número cresça para um valor a rondar os 68%. O crescimento urbano está normalmente associado ao crescimento social, económico e ambiental [Caragliu A. et al. 2011]. Tendo em conta este mesmo crescimento urbano é fundamental desenvolver e estabelecer mecanismos que tenham em consideração não só o crescimento da população como também as suas necessidades [Fernandes S. 2017]. [Silva J. et al.2012] deixa bem claro a distinção entre Smart Cities e Intelligent Cities: - enquanto as Smart Cities incidem sobretudo em embebed systems, sensores e conteúdo interativo, no caso das intelligent cities verifica-se uma maior inteligência cooperativa / colaborativa bem como sistemas inovadores e espaços mais vocacionados para a web [Steventon A. e Wight S. 2006]. [Caragliu et al. 2011] utiliza a seguinte definição para uma smart city: “Uma cidade é inteligente quando se investe em capital humano, social e tradicional (transporte) e em infraestruturas de comunicação moderna, que impulsionam um sustentável desenvolvimento económico e alta qualidade de vida, com uma gestão sábia dos recursos naturais através de uma gestão participativa”.

2.1.2 *Sistemas de Informação*

Nos dias de hoje, e tendo em conta o crescimento alucinante da informação diariamente, é cada vez mais importante o estudo dos sistemas de informação, uma vez que o processamento da informação tornou-se imprescindível para o desenvolvimento das empresas.[Lucas H. 1990] Antes de obtenção da informação em si,

é necessário haver uma recolha de dados. Os dados são factos isolados que por si só não aumentam o conhecimento de quem os obtém, mas a sua utilização poderá tornar-se pertinente a um determinado momento [Varajão J. 1998], sendo a informação o resultado da interpretação desses mesmos dados [Wigan M. 1992], tornando-se, por isso, fundamental que se estabeleça relação entre eles. Assim a informação pode ser definida como um grupo de dados sobre um determinado contexto útil e com significado, acrescentando um valor real e fornecendo um suporte nas ações ou decisões de quem os utiliza [Reis C. 1993]. Surge assim o conceito de conhecimento como uma combinação de instintos, ideias, regras e procedimentos que fornecem um suporte na tomada de ações ou decisões [Rascão J. 2001]. Segundo [Rosnay J. 1977] um sistema, por sua vez, pode ser definido como um conjunto de elementos que interagem dinamicamente entre si em função de um determinado objetivo. Como tal um sistema de informação de uma organização deve possuir sempre uma etapa onde é feita uma recolha de dados, seguidamente deve-se proceder ao armazenamento e processamento desses mesmos dados para que se obtenha uma informação eficaz, aumentando assim o conhecimento de quem tem acesso a essa informação.

2.1.3 *Inteligência artificial*

Segundo a Association for the Advancement for Artificial Intelligence (AAAI), a inteligência artificial pode ser definida como uma "Compreensão científica dos mecanismos que são fundamentais ao pensamento e ao comportamento inteligente e respetiva implementação nas máquinas". Inteligência artificial pode ainda ser definida como a capacidade de uma máquina executar determinadas ações geralmente associadas a mentes humanas tais como a aprendizagem e a resolução de problemas [Chui M. e Malhotra S. 2018].

2.1.4 *Machine Learning*

Machine Learning é um ramo da inteligência artificial onde, com recurso a algoritmos e técnicas matemáticas, é possível que os computadores obtenham conhecimento de uma forma automática [Murphy K. 2013]. Estes algoritmos têm como objetivo sintetizar a informação e analisá-la de forma a obter conhecimento da mesma. A inteligência artificial tem uma maior facilidade da resolução de problemas que podem ser descritos matematicamente, mas existe muita dificuldade no que toca a problemas reais do dia-a-dia, que dificilmente podem ser descritos formalmente e que nós humanos conseguimos intuitivamente resolver, tais como o reconhecimento de imagem ou de voz. Para tal, e à semelhança do que acontece com o conhecimento humano, é necessário fornecer um conjunto de experiências para permitir que o computador aprenda e se adapte a novas circunstâncias através de uma abordagem estatística [Goodfellow I. et al. 2016]. Os algoritmos de *machine learning* são usados essencialmente em duas tarefas: classificação e regressão [Jordan M. et al. 2016]. No contexto desta dissertação é utilizado classificação com o intuído de prever a ocorrência de chuva. Independentemente do tipo de problema, são considerados 3 tipos de abordagens:

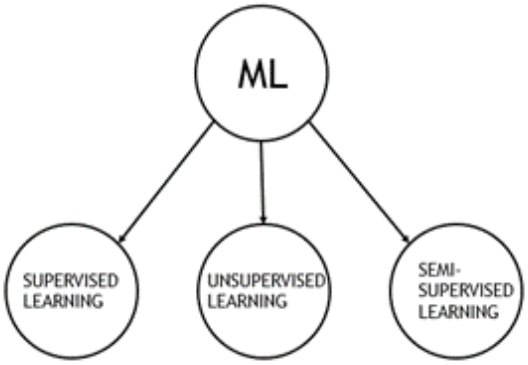


Figura 1: Tipos de Abordagem ML
Fonte: Adaptado de TowardsDataScicence.com 2018

- Supervised Learning: O modelo aprende a partir de dados previamente conhecidos de *input* e de *output*. Este é a forma mais habitual de aprendizagem, onde é produzido um *output* computacionalmente e é comparado com o *output* esperado, e o sistema é constantemente adaptado de forma a estar cada vez mais próximo do output esperado.
- Unsupervised Learning: Neste caso o modelo aprende apenas com os dados de entrada. Esta abordagem é muito útil em contexto prático uma vez que na maioria dos casos os dados não estão previamente rotulados. Esta técnica é utilizada essencialmente para estudar a existência de aglomerados de dados como, por exemplo, a existência de clusters.
- Semi-Supervised Learning: Este caso funciona como uma conjugação dos modelos anteriores. Aqui são utilizados ambos os tipos de dados para treinar o modelo (supervisionados e não supervisionados). Neste caso o modelo é treinado com dados supervisionados e em seguida aprimorado com dados não supervisionados ou vice-versa. É possível treinar o modelo recorrendo aos dois tipos de dados ao mesmo tempo, mas esta prática é menos frequente.

2.1.5 *Árvores de decisão (Decision Trees)*

Um dos objetivos subjacentes a este trabalho consiste na utilização de modelos preditivos. Um desses modelos é a árvore de decisão. As árvores de decisão (DT) são das técnicas de *data mining* mais utilizadas devido à sua simplicidade de implementação e compreensão do resultado final [Quinlan J. 1986]. As árvores de decisão representam os dados em forma de árvore, onde o seguimento de cada um dos nós é feito de forma recursiva. Numa DT cada folha (nó final) representa uma classe, um nó intermédio representa um teste e cada resultado do teste origina por si uma nova subárvore [Quinlan J. 1987]. O processo de classificação é feito desde a raiz até a uma folha da árvore, sendo esse item classificado consoante a classe representada por essa folha.

Figura 2: Exemplo de árvore de decisão
Fonte: Técnico Lisboa 2019

No exemplo da Figura 2 é representada uma árvore de decisão bastante rudimentar e não modela um problema real. As árvores podem atingir níveis de complexidade bastante elevados quando representam um elevado número de variáveis ou classes. No exemplo da Figura 2 é modelada a decisão de "Vou para a praia?" mediante dois atributos desse dia. Assim sendo as classes possíveis são o conjunto Vou para a praia, Não vou para a praia e os atributos são o conjunto Vento, Sol. O processo de classificação/decisão inicia-se sempre pela raiz onde é a analisada a existência de sol, sendo possível apenas dois valores, "Sim" ou "Não", sendo que no caso de não estar sol, o caso em análise é imediatamente classificado como "Não vou para a praia". Por outro lado, se estiver sol, então o processo de classificação avança para a subárvore seguinte, na qual é verificada a existência de vento. Em caso afirmativo é classificado como "Não vou para a praia", no entanto em caso negativo, é classificado como "Vou para a praia". As árvores de decisão constituem um método de aprendizagem supervisionada (*supervised learning*) uma vez que utiliza um *dataset* de treino como forma de aprendizagem, ou seja, em alguns dos casos a variável de output já é conhecida e é utilizada na aprendizagem. As árvores de decisão podem ser divididas em dois grandes tipos:

- árvores de classificação;
- árvores de decisão.

As árvores de decisão são utilizadas em problemas onde a variável de output é categórica como por exemplo "chuva", "não chuva", já as árvores de regressão são utilizadas em problemas de regressão onde a variável de output é um valor real como por exemplo "euros", ou "peso".

Vantagens da utilização de árvores de decisão:

As árvores de decisão têm um conjunto de vantagens relativamente a outros métodos de classificação Lewis R. [2000]:

- fácil compreensão e interpretação de resultados;

- o resultado obtido é facilmente comprovado através de operações lógicas conforme exemplificado na Figura 2;
- boa performance na análise de grandes volumes de dados;
- pode ser utilizada não só em dados numéricos, mas também em dados categóricos.

Desvantagens da utilização das árvores de decisão:

- se os inputs fornecidos para treino forem de baixa qualidade, existe uma maior propensão para a existência de *overfitting* [Papagelis A. e Kalles D. 2001], ou seja, nos dados de treino o modelo tem muito bom desempenho, porém quando utilizados os dados de teste o desempenho é desastroso, uma vez que neste caso o modelo não aprendeu com os dados de treino, mas sim tenta adivinhar o que deveria ser feito;
- a decisão é tomada ao nível do nó que está sob processamento, não havendo assim a garantia que a árvore final seja ótima [Hyafil L. e Rivest R. 1976] [Murthy S. 1998];
- sensibilidade a pequenas perturbações no conjunto de treino.

2.1.6 *Random Forest Classifier*

As *random forest classifiers* como o próprio nome indica, consistem num conjunto de árvores de decisão que operam em conjunto. Cada árvore na *Random Forest* calcula uma previsão para a classe em estudo, e a classe com mais votos torna-se a previsão do modelo. As *Random Forests* foram introduzidas por [Breiman L. 2001] e são uma extensão da ideia previamente definida por [Amit Y. e Geman D. 1997]. O *random forest classifier* baseia-se num princípio muito simples que é a sabedoria das multidões, isto significa que um grande número de modelos não correlacionados (cada uma das árvores) opera de forma cooperativa e como tal superará qualquer um dos modelos constituintes que atua de forma individual, isto acontece porque cada uma das árvores protege as restantes de forma a evitar a existência de erros individuais (desde que elas não errem constantemente na mesma direção) uma vez que enquanto algumas árvores poderão estar erradas, um conjunto de outras estarão certas movendo as erradas na direção correta. Uma diferença muito importante relativamente às DT é que as *Random Forest* evitam *overfitting* devido ao facto de serem utilizados subárvores aleatórias de menores dimensões. Um problema inerente a esta abordagem é o facto de ser necessário um maior tempo e poder computacional, dependendo do número de árvores que constituem a *random forest*.

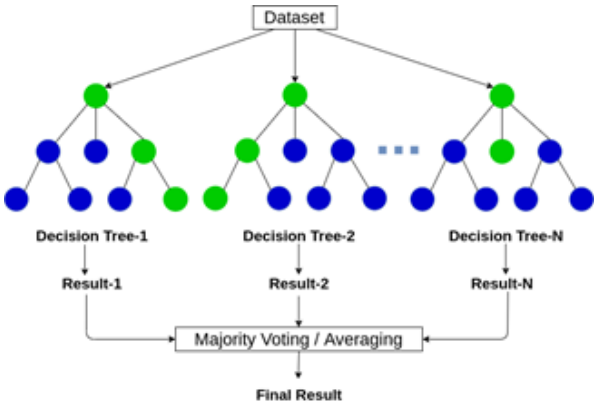


Figura 3: Exemplificação de funcionamento das Random Forest
Fonte : ResearchGate 2016

Vantagens da utilização das random forests:

- pode ser utilizado tanto para regressão como para classificação;
- é um algoritmo de fácil utilização uma vez que os hiperparâmetros com os valores *default* geralmente produzem bons resultados;
- dificilmente ocorrerá *overfitting*.

Desvantagens da utilização das random forests:

- grande necessidade de poder computacional dependendo do número de árvores;

- ineficiente em previsões em tempo real.

2.1.7 *Logistic Regression*

Muitas das vezes modelos utilizados para explorar a relação de uma ou várias variáveis independentes e uma variável dependente, são os modelos de regressão. Neste caso a regressão logística pode ser definida como um método de classificação supervisionado e trata-se de um caso particular dos modelos lineares generalizados [Mccullagh P. e Nelder J. 1989]. Os modelos de regressão logística são particularmente utilizados quando a variável de resposta (dependente) apresenta duas categorias, como por exemplo, no contexto deste projeto, a análise da existência de chuva, pode ser vista como uma variável dicotomizada (Chuva, Não Chuva). A regressão logística não requer que todas as variáveis sejam completamente independentes para que seja linearmente relacionado, nem é necessário que ocorra uma variância igual dentro de cada grupo, o que torna o procedimento menos rigoroso na análise estatística [Starkweather J. e Moske A. 2011].

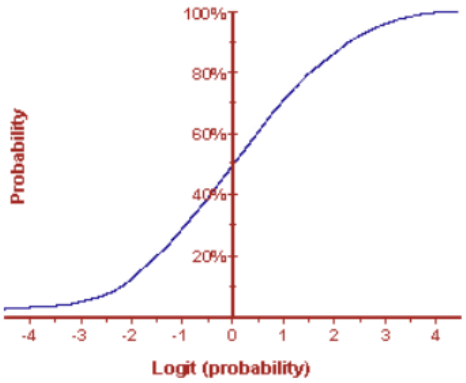


Figura 4: Curva logística
Fonte: Adaptado de Sharma, S. 1996

Vantagens do logistic regression:

- alto grau de confiabilidade;
- facilidade de classificação dos elementos em categorias;
- facilidade de lidar com variáveis independentes categóricas.

Desvantagens do logistic regression:

- necessidade de utilização de métodos numéricos para obtenção da solução de eficácia;
- se o número de observações for inferior ao número de *features* existe uma forte possibilidade de ocorrer *overfitting*.

2.2 TRABALHO RELACIONADO

Nesta secção são identificados os produtos que de alguma forma poderiam fornecer uma solução para o problema identificado. Nas pesquisas realizadas online foram utilizados motores de busca generalistas e a base de dados do Gabinete Europeu de Patentes [Branchi P. et al. 2014]. A maioria destes sistemas utilizam microcontroladores como Arduino ou Rasberry Pi, combinados com um conjunto de sensores no terreno.

Existem poucas soluções que se comparam à apresentada neste projeto, mas todas elas têm especificidades que as tornam de certa forma distintas ao protótipo apresentado, como o Smart Garden Hub da GreenIQ [2021], Rachio 3 da [Rachio 2020]. O problema com estas soluções prende-se com o facto de serem caras, proprietárias e não são escaláveis, uma vez que foram projetadas para quintais e pequenos jardins e todas elas apenas têm em consideração valores meteorológicos no momento de ativação do sistema de rega.

Existe um projeto chamado “Smart Home Garden Irrigation System with Raspberry Pi” [Ishak S. 2008] que, através de uma aplicação Android e de um Raspberry Pi, possibilita, ao proprietário de um determinado terreno, controlar o fluxo de água do sistema baseado em sensores de humidade do solo e de chuva. [Harishankar S. et al. 2014] implementaram um sistema que consiste em uma bomba de água movida a energia solar juntamente com um controlo automático de fluxo de água, usando um sensor de humidade. Um projeto bastante interessante é o Real-Time Monitoring Of Agricultural Activities Using Wireless Sensor Network [Saleemmaleekh A. e Sudhakar K. 2015], em que são utilizados uma rede de sensores no sector agrícola indiano para a obtenção de valores de humidade do ar, humidade do solo e ph e, mediante estes valores, é enviada uma SMS ao agricultor para ativar ou não o sistema de rega. A grande desvantagem deste projeto é o facto da ativação do sistema de rega não ser feito de forma automática, necessitando ainda da intervenção humana para que o mesmo aconteça.

Em Portugal já existe um sistema de rega implementado em Castelo Branco, que funciona da seguinte forma: - foram instalados vários pluviómetros em locais específicos que, ao detetar chuva, desligam automaticamente a rega. Além disso, o sistema permite acionar o sistema de irrigação remotamente e emitir alertas em caso de mau funcionamento do equipamento. [Allbesmart 2021]

2.3 CARACTERÍSTICAS INOVADORAS DO PROTÓTIPO A IMPLEMENTAR

Embora essas soluções tenham o seu lugar e sejam bem pensadas, carecem de recursos importantes, nomeadamente na sua maioria são soluções estanques que não permitem adaptar-se às necessidades do cliente em específico. Por exemplo as necessidades de um utilizador que possua um pequeno quintal, não serão exatamente as mesmas de um utilizador que possua uma quinta com vários hectares. Assim sendo, considerou-se fundamental o desenvolvimento de uma solução altamente modular que fosse capaz de atender as necessidades especificas de cada cliente. A título de exemplo, no protótipo apresentado, o cliente poderá não só por optar pela existência ou não de sensores reais, bem como escolher que sensores pretende que sejam implementados mediante as suas necessidades. Outro fator diferenciador é o facto da solução apresentada não se basear apenas na obtenção de dados no presente, mas também ter em consideração o histórico de leituras, bem como as previsões meteorológicas, o que permitirá uma tomada de decisão muito mais eficaz e ponderada.

2.4 RESUMO DO CAPÍTULO

Este capitulo teve sobretudo como objetivo definir alguns conceitos que serão importantes para o entendimento da solução apresentada para a otimização dos sistemas de rega em espaços verdes. Neste capitulo, foi explicado de que forma os algoritmos de *machine learning* aplicados neste trabalho atuam. Finalmente foram apresentadas algumas solução já existentes no mercado, as suas vantagens e desvantagens e de que forma o SIS (Smart irrigation system) se pode destacar das demais.

3

ANÁLISE E PLANEAMENTO DA SOLUÇÃO

Neste capítulo é apresentado o conjunto de etapas de análise e planeamento da solução de forma a garantir que a solução apresentada se propõe a agir e a solucionar os problemas existentes atualmente, bem como fornecer uma ferramenta relevante de apoio à tomada de decisão. Para isso é necessário ter uma metodologia de desenvolvimento de software que seja rigorosa, concisa e que atenda as necessidades do mercado Kerzner H. [2001]. Desta forma é fundamental que o processo de desenvolvimento de software seja detalhado, planeado e disciplinado e, para tal, é indispensável que sejam utilizadas as melhores práticas de engenharia de software e de conceção de sistemas de informação.

3.1 REQUISITOS FUNCIONAIS DO SISTEMA

Na primeira etapa, antes ainda do desenvolvimento do software, é necessário especificar corretamente o que sistema deverá fazer de forma a poupar tempo e dinheiro à posteriori.

Perspetivando-se que o sistema principal será subdividido em três subsistemas que serão definidos de forma mais detalhada no próximo capítulo, considerou-se que faria todo o sentido dividir os requisitos do Smart Irrigation System (SIS) nos três subsistemas de forma a permitir uma melhor organização dos mesmos.

Seguidamente apresentam-se os requisitos de alto nível levantados e subdivididos pelos seguintes subsistemas:

- requisitos da aplicação web;
- requisitos do *Automatic service*
- requisitos do algoritmo de *Machine learning*.

Neste caso optou-se por não colocar uma descrição detalhada de cada um dos requisitos de forma a não prolongar demasiado esta secção. No entanto após cada bloco de requisitos é incluído uma breve descrição.

3.1.1 *Aplicação web*

Este módulo do sistema é aquele em que o utilizador terá contacto direto e todas as funcionalidades relacionadas a ele terão sempre como finalidade fornecer ou obter informação por parte do utilizador. Neste sentido, este módulo terá sobretudo uma componente de apresentação de dados, não só relativa aos nós em si, mas também aos valores que estes recolhem no terreno. Neste caso, a informação pode ser no sentido Utilizador -> Sistema, no caso do registo de um novo nó e na ordem manual de ativação/desativação do sistema de rega, mas será sobretudo no sentido Sistema -> Utilizador. Este último caso será o que ocorrerá mais frequentemente uma vez que o principal intuito da aplicação web é fornecer, de forma clara e intuitiva, uma representação das métricas e dos valores recolhidos no terreno, de forma a ser um suporte a tomada de decisão por parte do utilizador.

