

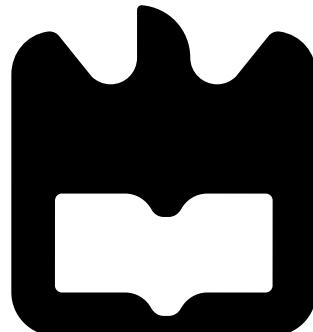


Rui Pedro dos  
Santos Oliveira

**Sistema de monitorização e controlo da produção  
de Salicornia na Ria de Aveiro**

**Monitoring and control system for the production  
of Salicornia in the estuary of Aveiro**

# **DOCUMENTO PROVISÓRIO**







Rui Pedro dos  
Santos Oliveira

**Sistema de monitorização e controlo da produção  
de Salicórnia na Ria de Aveiro**

**Monitoring and control system for the production  
of Salicornia in the estuary of Aveiro**

# **DOCUMENTO PROVISÓRIO**





Rui Pedro dos  
Santos Oliveira

**Sistema de monitorização e controlo da produção  
de Salicórnia na Ria de Aveiro**

**Monitoring and control system for the production  
of Salicornia in the estuary of Aveiro**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Joaquim Manuel Henriques de Sousa Pinto, Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor José Alberto Gouveia Fonseca, Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.



## **o júri / the jury**

presidente / president

### **ABC**

Professor Catedrático da Universidade de Aveiro (por delegação da Reitora da Universidade de Aveiro)

vogais / examiners committee

### **DEF**

Professor Catedrático da Universidade de Aveiro (orientador)

### **GHI**

Professor associado da Universidade J (co-orientador)

### **KLM**

Professor Catedrático da Universidade N



**agradecimentos /  
acknowledgements**

Quero expressar os meus agradecimentos a todos aqueles que direta e indiretamente contribuíram para a concretização da presente dissertação. Um agradecimento especial aos meus pais e irmão por todo o apoio prestado ao longo deste meu percurso académico, como também a toda a minha família.

A todos os meus colegas e amigos, que conheci ao longo destes 5 anos na Universidade de Aveiro, um grande obrigado



**palavras chave**

Cultivo da salicórnia, irrigação, sensores, atuadores, web, monitorização, atuação remota.

**resumo**

A evolução da Internet tem proporcionado o crescimento de novos modelos de negócio suportados por tecnologias de informação e comunicações e tem possibilitado o desenvolvimento de novos modelos de serviços baseados no cloud computing.

A exploração de novas descobertas na área da percepção visual, nomeadamente no que se refere à apreciação de obras de arte geniais, . . .



**keywords** Cultivo da salicórnia, irrigação, sensores, atuadores, web, monitorização, atuação remota.

**abstract** Nowadays, it is usual to evaluate a work ...



# Conteúdo

|   |     |
|---|-----|
| <b>Lista de Figuras</b>                                 | v   |
| <b>Lista de Tabelas</b>                                 | vii |
| <b>Acrónimos</b>  | ix  |
| <b>1 Introdução</b>                                     | 1   |
| 1.1 Motivação . . . . .                                 | 1   |
| 1.2 Objetivos . . . . .                                 | 2   |
| 1.3 Organização do documento . . . . .                  | 2   |
| <b>2 Conceito de IoT no cultivo da Salicornia</b>       | 5   |
| 2.1 Características da planta . . . . .                 | 5   |
| 2.2 Condições ideais de cultivo da Salicornia . . . . . | 7   |
| 2.3 Importância da planta . . . . .                     | 7   |
| 2.4 Evolução tecnológica: o IoT . . . . .               | 7   |
| 2.5 Considerações finais . . . . .                      | 9   |
| <b>3 Estado da arte</b>                                 | 11  |
| 3.1 Conceitos tecnológicos . . . . .                    | 11  |
| 3.2 Sistema de Gestão de Base de Dados . . . . .        | 11  |
| 3.2.1 MySQL . . . . .                                   | 12  |
| 3.2.2 SQL server . . . . .                              | 12  |
| 3.2.3 PostgreSQL . . . . .                              | 12  |
| 3.2.4 Comparação e solução adotada . . . . .            | 12  |
| 3.3 Desenvolvimento web . . . . .                       | 13  |
| 3.3.1 ASP.net . . . . .                                 | 13  |
| 3.3.2 Flask . . . . .                                   | 13  |

|          |   |           |
|----------|---|-----------|
| 3.3.3    | Django . . . . .  | 13        |
| 3.3.4    | Conclusões e solução adotada . . . . .                              | 13        |
| 3.4      | Desenvolvimento mobile . . . . .                                    | 14        |
| 3.4.1    | Plataformas nativas . . . . .                                       | 14        |
| 3.4.2    | Multi-plataforma . . . . .  | 14        |
| 3.4.3    | Conclusões e solução adotada . . . . .                              | 14        |
| 3.5      | REST Frameworks . . . . .   | 15        |
| 3.5.1    | Django Rest Framework . . . . .                                     | 15        |
| 3.5.2    | Flask-RESTful . . . . .   | 15        |
| 3.5.3    | Restlet . . . . .   | 15        |
| 3.5.4    | Conclusões e solução adotada . . . . .                              | 15        |
| 3.6      | Micro-controladores . . . . .                                       | 16        |
| 3.6.1    | Arduino . . . . .   | 16        |
| 3.6.2    | Raspberry Pi . . . . .  | 17        |
| 3.7      | Sensores . . . . .  | 19        |
| 3.7.1    | Sensor de salinidade . . . . .                                      | 19        |
| 3.7.2    | Sensor de temperatura . . . . .                                     | 19        |
| 3.7.3    | Sensor de luminosidade . . . . .                                    | 20        |
| 3.8      | Tecnologias de comunicação . . . . .                                | 21        |
| 3.8.1    | Zigbee . . . . .  | 21        |
| 3.8.2    | LoRa . . . . .  | 21        |
| 3.8.3    | Sigfox . . . . .  | 21        |
| 3.8.4    | Bluetooth (BLE) . . . . .   | 21        |
| 3.8.5    | Wi-Fi (IEEE 802.11) . . . . .                                       | 22        |
| 3.8.6    | Comparação de tecnologias de comunicação . . . . .                  | 22        |
| 3.9      | Aplicações relacionadas . . . . .                                   | 23        |
| 3.9.1    | Multi-monitorização de estufas agrícolas . . . . .                  | 23        |
| 3.9.2    | Agroopar . . . . .  | 23        |
| 3.9.3    | outras que vale a pena para comparacao.. . . . .                    | 23        |
| <b>4</b> | <b>Sistema de controlo e monitorização: arquitetura e modelação</b> | <b>25</b> |
| 4.1      | Descrição global do sistema . . . . .                               | 25        |
| 4.2      | Componentes . . . . .   | 27        |
| 4.2.1    | <i>Sensor Module</i> . . . . .                                      | 27        |
| 4.2.2    | <i>Controller Module</i> . . . . .                                  | 28        |
| 4.3      | Análise de requisitos . . . . .                                     | 29        |

|   |  |           |
|---|--|-----------|
| 4.3.1   | Requisitos funcionais . . . . .                          | 30        |
| 4.3.2   | Requisitos não funcionais . . . . .                      | 32        |
| 4.4   | Modelação . . . . .                                      | 33        |
| 4.4.1   | Entidades envolvidas . . . . .                           | 33        |
| 4.4.2   | Casos de uso . . . . .                                   | 33        |
| 4.4.3   | Modelo de dados . . . . .                                | 38        |
| 4.5   | Arquitetura lógica . . . . .                             | 41        |
| 4.6   | Arquitetura física . . . . .                             | 42        |
| 4.6.1   | Sistema de informação . . . . .                          | 43        |
| Aplicação web . . . . .                                   | 43   |           |
| API REST . . . . .  | 45   |           |
| Documentação interativa . . . . .                         | 47   |           |
| Implementação do sistema . . . . .                        | 48   |           |
| Aplicação mobile . . . . .                                | 48   |           |
| 4.6.2   | Simulação em <i>hardware</i> . . . . .                   | 49        |
| Sensores utilizados . . . . .                             | 50   |           |
| Comunicação . . . . .                                     | 52   |           |
| 4.6.3   | Sistema de deteção de intrusos . . . . .                 | 54        |
| Biblioteca para processamento de imagem: OpenCV . . . . . | 54   |           |
| 4.7   | Diagrama de componentes . . . . .                        | 57        |
| 4.8   | Considerações finais . . . . .                           | 58        |
| <b>5</b>  | <b>Implementação</b> . . . . .                           | <b>59</b> |
| 5.1   | Sistema de informação . . . . .                          | 59        |
| 5.1.1   | Sistema de registo e autenticação . . . . .              | 60        |
| 5.1.2   | Geração de alarmes . . . . .                             | 62        |
| 5.1.3   | Visualização dos dados e cálculos estatísticos . . . . . | 63        |
| 5.1.4   | Application Programming Interface (API) . . . . .        | 64        |
| 5.1.5   | Documentação da API . . . . .                            | 66        |
| 5.1.6   | Aplicação web . . . . .                                  | 66        |
| 5.1.7   | <i>Deploy</i> do projecto . . . . .                      | 67        |
| 5.1.8   | Aplicação mobile . . . . .                               | 69        |
| 5.2   | Simulação em <i>hardware</i> . . . . .                   | 70        |
| 5.2.1   | Arduino . . . . .  | 70        |
| 5.2.2   | Sensores . . . . .                                       | 70        |
| 5.2.3   | Comunicação . . . . .                                    | 71        |

|   |            |
|---|------------|
| Raspberry Pi . . . . .                            | 71         |
| Comunicação . . . . .                             | 71         |
| 5.2.4 Considerações finais . . . . .              | 72         |
| 5.3 Sistema de deteção de intrusos . . . . .      | 73         |
| 5.3.1 Algoritmos de deteção de intrusos . . . . . | 73         |
| 5.3.2 Testes . . . . .                            | 74         |
| 5.3.3 Implementação . . . . .                     | 74         |
| 5.4 Considerações finais . . . . .                | 74         |
| <b>6 Resultados</b>                               | <b>75</b>  |
| 6.1 Interface web . . . . .                       | 75         |
| 6.2 Interface mobile . . . . .                    | 75         |
| 6.3 Testes com API REST . . . . .                 | 75         |
| 6.4 Simulação em hardware . . . . .               | 75         |
| 6.5 Sistema de deteção de intrusos . . . . .      | 75         |
| <b>7 Conclusão e trabalho futuro</b>              | <b>77</b>  |
| 7.1 Conclusão . . . . .                           | 77         |
| 7.2 Trabalho futuro . . . . .                     | 77         |
| 7.3 Considerações finais . . . . .                | 77         |
| <b>A Application Programming Interface REST</b>   | <b>83</b>  |
| <b>B Mockups da aplicação mobile</b>              | <b>87</b>  |
| <b>C Trigger SQL</b>                              | <b>89</b>  |
| <b>D Resultados processamento de imagem</b>       | <b>91</b>  |
| D.1 Frame 1 . . . . .                             | 92         |
| D.2 Frame 2 . . . . .                             | 94         |
| D.3 Frame 3 . . . . .                             | 96         |
| D.4 Frame 4 . . . . .                             | 98         |
| <b>E Interface gráfica</b>                        | <b>101</b> |
| <b>F Descrição formal dos casos de uso gerais</b> | <b>103</b> |
| <b>G Interligação de componentes</b>              | <b>105</b> |

# **Lista de Figuras**

|      |  |    |
|------|--|----|
| 1.1  | Salicornia proveniente da ria de Aveiro . . . . .  | 1  |
| 2.1  | Coloração da planta <i>Salicornia ramosissima</i> na primavera (à esquerda) e no outono (à direita) (Fotografia por José M. G. Pereira) . . . . .  | 6  |
| 2.2  | Esquema representativo do ciclo de vida da <i>Salicornia ramosissima</i> . A - semente incluída no sedimento; B - jovens plantas e plantas senescentes do ano anterior; C - planta no estado vegetativo, caule carnudo e articulado; D - planta no estado de floração; E - planta no estado senescente. (Fotografias por Helena Silva) . . . . . | 6  |
| 2.3  | Evolução da internet em cinco fases . . . . .  | 8  |
| 2.4  | Pirâmide do conhecimento: modelo DIKW . . . . .  | 9  |
| 3.1  | Arduin Nano . . . . .  | 16 |
| 3.2  | Identificação dos pinos no Arduino Nano . . . . .  | 16 |
| 3.3  | Raspberry Pi 3 . . . . .   | 18 |
| 3.4  | Identificação dos principais componentes no Raspberry Pi 3 . . . . .   | 18 |
| 4.1  | Ilustração dos principais componentes do sistema . . . . .   | 25 |
| 4.2  | Ilustração da distribuição dos módulos em duas leiras . . . . .  | 26 |
| 4.3  | Esquema de componentes e respetiva comunicação entre três <i>Sensor Module</i> (SM) e um <i>Controller Module</i> (CM) . . . . .   | 27 |
| 4.4  | Fase de desenvolvimento de um software . . . . .   | 29 |
| 4.5  | Casos de uso para a aplicação web (dashboard) . . . . .  | 34 |
| 4.6  | Casos de uso para a aplicação mobile . . . . .   | 34 |
| 4.7  | Esquema relacional da estrutura da base de dados . . . . .   | 38 |
| 4.8  | Arquitetura lógica . . . . .   | 41 |
| 4.9  | Arquitetura física (blocos) . . . . .  | 42 |
| 4.10 | Arquitetura do sistema de informação ( <i>dashboard</i> , base de dados e API) . . . . .   | 44 |

|      |   |     |
|------|---|-----|
| 4.11 | Processo de autenticação em HTTP através de token (adaptado de [1]) . . . . . | 46  |
| 4.12 | Arquitetura da aplicação mobile . . . . .                                     | 48  |
| 4.13 | Sensor TTC 104 NTC . . . . .  | 50  |
| 4.14 | Esquema eletrotécnico da ligação do sensor de temperatura . . . . .           | 50  |
| 4.15 | Sensor foto-resistência GL5528 . . . . .                                      | 51  |
| 4.16 | Esquema eletrotécnico da ligação do sensor de luminosidade . . . . .          | 51  |
| 4.17 | <i>Water Level Switch Liquid Level Sensor Plastic Ball Float</i> . . . . .    | 52  |
| 4.18 | Esquema eletrotécnico da ligação do sensor de nível líquido . . . . .         | 52  |
| 4.19 | Led . . . . .   | 52  |
| 4.20 | Esquema eletrotécnico da ligação do led . . . . .                             | 52  |
| 4.21 | Comunicação entre componentes da simulação em hardware . . . . .              | 53  |
| 4.22 | Módulo bluetooth HC-06 . . . . .  | 53  |
| 4.23 | Esquema eletrotécnico da ligação do módulo bluetooth . . . . .                | 53  |
| 4.24 | Logótipo OpenCV . . . . .   | 54  |
| 4.25 | Raspberry Pi Camera Board V2 8MP 1080p . . . . .                              | 55  |
| 4.26 | Arquitetura do sistema de video stream . . . . .                              | 56  |
| 4.27 | Diagrama final de componentes do sistema . . . . .                            | 57  |
| 5.1  | Painel administrativo do Django . . . . .                                     | 60  |
| 5.2  | Diagrama de atividades do processo de registo e autenticação . . . . .        | 61  |
| 5.3  | Diagrama de fluxo para geração de alarmes . . . . .                           | 62  |
| 5.4  | Documentação da API REST com a ferramenta Swagger . . . . .                   | 66  |
| D.1  | Pirâmide do conhecimento: modelo DIKW . . . . .                               | 92  |
| D.2  | Pirâmide do conhecimento: modelo DIKW . . . . .                               | 94  |
| D.3  | Pirâmide do conhecimento: modelo DIKW . . . . .                               | 96  |
| D.4  | Pirâmide do conhecimento: modelo DIKW . . . . .                               | 98  |
| E.1  | Pirâmide do conhecimento: modelo DIKW . . . . .                               | 101 |
| E.2  | Pirâmide do conhecimento: modelo DIKW . . . . .                               | 102 |
| G.1  | Pirâmide do conhecimento: modelo DIKW . . . . .                               | 105 |

# **Lista de Tabelas**

|     |   |    |
|-----|---|----|
| 3.1 | Características do Arduino Nano . . . . .   | 17 |
| 3.2 | Comparação entre versão 2 e 3 do Raspberry Pi . . . . .                                     | 18 |
| 4.1 | Especificação das tabelas existentes no sistema . . . . .                                   | 39 |
| 4.2 | Especificação das tabelas existentes no sistema (continuação) . . . . .                     | 40 |
| 4.3 | Endpoints da API REST e respetivos métodos a implementar . . . . .                          | 47 |
| 4.4 | Características do sensor TTC 104 . . . . .   | 50 |
| 4.5 | Características do sensor GL5528 . . . . .  | 51 |
| 4.6 | Características do módulo bluetooth HC-06 . . . . .   | 54 |
| 4.7 | Características do módulo bluetooth HC-06 . . . . .   | 55 |
| 5.1 | Estrutura do ficheiro do tipo Comma-Separated Values (CSV) possível de exportação . . . . . | 64 |
| B.1 | Um nome qualquer . . . . .  | 87 |
| D.1 | Your caption here . . . . .   | 93 |
| D.2 | Your caption here . . . . .   | 95 |
| D.3 | Your caption here . . . . .   | 97 |
| D.4 | Your caption here . . . . .   | 99 |



# Acrónimos

|             |                                    |
|-------------|------------------------------------|
| <b>API</b>  | Application Programming Interface  |
| <b>CGI</b>  | Common Gateway Interface           |
| <b>CM</b>   | <i>Controller Module</i>           |
| <b>CSS</b>  | Cascading Style Sheets             |
| <b>CSV</b>  | Comma-Separated Values             |
| <b>FK</b>   | Foreign Key                        |
| <b>GPS</b>  | Global Positioning System          |
| <b>HTML</b> | HyperText Markup Language          |
| <b>HTTP</b> | HyperText Transfer Protocol        |
| <b>I/O</b>  | Input/ Output                      |
| <b>IDE</b>  | Integrated Development Environment |
| <b>IoT</b>  | <i>Internet of Things</i>          |
| <b>LDR</b>  | Light Dependent Resistor           |
| <b>NTC</b>  | Negative Temperature Coefficient   |
| <b>ORM</b>  | Object Relational Mapper           |
| <b>PK</b>   | Primary keys                       |
| <b>REST</b> | Representational State Transfer    |
| <b>REST</b> | Representational State Transfer    |