ID	**Requisito do Sistema**
W1	Deverá ter uma área reservada para login.
W2	Deverá permitir a criação de um nó.
W3	Deverá permitir associar um conjunto de sensores a um nó
W4	Deverá ser capaz de encontrar a estação meteorológica mais próxima no caso do nó não possuir sensores reais.
W5	Deverá permitir apresentar na interface todos os nós ativos associados ao utilizador.
W6	Deverá permitir uma apresentação gráfica de todas as métricas (temperatura, humidade...).
W7	Deverá permitir ativar/desativar manualmente o sistema de rega bem como os sensores, câmaras de filmar e luzes.
W8	Deverá apresentar todos os erros registados pelo Automatic Service.
W9	O sistema deverá permitir marcar um erro como 'Resolvido'.
W10	O sistema deverá ser capaz de fazer *geocoding* bem como *reverse geocoding*.
W11	O sistema deverá ser capaz de permitir a inserção de imagens da vegetação de forma a permitir "alimentar" o algoritmo de *machine learning*.

Tabela 1: Requisitos funcionais da aplicação web

3.1.2 *Automatic Service*

Este módulo representa um ponto fundamental no sistema e é aqui que estão todos os automatismos do sistema. Fundamentalmente é aqui que reside o *core* do mesmo. Este módulo é executado de forma periódica e a única interação humana que tem é na declaração dos parâmetros que serão fundamentais para a definição de um determinado terreno, com base no tipo de vegetação (evapotranspiração) e precipitação (Capítulo 4.3).

ID	Requisito do Sistema
S1	Definir uma periocidade com o qual se pretende que seja ativado automaticamente.
S2	Ser automaticamente ativado tendo em consideração uma periocidade definida.
S3	Ser capaz de obter todos os nós ativos.
S4	Recolher de forma autónoma os dados meteorológicos das últimas 24 horas para todos os nós ativos.
S5	Ser capaz de obter a hora de pôr de sol e de nascer do sol para um determinado local.
S6	Verificar automaticamente se a hora atual é a ideal para a rega.
S7	Verificar a previsão meteorológica para o presente dia.
S8	Verificar a previsão meteorológica para o dia seguinte.
S9	Calcular a média de temperatura das últimas 24 horas para um determinado local.
S10	Verificar se a temperatura é a ideal para a rega.
S11	Obter a precipitação diária do local onde está presente um determinado nó.
S12	Obter a evapotranspiração da vegetação para o concelho onde está localizado determinado nó.
S13	Calcular o tempo de rega baseado nas métricas definidas.
S14	Calcular a moda e a média das leituras para uma determinada métrica.
S15	Verificar a existência de outliers na leitura.
S16	Na presença de *outliers*, o sistema deverá ser capaz de automaticamente notificar o utilizador via email.
S17	Para todos os nós ativos, o sistema deverá periodicamente manter todas as tabelas de leituras atualizadas.
S18	Relativamente ao requisito S9, no caso de o nó possuir um sensor real, o sistema deverá ser capaz de fazer a leitura do ficheiro excel produzido e colocar a informação presente no ficheiro na respetiva tabela.
S19	Relativamente ao requisito S9, no caso de o nó não possuir um sensor real, o sistema deverá ser capaz de fazer uma requisição a API com o intuito de obter as informações de leitura da estação meteorológica mais próxima.

Tabela 2: Requisitos funcionais automatic service

3.1.3 *Algoritmo machine learning*

Este módulo do sistema representa uma componente autónoma do sistema, mas é um módulo tão importante como os anteriores, uma vez que fornece informações muito valiosas. A principal função é, para um determinado terreno, prever a existência de chuva para os dias subsequentes. Esta previsão deve ser de forma mais rápida e eficaz possível, de modo a ser um parâmetro a ter em consideração na ativação do sistema de rega num dado momento.

ID	**Requisito do Sistema**
M1	Permitir a utilização de imagens inseridas pelo utilizador sobre a vegetação como fonte de dados para o algoritmo
M2	Utilizar um dataset de forma a alimentar o algoritmo de *machine learning*
M3	Prever as condições meteorológicas (Chove / Não Chove)
M4	Ser capaz de recorrer a um conjunto de bibliotecas públicas e *open source* de forma a executar o algoritmo
M5	Fazer normalização e pré-processamento dos dados de *input*
M6	Permitir um conjunto de hiperparametros de forma a melhorar a eficácia do algoritmo
M7	Apresentar a eficácia associada ao algoritmo

Tabela 3: Requisitos funcionais algortimo de machine learning

3.2 DEFINIÇÃO DE ENTIDADES E RELACIONAMENTOS

3.2.1 *Contextualização*

Tendo em conta que o projeto atual apresenta algumas regras de negócio bastante especificas, considerou-se relevante a representação dessas regras de uma forma esquemática. Para tal o esquema mais amplamente aceite e utilizado na modelação de regras de negócio é o modelo entidade relacionamento (MER) ou diagrama entidade relacionamento (DER). Este diagrama é um tipo de fluxograma que representa a forma como as "entidades" se "relacionam" entre si num determinado sistema. Estes diagramas são amplamente utilizados na projeção de bases de dados relacionais nas áreas de sistemas de informação e engenharia de software. A utilização destes diagramas permite detetar à priori problemas de lógica ou de implementação, bem como ajuda a ter uma melhor perceção das regras de negócio.

Componentes de um diagrama de entidade relacionamento

Os diagramas de entidade relacionamento são constituídos sobretudo por entidades, relacionamentos e atributos.

- entidade - Algo que pode ser definido e pode ter dados armazenados sobre si

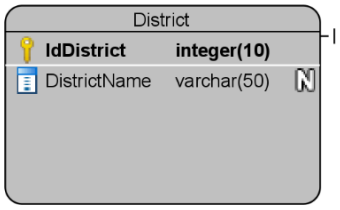


Figura 5: Exemplo de entidade

- relacionamentos - Representam a forma como as entidades estão associadas entre si. Essencialmente os relacionamentos podem ser vistos como verbos sobre determinadas entidades

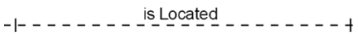


Figura 6: Exemplo de relacionamento

- atributo - O atributo é sobretudo a propriedade ou característica de uma determinada entidade

Figura 7: Exemplo de atributo

Existe ainda mais um termo característico dos diagramas entidade relacionamento que é designado de cardinalidade. Este define os atributos numéricos da relação entre duas entidades ou conjunto de entidades. A cardinalidade pode ser definida de três formas:

- um para um - uma entidade apenas se relaciona com uma outra entidade

Figura 8: Relacionamento um para um

- um para muitos - em que uma entidade tem várias relações com uma outra entidade

Figura 9: Relacionamento de um para muitos

- muitos para muitos - Muitos elementos de uma entidade estão associados a muitos elementos de uma outra entidade

Figura 10: Relacionamento de muito para muitos

3.2.2 *Modelo entidade relacionamento aplicado ao projeto*

No contexto do projeto foi definido o seguinte modelo lógico:

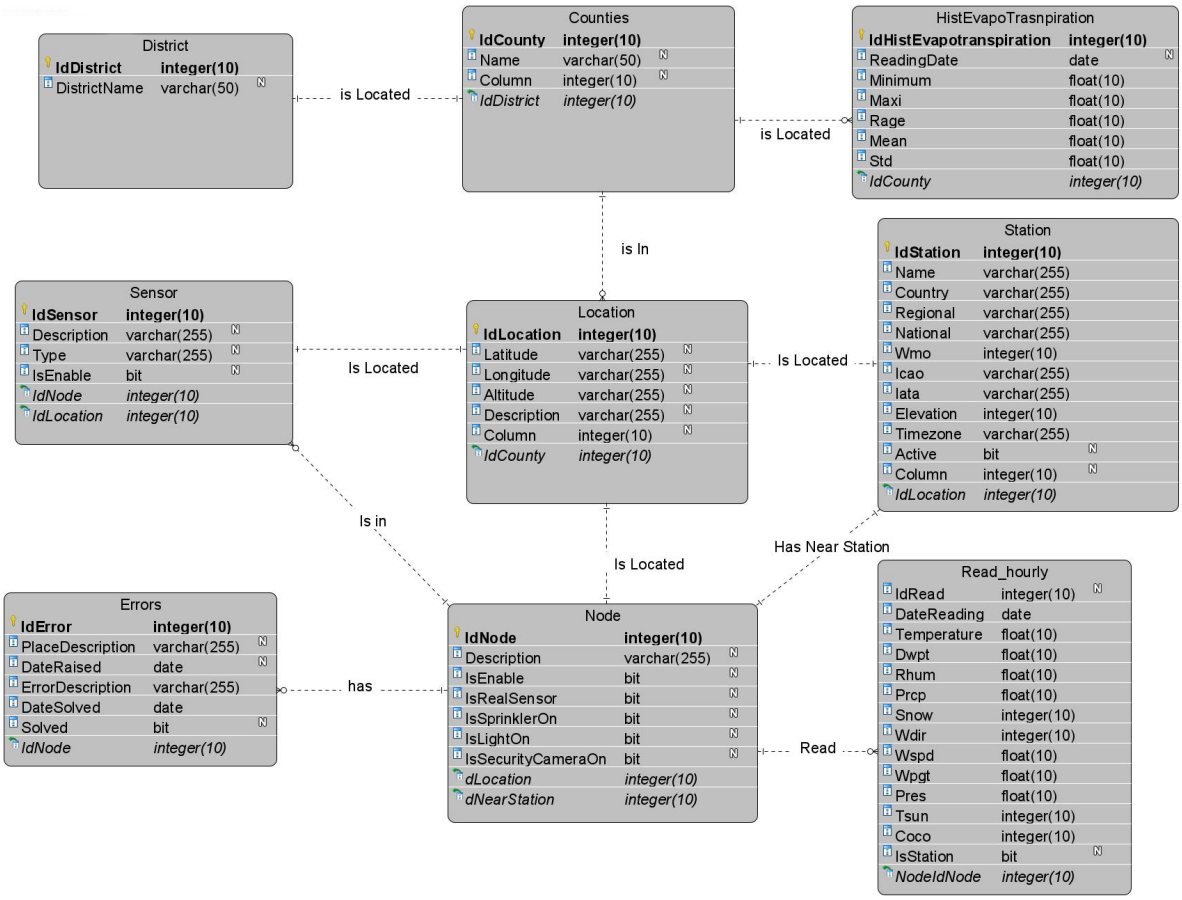


Figura 11: Diagrama entidade relacionamento

Neste contexto foram definidas as seguintes entidades: 1. District; 2. Counties; 3. Location; 4. Sensor; 5. Station; 6. Node; 7. Errors; 8. Readhourly

No caso da entidade District, será responsável por conter toda a informação relativa aos distritos suportados pelo sistema. Inicialmente esta entidade armazenará todos os distritos nacionais.

A entidade Counties terá toda a informação relativa aos concelhos suportados pelo sistema. Cada concelho apenas pertence a um e um só distrito. Por outro lado, cada distrito possui vários concelhos.

A entidade HistEvapotranspiration será responsável por armazenar toda a informação de histórico de evapotranspiração dos últimos dois meses de cada concelho que possua pelo menos um nó ativo no sistema. Uma linha de histórico de evapotranspiração apenas está associada a um determinado concelho, no entanto um concelho terá vários históricos de evapotranspiração. Assim sendo a relação é de 1:N no sentido HistEvapotranspiration(N) -> Counties(1).

A entidade Location ocupa uma posição central neste diagrama uma vez que praticamente todas entidades terão uma e uma só localização associada. Esta entidade está ainda relacionada com a entidade Counties uma vez que uma localização terá sempre um concelho associado, mas um concelho pode estar associado a várias localizações.

A entidade Station é responsável por armazenar todos os dados relativos às estações metereológicas. Cada estação meteorológica apenas existe numa determinada localização e uma localização apenas possui uma ou nenhuma estação metereológica.

A entidade Errors é responsável por armazenar todos os erros de leituras (outliers) existentes na entidade Node. Um nó pode ter vários erros associados, no entanto um erro só está associado a um e um só nó.

A entidade Sensor é muito semelhante à entidade Station, e é responsável por armazenar informação relativa aos sensores implementados num determinado nó. Neste caso, e ao contrário do que acontece com a entidade Station, a relação não é de 1:1 mas sim de 1:N com a entidade Node, uma vez que um sensor apenas está presente num nó, no entanto um nó pode ter um ou vários sensores associados (Temperatura, Humidade...).

A entidade ReadHourly é responsável por armazenar todas as leituras de uma entidade responsável por fazer leituras (Station ou Sensor). Um Node pode ter várias leituras associadas, no entanto uma leitura apenas pertence a um Node.

A entidade Node é também uma das entidades mais importantes do sistema. Esta entidade relaciona-se com a entidade Location uma vez que um nó tem apenas uma localização associada, no entanto uma localização pode ter um ou nenhum nó associado. Esta entidade pode ter um ou nenhum Station associado ou zero, um, ou vários Sensor associados. Esta condição verifica-se no atributo isRealSensor. Se o valor deste atributo for verdadeiro, então o nó terá de ter pelo menos um sensor associado, caso contrário, terá de ter um Station associado.

3.3 DIAGRAMA DE ATIVIDADES

3.3.1 *Contextualização*

O objetivo do diagrama de atividades é demonstrar o fluxo de atividades (comportamento) de um determinado processo do ponto de vista funcional. O diagrama demonstra, sobretudo, como uma atividade depende de outra [Jablonski S. e Bussler C. 1996].

O diagrama de atividade tem um conjunto de símbolos que o caracteriza, entre os quais vale a pena destacar os seguintes:

- Ponto inicial: Um pequeno círculo preenchido seguido por uma seta representa o estado de ação inicial ou o ponto inicial para qualquer diagrama de atividades.

Figura 12: ponto inicial

- atividade: Um estado de ação representa a ação ininterrupta de objetos

Figura 13: Atividade

- fluxo de ação: Ilustra a transição de um estado de ação para outro

Figura 14: Fluxo de ação

- decisões: É utilizado quando uma atividade requer uma decisão antes de passar para a próxima atividade

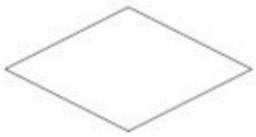


Figura 15: Decisões

3.3.2 *Diagrama de atividades no contexto do projeto*

Diagrama de atividade de ativação do sistema de rega (automatic service)

O protótipo do SIS apresenta um conjunto de atividades que são de relativa fácil compreensão e de baixa complexidade, no entanto algumas operações podem ser um pouco mais complexas e não tão transparentes. Por exemplo, no caso do automatic service é necessario que um conjunto de premissas sejam cumpridas para que ocorra a ativação do sistema de rega. Basta que uma delas falhe para que o sistema não seja ativado. Nesse sentido, e tendo em conta que este processo poderá ser o menos transparente e de difícil compreensão, foi necessário proceder à elaboração de um diagrama de atividades. Este diagrama apenas representa todas as atividades que são executadas para que o processo seja executado (neste caso a ativação do sistema de rega).

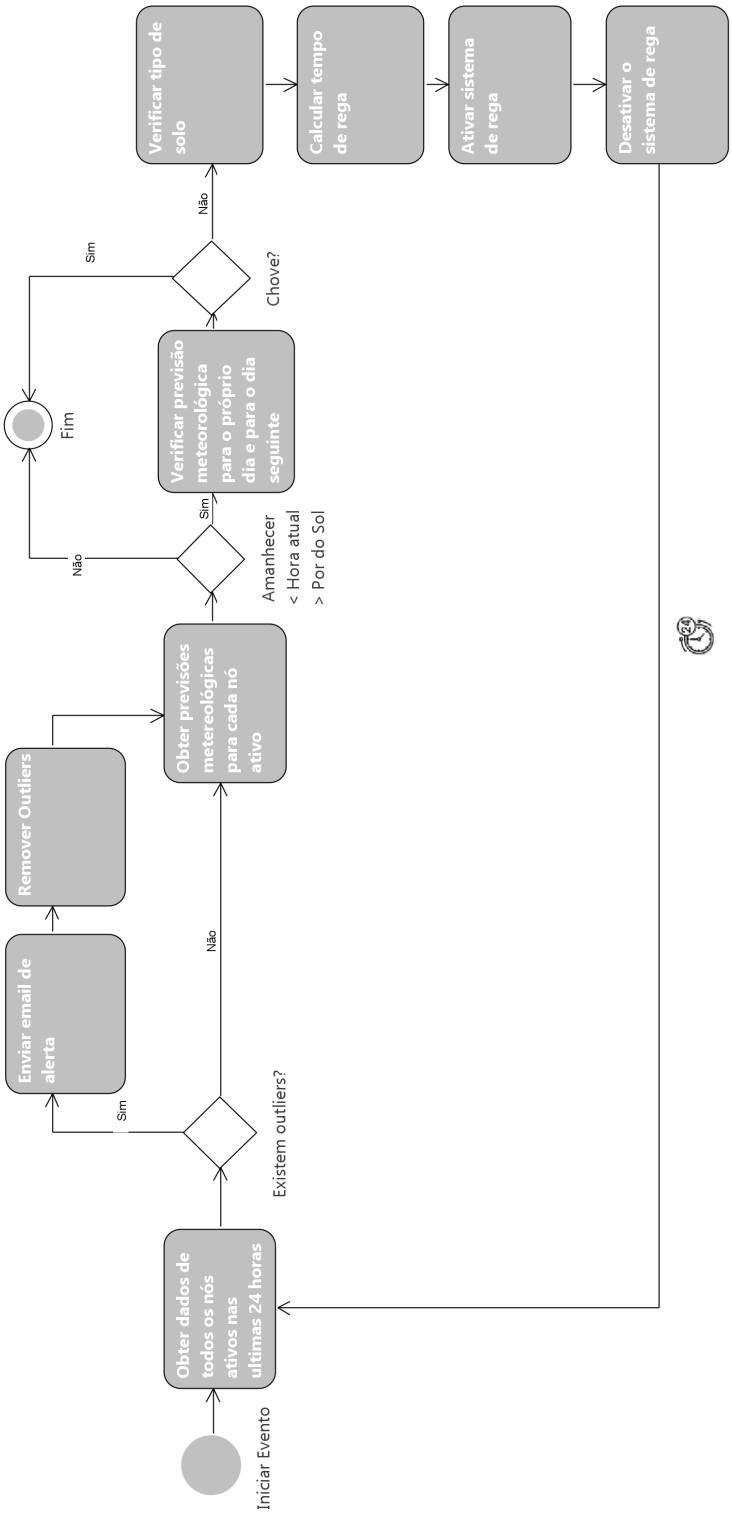


Figura 16: Diagrama de atividades ativação do sistema de rega

A ativação do sistema de rega acontece periodicamente, a uma data, hora e com uma frequência previamente definida pelo administrador. Quando essa data e hora são atingidas, o sistema "acorda", e é feita uma requisição à base de dados com todas as informações dos nós que se encontram ativos, bem como as leituras das últimas vinte e quatro horas para cada um desses nós. Depois disso, para cada um dos nós é verificada a existência de *outliers* por nó.

Se existirem *outliers*, então é enviado um email de alerta ao administrador, notificando que poderá existir um erro num ou mais sensores, e esses valores são removidos da lista de leituras. Caso não existam *outliers*, é feita uma requisição HTTP com o intuito de obter as previsões meteorológicas para cada nó ativo.

Como resposta dessa requisição, são devolvidas as previsões meteorológicas para o dia atual (em que o sistema esta a ser executado) e para o dia seguinte. Se a hora a que o sistema está a ser executado estiver compreendido entre o amanhecer e o anoitecer, então o sistema volta a "adormecer" até que o sol se ponha. Caso o sol já se tenha posto, então o sistema verificará as previsões meteorológicas para o próprio dia e para o dia seguinte: se nesse período estiver previsto a existência de aguaceiros ou chuva, então o sistema volta a "adormecer".