---

|             |                                    |
|-------------|------------------------------------|
| <b>SDLC</b> | Systems Development Life Cycle     |
| <b>SGBD</b> | Sistema de Gestão de Base de Dados |
| <b>SM</b>   | <i>Sensor Module</i>               |
| <b>SQL</b>  | Structured Query Language          |
| <b>URL</b>  | Uniform Resource Locator           |
| <b>WSGI</b> | Web Server Gateway Interface       |
| <b>WWW</b>  | World Wide Web                     |
| <b>JS</b>   | JavaScript                         |
| <b>ORM</b>  | Object-Relational Mapping          |
| <b>CPU</b>  | Central Processing Unit            |
| <b>RAM</b>  | Random Access Memory               |
| <b>DIKW</b> | Data-Information-Knowledge-Wisdom  |
| <b>ISM</b>  | Industrial, Scientific, Medical    |
| <b>LED</b>  | Light Emitting Diode               |
| <b>IP</b>   | Internet Protocol                  |
| <b>EDR</b>  | Enhanced Data Rate                 |
| <b>CGI</b>  | Common Gateway Interface           |
| <b>SMTP</b> | Simple Mail Transfer Protocol      |
| <b>TCP</b>  | Transmission Control Protocol      |
| <b>JSON</b> | JavaScript Object Notation         |
| <b>VPS</b>  | Virtual Private Server             |

# Introdução

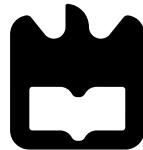


Figura 1.1: Salicornia proveniente da ria de Aveiro

## 1.1 Motivação

<http://eusougourmet.blogspot.pt/2011/09/compre-o-que-e-nosso-salicornia.html>

HyperText Markup Language (HTML)

\* O gênero salicornia HTML inclui cerca de 117 espécies, sendo Salicornia herbacea, Salicornia bigelovii, Salicornia europea, Snicornia prostata, Salicornia mmosissima e Salicornia virginica aquelas com maior ocorrência. [2] [3]

A que serve de mote a esta dissertação ...

Os recursos naturais, nomeadamente, plantas, animais e minerais, são utilizados desde a antiguidade pelo ser humano, não apenas como fonte de alimentos mas também para o tratamento de diversas doenças []. Muitas das espécies que nascem em todo o mundo inicialmente são consideradas pragas, contudo e após alguns estudos intensivos à espécie são descobertas verdadeiras pérolas. Um exemplo disso é a salicornia.

A salicornia é a planta que iremos dar destaque durante este projeto. Esta planta é por vezes utilizada como substituta do sal marinho[] e utilizada para os mais diversos fins. Iremos abordar alguns deles mais à frente.

A salicornia nasce e cresce naturalmente ao longo dos estuários e sapais (salinas) costeiras do Mediterrâneo].

Esta é uma planta suculenta adaptada a ambientes salinos (halófita) que se desenvolve maioritariamente em ambientes aquáticos com elevado teor de sal.]

Existem mais de 20 espécies de Salicornia, das quais as mais comuns são:

Existem cerca de 20 espécies de Salicornia, das quais as mais comuns encontram-se destacadas de seguida:

Salicornia virginica: é uma planta com flor e pode ser encontrada na região mediterrânica  
Salicornia europaea: cresce em várias zonas de entre-marés salinas  
Salicornia maritima: Salicornia bigelovii: Salicornia perennis: Salicornia ramosissima:

A evolução tecnológica é algo que sempre esteve presente na vida do ser humano desde os seus primórdios até aos dias atuais, sendo que se tem verificado um aumento desta relação com o humano e principalmente com o ritmo da própria evolução. As tecnologias, de uma maneira geral, são todas as invenções produzidas pelo homem, para aumentar a sua atividade no planeta e simplificar o modo de vida que quem o habita [1]. O conceito de “Internet das coisas” (do inglês “Internet of Things”, IoT) é fruto desta evolução tecnológica, já que permite a ligação dos mais diversos dispositivos eletrónicos à Internet.

## 1.2 Objetivos

Este trabalho tem como objetivo o desenvolvimento

- Criação de uma plataforma web que permita:
  - Disponibilizar a leitura dos mais diversos sensores de sensores (temperatura, salinidade...)
  - Permitir gerar alarmes de inundação, sendo estes enviados via SMS ou email para o cliente.
  - Atuar remotamente para drenagem de água em excesso existente nas leiras
  - Sistema de transmissão de vídeo disparada por eventos gerados pelos sensores
- Criação de uma aplicação móvel que permita receber alarmismos de situações anómalas.

## 1.3 Organização do documento

A presente dissertação está dividida em 7 capítulos: ...

O primeiro capítulo descreve e enfatiza a importância

De seguida, no Estado de Arte, é

No Capítulo 2 apresenta-se

o projeto CAMBADA e identifica-se os pontos chave tanto do software como do hardware.

No Capítulo 3

No Capítulo 4 é....

Para finalizar, no Capítulo 5 apresentam-se conclusões sobre o trabalho desenvolvido e eventuais melhorias para o futuro.

Resumo As elevadas propriedades nutricionais da *Salicornia L.* tornam-na atrativa para aplicações culinárias e para o tratamento e prevenção de algumas doenças.



# Conceito de IoT no cultivo da Salicórnia

A *Salicornia ramosissima J. Woods* (*S. ramosissima*)[4] que impulsionará toda esta dissertação, é uma espécie do género *Salicornia L.*, pertencente à família das beterrabas denominada de *Chenopodiaceae*[5]. Neste capítulo será apresentada a planta, as suas principais características e respetivas propriedades, bem como as suas diferentes aplicações medicinais e alimentares. Este capítulo servirá ainda como uma pequena introdução ao conceito de *Internet of Things* (IoT) e a sua respetiva importância no contexto deste projeto.

## 2.1 Características da planta

A Salicórnia é uma espécie halófita, adaptada a viver em ambientes com elevado teor de sais[6], sendo uma das mais evoluídas da sua família. É uma planta anual de dimensão pequena, aparentemente sem folhas, ereta, os seus caules são carnudos e suculentos, simples e/ou extremamente ramificados, segmentados por articulações[7], geralmente com menos de 30 cm de altura[2]. Esta planta tem uma coloração durante a maior parte do ano verde-escuro mas a sua ramagem torna-se verde-amarelado ou mesmo vermelho-púrpura no outono[7] (figura 2.1).

A *Salicornia ramosissima* desenvolve-se preferencialmente no litoral costeiro, em pântanos e sapais salgados ou em margens de salinas temporariamente alagadas. Encontra-se distribuída maioritariamente na parte oeste da Europa e a oeste da região do Mediterrâneo, sendo uma das espécies mais abundantes[8]. Em Portugal, onde é vulgarmente conhecida como erva-salada, sal verde e/ou espargos do mar[9], é encontrada frequentemente nas mar-

gens dos canais da Ria de Aveiro e Ria Formosa, no Algarve[9] sendo encontrada com menos frequência na região do Minho[7]. Na Inglaterra, a salicórnia é conhecida como *purple glasswort*, podendo este nome estar na origem desta pigmentação caraterística[10].



Figura 2.1: Coloração da planta *Salicornia ramosissima* na primavera (à esquerda) e no outono (à direita) (Fotografia por José M. G. Pereira)

Esta planta é uma das menos estudadas pelos cientistas[8], sabendo-se apenas que possui um ciclo de vida anual bem definido, com gerações discretas e as suas sementes são hermafroditas[11]. A salicórnia cresce habitualmente entre março, início da sementeira (A da figura 2.2) com respetivo crescimento (B da figura 2.2) e novembro fechando assim o ciclo com a produção de sementes (E da figura 2.2). Entre maio e agosto decorre a colheita da planta[9] (C da figura 2.2) que pode ser utilizada para os mais diversos fins. A floração ocorre fundamentalmente no mês de outubro[8] (D da figura 2.2).

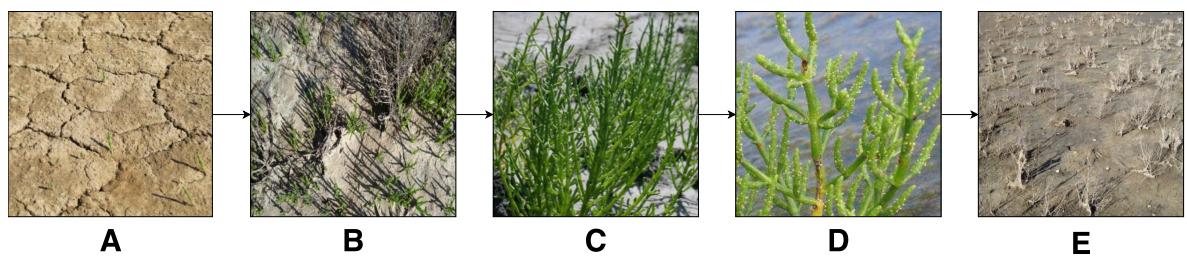


Figura 2.2: Esquema representativo do ciclo de vida da *Salicornia ramosissima*. A - semente incluída no sedimento; B - jovens plantas e plantas senescentes do ano anterior; C - planta no estado vegetativo, caule carnudo e articulado; D - planta no estado de floração; E - planta no estado senescente. (Fotografias por Helena Silva)

## 2.2 Condições ideais de cultivo da Salicórnia

O crescimento da *Salicornia ramosissima* é influenciado por diversos fatores ambientais, sendo a salinidade um dos mais importantes, já que influencia a distribuição, a abundância e a fisiologia da planta. Um estudo realizado por *Silva et al.*[11] comprova que esta planta halófita apresenta um crescimento ideal em salinidades baixas ou moderadas. Este estudo permite considerar esta planta como uma halófita não obrigatória, já que o seu crescimento ideal não acontece em condições de salinidade elevada. Embora o crescimento ideal ocorra a baixas salinidades, a Salicornia é capaz de tolerar níveis elevados de salinidade no meio de cultivo[12].

## 2.3 Importância da planta

Desde a antiguidade que as espécies do género *Salicornia L.* estão incluídas na alimentação humana. Normalmente consome-se crua, cozinhada ou seca. Quando crua é usada como acompanhamento das mais diversas refeições enquanto que seca ou triturada é usada como especiaria, para tempero na confeção de peixes, marisco ou carnes. O sal verde é um grande substituto do sal comum, pois é rico em substâncias depurativas e diuréticas. Os seus caules carnudos são bastante requisitados para cozinhas *gourmet*, não só pelo seu sabor salgado, mas também pelo seu elevado valor nutricional[13], nomeadamente pelos níveis de minerais e vitaminas antioxidantes, como a vitamina C e o  $\beta$ -caroteno. A Salicórnia é também uma fonte de proteínas e possui alto teor de ácidos gordos, destacando-se o ómega-3[14].

A nível medicinal, existem inúmeros estudos que revelam que as propriedades químicas da planta, tornam-na eficiente na prevenção e tratamento de algumas doenças, tais como, a hipertensão, cefaleias e escorbuto, diabetes, obesidade, cancro, entre outras[15].

Tendo em conta todas estas propriedades alimentares e medicinais da Salicórnia, torna-se fulcral controlar o seu cultivo, a fim de otimizar a produção para tirar maior partido da sua importância biológica. Este controlo pode ser feito recorrendo à evolução tecnológica, nomeadamente ao conceito de IoT, tal como será descrito nas próximas secções deste capítulo.

## 2.4 Evolução tecnológica: o IoT

Antes de descrever a importância e o conceito de IoT, é necessário entender as diferenças entre os termos Internet e Web ( World Wide Web, WWW), que são usados indistintamente pela sociedade. A Internet é a camada ou rede física composta por *switches*, *routers* e outros equipamentos[16]. A sua principal função é transportar informações de um ponto para outro de forma rápida, confiável e segura. Por outro lado, a Web pertence à camada de aplicações

que opera sobre a Internet cuja principal função é oferecer uma interface que transforme as informações que fluem pela Internet em algo útil. Ao longo do tempo, a Web passou e continua a passar por várias etapas evolucionárias, identificadas como Web 1.0, Web 2.0 e Web 3.0, explicadas nas próximas secções.

- **Web 1.0 - passado:** esta primeira etapa foi inventada por Tim Berners Lee em 1989[17]. Nesta fase surgiram os principais conceitos que conhecemos da Internet atual: Localizador Uniforme de Recursos (do inglês Uniform Resource Locator (URL)), Linguagem de Marcação de Hipertexto (do inglês HTML) e Protocolo de Transferência de Hipertexto (do inglês HyperText Transfer Protocol (HTTP)). Ainda nesta primeira fase, mas mais tarde, em 1998 foi criado por Larry Page e Sergey Brin o Google que criou simplicidade nas pesquisas na Web[18].
- **Web 2.0 - presente:** a Web cresceu muito e muito rapidamente. Atualmente é considerada a versão mais próxima da visão de Tim Berners Lee (colaborativa), usado como meio de interação, comunicação global compartilhamento de informação.
- **Web 3.0 - futuro:** para o futuro prevê-se que os conteúdos *online* possam vir a estar organizados de forma semântica, muito mais personalizados para cada utilizador, sites, aplicações inteligentes e/ou publicidade baseada nas pesquisas e nos comportamentos.

A primeira evolução real da Internet foi o aparecimento do IoT, que já transformou a Internet em algo sensorial, através da medição de diferentes características, como por exemplo a temperatura, a pressão, as vibrações, a iluminação, a umidade, o *stress*, entre outras. No futuro, ao desenvolvimento de aplicações revolucionárias com potencial para melhorar significativamente a forma como a sociedade vive, aprende, trabalha e se diverte.

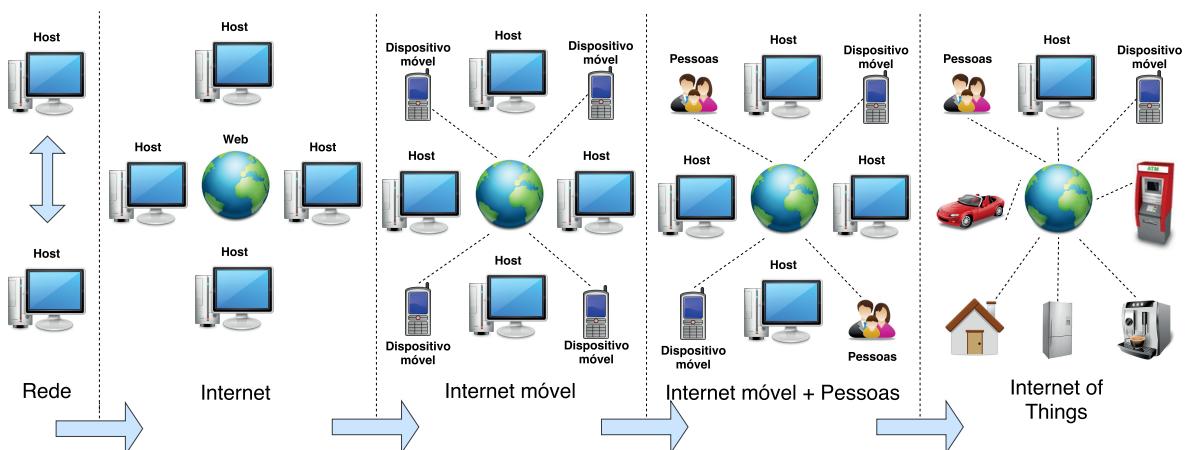


Figura 2.3: Evolução da internet em cinco fases (Adaptado de [19])

A figura 2.3 representa a evolução da Internet em cinco fases. Inicialmente surge a conexão entre dois computadores que permite a criação de uma rede, posteriormente nasce o conceito de World Wide Web (WWW) ligando um grande número de computadores entre si. Seguidamente, surgiu a Internet móvel que permitiu conectar dispositivos moveis à Internet, possibilitando a ligação da sociedade através das redes sociais. Finalmente, a internet está a evoluir para o IoT, permitindo ligar objetos do quotidiano ao sistema global de redes de computadores[19].

Uma das principais vantagens do IoT é a sua ligação evidente a todos os objetos, o que por si só é uma ideia avassaladora. O volume de dados gerado por este tipo de ligação pode ser interpretado pelo modelo Data-Information-Knowledge-Wisdom (DIKW)[20]. Este modelo, também conhecido como pirâmide do conhecimento (Figura 2.4), é uma hierarquia informacional utilizada especialmente nas áreas da ciência da informação e na gestão do conhecimento, onde cada camada acrescenta certos atributos sobre a anterior.

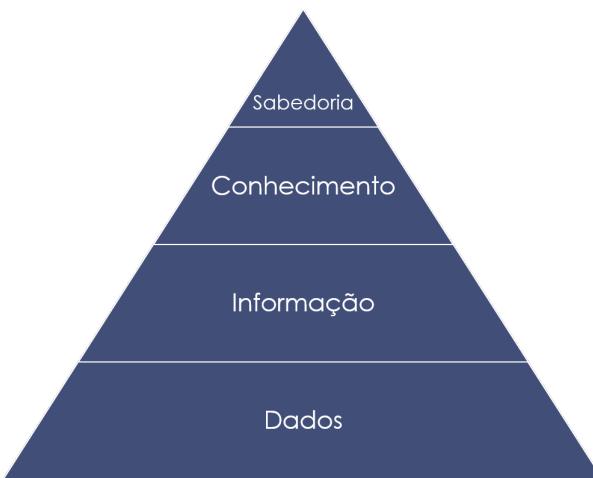


Figura 2.4: Pirâmide do conhecimento: modelo DIKW

A ligação dos objetos à Internet acarreta benefícios visíveis à nossa sociedade, possibilitando um maior controlo e entendimento de como os sistemas interagem entre si e proporcionando uma melhor qualidade de vida a todos. Embora as vantagens se sobreponham às desvantagens não nos podemos esquecer que existem alguns problemas a nível segurança, privacidade, legislação e identidade.

## 2.5 Considerações finais

Como vimos neste capítulo, as propriedades alimentares e terapêuticas da Salicornia têm conduzido a um elevado interesse económico e ao aumento do seu desenvolvimento comercial.