Caso não esteja previsto chuva nem aguaceiros, então o sistema irá consultar o ficheiro de configuração (JSON) com o intuito de verificar o tipo de solo para calcular o tempo de rega. Seguidamente o sistema de rega é ativado durante o período de tempo definido anteriormente. Quando esse período de tempo for atingido, os aspersores são desligados e o sistema volta a "adormecer". Este conjunto de ações é repetido ciclicamente.

Diagrama de atividades para atualização automática das bases de dados (automatic service)

Periodicamente é fundamental que o automatic service execute a função de manter a base de dados atualizada, principalmente as tabelas de evapotranspiração e a tabela de leituras horárias. Este processo é fundamental para que a automatização de ativação do sistema de rega ocorra com os dados mais atuais e consistentes.

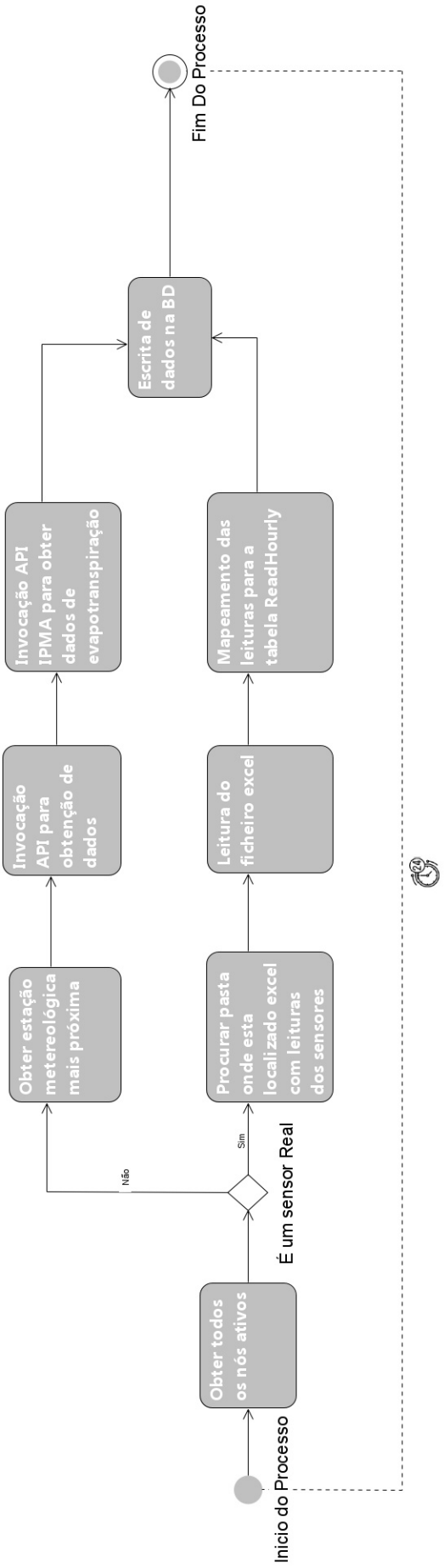


Figura 17: Diagrama de atividades para atualização automática das bases de dados

Há semelhança do que acontece no processo anterior, este também ocorre de forma cíclica e com uma periocidade previamente definida de 24 horas. Assim sendo, a cada 24 horas o sistema “acorda” e é feita uma requisição à base de dados com o intuito de obter a informação de todos os nós que se encontram ativos.

Para cada um desses nós é verificado se o nó possui sensores reais ou não, através do atributo *isRealSensor*. Em caso de resposta negativa, é efetuada uma requisição à API pública DevMeteoStat com o objetivo de ver retornada a informação meteorológicas das últimas 24 horas dessa estação meteorológica. Por outro lado, caso o nó possua sensores reais, então o sistema automaticamente irá procurar numa pasta o ficheiro cujo o nome respeite a seguinte regra:

{AnoMesDia da leitura}_Node{idNode}.xlsx

Exemplo:

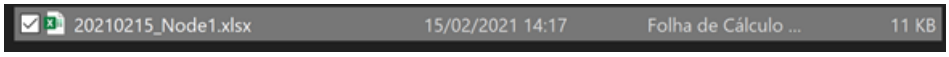
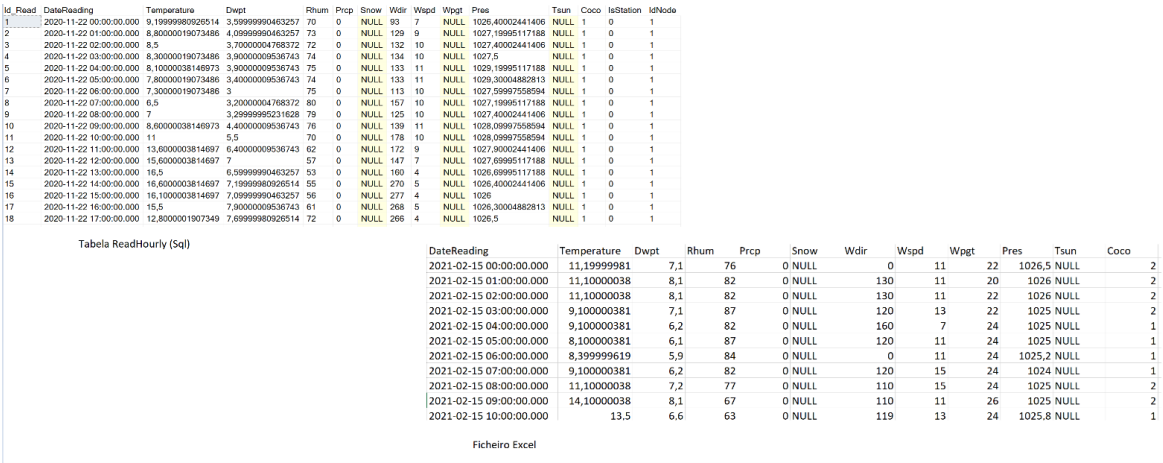


Figura 18: Exemplo de Nome de ficheiro com dados dos sensores

Após ser feita a leitura do ficheiro é feito o mapeamento dos valores lidos para a Tabela Read_Hourly:



Id_Read	DateReading	Temperature	Dwpt	Rhum	Prcp	Snow	Wdir	Wspd	Wpgt	Pres	Tsun	Coco	IsStation	IdNode
1	2020-11-22 00:00:00.000	9,19999980926514	3,59999990463257	70	0	NULL	93	7	NULL	1026,40002441406	NULL	1	0	1
2	2020-11-22 01:00:00.000	8,80000019073486	4,09999990463257	73	0	NULL	129	9	NULL	1027,19995117188	NULL	1	0	1
3	2020-11-22 02:00:00.000	8,5	3,70000004768372	72	0	NULL	132	10	NULL	1027,40002441406	NULL	1	0	1
4	2020-11-22 03:00:00.000	8,30000019073486	3,90000009536743	74	0	NULL	134	10	NULL	1027,5	NULL	1	0	1
5	2020-11-22 04:00:00.000	8,10000038146973	3,90000009536743	75	0	NULL	133	11	NULL	1029,19995117188	NULL	1	0	1
6	2020-11-22 05:00:00.000	7,80000019073486	3,40000009536743	74	0	NULL	133	11	NULL	1029,30004882813	NULL	1	0	1
7	2020-11-22 06:00:00.000	7,30000019073486	3	75	0	NULL	113	10	NULL	1027,59997558594	NULL	1	0	1
8	2020-11-22 07:00:00.000	6,5	3,20000004768372	80	0	NULL	157	10	NULL	1027,19995117188	NULL	1	0	1
9	2020-11-22 08:00:00.000	7	3,29999995231628	79	0	NULL	125	10	NULL	1027,40002441406	NULL	1	0	1
10	2020-11-22 09:00:00.000	8,60000038146973	4,40000009536743	76	0	NULL	139	11	NULL	1028,09997558594	NULL	1	0	1
11	2020-11-22 10:00:00.000	11	5,5	70	0	NULL	178	10	NULL	1028,09997558594	NULL	1	0	1
12	2020-11-22 11:00:00.000	13,6000003814697	6,40000009536743	62	0	NULL	172	9	NULL	1027,90002441406	NULL	1	0	1
13	2020-11-22 12:00:00.000	15,6000003814697	7	57	0	NULL	147	7	NULL	1027,69995117188	NULL	1	0	1
14	2020-11-22 13:00:00.000	16,5	6,59999990463257	53	0	NULL	160	4	NULL	1026,69995117188	NULL	1	0	1
15	2020-11-22 14:00:00.000	16,6000003814697	7,19999980926514	55	0	NULL	270	5	NULL	1026,40002441406	NULL	1	0	1
16	2020-11-22 15:00:00.000	16,1000003814697	7,09999990463257	56	0	NULL	277	4	NULL	1026	NULL	1	0	1
17	2020-11-22 16:00:00.000	15,5	7,90000009536743	61	0	NULL	268	5	NULL	1026,30004882813	NULL	1	0	1
18	2020-11-22 17:00:00.000	12,8000001907349	7,69999980926514	72	0	NULL	266	4	NULL	1026,5	NULL	1	0	1

Tabela ReadHourly (Sql)

DateReading	Temperature	Dwpt	Rhum	Prcp	Snow	Wdir	Wspd	Wpgt	Pres	Tsun	Coco
2021-02-15 00:00:00.000	11,19999981	7,1	76	0	NULL	0	11	22	1026,5	NULL	2
2021-02-15 01:00:00.000	11,10000038	8,1	82	0	NULL	130	11	20	1026	NULL	2
2021-02-15 02:00:00.000	11,10000038	8,1	82	0	NULL	130	11	22	1026	NULL	2
2021-02-15 03:00:00.000	9,100000381	7,1	87	0	NULL	120	13	22	1025	NULL	2
2021-02-15 04:00:00.000	9,100000381	6,2	82	0	NULL	160	7	24	1025	NULL	1
2021-02-15 05:00:00.000	8,100000381	6,1	87	0	NULL	120	11	24	1025	NULL	1
2021-02-15 06:00:00.000	8,399999619	5,9	84	0	NULL	0	11	24	1025,2	NULL	1
2021-02-15 07:00:00.000	9,100000381	6,2	82	0	NULL	120	15	24	1024	NULL	1
2021-02-15 08:00:00.000	11,10000038	7,2	77	0	NULL	110	15	24	1025	NULL	2
2021-02-15 09:00:00.000	14,10000038	8,1	67	0	NULL	110	11	26	1025	NULL	2
2021-02-15 10:00:00.000	13,5	6,6	63	0	NULL	119	13	24	1025,8	NULL	1

Ficheiro Excel

Figura 19: Exemplo de mapeamento entre tabela e ficheiro excel

Conforme é possível observar na Figura 19, o mapeamento é feito diretamente porque o nome das colunas da Tabela Read_Hourly e o nome das colunas do ficheiro Excel é idêntico. No entanto, no caso da coluna IsStation, e tendo em conta que a leitura é feita a partir de sensores reais, este valor é automaticamente igual a 0. A coluna IdNode é preenchida pelo automatic service tendo em conta a associação que é feita entre o ficheiro aberto e o nó correspondente.

Diagrama de atividades para inserção de um novo nó

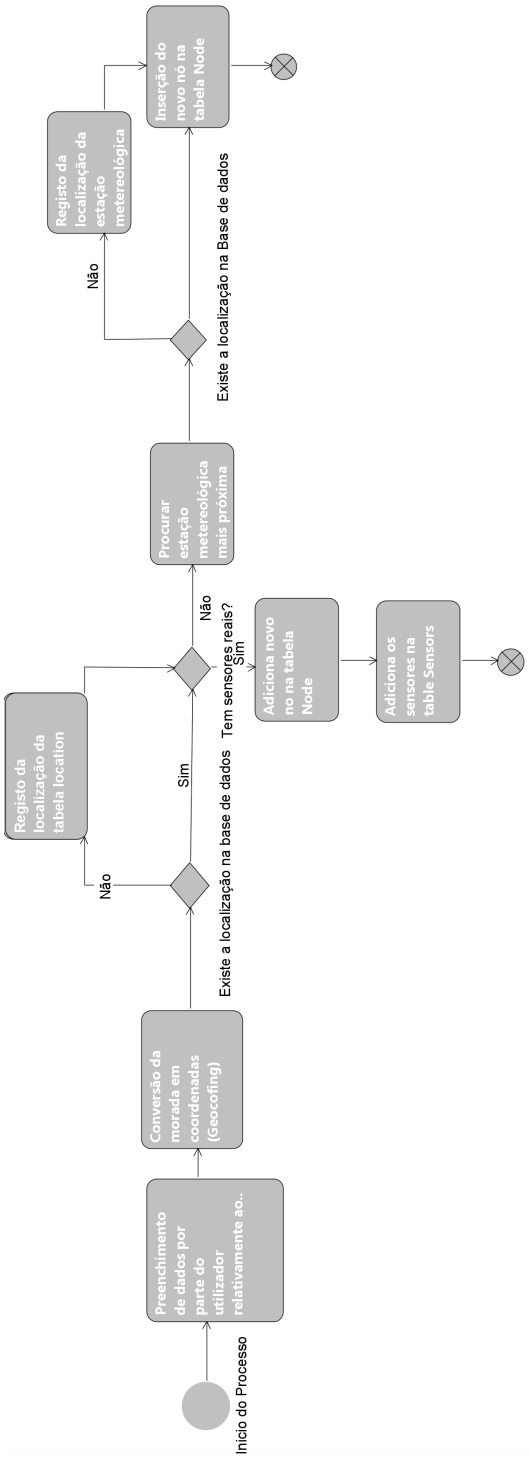


Figura 20: Diagrama de atividades para inserção de um novo nó

A inserção de um novo nó (Figura 20) inicia-se com o preenchimento, por parte do utilizador, de um formulário em que são registados os detalhes do terreno no qual é pretendido fazer a inserção de um nó. Seguidamente, e após a submissão deste formulário, o *backend* da aplicação web irá transformar morada introduzida em coordenadas geográficas, processo esse designado de *Geocoding*.

Após essa transformação, o sistema irá verificar se a localização pretendida já se encontra na Tabela *Location*. Em caso negativo, a localização é inserida.

Em seguida, é verificado se o Nó que se pretende inserir possui sensores reais. Sendo assim existem duas possibilidades:

- em caso negativo é invocada uma API (DevMeteoStat) que, tendo como argumento as coordenadas geográficas, irá retornar a estação meteorológica mais próxima. Em seguida é feita a inserção da localização da estação meteorológica na Tabela *Location*, bem como da estação meteorológica e do Nó nas tabelas *Station* e *Node*, respetivamente;
- em caso afirmativo, primeiramente é adicionado um novo nó (com o valor de *IsRealSensor* igual a 0) e em seguida é adicionado cada um dos sensores na Tabela *Sensors*.

3.4 RESUMO DO CAPÍTULO

No inicio deste capítulo foi feita uma breve contextualização teórica sobre o que são os diagramas de entidade relacionamento, para que são utilizados e quais os seus elementos. Foi ainda demonstrado que forma os mesmos são aplicados no âmbito do trabalho em questão, com o intuito de explorar de que forma as atividades mais complexas do sistema se processam, fornecendo assim uma visão de alto nível do comportamento do sistema.

4

MODELO SISTEMA DE REGA INTELIGENTE

4.1 OBJETIVOS

O objetivo do Smart Irrigation System (SIS) é ampliar a informação disponível sobre as condições ambientais das áreas agrícolas ou de lazer de um ou mais municípios, de forma que a ativação do sistema de rega seja feito de forma inteligente.

Toda a informação relativa a estas condições será armazenada de forma centralizada com o intuito de viabilizar a consulta e análise dos dados bem como o suporte à tomada de decisão, permitindo assim que a rega seja feita da forma mais eficiente possível, diminuindo consideravelmente a água desperdiçada. É fundamental ainda, que o sistema tenha um conjunto de mecanismos que permita a identificação de potenciais erros nos equipamentos de leitura, para que a entidade responsável possa proceder a reparação ou substituição dos mesmos, caso se justifique.

4.2 ARQUITETURA

Numa primeira fase deste projeto, foi estabelecido um conjunto de objetivos para as diferentes fases de desenvolvimento. Assim sendo, como ilustrado na Figura 21 a arquitetura do sistema de rega pode ser dividida em 3 camadas:

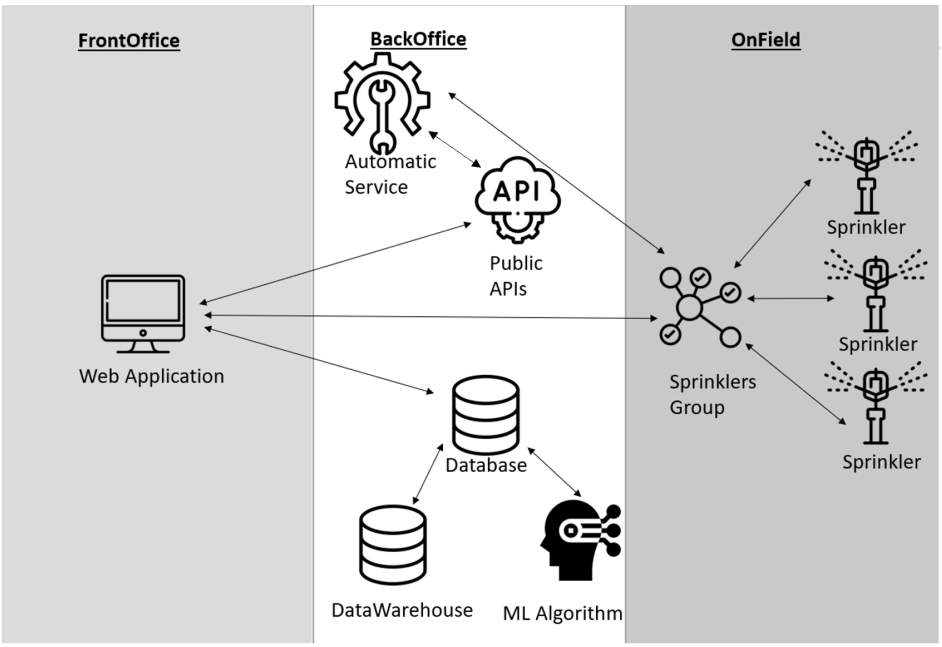


Figura 21: Proposta de arquitetura do sistema de rega inteligente

FrontOffice

O FrontOffice é a camada mais próxima do cliente onde está inserida a aplicação web. A aplicação web é responsável pela representação de gráficos e estatísticas baseadas nos dados recolhidos e armazenados na base de dados. A aplicação web permite ainda que o cliente ative ou desative o sistema de rega de forma manual, bem como a comunicação com as APIs públicas de forma a fazer as operações de CRUD relativamente aos locais em que se pretende fazer a monitorização.

BackOffice

O BackOffice é invisível para o cliente, mas é aqui que estão inseridos todos os motores responsáveis por alimentar não só o dashboard, mas também os processos passiveis de serem automatizados e que não necessitam de intervenção humana direta (Automatic Service), tais como a ativação do sistema de rega de forma automática baseado em métricas (temperatura, humidade data e hora), invocação de APIs públicas e armazenamento dos dados na base de dados.

Nesta camada estão ainda todas as comunicações com as APIs públicas de forma a obter o histórico de dados meteorológicos (DevMeteoStat API), dados de evapotranspiração dos últimos 3 meses (IPMA API), (Reverse) Geocoding (PositionStack API) e as previsões meteorológicas para um determinado local (OpenWeather API). Nesta camada encontra-se ainda presente um algoritmo de *machine learning*, que tendo como base o histórico de dados meteorológicos para um dado local, irá prever a possibilidade de existência de chuva ou aguaceiros.

OnField

O OnField é a camada onde está inserido todo o hardware do sistema (sensores e aspersores). Cada um dos aspersores estará inserido num grupo (Node) de forma a ser possível fazer a divisão de um determinado terreno em diferentes zonas. Cada grupo representará uma zona especifica. Cada Nó poderá ou não ter associado a si um conjunto de sensores reais.

4.3 PARÂMETROS EM ANÁLISE

Neste trabalho pretende-se que estejam disponíveis um conjunto de dados obtidos recorrendo a APIs públicas.

No caso das APIs que permitem receber dados meteorológicos, a obtenção dos mesmos pode ser feita através da obtenção das condições atmosféricas numa determinada hora ou intervalo de horas, que poderá ir até 9 dias a partir da data de requisição. Quanto a obtenção de dados por hora, são retornadas as seguintes métricas:

Parameter	Description	Type
time	UTC time stamp (format: YYYY-MM-DD hh:mm:ss)	String
time_local	Local time stamp (format: YYYY-MM-DD hh:mm:ss); only provided if tz is set	String
temp	The air temperature in °C	Float
dwpt	The dew point in °C	Float
rhum	The relative humidity in percent (%)	Integer
prcp	The one hour precipitation total in mm	Float
snow	The snow depth in mm	Integer
wdir	The wind direction in degrees (°)	Integer
wspd	The average wind speed in km/h	Float
wpgt	The peak wind gust in km/h	Float
pres	The sea-level air pressure in hPa	Float
tsun	The one hour sunshine total in minutes (m)	Integer
coco	The weather condition code	Integer

Figura 22: Parâmetros retornados na invocação da API (DevMeteoStat)

Exemplo de resposta:

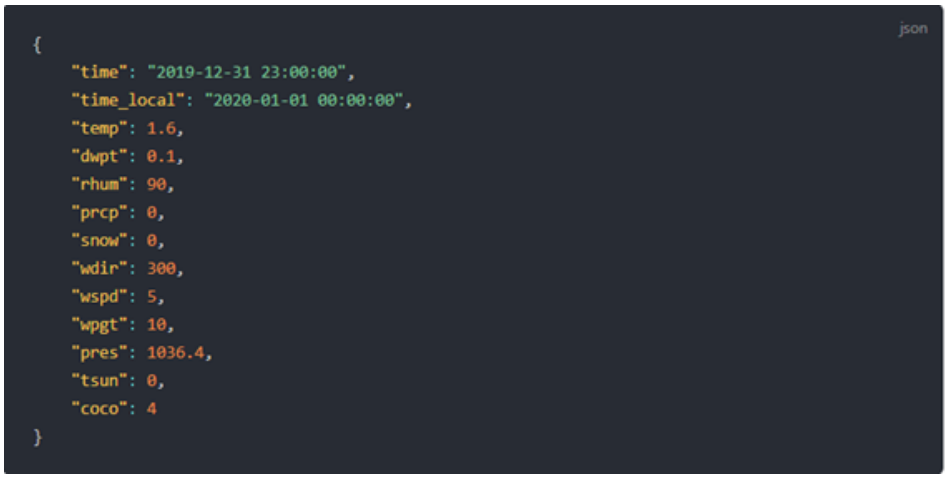


Figura 23: Exemplo de resposta da API (DevmeteoStat)

Para além destes valores, é fundamental ter em consideração os valores de evapotranspiração diária por concelho dos últimos 2 meses, que são obtidos através da invocação da API do IPMA.

Como output desta invocação são obtidas as seguintes propriedades:

- date: data dos valores de referência;
- minimum: valor diário mínimo de Evapotranspiração em mm;
- maximum: valor diário máximo de Evapotranspiração em mm;
- range: valor diário da amplitude de Evapotranspiração em mm;
- mean: valor diário da mediana de Evapotranspiração em mm;
- std: valor diário do desvio padrão de Evapotranspiração em mm.

Tendo por base toda esta informação, as variáveis que se consideram mais relevantes para a ativação do sistema de rega são:

1. temperatura (T): considerou-se que a melhor temperatura do solo para irrigar estava na faixa de 15 ºC e 26 ºC [Rosário L. 2004]. Se o solo estiver abaixo desta temperatura poderá estar muito frio, dificultando a infiltração da água. Acima da faixa, é possível que a evaporação da água seja muito rápida e o solo permaneça seco [Ipma.pt 2021.
2. precipitação (P) Tendo como base dos dados compilados em [Santos et al. 2003] considerou-se a divisão do solo em 3 categorias:

   Conforme é possível observar na Tabela 4, a caracterização das regiões climáticas pode ser feita segundo quatro categorias: Semiárida, Sub-húmida seca, Sub-húmida e Húmida.

Regiões Climáticas	Precipitação diária em mm (verão)
Semiárida	0
Sub-húmida seca	0.55
Sub-húmida	1.11
Húmida	1.66

Tabela 4: Categorização dos solos baseado na precipitação diária (mm)

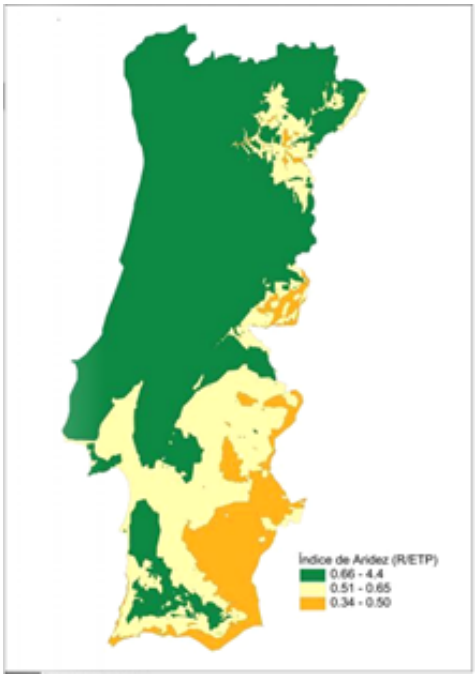


Figura 24: Mapa de Portugal com regiões climáticas baseadas no Índice de Aridez

Na Figura 24 é ilustrado de que forma o índice de aridez é distribuído ao longo do território nacional. É possível observar que a região do Alentejo e Algarve são as mais áridas a nível nacional.

3. evapotranspiração (Etp) é a perda de água do solo por evaporação. Tendo novamente por base dados compilados em e [Santos F. et al. 2003] as categorias definidas anteriormente:

Regiões Climáticas	**Evapotranspiração potencial diária em mm (verão)**
Semiárida	6.67
Sub-húmida seca	6.11
Sub-húmida	5
Húmida	4.44

Tabela 5: Categorização dos solos baseada na evapotranspiração

4. data e hora Considerou-se que o ideal seria a rega ser feita entre as 19h e a 7h de forma a evitar a que a mesma se evapore mesmo antes de ser absorvida pelo solo

5. necessidade de água Seguindo a mesma categorização feita na evapotranspiração e precipitação diária, se a necessidade de água for inferior a 6.67 m^3/ha então a região climática é considerada Semi-árida, se for entre 6.11 m^3/ha e 5 m^3/ha, então e considerada Sub-húmida seca. No entanto se for superior a 4.44 m^3/ha então trata-se de um solo húmido e com muito menos necessidade de água

Regiões Climáticas	**Necessidade de água em m^3/ha(verão)**
Semiárida	6.67
Sub-húmida seca	6.11
Sub-húmida	5
Húmida	4.44

Tabela 6: Categorização dos solos baseada na evapotranspiração

Para além das métricas definidas anteriormente, é fundamental ter em consideração as previsões meteorológicas para as 6h seguintes uma vez que, mesmo que o solo esteja relativamente seco, se houver previsão de aguaceiros nas horas seguintes, não há a necessidade de se ativar o sistema de rega.

4.4 ARQUITETURA DA APLICAÇÃO WEB

Tendo em conta o problema que este trabalho se propõe solucionar, chegou-se à conclusão de que o desenvolvimento de projeto deveria incidir sobretudo numa abordagem em que o domínio de negócio teria de ocupar uma posição central, onde a maior preocupação aplicar-se-ia em representar da melhor forma as necessidades de negócio em classes e comportamentos.

Sendo assim, optou-se por utilizar um desenvolvimento baseado em DDD (Domain driven Design). A ideia inicial do DDD é utilizar uma modelagem orientada aos objetos de uma forma mais pura, ou seja, as classes modeladas e os seus métodos devem representar as regras de negócio no contexto atual. A persistência dos dados é colocada em segundo plano sendo apenas uma camada complementar. [Evans E. 2014]

Em cima desta abordagem de desenvolvimento, a arquitetura do software foi pensada com base numa arquitetura em camadas, designada *Onion Architecture* uma vez que se considerou fundamental a diminuição do acoplamento entre camadas de forma a permitir que o sistema funcione de forma mais independente e modular possível.

4.4.1 *Onion Architecture*

A Onion Architecture é baseada no princípio de inversão de controlo. A Onion Architecture é composta por várias camadas concêntricas onde é feita a invocação entre elas através da utilização de interfaces, a partir das camadas mais externas e em direção ao núcleo que representa o domínio. A arquitetura não depende da camada de dados como nas arquiteturas multicamadas clássicas, mas dos modelos de domínio reais.

Problema e Solução

Numa arquitetura tradicional, a camada mais externa User Interface (UI) inter-relaciona-se com a camada de Business Logic (BL) e a camada de BL comunica com a camada de dados - todas estas camadas estão fortemente acopladas e dependem umas das outras, ou seja, nenhuma destas camadas é independente. Estes sistemas são muito difíceis de compreender e manter, pelo que normalmente uma alteração numa destas camadas terá inevitavelmente implicação nas demais.

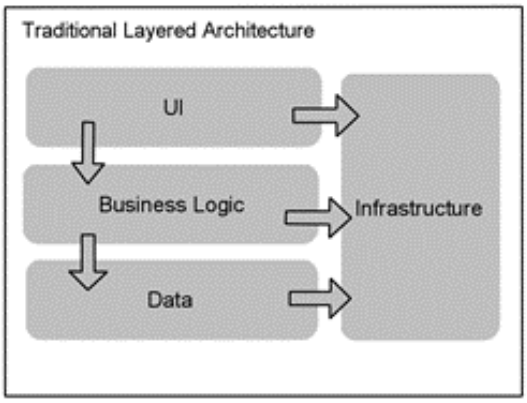


Figura 25: Arquitetura tradicional do Software em camadas

Este problema pode ser resolvido tendo por base um conjunto de camadas que vão desde o núcleo até à infraestrutura. A grande característica desta abordagem é que o acoplamento é movido em direção ao centro.

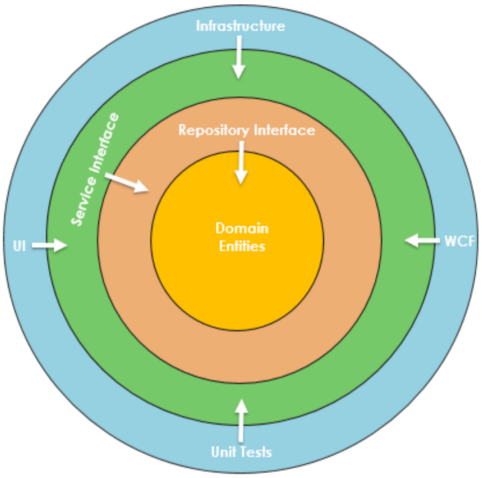


Figura 26: Arquitetura do Software em camadas (Onion Architecture)

No centro da Onion Architecture, existe a camada de domínio. Esta camada representa os objetos e comportamentos de negócio. O principal objetivo é ter todos os objetos de domínio de forma centralizada no núcleo. Além dos objetos, nesta camada estão ainda presentes interfaces de forma a permitir que haja abstração entre esta camada e a camada superior.

Camada de Domínio

No centro da Onion Architecture, existe a camada de domínio. Esta camada representa os objetos e comportamentos de negócio. O principal objetivo é ter todos os objetos de domínio de forma centralizada no núcleo. Além dos objetos, nesta camada estão ainda presentes interfaces de forma a permitir que haja abstração entre esta camada e a camada superior.

Camada de Repositório

A camada de repositório cria uma abstração entre as entidades de domínio e a lógica de negócio da aplicação. Nesta camada estão definidas um conjunto de interfaces que permitem guardar e utilizar um conjunto de objetos envolvidos na base de dados. É criado geralmente um repositório genérico, onde são utilizadas *queries* para retribuir os dados, é feito o mapeamento dos dados a partir da fonte para uma entidade de negócio e onde são persistidas as alterações da entidade de negócio para a fonte de dados.

Camada de Serviço

A camada de serviço mantém interfaces com operações comuns, como criar, ler, editar e eliminar. Além disso, esta camada é usada para a comunicação entre a camada de UI e a camada de repositório. A camada de serviço também pode conter a lógica de negócio para uma entidade. Nesta camada, as interfaces de serviço são mantidas separadas da sua implementação, mantendo um baixo acoplamento e a separação de funções.

Camada de UI

A camada de UI é a mais externa e é a camada mais próxima do cliente. Para uma aplicação Web, representa a API ou o projeto de testes unitários. Esta camada tem uma implementação seguindo o princípio de injeção de dependência para que seja construída uma aplicação com uma estrutura fracamente acoplada e que permita a comunicação com a camada imediatamente abaixo através de interfaces.

4.5 RESUMO DO CAPÍTULO

Neste capitulo foi feita a modulação do sistema de rega inteligente, inicialmente é apresentada uma proposta de arquitetura para o sistema a nível geral. Seguidamente são apresentadas as métricas necessárias a ter em conta na tomada de decisão do sistema e finalmente foi esmiuçada a arquitetura de software que vai ser utilizada pela aplicação web. Esta definição conceptual é fundamental para evitar correções arquiteturais em fases posteriores do desenvolvimento do sistema

5

DEFINIÇÃO E CONCEÇÃO DO SISTEMA

Este capitulo incide sobretudo no desenvolvimento prático do sistema. Inicialmente são definidas as ferramentas a utilizar em cada um dos módulos e porque razão se optou por cada uma delas. De seguida cada um dos módulos é analisado em detalhe de modo a descrever como o sistema foi concebido a nível prático. No caso do ambiente de sensorização, são propostos um conjunto de sensores que se consideram ser os ideais relativamente à razão custo/benefício.

5.1 ESCOLHA DE FERRAMENTAS

Tendo em conta a modularidade do sistema, foi fundamental ter em consideração uma diversidade de ferramentas, tendo em mente aquelas que melhor poderiam dar resposta às necessidades de cada um dos módulos

Aplicação web

No caso da aplicação web, a escolha de ferramentas foi feita tendo em consideração o facto se serem tecnologias *open-source*, de fácil utilização, bem consolidadas no mercado e que fossem gratuitas.

Como tal foram escolhidas as seguintes tecnologias:

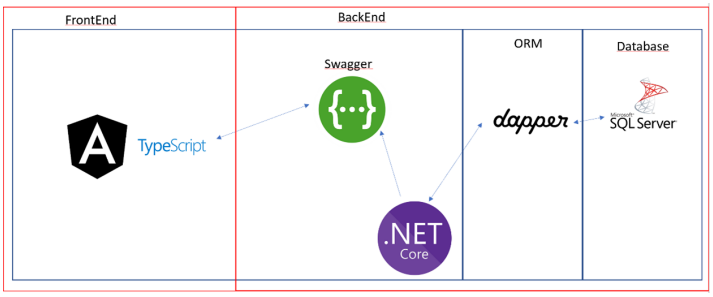


Figura 27: Tecnologias utilizadas na aplicação web

No caso do *frontend* foi escolhido Angular 9. O Angular é uma plataforma e *framework* cujo objetivo é permitir a construção de Single Page Applications (SPA) utilizando HTML, CSS e Typescript. O Angular é escrito em Typescript e foi criada pela Google. Esta *framework* contém um conjunto de elementos que são

bastante característicos tais como componentes, templates, directives, routing, módulos, serviços, dependency injection e ferramentas de automatização de tarefas. Para além desta ferramenta ser *open-source*, possui uma grande comunidade, e é amplamente aceite ao nível empresarial. Para além disso esta *framework* está muito bem documentada.

No caso do *backend* como *framework* de desenvolvimento utilizado foi o .Net Core 3.1 da Microsoft em C# que possui uma abordagem multi-plataforma. É uma *framework* com uma boa arquitetura e simples. No âmbito deste projeto foi utilizado o padrão de desenvolvimento Web Api. Tendo em conta que o desenvolvimento incide sobretudo em APIs, optou-se pela utilização do Swagger que é uma *interface description language* que permite fazer a documentação das APIs REST desenvolvidas. Atendendo que o ambiente de desenvolvimento é baseado em tecnologias Microsoft então, para que haja uma maior integração e coerência de escolha de ferramentas, optou-se pela utilização de uma base de dados relacional Microsoft SQL Server. Como "ponte" entre a base de dados e a *framework* de *backend* utilizou-se uma ORM (Object Relational Mapping) chamada dapper. As grandes vantagens desta ORM são o facto de ser uma micro ORM, ou seja, é bastante leve, o mapeamento é feito de forma bastante simples, de fácil integração e liberdade de configuração.

No caso do Automatic Service, e tendo em conta que se trata de um serviço que apenas possui *backend*, as tecnologias utilizadas são exatamente as mesmas usadas no *backend* da aplicação web.

Algoritmo de Machine Learning

No que toca ao algoritmo de *Machine Learning* foi utilizada a linguagem Python, uma vez que esta é a linguagem que possui um maior número de bibliotecas de apoio aos algoritmos de *Machine Learning*, o que permite que o desenvolvimento do código ocorra de uma forma mais rápida. Como ambiente de desenvolvimento foi necessário fazer a instalação do software Anaconda.

Figura 28: Ferramentas usadas no algoritmo de machine learning

Numpy: O Numpy é uma biblioteca de Python que permite a operação sobre multi-arrays e matrizes grandes e multi-dimensionais, fornecendo uma grande coleção de operações matemáticas para operar sobre esses arrays.

Seaborn: Esta biblioteca permite converter qualquer objeto de visualização de dados baseado no MatplotLib, permitindo a definição de gráficos estatísticos.

Sklearn: É uma biblioteca de *machine learning* que dá suporte a uma grande variedade de algoritmos de *machine learning*, tais como: k-means, gradient boosting, random forests...

MatplotLib: Permite a elaboração de gráficos para uma melhor representação dos dados.

Pandas: É uma biblioteca orientada para a manipulação e análise de dados, oferecendo estruturas de forma a manipular tabelas numéricas.

5.2 CONCEÇÃO DO SISTEMA

Nesta secção é definido ao detalhe, e tendo em conta uma vertente mais prática, de que forma o sistema foi concebido. Esta secção está dividido em subsecções, de forma a descrever em detalhe o funcionamento de cada um dos módulos.

5.2.1 *Automatic Service*

Conforme foi definido no Capítulo 4.2, o Automatic service está localizado na camada de BackOffice, uma vez que este módulo não possui qualquer tipo de interação direta com o cliente, mas sim diretamente com os dados recolhidos pelas APIs e também com o hardware do sistema de rega em si.

Assim sendo, o Automatic Service tem três funções que são essenciais no sistema:

- Periodicamente, manter os dados da base de dados atualizados.
- Periodicamente, observar os dados que estão armazenados na base de dados e ativar automaticamente o sistema de rega caso se considere necessário.
- Verificar a existência de *outliers* e notificar utilizador caso existam

É importante salientar que as três funções do Automatic Service são completamente independentes entre si, e a periodicidade com que cada uma das funções é ativada é completamente personalizável por parte do cliente.

Atualização dos dados da base de dados

No caso desta função, foi definida que com uma periocidade de 24h é feita uma requisição a API DevMeteoStat, de modo a obter as informações relativas as condições meteorológicas para cada um dos nós ativos.