Existem inúmeras empresas a cultivar esta espécie para que possa ser comercializada para os mais diversos fins, sendo que grande parte já é exportada.

Uma vez que a salicornia carece de um controlo minucioso de certos parâmetros durante o seu cultivo, existe necessidade de criar um sistema que monitorização de forma a melhor as condições de produção desta espécie.

O conceito de IoT pode ser aplicado neste contexto, uma vez que possibilitará a interligação de equipamentos eletrónicos que melhorem a eficiência de produção da espécie através da colocação de sensores, atuadores e respetiva atuação remota.

# 3

## Estado da arte

Nesta secção, são apresentados os resultados da pesquisa bibliográfica sobre as ferramentas e funcionalidades que poderão estar presentes no sistema a desenvolver. Pretende-se apresentar de forma geral todas as tecnologias possíveis de utilização e respetiva comparação.

### 3.1 Conceitos tecnológicos

- HTML:
- Cascading Style Sheets (CSS):
- JavaScript (JS):
- Python:
- API:

### 3.2 Sistema de Gestão de Base de Dados

Um Sistema de Gestão de Base de Dados (SGBD) é um conjunto de software responsáveis pela gestão de uma base de dados.

### 3.2.1 MySQL

### 3.2.2 SQL server

### 3.2.3 PostgreSQL

O PostgreSQL é um sistema de gestão de base de dados do tipo objeto-relacional uma vez que permite um modelo de dados orientado a objetos, isto é, possibilita a manipulação de objetos, classes e heranças diretamente no esquemas da base de dados. Segundo o site oficial do PostgreSQL este é considerado um SGBD bastante poderoso e com desenvolvimento *open sources* [21].

### 3.2.4 Comparação e solução adotada

Os próprios criadores do Django recomendam a utilização do PostgreSQL, indicando que alcança um bom equilíbrio entre custo, características, rapidez e estabilidade.

No entanto, é pertinente fazer uma comparação entre o PostgreSQL e outras ferramentas open-source como o MySQL. Embora as diferenças entre as duas ferramentas não sejam muito grandes, podemos ter também em conta a performance de uma e outra. Uma comparação feita usando o benchmark TPC-H 8 mostra que a performance do PostgreSQL é ligeiramente superior à do MySQL na maioria das queries [22].

### 3.3 Desenvolvimento web

Para o desenvolvimento da dashboard poderiam ser adotadas duas estratégias distintas para o desenvolvimento web:

A criação de sites dinâmicos que se adaptam ao cliente podem ser alcançados de dois modos:

- Manipulação local usando javascript do DOM.
- Acesso ao servidor que serve conteúdos criados em função dos pedidos do cliente

Neste contexto poderiam ser utilizados

Angular, React

Servidor serve conteudos criados em função dos pedidos do cliente

#### 3.3.1 ASP.net

#### 3.3.2 Flask

#### 3.3.3 Django

Assim, e de acordo com as explicações dos autores da ferramenta [18], as principais vantagens tiradas da utilização da framework Django são: Boa documentação; Facilidade e rapidez de desenvolvimento e deployment; Estabilidade; Escalabilidade.

Models

Views

Templates django

#### 3.3.4 Conclusões e solução adotada

## 3.4 Desenvolvimento mobile

<http://bloomidea.com/blog/aplicacoes-nativas-vs-hibridas-qual-escolher-para-o-seu-projeto/>

### 3.4.1 Plataformas nativas

uma aplicação móvel nativa é uma app que foi desenvolvida para ser utilizada numa plataforma ou dispositivo específico (iOS ou Android), usando as ferramentas e a linguagem de desenvolvimento correspondentes àquelas que o sistema em questão suporta. Uma app nativa pode assim interagir e tirar partido das funcionalidades do próprio sistema operativo e de outro software que esteja instalado nessa plataforma, o que faz desta opção uma excelente aposta.

- **Performance:**
- **Desenvolvimento:**
- **Manutenção:**
- **Interface:**
- **Recursos disponíveis:**
- **Plug-ins:**
- **Segurança:**

### 3.4.2 Multi-plataforma

<http://websocialdev.com/lista-de-frameworks-para-desenvolvimento-mobile/>

- **Performance:**
- **Desenvolvimento:**
- **Manutenção:**
- **Interface:**
- **Recursos disponíveis:**
- **Plug-ins:**
- **Segurança:**

### 3.4.3 Conclusões e solução adotada

## 3.5 REST Frameworks

### 3.5.1 Django Rest Framework

Django REST Framework é uma ferramenta considerada 'poderosa e flexível para a construção de APIs Web' [], que pode ser usada juntamente com a framework de desenvolvimento de aplicações Web Django, que quando integrada no desenvolvimento de um determinado *backend* permite a implementação de serviços do tipo REST.

A API navegável Web é uma vitória usabilidade enorme para os desenvolvedores.

Políticas de autenticação , incluindo pacotes para OAuth1a e OAuth2 .

Serialização que suporta tanto ORM e não ORM fontes de dados.

Customizável todo o caminho - basta usar vistas regulares baseadas na função , se você não precisar dos mais poderosos recursos .

Extensa documentação , e grande apoio da comunidade .

Utilizado e confiável por empresas internacionalmente reconhecidas, incluindo Mozilla , Red Hat , Heroku , e Eventbrite .

### 3.5.2 Flask-RESTful

### 3.5.3 Restlet

### 3.5.4 Conclusões e solução adotada

com autenticação via token

app mobile microcontroladores -*i* controller modulers

documentação com swagger

## 3.6 Micro-controladores

### 3.6.1 Arduino

O Arduino é fruto da evolução de um projeto italiano desenvolvido no ano de 2005, cujo o objetivo foi ser utilizado em projetos escolares de forma a ter um orçamento menor que outros sistemas de prototipagem disponíveis naquela época.

Tal como descrito no seu site oficial, um Arduino consiste numa plataforma *open-source* de prototipagem eletrónica com *hardware* e *software* flexíveis e com elevada facilidade utilização]. O Arduino é utilizado para projetos especialmente no contexto do IoT e da robótica educativa. A este micro-controlador, podem ser estendidos vários módulos, dependendo da tarefa que se quer que seja executada.

O Arduino possui um conjunto de pinos que podem ser programados para funcionarem como entradas ou saídas fazendo com que o Arduino interaja com o meio externo para os mais diversos fins. Para além dos pinos de I/O existem pinos de alimentação que fornecem diversos valores de tensão que podem ser utilizados para transmitir energia elétrica aos diferentes componentes de um projeto.

A versão Nano do Arduino,

Na figura 3.1 e 3.2 apresenta-se uma imagem do arduino utilizado e a identificação dos diferentes pinos existentes, respectivamente. Na tabela 3.1 encontram-se as principais características desta versão do Arduino.

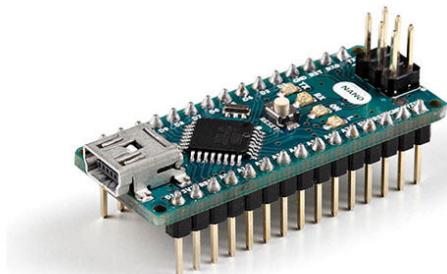


Figura 3.1: Arduin Nano

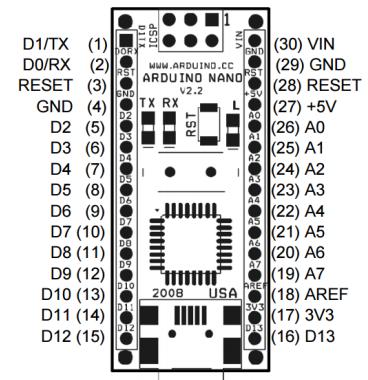


Figura 3.2: Identificação dos pinos no Arduino Nano

|  |                                     |
|--|-------------------------------------|
| Microcontrolador                       | ATmega328                           |
| Tensão de operação                     | 5V                                  |
| Tensão de entrada                      | 7-12V                               |
| Portas digitais                        | 14 (6 podem ser usadas como PWM)    |
| Portas analógicas                      | 8                                   |
| Corrente nos pinos Input/ Output (I/O) | 40mA                                |
| Memória Flash                          | 32KB (2KB usado no bootloader)      |
| Memória RAM (SRAM)                     | 2KB                                 |
| EEPROM                                 | 1KB                                 |
| Velocidade do Clock                    | 16MHz                               |
| Dimensões                              | 45 x 18mm                           |
| LED interno                            | Pino digital 13                     |
| Ligaçāo USB                            | Ligaçāo ao computador e alimentação |

Tabela 3.1: Características do Arduino Nano

### 3.6.2 Raspberry Pi

O Raspberry Pi (figura 3.3) é considerado um micro-computador do tamanho de um cartão de crédito que possui um conjunto de *hardware* integrado que tal como Arduino possibilita uma interação com o meio exterior. O principal objetivo deste poderoso componente consistiu em promover o ensino da ciência da computação em escolas de ensino básico. O Raspberry Pi foi desenvolvido no Reino Unido pela *Raspberry Pi Foundation*.

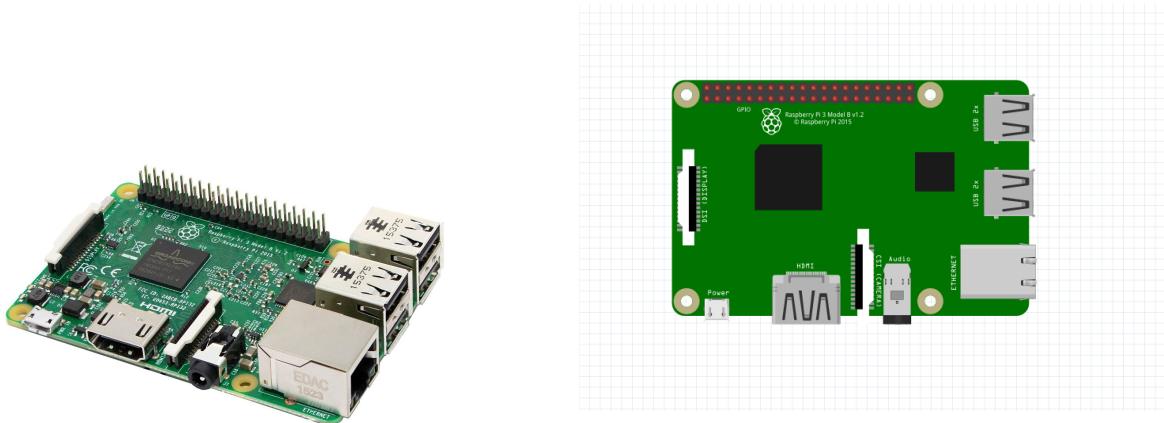


Figura 3.3: Raspberry Pi 3

Figura 3.4: Identificação dos principais componentes no Raspberry Pi 3

|                               | Raspberry Pi 3 Model B   | Raspberry Pi 2 Model B 1.2   |
|-------------------------------|--|--|
| <b>Processor Chipset</b>      | Broadcom BCM2837<br>64Bit Quad Core<br>Processor powered<br>Single Board Computer<br>running at 1.2GHz | Broadcom BCM2837 64Bit<br>Quad Core Processor<br>powered Single Board<br>Computer running at<br>900MHz |
| <b>Processor Speed</b>        | QUAD Core @1.2 GHz   | QUAD Core @900 MHz   |
| <b>RAM</b>                    | 1GB SDRAM @ 400 MHz  | 1GB SDRAM @ 400 MHz  |
| <b>Storage</b>                | MicroSD  | MicroSD  |
| <b>USB 2.0</b>                | 4x USB Ports   | 4x USB Ports   |
| <b>Max Power Draw/voltage</b> | 2.5A @ 5V  | 1.8A @ 5V  |
| <b>GPIO</b>                   | 40 pin   | 40 pin   |
| <b>Ethernet Port</b>          | Yes  | Yes  |
| <b>WiFi</b>                   | Built in (802.11n)   | No   |
| <b>Bluetooth LE</b>           | Built in (4.1)   | No   |

Tabela 3.2: Comparação entre versão 2 e 3 do Raspberry Pi

## 3.7 Sensores

Esta secção tem como objetivo fazer um estudo comparativo entre diferentes tecnologias usadas para a medição dos vários parâmetros ambientais necessários ao controlo e monitorização da salicornia. Todas as soluções adaptadas tem termos de hardware escolhidas devido à possuir-las.

### 3.7.1 Sensor de salinidade

### 3.7.2 Sensor de temperatura

Existem vários tipos de sensores de temperatura baseados em princípios de funcionamento distintos.

- **Termopares:**
- **RTDs:**
- **Termístor:**
- **Circuito integrado:**

### 3.7.3 Sensor de luminosidade

O LDR (Light Dependent Resistor) é um componente cuja resistência varia de acordo com a intensidade da luz. Quanto mais luz incidir sobre o componente, menor a resistência. Este sensor de luminosidade pode ser utilizado em projetos com arduino e outros microcontroladores para alarmes, automação residencial, sensores de presença e etc.

## 3.8 Tecnologias de comunicação

Nesta secção serão apresentados alguns das tecnologias de comunicação mais utilizados em *Internet of Things* que permite a troca de informações entre dispositivos e respetiva comparação entre eles.

### 3.8.1 Zigbee

Zigbee designa um conjunto de especificações para a comunicação sem-fio entre dispositivos eletrônicos, com ênfase na baixa potência de operação, na baixa taxa de transmissão de dados e no baixo custo de implementação. Tal conjunto de especificações define camadas do modelo OSI subsequentes àquelas estabelecidas pelo padrão IEEE 802.15.4.

### 3.8.2 LoRa

A tecnologia Lora

Wide-Area Network Low-Power ( LPWAN ) ou Low-Power Rede ( LPN ) é um tipo de telecomunicações sem fio de rede projetada para permitir comunicações de longo alcance em uma baixa taxa de bits entre as coisas (objetos relacionados), tais como sensores operados em uma bateria.

As tecnologias WAN de baixa potência são projetadas para ambientes de rede máquina a máquina (M2M). Com a diminuição dos requisitos de energia, maior alcance e menor custo do que uma rede móvel, os LPWANs são pensados para permitir uma gama muito mais ampla de aplicativos M2M e Internet of Things (IoT), que foram limitados por orçamentos e problemas de energia.

### 3.8.3 Sigfox

Uma empresa francesa que constrói redes sem fio para conectar objetos de baixa energia, como medidores de energia elétrica , smartwatches e máquinas de lavar, que precisam estar continuamente ligados e emitindo pequenas quantidades de dados. Sua tecnologia é voltada para a Internet das Coisas (IoT).

### 3.8.4 Bluetooth (BLE)

Bluetooth é uma especificação de rede sem fio de âmbito pessoal (Wireless personal area networks – PANs) consideradas do tipo PAN ou mesmo WPAN

### 3.8.5 Wi-Fi (IEEE 802.11)

rede sem fio IEEE 802.11, que também são conhecidas como redes Wi-Fi ou wireless, foram uma das grandes novidades tecnológicas dos últimos anos. Atuando na camada física, o 802.11 define uma série de padrões de transmissão e codificação para comunicações sem fio, sendo os mais comuns: FHSS (Frequency Hopping Spread Spectrum), DSSS (Direct Sequence Spread Spectrum) e OFDM (Orthogonal Frequency Division Multiplexing). Atualmente, é o padrão de fato em conectividade sem fio para redes locais. Como prova desse sucesso pode-se citar o crescente número de Hot Spots e o fato de a maioria dos computadores portáteis novos já saírem de fábrica equipados com interfaces IEEE 802.11. A Rede IEEE possui como principal característica transmitir sinal sem fio através de ondas!

### 3.8.6 Comparação de tecnologias de comunicação

## 3.9 Aplicações relacionadas

Seja para comparar, seja para replicar boas funcionalidades, ou seja para conseguir oferecer algo mais ao utilizador final, quando se pretende desenvolver uma determinada aplicação, é importante proceder a uma avaliação de aplicações da mesma área se encontram no mercado. Assim, são aqui abordadas algumas das aplicações relacionadas que são mais utilizadas ou que mais se aproximam daquilo que se pretende para a aplicação a desenvolver neste projeto, tendo em conta os diferentes sistemas operativos.

### 3.9.1 Multi-monitorização de estufas agrícolas

<https://repositorio.ipcb.pt/bitstream/10400.11/949/1/Multimonitorizacao>

### 3.9.2 Agroopar

<http://www.vidarural.pt/agroopar-os-custos-na-mao-do-agricultor/>

### 3.9.3 outras que vale a pena para comparacao..

<http://www.anje.pt/preview/perfil-coolfarm>

[22]



# Sistema de controlo e monitorização: arquitetura e modelação

Este capítulo tem como principal objetivo a descrição do sistema que resultou do trabalho prático desta dissertação. Para isso, cada elemento do sistema é caracterizado de acordo com as suas funções, especificidades e arquitetura, bem como a forma como os elementos interagem entre si. Para além disso, é apresentado todo o processo de modelação do sistema tendo por base os requisitos do cliente.

## 4.1 Descrição global do sistema

Este sistema tem como objetivo a supervisão remota da produção de Salicórnia, permitindo não só a monitorização dos dados adquiridos pelos sensores, como também a atuação remota de determinados comandos. Neste contexto, também será possível a aquisição de imagens que possibilitará a deteção de intrusos nas quintas onde se realiza a produção desta espécie. O esquema da figura 4.1 ilustra de um modo geral todos os componentes e as diferentes plataformas com que o cliente pode interagir.