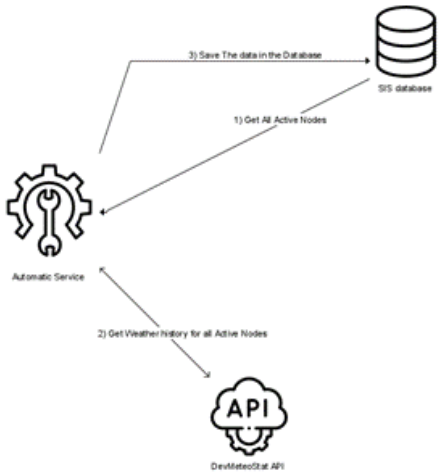


Figura 29: Automatic Service, Update de base de dados diário

Assim sendo, a cada 24 horas, é feita uma requisição à base de dados do sistema de forma a obter todos os nós ativos, bem como as suas localizações. Depois disso, e para cada um dos nós ativos, é feita uma requisição à API DevMeteoStat, de forma a obter o histórico de dados meteorológicos.

```
 public RootWeatherDataModel<HourlyDataModel> GetHourlyDataOfStation(
     HourlyDataOfStationQueryParams hourlyDataOfStationParams)
{
            RestClient client = new RestClient($"{_config.GetConfiguration("
                DevMeteoStatApi:APIBASICURI")}stations/hourly");
            var request = new RestRequest();
            request.AddHeader("x-api-key", _config.GetConfiguration("
                DevMeteoStatApi:APIKEY"));
            request.AddHeader("Accept-Encoding", "gzip, deflate");
            request.AddHeader("User-Agent", "runscope/0.1");
            request.AddHeader("Accept", "*/*");
            request.Method = Method.GET;
            request.AddParameter("station", hourlyDataOfStationParams.Station,
                ParameterType.QueryString);
            request.AddParameter("start", hourlyDataOfStationParams.Start,
                ParameterType.QueryString);
            request.AddParameter("end", hourlyDataOfStationParams.End,
                ParameterType.QueryString);
            request.RequestFormat = DataFormat.Json;


            var response = client.Execute(request);
            return
}
```

Listagens 1: Requisição API DevMeteoStat

Após a requisição ser bem-sucedida, é feita a escrita dos dados retornados pela API, tendo em atenção a não replicação dos dados existentes, sendo para tal utilizado o seguinte método:

```
public int AddReadHourly(RootWeatherDataModel<HourlyDataModel>
    rootWeatherDataModel, bool isStation, int? idStation, int idNode)
        {
            int rowsupdated = 0;
            using (IDbConnection db = new SqlConnection(_connectionString))
            {

                foreach (var item in rootWeatherDataModel.Data)
                {
                    string sql =
                        $"IF NOT EXISTS (SELECT * FROM [dbo].[Read_Hourly] WHERE
                            DateReading = @DateReading and IdNode = @IdNode) " +
                        $"Insert into [dbo].[Read_Hourly] (DateReading,
                            Temperature,Dwpt, Rhum, Prcp, Snow, Wdir, Wspd, Wpgt,
                            Pres, Tsun, Coco, IsStation, IdNode) values (
                            @DateReading, @Temperature, @Dwpt, @Rhum, @Prcp, @Snow
                            , @Wdir, @Wspd, @Wpgt, @Pres, @Tsun, @Coco, @IsStation
                            , @IdNode)";
                    rowsupdated = db.Execute(sql,
                         new
                         {
                             DateReading = item.Time,
                             Temperature = item.Temp,
                             Dwpt = item.Dwpt,
                             Rhum = item.Rhum,
                             Prcp = item.Prcp,
                             Snow = item.Snow,
                             Wdir = item.WDir,
                             Wspd = item.WSpd,
                             Wpgt = item.WPgt,
                             Pres = item.Pres,
                             Tsun = item.Tsun,
                             Coco = item.Coco,
                             isStation = isStation,
                             idNode = idNode
                         });
                }
            }

            return rowsupdated;
        }
```

Listagens 2: Atualização dos dados na Tabela Read_Hourly

Para que não haja duplicação de dados, é fundamental fazer uma validação prévia se os dados já existem na Tabela Read_Hourly.

Deteção de outliers e sistema de alertas

Os *outliers* são dados que se distanciam da normalidade, podem ser considerados atípicos, desvios, anomalias, exceções, ruido, etc [40]. Estes dados poderão e irão certamente causar problemas no nosso sistema, uma vez que não irão refletir dados reais, aumentando assim a possibilidade de erro. No contexto deste projeto, os *outliers* podem ocorrer devido a falhas no hardware (sensores), e como tal, assim que seja detetada uma falha nestes equipamentos, é fundamental proceder-se a reparação ou substituição dos mesmos, de forma a não prejudicar o funcionamento do sistema.

Para que seja detetada a presença de *outliers* é necessário:

- organizar os dados de forma crescente;
- calcular a mediana do conjunto de dados;
- calcular o quartil inferior;
- calcular o quartil superior;
- calcular as barreiras internas e externas.

A organização dos dados é feita facilmente através invocação do método .Sort() nativo do C#. Após a organização dos dados a mediana é calculada da seguinte forma:

- se o número de elementos for par é retornada a média dos elementos ao centro

1	2	3	4	5	6	7	8	9	10

Tabela 7: Lista de Elementos Par

Neste caso a mediana é (5 + 6) / 2 = 5.5

- se o número de elementos for ímpar, é retornado o elemento localizado no centro da lista

1	2	3	4	5	6	7	8	9	10

Tabela 8: Lista de Elementos Impar

Neste caso a mediana é 5

```
private static double getMedian(List<double> data)
{
            if (data.Count % 2 == 0)
                return ((data[data.Count / 2]) + (data[data.Count / 2 - 1]))
                     / 2;
            else
                return data[data.Count / 2];
        }
```

Listagens 3: Cálculo da mediana

Após ser feito o cálculo da mediana, são calculados os quartis, através da repetição do processo de cálculo de medianas para cada um dos subconjuntos gerados no passo anterior. De seguida é calculado o *interquartile range* (IQR), através da diferença entre o primeiro (Q1) e o terceiro quartis (Q3), e depois são calculadas as barreiras. No caso da barreira inferior é calculada tendo em conta a seguinte fórmula:

$$\text{Lowerfence} = Q1 - 1.5 * IQR$$

Já a barreira superior é calculada tendo em conta a seguinte fórmula:

$$\text{Upperfence} = Q3 + 1.5 * IQR$$

```
public static List<double> getOutliers(List<double> input)
        {
            List<double> output = new List<double>();
            List<double> data1 = new List<double>();
            List<double> data2 = new List<double>();
            input.Sort();
            if (input.Count % 2 == 0)
            {
                data1 = input.GetRange(0, input.Count / 2);
                data2 = input.GetRange(input.Count / 2, (input.Count) - (
                    input.Count / 2));
            }
            else
            {
                data1 = input.GetRange(0, input.Count / 2);
                data2 = input.GetRange(input.Count / 2 + 1, input.Count-1);
            }

            double q1 = getMedian(data1);
            double q3 = getMedian(data2);
             double iqr = q3 - q1;
            double lowerFence = q1 - 3 * iqr;
            double upperFence = q3 + 3 * iqr;
            for (int i = 0; i < input.Count; i++)
            {
```

```
                if (input[i] < lowerFence || input[i] > upperFence)
                    output.Add(input[i]);
            }

            return output;

        }
```

Listagens 4: Cálculo dos outliers

Assim sendo, para cada um dos valores é verificado se é maior que a barreira superior ou menor que a barreira inferior. Em caso afirmativo, significa que estamos perante a presença de um *outlier*. Após a deteção de um *outlier* é enviado um email ao administrador do sistema para que proceda a verificação do hardware:

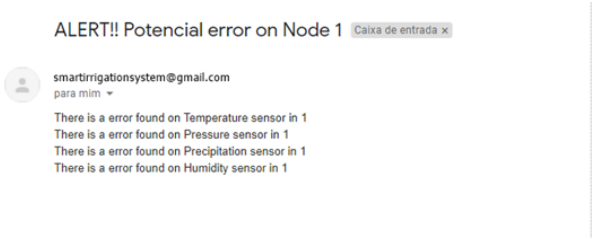


Figura 30: Exemplo de email de notificação de erros

É importante salientar que todo o email é passível de ser customizado mediante as necessidades de informação por parte do cliente.

Ativação do sistema de rega

Conforme já foi enunciado anteriormente, este módulo possui uma função que se pretende que seja uma mais valía em todo o protótipo. Assim sendo, essa função tem como principal objetivo permitir a ativação do sistema de rega de forma automática tendo em consideração um conjunto de métricas. Estas métricas, assim como qualquer parametrização deste protótipo pode ser feita de forma completamente personalizável, com o intuito de atender da melhor forma possível as necessidades do cliente, tendo em conta, por exemplo, a sua localização geográfica. Por *default* esta função é ativada de hora em hora.

Antes do desenvolvimento da funcionalidade em si, foi necessário proceder-se à criação de um ficheiro .JSON com as caracterizações do tipo de terreno e do ar (definidos no Capitulo 4.3). Estas parametrizações serão fundamentais para o cálculo de tempo de rega. O tempo de rega para cada um dos tipos de solo também estão definidos no ficheiro JSON.

```
{
  "Evapotranspiracao": {
    "Semiarida": 6.67,
    "SubHumidaSeca": 6.11,
```

```
    "SubHumida": 5,
    "Humida": 4.44

  },

  "Precipitacao": {
    "Semiarida": 0,
    "SubHumidaSeca": 0.55,
    "SubHumida": 1.11,
    "Humida": 1.66
  },

  "TempoDeRegaEmMin": {
    "Semiarida": 30,
    "SubHumidaSeca": 20,
    "SubHumida": 10,
    "Humida": 5
  }
}
```

Listagens 5: Ficheiro JSON com a caracterização dos solos

Conforme é possível observar na Figura 16, para que a ativação do sistema de rega aconteça de forma automática, um conjunto de parâmetros têm de se verificar obrigatoriamente.

- a temperatura atual tem de estar compreendida num intervalo de +3/-3 graus em relação à temperatura média do dia;
- a hora atual tem de ser superior à hora do por do sol e inferior à hora de nascer do sol;
- a previsão meteorológica para o dia seguinte não pode ser de chuva.

A ativação do sistema de rega de forma automática acontece se e só se todos os parâmetros definidos acima se concretizarem.

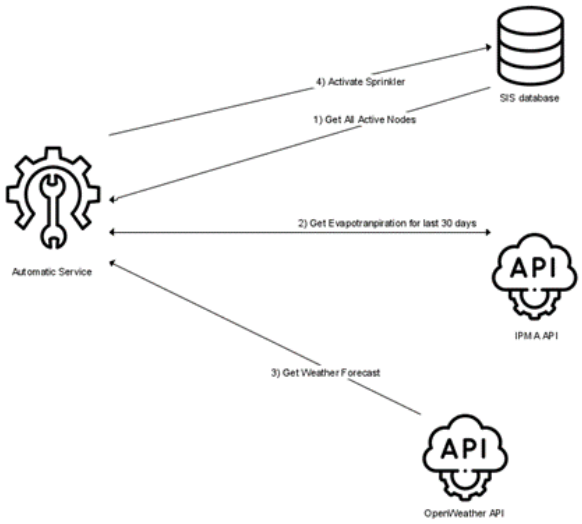


Figura 31: Fluxo de informação necessária para ativação do sistema de rega

Inicialmente é necessário fazer uma requisição à base de dados com o intuito de obter todas os nós ativos, uma vez que a ativação do sistema de rega apenas ocorrerá nos nós ativos. Após esta requisição, e para cada um dos nós ativos, é feito uma requisição à API do IPMA com o intuito de obter a evapotranspiração dos últimos 30 dias para cada concelho.

```
 public string[] GetHistoryEvaporationByCountyName(County county, string
     districtName)
{
            RestClient client = new RestClient($"https://api.ipma.pt/open-data/
                observation/climate/evapotranspiration/{districtName.ToLower()}/
                et0-{county.CountyId}-{county.Name.ToLower()}.csv");
            var request = new RestRequest();

            request.AddHeader("Accept", "*/*");
            request.AddHeader("Accept-Encoding", "gzip, deflate");
            request.AddHeader("User-Agent", "runscope/0.1");
            request.Method = Method.GET;
            var response = client.Execute(request).Content.Split("\n");


            return response;

}
```

Listagens 6: Requisição dos dados Evapotranspiração (IPMA API)

Como resposta a esta requisição é produzido um ficheiro .CSV com o seguinte formato:

date	minimum	maximum	range	mean	std
02/12/2020	1.128679	1.30822	0.179536	1.21428	0.045226
03/12/2020	1.01566	1.20227	0.186605	1.117089	0.044089
04/12/2020	0.899726	1.07874	0.179015	0.98555	0.044666
05/12/2020	0.716835	0.785314	0.068478	0.758387	0.017043
06/12/2020	0.557084	0.613225	0.05614	0.583675	0.012885
07/12/2020	0.862991	1.05911	0.196121	0.972791	0.03966
08/12/2020	0.959586	1.091439	0.13185	1.022259	0.035836
09/12/2020	0.650216	0.801123	0.150905	0.71915	0.03888
10/12/2020	0.555908	0.586872	0.030964	0.570608	0.006815
11/12/2020	0.805809	0.895623	0.089814	0.841352	0.025461
12/12/2020	1.092079	1.24343	0.15135	1.14978	0.03528
13/12/2020	0.670251	0.708416	0.038165	0.683164	0.006243
14/12/2020	0.700227	0.725054	0.024826	0.711836	0.005401
15/12/2020	0.716242	0.770246	0.054003	0.744549	0.011865
16/12/2020	0.805373	0.838324	0.032951	0.822793	0.007113
17/12/2020	0.723016	0.906448	0.18343	0.808758	0.045361

Figura 32: Exemplo ficheiro produzido pela API IPMA

Após a obtenção dos dados da API do IPMA, é feita uma nova requisição a API do OpenWeatherMap com o intuito de obter os dados de previsão meteorológica:

```
public RootWeatherForecast<Daily> GetWeatherForecast(string latitude, string
    longitude)
{
    RestClient client = new RestClient($"{_config.GetConfiguration("
        Openweathermap:APIBASICURI")}onecall");
    var request = new RestRequest();

    request.AddHeader("Accept", "*/*");
    request.AddHeader("Accept-Encoding", "gzip, deflate");
    request.AddHeader("User-Agent", "runscope/0.1");
    request.Method = Method.GET;
    request.AddParameter("lat", latitude, ParameterType.QueryString);
    request.AddParameter("lon", longitude, ParameterType.QueryString);
    request.AddParameter("exclude", "current,minutely,hourly,alerts",
        ParameterType.QueryString);
    request.AddParameter("appid", _config.GetConfiguration("Openweathermap:APIKEY
        "), ParameterType.QueryString);
    request.AddParameter("units", "metric", ParameterType.QueryString);

    var response = client.Execute(request);
        try
    {
        var x = JsonConvert.DeserializeObject<RootWeatherForecast<Daily>>(
            response.Content);
        return x;
    }
```

```
    catch (Exception ex)
    {
        return new RootWeatherForecast<Daily>();

    }
}
```

Listagens 7: Invocação API OpenWeatherMap

Esta API é bastante completa, e o output da invocação da mesma é um JSON com bastante informação, mas como apenas é pretendido saber se a previsão meteorológica para o dia seguinte é de chuva ou não, apenas devemos concentrar a nossa atenção nestas entradas:

```
{
        "weather": [
            {
                "id": 804,
                "main": "Clouds",
                "description": "overcast clouds",
                "icon": "04d"
            }
        ]
}
```

Listagens 8: Exemplo de parte da resposta da API OpenWeatherMap

Tendo toda a informação necessária por parte das APIs públicas é necessário fazer uma caracterização do tipo de solo, baseado nos dados obtidos na invocação da API do IPMA e tendo como base o ficheiro JSON especificado acima.

```
if (EvapoTranspirationForLast30Days <= float.Parse(config["Evapotranspiracao:
    Humida"],
  CultureInfo.InvariantCulture.NumberFormat))
  timeSprinklerActivated = int.Parse(config["TempoDeRegaEmMin:Humida"],
    CultureInfo.InvariantCulture.NumberFormat);

if (EvapoTranspirationForLast30Days > float.Parse(config["Evapotranspiracao:
    Humida"],
    CultureInfo.InvariantCulture.NumberFormat) && EvapoTranspirationForLast30Days
        <=
  float.Parse(config["Evapotranspiracao:SubHumida"]))
  timeSprinklerActivated = int.Parse(config["TempoDeRegaEmMin:SubHumida"],
    CultureInfo.InvariantCulture.NumberFormat);


if (EvapoTranspirationForLast30Days > float.Parse(config["Evapotranspiracao:
    SubHumida"],
```

```
    CultureInfo.InvariantCulture.NumberFormat) && EvapoTranspirationForLast30Days
        <=
  float.Parse(config["Evapotranspiracao:SubHumidaSeca"]))
  timeSprinklerActivated = int.Parse(config["TempoDeRegaEmMin:SubHumidaSeca"],
    CultureInfo.InvariantCulture.NumberFormat);

if (EvapoTranspirationForLast30Days > float.Parse(config["Evapotranspiracao:
    SubHumidaSeca"],
  CultureInfo.InvariantCulture.NumberFormat))
  timeSprinklerActivated = int.Parse(config["TempoDeRegaEmMin:Semiarida"],
    CultureInfo.InvariantCulture.NumberFormat);
```

Listagens 9: Caracterização do solo e cálculo do tempo de rega

Finalmente, é feita a verificação de todos os parâmetros com o intuito de validar se estão reunidas todas as condições para que se proceda à ativação do sistema de rega pelo período de tempo definido no ficheiro .JSON.

```
if (metricsService.IsTheTemperatureOkToSprinkle(activeNodesWith24HoursReadings))
{
  DateTime sunset =
    DatetimeService.UnixTimeStampToDateTime(
      double.Parse(forecast.Daily.FirstOrDefault().Sunset));
  DateTime sunrise =
    DatetimeService.UnixTimeStampToDateTime(
      double.Parse(forecast.Daily.FirstOrDefault().Sunrise));
   //checks if the time that is running is after sunset and before sunrise
   if (DateTime.Now < sunrise && DateTime.Now > sunset)
  {
    if (!forecast.Daily.FirstOrDefault().Weather.FirstOrDefault().Main.ToLower()
        .Contains("rain") ||
      !forecast.Daily.FirstOrDefault().Weather.FirstOrDefault().Main.ToLower()
        .Contains("rain"))
    {
       //activate sprinkler
       objsync.ActivateSprinkler(node.Select(z => z.IdNode).FirstOrDefault());
      Thread.Sleep(timeSprinklerActivated * 60 * 1000);
      objsync.DeactivateSprinkler(node.Select(z => z.IdNode).FirstOrDefault());
```

Listagens 10: Validação dos parâmetros e ativação do sistema de rega

5.2.2 *Ambiente de Sensorização*

Apesar do desenvolvimento do ambiente de sensorização estar fora daquilo que é o desígnio deste projeto, é fundamental dotar o sistema de forma que este esteja preparado para receber dados de sensores reais. Para tal, foi fundamental um trabalho de pesquisa de forma a compreender que sensores poderiam ser utilizados, com o objetivo de obter leituras reais de dados meteorológicos no terreno, para que posteriormente estes possam ser utilizados com o intuito de alimentar o sistema. Conforme definido anteriormente, todo o ambiente de sensorização é constituído por um conjunto de nós (*node*). Cada nó por sua vez é constituído por um conjunto de sensores que lhe estão associados e um microcontrolador. Para que a conceção do ambiente de sensorização seja viável é fundamental ter em consideração os seguintes pontos:

- os dispositivos deverão ter um custo reduzido;
- os dispositivos deverão ter um tamanho reduzido de forma a minimizar o impacto na instalação;
- a ativação deverá poder ser feita de forma manual, tendo em consideração os valores de *input* da aplicação web;
- deverão ter um baixo consumo energético;
- deverão permitir a interação com diversos sensores.

Para tal, e tendo em consideração os pontos referidos acima, é necessário em primeiro lugar utilizar microcontrolador [Tanenbaum A. 2006] para que seja possível a interação com os dispositivos de entrada e saída, como por exemplo os sensores. Assim sendo, o microcontrolador é responsável por obter os valores dos sensores, guardar/ler os dados da memória flash, ler o nível de bateria e comunicar essas informações com a base de dados. Os sensores são responsáveis por detetar a grandeza do mundo físico e transformá-la num sinal elétrico (analógico ou digital), permitindo a leitura do valor pelo microcontrolador. Todo o sistema deverá ser desenhado de forma a aumentar o máximo possível a eficácia energética e, portanto, estará “adormecido a maioria” do tempo, uma vez que a verificação da necessidade de ativação do sistema de rega é feita de 5 em 5 minutos (parâmetro configurável). Obviamente que quanto menor for o tempo entre verificações de necessidade de ativação do sistema de rega por parte do microcontrolador, menor será a duração da bateria. É necessário que o nó possua também um módulo TCP/IP, de forma a enviar dados dos sensores e receber ordens/instruções. É fundamental a existência de um Real-Time Clock (RTC) para obter com facilidade a data e hora para serem inseridas na base de dados juntamente com os dados obtidos através dos sensores.

Já o microcontrolador em si, geralmente encontra-se embutido numa Placa de Circuito Impresso (PCI). Tendo em conta o crescimento da Internet of Things [Pfister C. 2011], podemos ter acesso uma enorme quantidade de microcontroladores que permitem com muita facilidade uma integração com outros componentes de Hardware.

Arduino

O Arduino é um pequeno computador de placa única, com uma interface amigável, o que facilita a utilização por parte de qualquer pessoa, mesmo não sendo especializadas em eletrónica, engenharia ou programação. É de baixo custo, multiplataforma e de fácil programação [Di Justo P. 2012]. Geralmente os dispositivos são baseados em microcontroladores da ATMEL. No caso do Arduino Uno, este possui um chip ATMEL AVR ATMEGA328P como microcontrolador principal

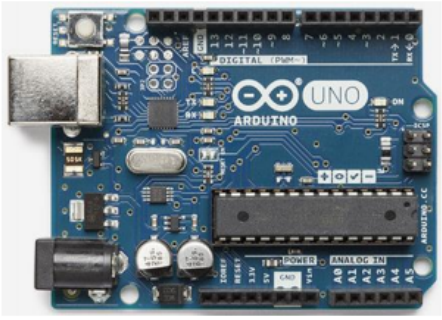


Figura 33: Arduino Uno

O arduino permite ainda aumentar os seus recursos adicionando placas adicionais que podem ser conectadas a placa base do arduino, como por exemplo *ethernet shield, Wifi shield, e Motor Shield*.

Sensores

De forma a ser possível monitorizar os valores meteorológicos no terreno, é necessário proceder a instalação de sensores que irão estar ligados ao controlador.

Sensor de temperatura e humidade relativa do ar

O sensor escolhido para a obtenção da temperatura e humidade relativa do ar (HR) foi o DHT22, por ser fácil de encontrar no mercado e ter desempenhos bastante bons apesar de ser básico e lento comparado com outros sensores no mercado. Com a utilização deste sensor também é possível obter a temperatura do ar Aosong Electronics. Este sensor tem as seguintes características:

- tempo de conversão a 12 bit (1 segundo);
- erro máximo de humidade relativa de +/- 5%;
- erro máximo de temperatura de +/-0.5 ºC;
- gama de Funcionamento: 0 a 100% de HR e -40ºC a +80ºC;
- tensão de alimentação de 3.3V a 6V.

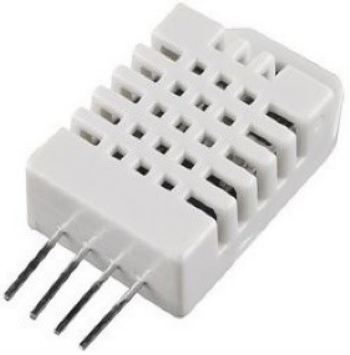


Figura 34: Sensor DHT22

Sensor de pressão

No caso do sensor de pressão, uma sugestão de sensor poderia incidir sobre o BMP180 pelas mesmas razões da escolha do sensor anterior. Este sensor tem as seguintes características:

- calibrado de fábrica;
- resolução de 0.02hPa (0.17m);
- gama de Pressão entre os 300hPa a 1100hPa;
- tensão de alimentação entre os 1.8V a 3.6V;
- dimensões reduzidas.

Figura 35: Módulo de sensor de pressão (BMP180)

Sensor de controlo de fluxo de Água

Este sensor tem como principal função medir o fluxo de água libertado a partir de uma mangueira ou conduta e permite que apenas a quantidade de água necessária seja expelida. O sensor escolhido foi o YF-S201 e conta com as seguintes características:

- caudal de água: 1 – 30L/min;
- diâmetro de saída 12mm;
- diâmetro de entrada: 9mm;

- tensão de alimentação entre 5V e 18V.

Figura 36: Módulo de fluxo de caudal YF-S201

Monitorização da Velocidade do Vento

Este instrumento tem como principal função medir a velocidade do vento. É composto por uma concha, cataventos e um circuito. Quando o vento passa por este sistema faz com que a concha gire, gerando corrente. As principais características deste instrumento são:

- erro associado +/- 3%;
- velocidade do vento inicial 0.4 - 0.8m/s;
- tensão de alimentação: 9V a 24V;
- faixa de medição efetiva: 0-30m / s.

Figura 37: Kit Anemômetro (SEN0170)

Módulo TCP/IP

Para a comunicação TCP/IP, é sugerido a utilização do ENC28J60 da Microship. Este é um controlador Ethernet 10BASE-T num circuito integrado de 28 pinos e suporta comunicação [SPI Microship 2012]. Este módulo conta com uma porta ethernet RJ45 e tem dimensões bastante reduzidas.

Figura 38: Módulo TCP/IP (ENC28J60)

5.2.3 *Algoritmo de Machine Learning*

Conforme foi descrito nos capítulos anteriores, este algoritmo tem como principal função fazer a previsão de ocorrência de chuva para um determinado local, recorrendo aos dados recolhidos através da API pública DevMeteoStat. Para tal recorreu-se a um conjunto de técnicas de *machine learning* utilizando uma metodologia CRISP-DM. Conforme descrito no Capítulo 2.1, esta metodologia assenta sobretudo nas seguintes etapas:

1. entendimento do problema (Business Understanding);
2. compreensão dos dados (Data understanding);
3. preparação dos dados (Data preparation);
4. modelação (Modeling);
5. avaliação (Evaluation);
6. aplicação (Deployement).

Neste capítulo é dado um maior ênfase sobretudo aos primeiros 4 itens, sendo que o quinto ponto será discutido em maior detalhe no Capítulo 6.

Entendimento do problema

De forma a permitir que o sistema de rega não seja ativado de forma desnecessária, um dos parâmetros que é fundamental para garantir a ativação do mesmo é a não ocorrência de aguaceiros ou chuva nas horas seguintes ao momento de análise. Para tal, tendo em consideração os dados de um determinado local, é necessário prever a possibilidade de existência de chuva, tendo em conta o histórico de leituras anteriores. Assim sendo, e para obter esses dados de histórico de leituras, foi necessário recorrer a uma API pública DevMeteoStat, em que são fornecidos como input as coordenadas geográficas do local em análise e o intervalo de datas segundo o qual se pretende obter o histórico de leituras, e como resultado é recebido um dataframe com o histórico de leituras.

```
//Encontrar Estacao metereologica atraves das coordenadas

stations = Stations()
stations = stations.nearby(41.3021183, -7.744898)
station = stations.fetch(1)


//Definicao das datas de inicio e de fim
start = datetime(1948, 1, 1)
end = datetime(2017, 12, 14)

// Obter dados diarios
data = Daily(station, start, end)
data = data.fetch()


dataframe =data[["DATE","PRCP","TMAX","TMIN","RAIN"]].copy()
dataframe.head()
```

Listagens 11: Exemplo de invocação APi DevMeteoStat a partir das coordenadas

Neste contexto em específico, foram enviados para a API as coordenadas geográficas da cidade de Vila Real, e é pretendido obter o histórico de leituras diário desde o dia 1/1/1948 até ao dia 14/12/2017.

Compreensão dos dados

Como resultado do exemplo de invocação enunciado no Capítulo 7.3.1, é obtido um dataframe com a seguinte estrutura

- DATE – Data em que foi efetuada a leitura
- PRCP – Precipitação diária total em mm
- TMIN – Temperatura do ar mínima registada no dia em questão (ºC)
- TMAX – Temperatura do ar máxima registada no dia em questão (ºC)
- RAIN – Indicação de existência de chuva

Out[328]:

	DATE	PRCP	TMAX	TMIN	RAIN
0	01/01/1948	0.47	10.555556	5.555556	True
1	02/01/1948	0.59	7.222222	2.222222	True
2	03/01/1948	0.42	7.222222	1.666667	True
3	04/01/1948	0.31	7.222222	1.111111	True
4	05/01/1948	0.17	7.222222	0.000000	True

Figura 39: Exemplo de resposta da Api DevMeteoStat (Vila Real)

Neste caso, é pretendido prever o valor da variável em estudo RAIN, tendo em consideração as variáveis DATE, PRCP, TMAX E TMIN. Este dataframe conta com 10900 entradas em que choveu, e 14648 entradas em que não foi verificada a existência de chuva.

Contagem de ocorrencias de chuva

```
In [175]: df['RAIN'].value_counts()
Out[175]: False    14648
          True     10900
          Name: RAIN, dtype: int64
```

Figura 40: Número de ocorrências de chuva

Seguidamente foi feita uma representação gráfica dos dados de modo a ter uma perceção de como os dados estão relacionados . Para tal recorreu-se a um *pairplot* da biblioteca seaborn.

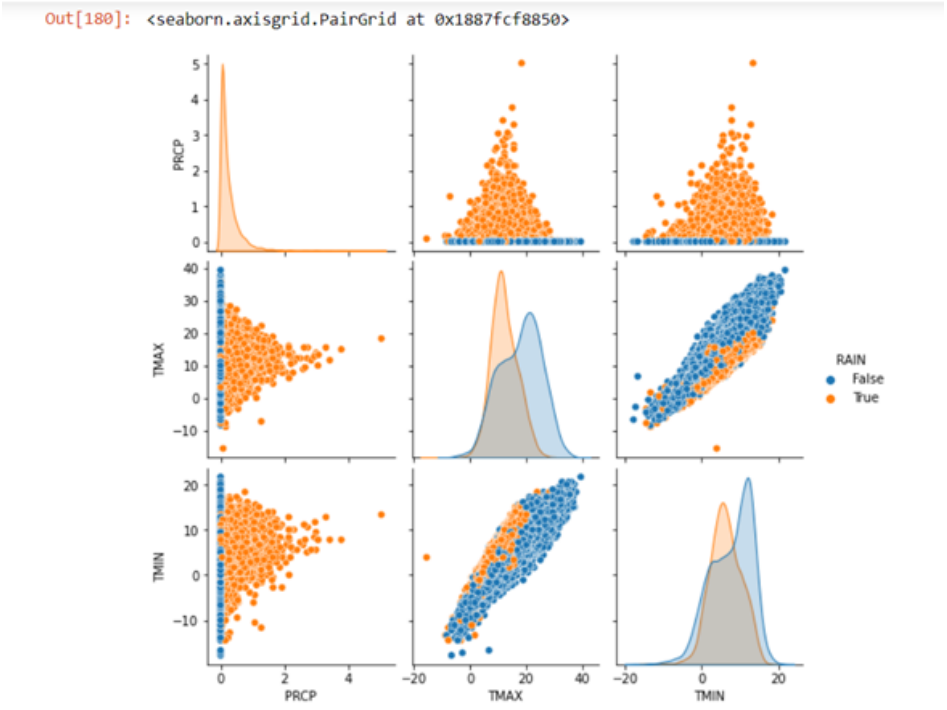


Figura 41: Pairplot dos dados

Através da utilização do *pairplot* é possível obter duas figuras básicas. Um histograma e um gráfico de dispersão (*scatter plot*). Com o histograma, na diagonal, permite-nos observar a distribuição de uma variável e os gráficos de dispersão, permitem-nos observar a relação entre as variáveis. É importante salientar que uma eventual relação entre duas varáveis não implica que haja uma relação causal entre elas. Para estudar uma relação de causalidade entre as varáveis é necessário recorrer a uma matriz de correlação:

```
In [182]: #Heatmap for correlation between variables
          import seaborn as sns
          plt.figure(figsize=(8, 8))
          sns.heatmap(df.corr(),annot=True)

Out[182]: <AxesSubplot:>
```

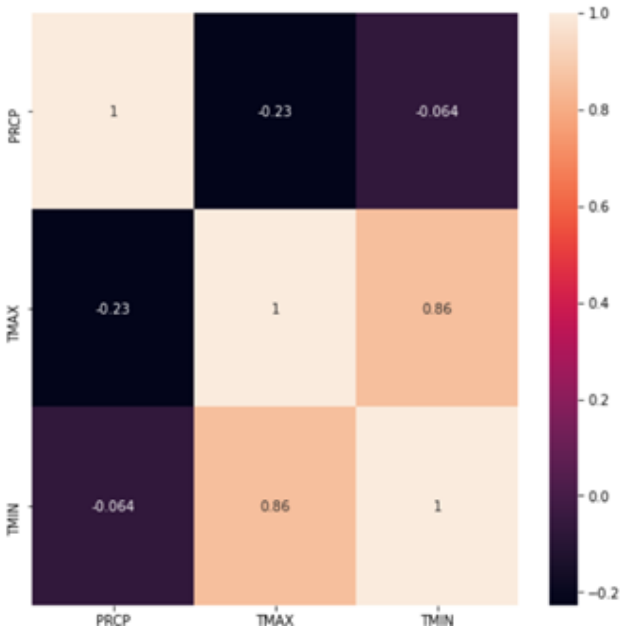


Figura 42: Matriz de correlação das variáveis em estudo

Através da análise da Figura 42 podemos concluir que TMIN e TMAX estão positivamente correlacionadas, ou seja, ambas variam numa relação diretamente proporcional.

Seguidamente os valores de RAIN foram convertidos para valores numéricos de forma a facilitar a visualização da *label*, bem como a coluna de DATE foi convertida num formato de data válido. Conforme é possível observar na Figura 43, existem 14648 dias em que não foi detetada chuva e 10900 dias em que foi efetivamente detetada chuva.

```
In [185]: sns.countplot(x=df['rain'].values)

Out[185]: <AxesSubplot:ylabel='count'>
```

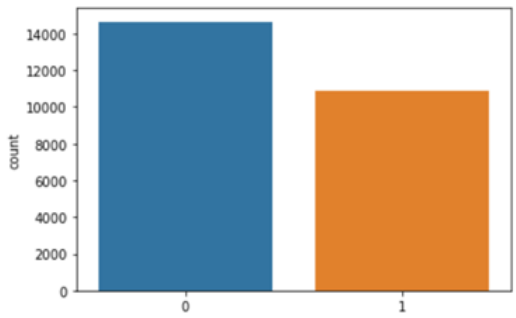


Figura 43: Número de ocorrências de chuva (gráfico)

Finalmente, através da Figura 44 podemos observar que há uma maior ocorrência de chuva nos meses correspondentes ao inverno, e, portanto, podemos concluir que há relação entre a coluna DATE e a variável em estudo.

```
In [189]: a=df.pivot_table(index=['rain'], columns='month', aggfunc='size', fill_value=
          key=a.loc[1.0].index
          value=a.loc[1.0].values

          import seaborn as sns
          ax = sns.barplot(x=key, y=value)
```

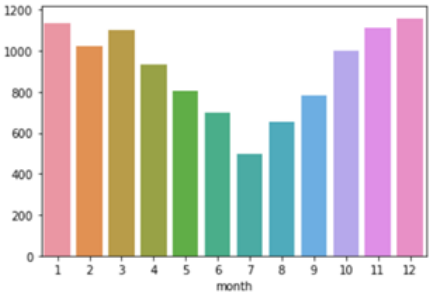


Figura 44: Distribuição da precipitação por mês

Preparação dos dados

O processo de preparação dos dados é fundamental, no que toca à execução de algoritmos de *machine learning*, uma vez que dados errados, ou a ausência dos mesmos, pode induzir o algoritmo em erro. Esta é uma situação que é facilmente evitável quando é feita uma boa preparação de dados. Nesse sentido, o primeiro passo consiste sobretudo em analisar a presença de eventuais valores nulos no *dataset*, para tal recorreu-se ao seguinte comando:

Contagem dos valores nulos

```
In [176]: for col in df:
              print(col)
              print(df[col].isnull().sum().sum())
```

```
DATE
0
PRCP
3
TMAX
0
TMIN
0
RAIN
3
```

Figura 45: Contagem dos valores nulos

Através da execução do comando representado na Figura 45, podemos concluir que neste *dataset* em específico existem 6 valores nulos, 3 relativos a precipitação e 3 relativos ao RAIN. Tendo em conta que o *dataset* é

bastante extenso e que o número de entradas nulas é residual, então optou-se por remover estas entradas no *dataset*.

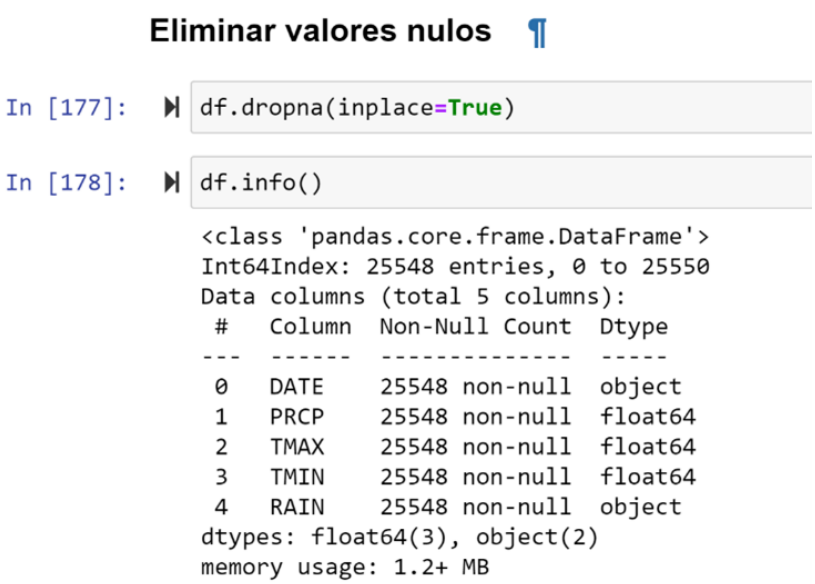


Figura 46: Remoção dos valores nulos do dataset

Modelação

Para o problema especificado, o *dataset* foi dividido em 25% para teste e 75% para treino . Uma vez que se trata de um *dataset* relativamente grande, foram utilizados ainda 3 algoritmos distintos de forma a ser possível obter uma comparação de resultados, tendo em consideração diferentes algoritmos. Os algoritmos utilizados foram o *logistic regression*, a *decision tree* e o *random forest classifier*.