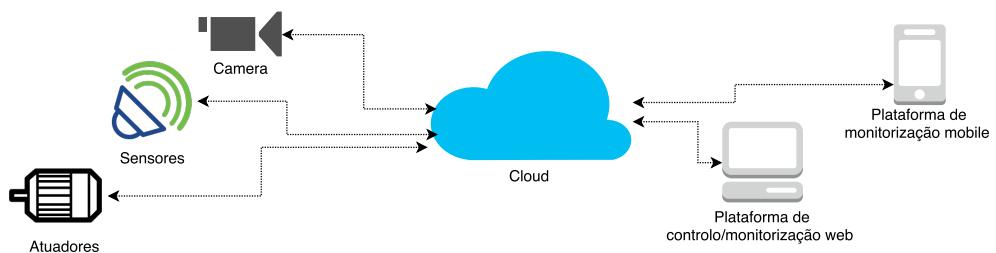


Figura 4.1: Ilustração dos principais componentes do sistema

Como vimos no capítulo 3, uma plantação de Salicornia carece de um controlo relativamente fino de certos parâmetros ambientais sobretudo da salinidade do terreno onde ela cresce, que depende, das chuvas, da salinidade da água dos canais da ria, entre outros. Nas quintas onde se cultiva salicornia, a produção faz-se numa espécie de leiras limitadas por pequenos canais de irrigação que podem ser cheios de água salgada proveniente dos esteiros que rodeiam a quinta. Esta operação implica a abertura de válvulas de admissão dessa água, medida do nível da maré nos canais, monitorização da qualidade e salinidade da água exterior.

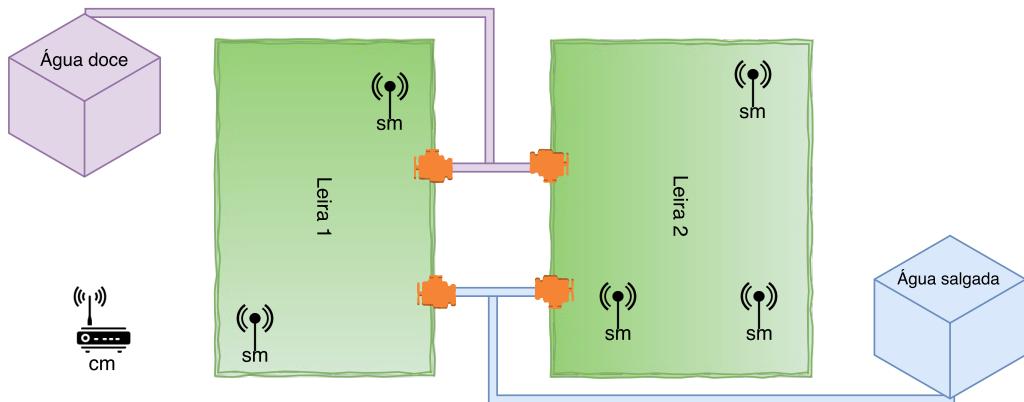


Figura 4.2: Ilustração da distribuição dos módulos em duas leiras

Tal como ilustrado na figura 4.2, pretende-se que sejam colocados módulos com sensores distribuídos estrategicamente por cada leira. Cada um desses módulos, irá comunicar com um módulo central originando uma topologia de rede em estrela. Por sua vez, cada um destes módulos centrais irá comunicar diretamente com o servidor que receberá todos os dados adquiridos tanto pelos sensores como por atuadores, permitindo que estes sejam guardados numa base de dados específica. Os atuadores, permitirão despoletar ações que autorizam a ativação ou desativação de bombas e/ou válvulas para transferências de águas de modo a melhorar as condições de cultivo da Salicornia. Pretende-se que os módulos centrais tenham acesso à camada protocolar TCP/IP (Internet) de modo a conseguirem a utilização da API REST via HTTP desenvolvida para o efeito.

No que diz respeito às plataformas de interação com o cliente, espera-se que exista uma *dashboard* e uma aplicação *mobile*. A *dashboard* disponibilizará uma interface que apresenta as informações mais importantes para o utilizador de forma apelativa, tornando mais fácil a sua interação e respetiva leitura, possibilitando ainda a gestão de todo o sistema e realização de operações de controlo remoto. Por outro lado, a aplicação *mobile* deverá permitir apenas a monitorização do cultivo da Salicornia e receção de alertas quando estes ocorrem.

## 4.2 Componentes

No contexto desta dissertação é necessário reter dois conceitos principais, são eles:

- **Sensor Module:** consiste num módulo responsável pela aquisição de dados provenientes dos mais diversos tipos de sensores.
- **Controller Module:** consiste num módulo responsável pela receção dos dados/estados do *Sensor Module* e respetivo envio para a *cloud*.

O cenário da figura 4.3 ilustra três *Sensor Modules* que comunicam com um *Controller Module*. Cada um desses *Sensor Module* possui um conjunto específico de sensores, podendo estes ser atuadores ou câmaras. Para a comunicação com o *Controller Module*, cada *Sensor Module* possui um determinado módulo de comunicação que permite a transferência dos dados adquiridos pelos sensores. Posteriormente, o *Controller Module* possui um determinado protocolo de comunicação (TCP/IP) que permite a utilização da API e respetivo envio ou atualização dos dados adquiridos pelo sistema.

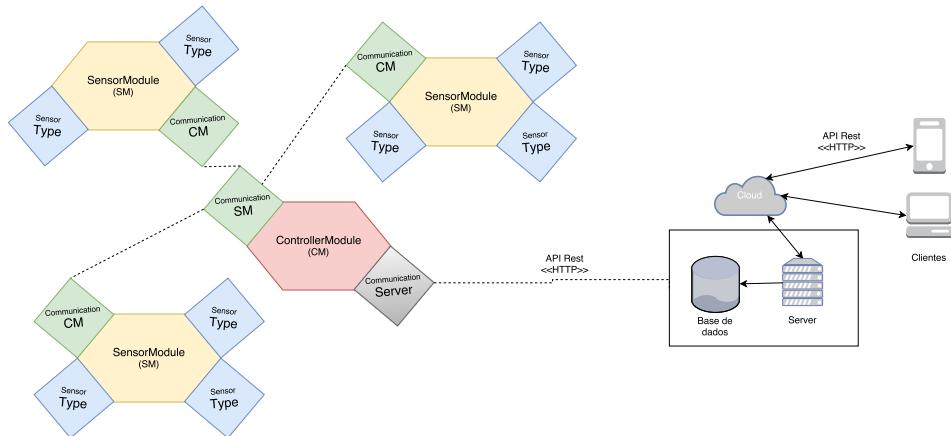


Figura 4.3: Esquema de componentes e respetiva comunicação entre três SM e um CM

Seguidamente serão especificados todos os detalhes de cada módulo, uma vez que serão considerados na modelação de todo o sistema.

### 4.2.1 Sensor Module

Um *Sensor Module* consiste num micro-controlador responsável pela aquisição de dados provenientes dos mais diversos tipos de sensores, podendo estes ser atuadores ou câmaras. No caso de se tratar de um atuador, isto é, válvulas, bombas, contadores, pás ou cancelas, apenas serão lidos valores binários. Caso se trate de uma câmara, todo o processamento é feito internamente nesta, sendo que o sistema apenas irá receber o IP da mesma.

Tal como referido anteriormente, cada *Sensor Module* terá que utilizar um determinado módulo de comunicação para possibilitar a transferência dos dados adquiridos para o módulo central. Para além disso, pretende-se que o *Sensor Module* em condições extremas, possa tomar decisões de atuação, isto é, caso seja lido um valor fora do padrão e que seja necessário a activação de um atuador, este deverá ser auto suficiente em tomar esta decisão, sem necessidade de intervenção do utilizador.

Pretende-se que este módulo seja identificado por um determinado nome, possua uma bateria que permita a sua mobilidade, tenha um ou vários módulos de comunicação acoplados que permitam comunicar com um módulo central, uma memória e um módulo Global Positioning System (GPS) que permita aceder à sua localização, identificando-o em caso de furto. Para além disso, um *Sensor Module* terá que possuir obrigatoriamente um ou vários sensores.

#### 4.2.2 *Controller Module*

Um *Controller Module* consiste num micro-controlador responsável pela receção dos dados provenientes dos vários *Sensor Modules*. Pretende-se que este módulo envie ou receba informações para os *Sensor Module* quando solicitados pelo utilizador. Após a receção dos dados, estes são enviados para um servidor em *cloud* através de uma API Representational State Transfer (REST) criada para o efeito. Sendo que a tecnologia REST opera sob o protocolo de comunicação HTTP, este componente tem que necessariamente estar ligado à rede Internet via Transmission Control Protocol (TCP)/Internet Protocol (IP).

Pretende-se que este módulo possua alguma capacidade de processamento, uma vez que poderá ter vários *Sensor Modules* a si associados e com necessidade de constante envio e receção de dados. Para além disso, pretende-se que o *Controller Module* seja identificado por um determinado nome, tenha um módulo de comunicação que possibilite o envio de dados para um servidor e outros para comunicação com os diferentes *Sensor Modules*. Tal como acontece com os *Sensor Modules*, existe necessidade de acoplado um módulo GPS que permita localizar o micro-controlador em caso de robo.

## 4.3 Análise de requisitos

Durante o desenvolvimento de *software* pressupõe-se que os seus intervenientes sigam determinadas metodologias para que o seu sistema possa revolucionar a vida de um grupo em específico ou até mesmo da sociedade.

O ciclo de vida do desenvolvimento de um software, também conhecido como Systems Development Life Cycle (SDLC), é composto genericamente por quatro fases principais: conceção, projeto, criação e implementação. Antes do SDLC, o processo de desenvolvimento do *software* foi tomado como atividade informal sem regras nem padrões formais. Este facto poderá originar vários problemas, tais como o atraso no desenvolvimento, aumento de custos e baixa qualidade do *software* criado. Existem inúmeros modelos e visões que propõem alguns padrões e etapas necessárias ao desenvolvimento de um sistema com qualidade. Na figura 4.4 encontram-se as várias etapas consideradas por *Munish Kaur et al.*[3].