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
    random_state = 42)
```

Listagens 12: Divisão do dataset

No caso do *Random Forest Classifier* e da *Decision Tree* foi ainda utilizado o método de *cross validation*, com o intuito de evitar a existência de *overfitting*.

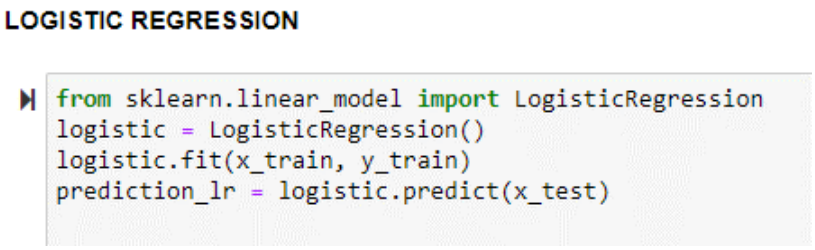


Figura 47: Logistic Regression

DECISION TREE

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier()
tree.fit(x_train, y_train)
prediction_dt = tree.predict(x_test)
cv_dict = cross_validate(tree, X, y, return_train_score=True)
```

Figura 48: Decision Tree com cross validation

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier()
clf.fit(x_train,y_train)
pred=clf.predict(x_test)
cv_dict = cross_validate(clf, X, y, return_train_score=True)
```

Figura 49: Random Forest com cross validation

5.3 RESUMO DO CAPÍTULO

Neste capitulo foi analisado ao nível prático cada um dos módulos em maior detalhe. De seguida foi apresentado o funcionamento do Automatic Service, analisando a forma como cada uma das suas funcionalidades se processava. O Automatic Service, tem em considerção um conjunto de métricas, a partir das quais tomará a decisão de ativar ou desativar o sistema de rega. Posteriormente, foi feita a análise e implementação do módulo de *machine learning*, foi exposto como foi feito o pré-processamento dos dados e a implementação de cada um dos algoritmos. Os resultados obtidos por cada um dos algoritmos são apresentados no próximo capítulo.

6

RESULTADOS OBTIDOS

Tendo em conta os diferentes graus de complexidade de cada um dos módulos, vale a pena analisar com maior detalhe os resultados obtidos do desenvolvimento da aplicação web e do algoritmo de *machine learning*, uma vez que estes módulos são os que apresentam maior complexidade.

6.1 APLICAÇÃO WEB

Conforme descrito no Capítulo 4, a aplicação web funciona como camada de apresentação do sistema e é o único módulo do sistema que tem contacto direto com o cliente final. É através deste módulo que o utilizador pode registar um novo Nó, consultar os dados recolhidos pelos sensores / estações meteorológicas bem como ativar de forma manual o sistema de rega ou qualquer hardware que esteja presente no terreno.

Login

Inicialmente o utilizador terá de se registar e fazer login de modo a poder consultar toda a informação que lhe esteja associada.

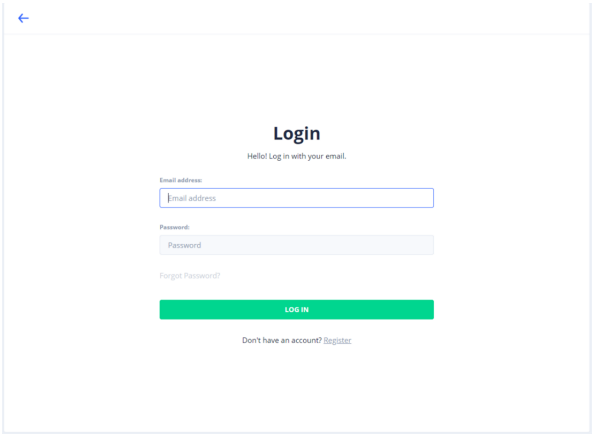


Figura 50: Área de login

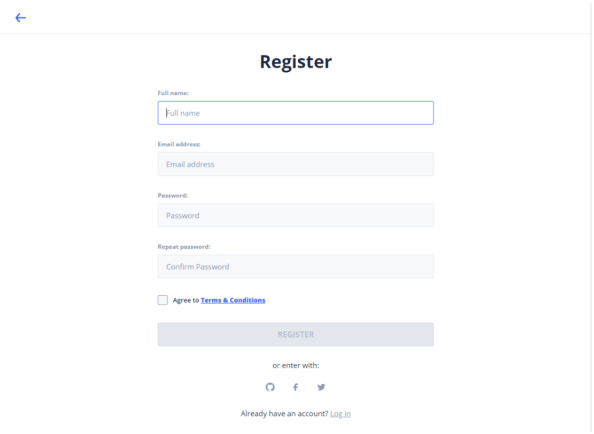


Figura 51: Área de registo

Dashboard

Após o utilizador autenticar-se com sucesso na aplicação terá acesso a página de *dashboard* da aplicação web

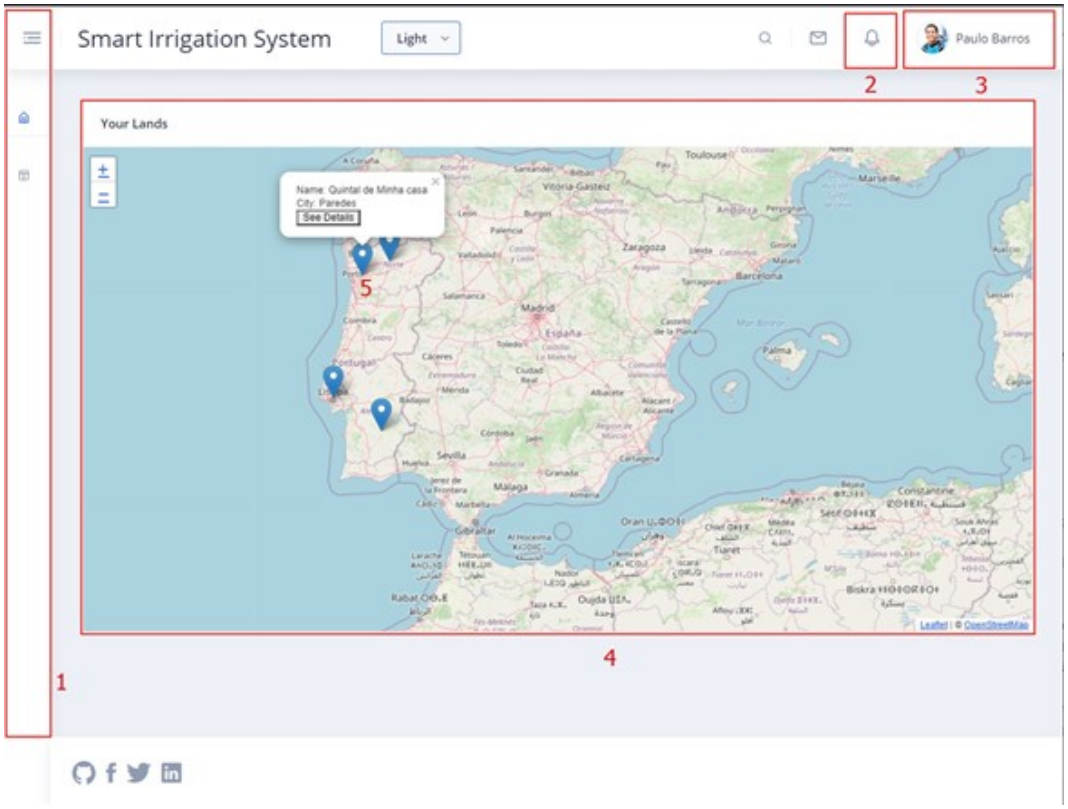


Figura 52: Dashboard Aplicação web

Na página de dashboard (Figura 52), o utilizador (3) terá acesso a um menu (1) onde poderá navegar na aplicação. Por predefinição, o menu permitirá ao utilizador registar um novo terreno ou consultar todos os Nós

que tem registados no sistema a partir de um mapa (4). Através do clique no marker do mapa, o utilizador verá uma pequena modal com a localização do Nó (5). Finalmente, o utilizador terá acesso a todas as notificações reportadas pelo automatic service clicando no sino (2). Caso seja reportado algum erro pelo automatic service, o utilizador verá o dashboard da seguinte forma:

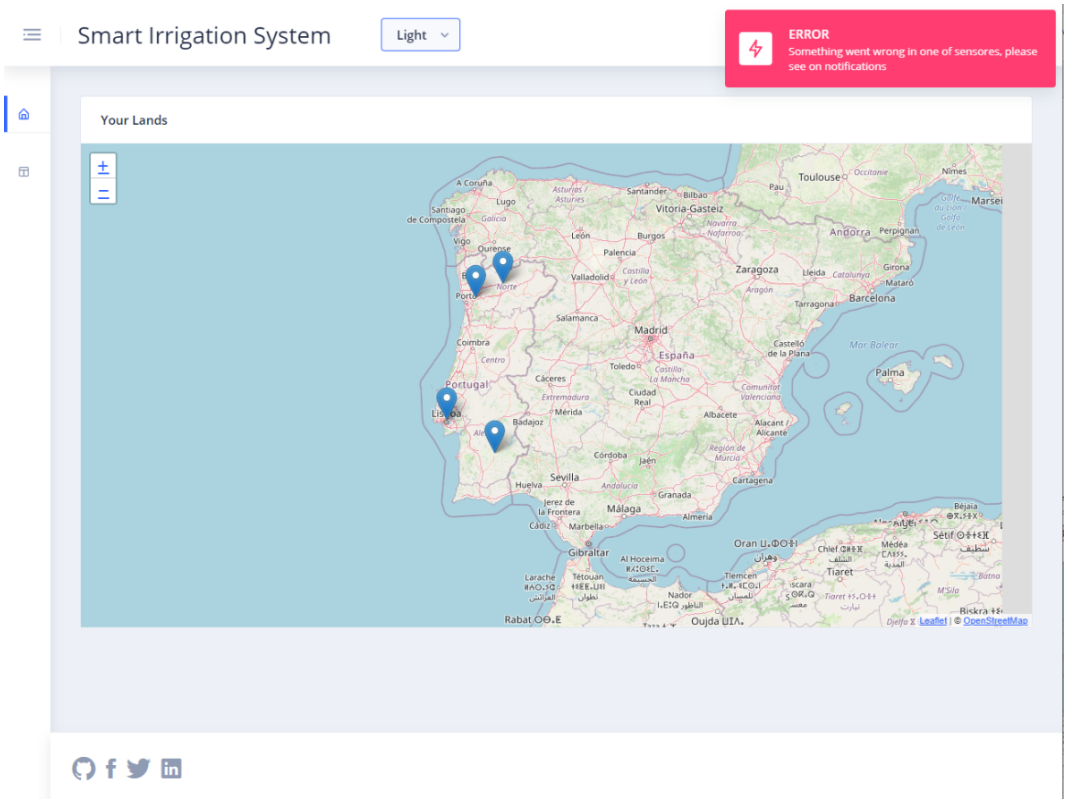


Figura 53: Dashboard com erros reportados pelo automatic service

Neste caso um aviso será mostrado sempre que o utilizador se encontre na página de dashboard, informando que existe um erro na leitura dos dados e que deverá consultar a página de notificações (Figura 52 ponto 1).

Notificações

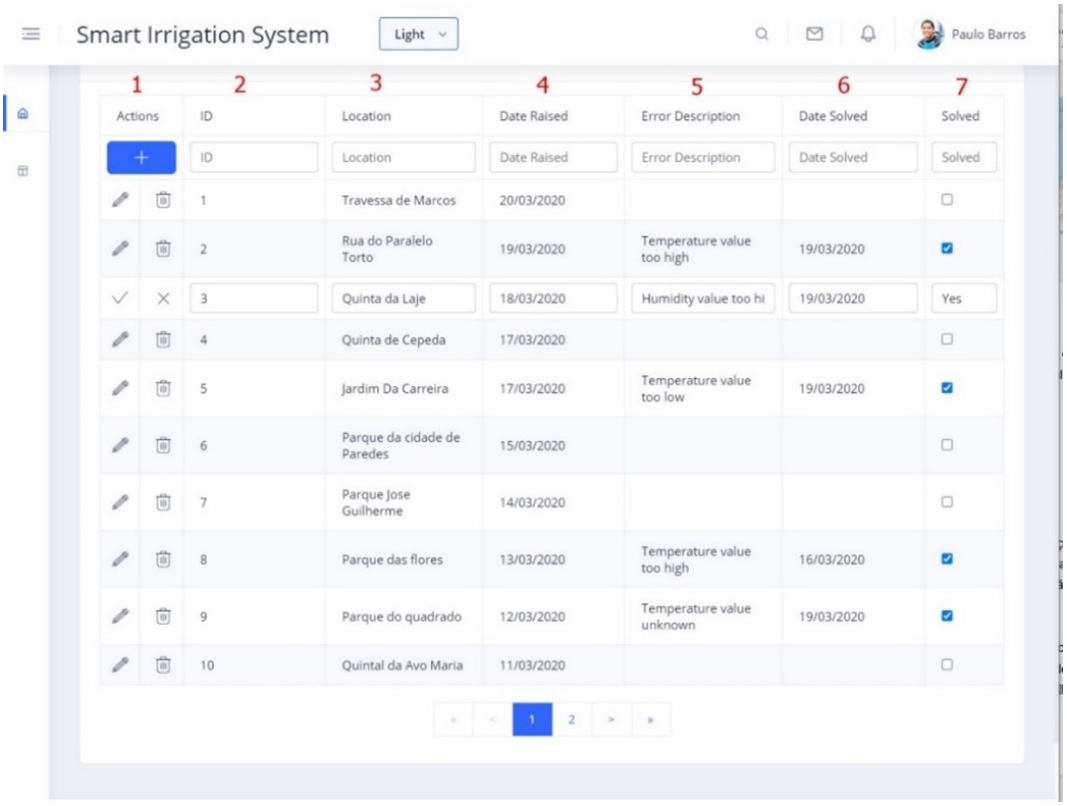


Figura 54: Página de Notificações Web Application

Nesta página o utilizador terá acesso a todas informações reportadas pelo automatic service, podendo editar e/ou eliminar erros.

Esta página permite que o utilizador:

1. elimine ou edite o erro;
2. obter ID do erro;
3. consultar a descrição da localização do nó cujo erro foi reportado;
4. consultar em que data o erro foi lançado;
5. consultar / Editar descrição do erro;
6. consultar / Editar data em que o erro foi resolvido;
7. consultar /Editar estado do erro (Resolvido / Não Resolvido).

Adicionar um novo nó

O utilizador poderá optar por adicionar um novo nó. Caso o terreno já esteja registado no sistema, então um novo nó é adicionado a esse terreno, caso contrário, um novo terreno é registado. Para isso o utilizador terá de inserir as informações do terreno no sistema (Figura 55)

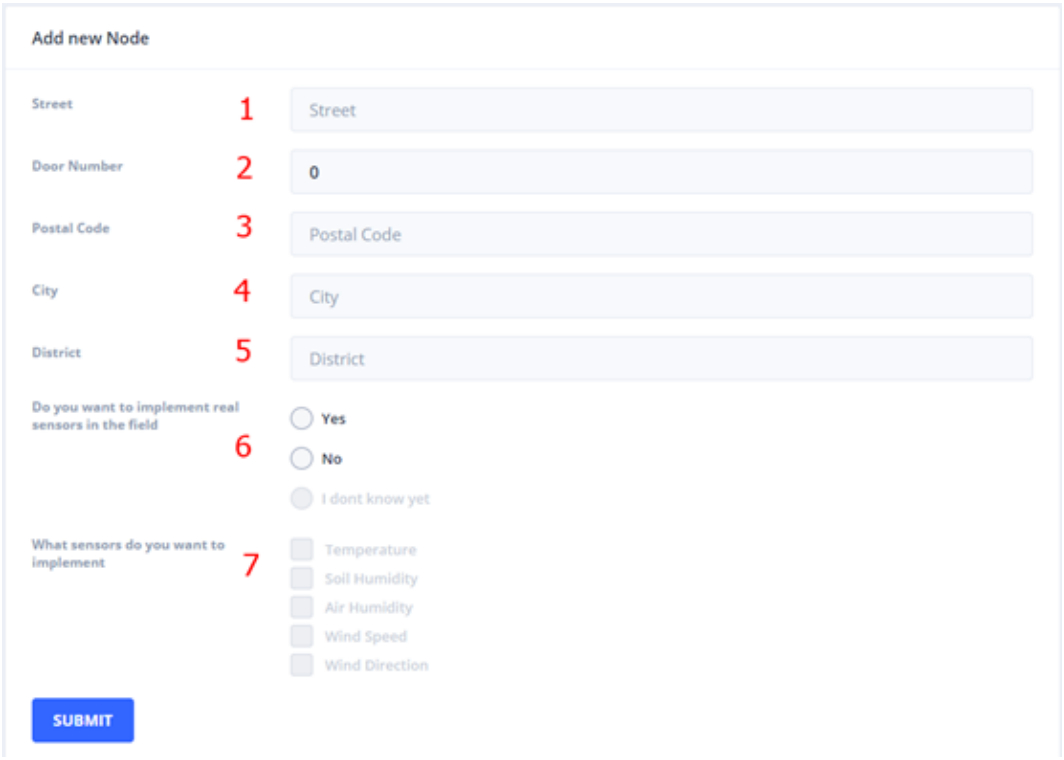


Figura 55: Registo de um novo nó

É importante salientar os pontos 6 e 7 da Figura 55. O utilizador poderá ou não optar por inserir sensores reais. Caso pretenda utilizar sensores reais, deve especificar no ponto 7 que sensores pretende que sejam inseridos. Caso contrário, então é localizada a estação meteorológica mais próxima, através da invocação da API DevMeteoStat, tendo em conta as coordenadas da localização inserida (Geocoding). Assim sendo, antes de se proceder à inserção de um novo nó, e conforme especificado no diagrama de atividades da Figura 20, é necessário invocar a API PositionStackAPI de forma a executar o geocoding:

Figura 56: Exemplificação de geocoding

Se o utilizador optar por recorrer a sensores reais para o nó criado, então uma nova entrada é criada na Tabela “Sensors” com uma breve descrição do sensor, o tipo, a localização, o Nó que esta acoplado ao sensor bem como se o mesmo se encontra ativo ou não. Por outro lado, se o utilizador não pretender recorrer a sensores reais, então é feita uma nova requisição a uma API pública de forma a obter os dados da estação meteorológica mais próxima das coordenadas obtidas anteriormente. Desta forma a API utilizada é novamente a DevMeteoStat.

Figura 57: Obtenção da estação meteorológica mais próxima

A resposta da API é um ficheiro .JSON com a seguinte estrutura:

Parameter	Description	Type
id	The Meteostat ID of the weather station	String
name	Object containing the name of the weather stations in different languages	Object
active	A boolean value which is true if the weather station reported data within the previous 90 days	Boolean
distance	The distance to the geo location defined in the request	Float

Figura 58: Estação Meteorológica mais próxima (Estrutura)

Após a obtenção da estação meteorológica mais próxima, a mesma é inserida na base de dados. É importante salientar que, quer a localização da estação meteorológica, quer a localização do novo nó, são inseridos na Tabela “Location ”.

Painel de Controlo

Se no Dashboard o utilizador clicar num dos markers definidos na Figura 53 e optar por ver mais detalhes, será reencaminhado para a página de Painel de Controlo do nó selecionado.

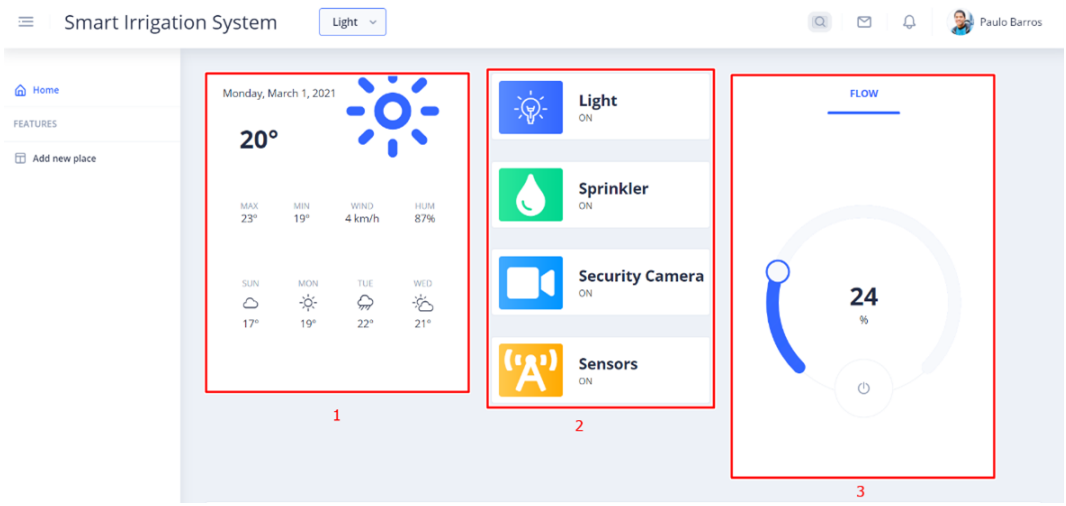


Figura 59: Painel de controlo (1)

Nesta primeira parte do Painel de Controlo, o utilizador pode consultar as previsões meteorológicas para os próximos dias no local onde o nó está localizado (1), pode ainda ativar ou desativar de forma manual um conjunto de controlos presentes no terreno, tais como os sensores, os aspersores, as luzes etc (2), e pode ainda controlar a percentagem de água que é expelida pelos aspersores.

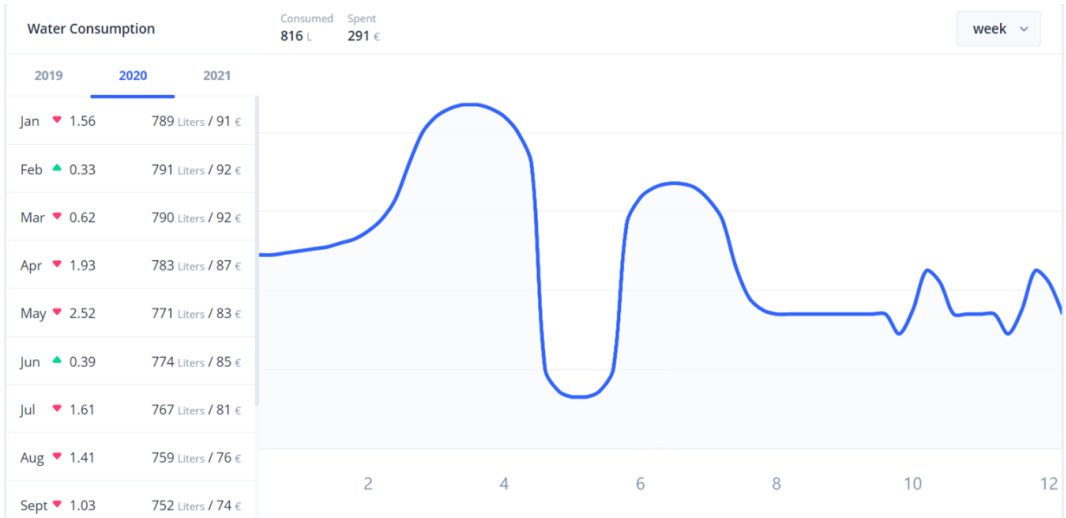


Figura 60: Painel de controlo (2)

Na segunda parte do Painel de Controlo é possível obter uma estimativa da quantidade de água despendida, bem como o custo associado a essa quantidade de água. É possível ainda obter uma comparação da quanti-

dade de água / custo gasto noutros meses e noutros anos, de forma a poder estabelecer uma comparação com os valores atuais.

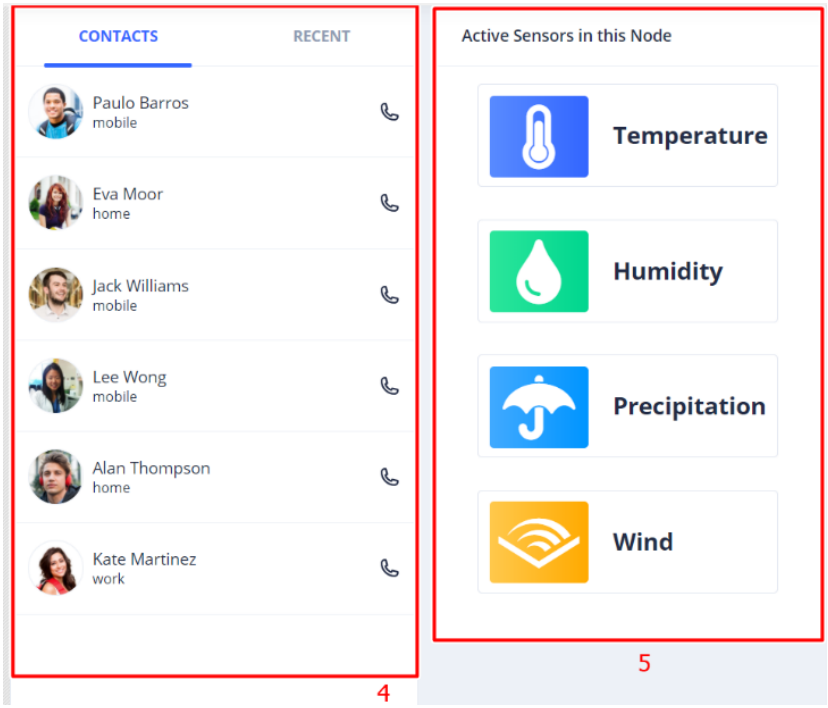


Figura 61: Painel de controlo (3)

Finalmente, no painel de controlo o utilizador pode encontrar os contactos telefónicos de pessoal que seja relevante para *troubleshooting*, por exemplo (4). No ponto 5, mediante os sensores que o utilizador tenha instalado, poderá ver de uma forma gráfica os detalhes das leituras para cada um dos sensores, bastando apenas clicar no sensor segundo o qual se pretende obter mais detalhes. É importante salientar que esta opção não estará disponível para os utilizadores que não possuam sensores reais implementados no terreno.

Detalhes de leituras

Conforme foi definido no capítulo anterior, o sistema permite que o utilizador possa consultar de forma gráfica os detalhes das leituras efetuadas pelos sensores. As informações contidas nestes gráficos são altamente personalizáveis com o objetivo de que o cliente tenha acesso à informação que considera mais relevante, tendo em conta as suas necessidades. Na Figura 62 é possível observar um conjunto de gráficos que podem ser obtidos na aplicação web.

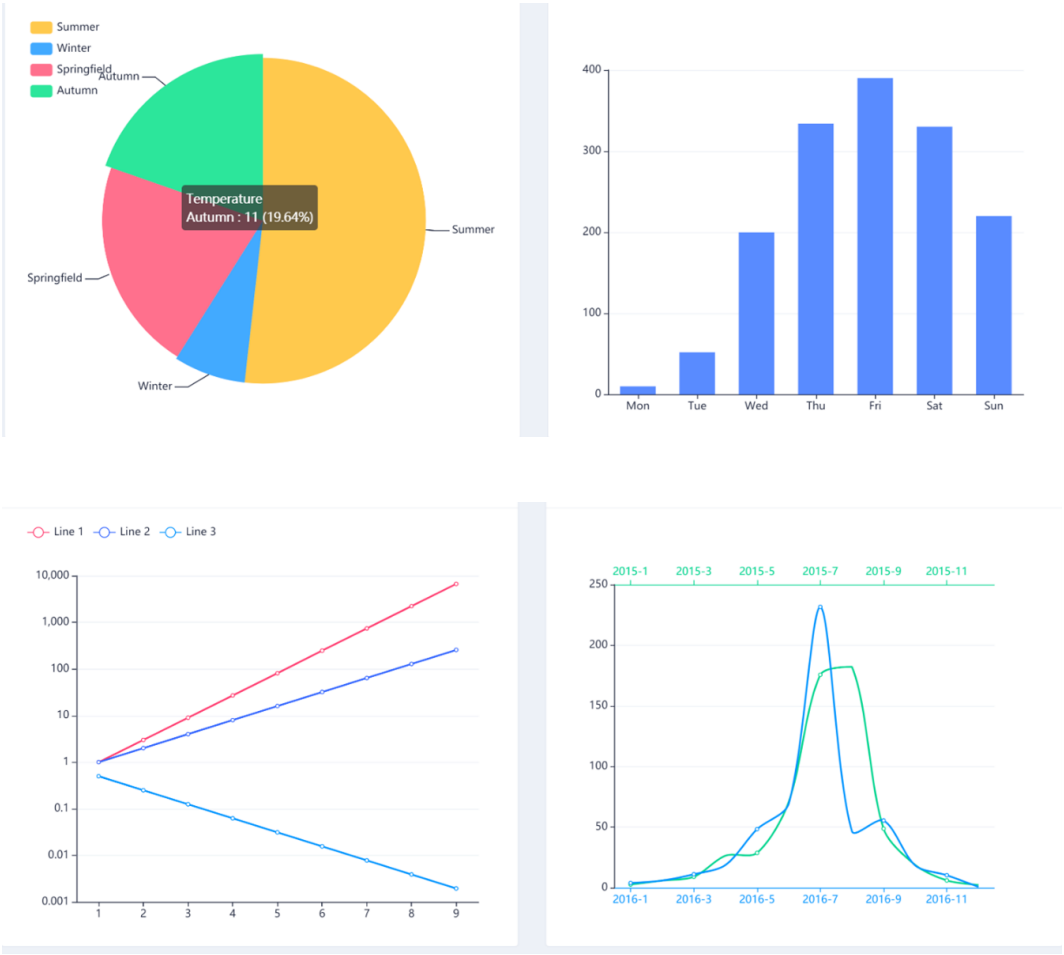


Figura 62: Exemplos de gráficos

No primeiro gráfico pode-se obter a informação relativa á temperatura, organizada por estação do ano. No segundo gráfico a mesma informação encontra-se organizada num gráfico de barras e está estruturada por dias da semana. No terceiro gráfico é possível estabelecer uma comparação entre as temperaturas do nó atual e dos demais nós localizados na mesma cidade. Finalmente no quarto gráfico é permitido estabelecer uma comparação entre a temperatura mensal no ano 2015 e 2016.

6.2 ALGORITMOS DE MACHINE LEARNING

No caso dos algoritmos de *Machine learning* os resultados obtidos foram bastante positivos, uma vez que se obteve uma percentagem de acerto de 100% tanto nas *decision trees* como no *random forest classifier* e uma eficácia de cerca de 93% no *logistic regression*. O facto de ser utilizado *cross validation* permite atenuar a possibilidade de existência de *overfitting*.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3683
1	1.00	1.00	1.00	2704
accuracy			1.00	6387
macro avg	1.00	1.00	1.00	6387
weighted avg	1.00	1.00	1.00	6387

Figura 63: Classification Report Decision Tree

Através da observação da Figura 63 podemos observar uma eficácia de 100%, significa que o algoritmo acertou 100% na previsão da existência/ausência de chuva, com uma precisão e um *recall* de 100%. Estes resultados foram semelhantes no *Random Forest classifier*.

```
array([[3683,    0],
       [   0, 2704]], dtype=int64)
```

Figura 64: Matriz de confusão Decision Tree

Segundo a matriz de confusão da Decision Tree (Figura 64) podemos, de uma outra perspetiva, observar que o algoritmo teve 0 falsos positivos e 0 falsos negativos, o que vem confirmar o resultado de 100% de eficácia observado no *classification report*. Tendo em consideração os resultados obtidos pela *decion tree* e pelo *random forest classifier*, o algoritmo que melhor se adequa ao contexto do problema é a *Decision Tree*, uma vez que este é bastante mais leve que o *Random Forest*, o que resulta num tempo de execução e numa necessidade de recursos computacionais muito mais baixa.

6.3 RESUMO DO CAPÍTULO

Neste capítulo foi apresentado em maior detalhe o funcionamento da aplicação web e dos algoritmos de *machine learning*. Inicialmente foi demonstrado de que forma a aplicação web se comporta, exibindo algumas das páginas onde serão feitas as operações de maior relevância para o sistema. Finalmente, neste capítulo foram apresentados e discutidos os resultados obtidos por parte de cada um dos algoritmos de *machine learning* concluindo que o algoritmo que melhor se adequaria as necessidades do sistema seria a *Decision Tree*

7

CONCLUSÕES E TRABALHO FUTURO

Esta dissertação aborda um dos grandes problemas existentes nos dias de hoje: a má gestão da irrigação e dos espaços verdes nas áreas públicas. Há muito desperdício desnecessário de água e na maioria das vezes o cuidado deficiente da vegetação. Com isso em mente foi criado um protótipo a partir de um sistema inteligente para otimizar o sistema de irrigação e obter dados de temperatura, evapotranspiração, data e hora, para que a rega da vegetação seja feita em condições o mais próximas das ideais possível.

A aplicação web revela-se um trunfo importante para os cuidadores dos espaços verdes, uma vez que disponibiliza os dados do solo numa interface simples e de fácil utilização, permitindo aos utilizadores fazer uma melhor gestão e otimização no cuidado da vegetação. O automatic service é o *core* da aplicação e é responsável por, tendo em conta o histórico de dados e a previsão meteorológica para um determinado local, tomar a decisão de ativar ou não o sistema de rega. Para além disso, o algoritmo de *machine learning* permite que haja uma maior eficácia de previsão de existência de chuva ou aguaceiros, o que reduz de forma substancial a possibilidade de ativação do sistema de rega em dias em que não seria necessário.

Como trabalho futuro pretende-se proceder à implementação real do módulo de sensorização, de forma que as leituras presentes na base de dados, bem como os dados apresentados nos *dashboards*, contenham leituras reais. Tendo em conta o grande volume de informações seria oportuno recorrer à implementação de um *datawarehouse* com o objetivo de permitir um acesso simplificado aos dados de histórico, uma unificação dos dados desconexos e tornar possível que os dados se encontrem mais centralizados, diminuindo as inconsistências nas informações que são solucionadas antes mesmo do carregamento dos dados. Finalmente, seria interessante permitir a possibilidade de acesso aos dados presentes na aplicação web, através de uma aplicação *mobile*, permitindo assim uma melhor experiência para o utilizador, facilitando o acesso aos dados.

REFERÊNCIAS BIBLIOGRÁFICAS

Allbesmart. Smart irrigation system in castelo branco, 2021. URL http://www.allbesmart.pt/smart_irrigation.php.

Amit Y. and Geman D. Shape Quantization And Recognition With Randomized Trees. *NEURAL COMPUTATION*, 9(7):1545–1588, 1997.

Aosong Electronics. Datasheet dht22. URL https://cdnshop.adafruit.com/datasheets/DHT22.pdf.

Arduino. Home. URL https://www.arduino.cc/.

ATMEL. Microcontrolador avr atmega328p. URL http://www.atmel.com/devices/ATMEGA328P.aspx.

Bigas H. *The Global Water Crisis: Addressing an Urgent Security Issue*. Papers for the InterAction Council, 2011.

Branchi P. and Fernández-Valdivielso C. and Matias I. Analysis matrix for smart cities. *Future Internet*, 6(1): 61–75, 2014.

Breiman L. Random forests. 45(1):5–32, 2001.

Caragliu A. and Del Bo C. and Nijkamp P. Smart Cities In Europe. *Journal of Urban Technology*, 18(2):65–82, 2011.

Chui M. and Malhotra S. *AI Adoption Advances, But Foundational Barriers Remain*. 2018.

Di Justo P. and Gertz E. *Environmental monitoring with arduino: building simple devices to collect data about the environment*. O'Reilly Media, Inc, 2012.

Diario Oficial República Portuguesa. Lei nº 277.2019, de 2 de outubro de 2009. dispõe sobre o programa do governo no tocante à modernização administrativa e à melhoria da qualidade dos serviços públicos com ganhos de eficiência, 2009. URL https://dre.pt/application/conteudo/491175.

Evans E. Domain-driven design reference: Definitions and pattern summaries. pages 44–100, 2014.

Fernandes S. *Smart Cities – Inclusão, Sustentabilidade*. Resiliência, 2017.

Goodfellow I. and Bengio Y. and Courville A. . *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

GreenIQ. Greeniq smart garden hub, 2021. URL https://produktinfo.conrad.com/datenblaetter/1500000-1599999/001574778-an-01-en-.

Harishankar S. and Sathish K. and Sudharsan K. P. and Vignesh U. and Viveknath T. '"solar powered smart irrigation system'. *Advance in Electronic and Electric Engineering*., 4:341–346, 2014.

Hyafil L. and Rivest R. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

IPMA.pt, 2021. URL https://www.ipma.pt/pt/agrometeorologia/mapas/diario/index.jsp?page=dts5_co.xml.

Ishak S. N. 'Smart Home Garden Irrigation System With Rasberry Pi'. 16:24–25, 2008.

Jablonski S. and Bussler C. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.

Jordan M. and Kleinberg J. and Schölkopf B. . Institute for systems and robotics – pushing science forward, 2016. URL http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop-PatternRecognitionAndMachineLearning-Springer2006.pdf.

Kerzner H. *Project Management: A Systems Approach To Planning, Scheduling, And Controlling*. 10th edition, 2001.

Lewis R. An Introduction To Classification And Regression Tree (CART) Analysis. *Annual Meeting Of The Society For Academy Emergency Medicine, San Francisco*, 16(1):503–505, 2000.

Lucas H. C. *Information Systems Concepts for Management*. McGraw-Hill, 1990.

Martyusheva O. *Smart Water Grid*. USA: Department of Civil and Enviromental Engineering, Colorado State University, 2014.

Mccullagh P. and Nelder J. A. *Generalized Linear Models*. 1989.

Microship. Datasheet dht22, 2012. URL http://ww1.microchip.com/downloads/en/DeviceDoc/39662e.pdf.

Moffitt, Sean. The top 30 emerging technologies (2018–2028), May 2018. URL https://medium.com/@seanmoffitt/the-top-30-emerging-technologies-2018-2028-eca0dfb0f43c.

Murphy K. P. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2013.

Murthy S. K. Automatic Construction Of Decision Trees From Data: A Multidisciplinary Survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

Papagelis A. and Kalles D. Breeding Decision Trees Using Evolutionary Techniques. *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 393–400, 2001.

Pfister C. Getting Started With The Internet of Things. *O'Reilly Media, Inc.*, 1, 2011.

Quinlan J. Introduction of Decision Trees. *Machine learning*, 1(1):81–106, 1986.

Quinlan J. Simplifying Decision Trees. *International Journal Of Man-Machine Studies*, 27(3):221–234, 1987.

Rachio. Rachio 3 smart sprinkler controller, 2020. URL https://www.rachio.com/rachio-3/.

Rascão José P. Sistemas de Informação Para As Organizações. *1*, 1, 2001.

Reis C. *Planeamento Estratégico de Sistemas de Informação*. Presença, 1993.

ResearchGate, 2016. URL https://www.google.pt/search?q=random+forest+classifier&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiw4LX5vanvAhWMzYUKHZWmCZcQ_AUoAXoECBUQAw&biw=1276&bih=957#imgrc=SzqDAcKCkkaZ_M).

Rosnay J. *O Macroscópio Para Uma Visão Global, Lisboa*. 1st edition, 1977.

Rosário L. Indicadores De Desertificação Para Portugal Continental. *Direcção-Geral dos Recursos Florestais. Núcleo de Desertificação.*, 65, 2004.

Saleemmaleekh A. and Sudhakar K. N. 'Real-Time Monitoring Of Agricultural Activities Using Wireless Sensor Network'. *International Journal of Science and Research (IJSR)*, 4(5):2843–2846, 2015.

Santos F. D. and Valente M.A. and Miranda P.M.A. and Aguiar A. and Azevedo E.B. and Tomé A. and Coelho F. . Climate change in portugal - siam scenarios - impacts and adaptation measures. *World Resources Review*, 14:69, 2003.

Sharma, S. . Applied multivariate techniques. new york: John wiley and sons. page 320, 1996.

Silva J. and Ferreira J. and Condessa B. and Loupa-Ramos I. Intelligent plans for intelligent cities. 07 2012.

Starkweather J. and Moske A. K. *Multinomial Logistic Regression*. 2011.

Steventon A. and Wight S. *'Intelligent Spaces: The Application Of Pervasive ICT'*. 2006.

Tanenbaum A. S. Structured Computer Organization. *Pearson Prentice Hall*, 5, 2006.

Techopedia. What is a smart city? - definition from techopedia, Nov 2015. URL https://www.techopedia.com/definition/31494/smart-city.

Tecnico Lisboa, 2019. URL https://www.google.com/search?q=arvores+de+decis%C3%A3o&sxsrf=ALeKk02oybjuDHQGiTpwXuga_gLnG5flig:1615485386078&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiMo86E6KjvAhViRBUIHR-DD7AQ_AUoAXoECAYQAw&biw=1280&bih=911#imgrc=x).

Towardsdatascience.com, 2018. URL https://towardsdatascience.com/what-is-machine-learning-a-short-note-on-supervised-unsupervised-semi-supe

Unesco. Water for a sustainable world: United nations educational, scientific and cultural organization, 2015. URL http://www.unesco.org/new/en/natural-sciences/environment/water/wwap/wwdr/2015-water-for-a-sustainable-world.

Varajão J. E. Q. *A Arquitectura da Gestão de Sistemas de Informação*. FCA, 1998.

Wigan M. R. Data Ownership. In Managing Information Technology's Organizational Impact II and R. e J. Cameron Clarke, editors, *Elsevier Science Publishers B. V.*, pages 157–169. 1992.