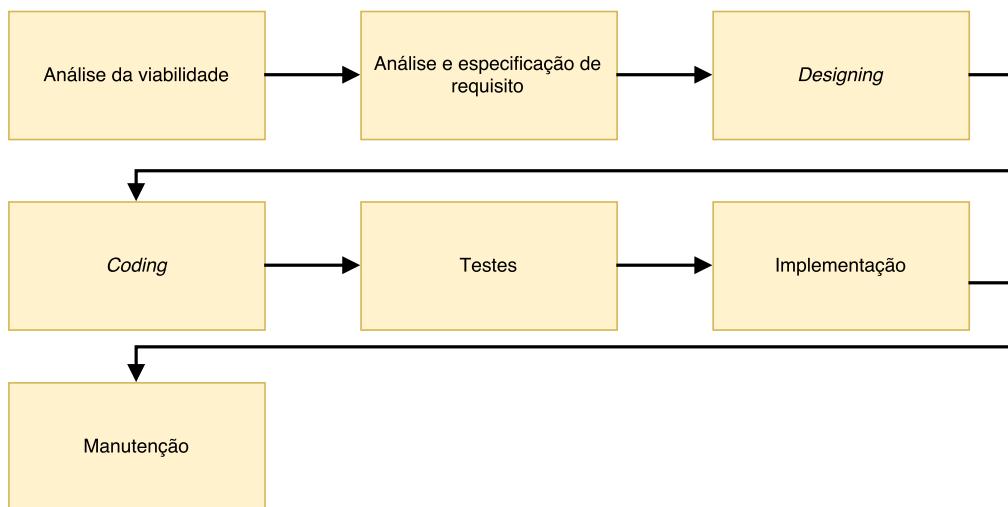


Figura 4.4: Fase de desenvolvimento de um software (Adaptado de [3])

- **Análise de viabilidade:** nesta fase são analisados os dados de entrada e saída, processamento necessário, análise de custos e planeamento do projeto. É ainda incluída a viabilidade técnica em termos de *software*, *hardware* e pessoas qualificadas.
- **Análise e especificação de requisitos:** nesta fase são recolhidos e analisados os requisitos necessário para a elaboração do *software*. No final, pretende-se que sejam conhecidos os vários requisitos do *software* e seja também criado um documento que os especifique.
- **Designing:** consiste na tradução dos requisitos especificados para uma estrutura lógica. No final prevê-se que seja elaborado um documento de especificação do *design*.

- **Coding:** a programação real é elaborada nesta fase. O documento de *design* é traduzido para o código-fonte numa determinada linguagem de forma a que possa ser executado.
- **Testes:** o código-fonte gerado na fase anterior é testado usando vários cenários de testes. São usadas várias técnicas de teste para corrigir e avaliar o *software*.
- **Implementação:** o *software* desenvolvido é implementado para que possa ser disponibilizado ao utilizador para uso real. Pretende-se que o utilizador do sistema possa reportar erros ou problemas quando encontrados.
- **Manutenção:** o *software* poderá sofrer alterações para solucionar problemas que tenham ocorrido. Esta fase é responsável pela pós-implementação e manutenção do *software* para o seu bom funcionamento.

Após o entendimento da descrição geral do sistema bem como todos os componentes envolvidos, segue a fase de efectuar o levantamento dos requisitos do sistema, levando-nos a entender o que o cliente deseja ou acredita precisar possibilitando a criação dos processos de negócio necessários ao sistema. Seguidamente apresentam-se os requisitos funcionais e não funcionais deste sistema.

#### 4.3.1 Requisitos funcionais

Os requisitos funcionais descrevem os critérios que devem ser usados para avaliar as funções específicas ou os comportamentos de um determinado sistema. Os requisitos funcionais que de seguida se apresentam foram propostos pelo cliente no contexto deste projeto para as duas plataformas disponíveis: *web (dashboard)* e *mobile*.

##### Aplicação web (*dashboard*)

- A interface do sistema deve permitir que o utilizador, seja ele qual for, entre ou faça *login* no sistema.
- A interface do sistema deve permitir que o utilizador, seja ele qual for, saia ou faça *logout* no sistema.
- O *dashboard* deverá permitir que qualquer utilizador possa recuperar a sua chave de acesso ao sistema.
- O sistema deve permitir que o utilizador possa adicionar e/ou gerir um ou vários módulos de sensores.

- A aplicação web deve permitir visualizar graficamente os dados que cada sensor obtém.
- O sistema deve permitir visualizar tabularmente (*dataset*) os dados que cada sensor obtém.
- Em cada uma das visualizações anteriormente descritas, deve ser possível efetuar uma filtragem por data.
- O sistema deverá permitir a exportação dos dados obtidos pelos sensores em formato CSV.
- O sistema deve ter a capacidade de gerar alarmes quando algum valor lido pelos sensores esteja fora do previsto.

De modo a tornar o sistema genérico e *standalone* foram impostos os seguintes requisitos adicionais:

- O sistema deve permitir que qualquer utilizador se possa registar no sistema, embora tenha que estar obrigatoriamente associado a uma empresa.
- O utilizador comum só terá acesso à sua área reserva após a validação por parte da empresa.
- A *dashboard* deverá permitir ao administrador a adição de novas empresas e a gestão de todos os utilizadores.
- O sistema deve permitir que qualquer utilizador possa adicionar, editar ou remover:
  - Tipos de sensores;
  - Tipos de comunicação;
  - *Controller Module* com as respetivas especificações e características;
  - Tipo de comunicação a um *Controller Module* que possibilite a sua comunicação como o servidor;
  - *Sensor Modules* a um determinado *Controller Module* com as suas respetivas especificações e características;
  - Um ou vários tipos de comunicação a um *Sensor Module* que permita a sua comunicação com um *Controller Module*.
  - Um ou vários sensores a um *Sensor Module* em que cada sensor poderá ser de um determinado tipo
- O sistema deverá disponibilizar uma API que permita a criação de novas aplicações com base nesta.

- Consultar a documentação da API de modo interativo.
- Gerar e consultar *token* de autenticação para utilização da API
- Alterar configurações base de utilizador

### Aplicação mobile

- A interface da aplicação mobile deve permitir que o utilizador, seja ele qual for, entre ou faça *login* no sistema.
- A interface da aplicação mobile deve permitir que o utilizador, seja ele qual for, saia ou faça *logout* no sistema.
- A aplicação mobile deve permitir visualizar graficamente os dados de cada sensor.
- A aplicação deve permitir receber alarmes quando um determinado valor lido pelo sensor esteja fora do pretendido.

#### 4.3.2 Requisitos não funcionais

Os requisitos não funcionais são todos os requisitos da aplicação relacionados com performance, escalabilidade, segurança, disponibilidade e usabilidade. Estes não são necessariamente pedidos pelo cliente, embora grande parte exista com devido

- O sistema deverá executar em qualquer plataforma, tanto web como mobile.
- A base de dados deve ser protegida para acesso apenas a utilizadores autorizados.
- O sistema deverá disponibilizar uma API para que possam ser criados novos produtos com base neste
- Deverá ser usada, na medida do possível, tecnologias sem qualquer custo para o cliente (*opensource*).
- Pretende-se que o sistema possa ser adaptado a qualquer outro contexto, não sendo especificamente restrito ao contexto da produção da Salicornia.

## 4.4 Modelação

Tal como vimos no início da secção 4.3, a conceção de uma arquitetura envolve o estudo e respetiva modelação dos componentes que são fundamentais para a realização da mesma, bem como a análise dos casos de uso e respetiva especificação, criação do modelo de dados, diagramas de fluxos, entre outros. Toda esta modelação será descrita neste secção permitindo entender melhor como emparelhar toda a estrutura pretendida.

### 4.4.1 Entidades envolvidas

As entidades envolvidas ou atores são os utilizadores do sistema, podendo ser humanos ou entidade máquina, que interagem com o sistema para executar uma determinada ação significativa. No contexto do sistema descrito existem três entidades distintas que são importantes descrever:

- **General user:** este ator poderá registar-se e associar-se a uma determinada empresa existente no sistema. Após a validação por parte da empresa, este utilizador poderá aceder à sua área reservada através da *dashboard* ou da aplicação *mobile*.
- **Company user:** este utilizador tem a possibilidade de gerir todos os *general users* que se possam associar à sua empresa. Deste modo, este utilizador poderá validar ou remover os *general users* que a si se associam.
- **Administrador:** vulgarmente denominado por *admin*. Pretende-se que apenas exista uma único administrador. Este ator tem a possibilidade de adicionar novas empresas ao sistema, isto é, criar novos utilizadores com permissões específicas.

### 4.4.2 Casos de uso

Nas figuras 4.5 e 4.6 encontram-se representados os esquemas dos casos de uso da *dashboard* (denominada por *saliDashboard*) e da aplicação *mobile* (denominada por *saliMobile*), respectivamente. Seguidamente serão descritos cada um dos casos de uso para as duas plataformas. De notar que todos os casos de uso na aplicação *mobile* também se encontram disponíveis através da *dashboard*.

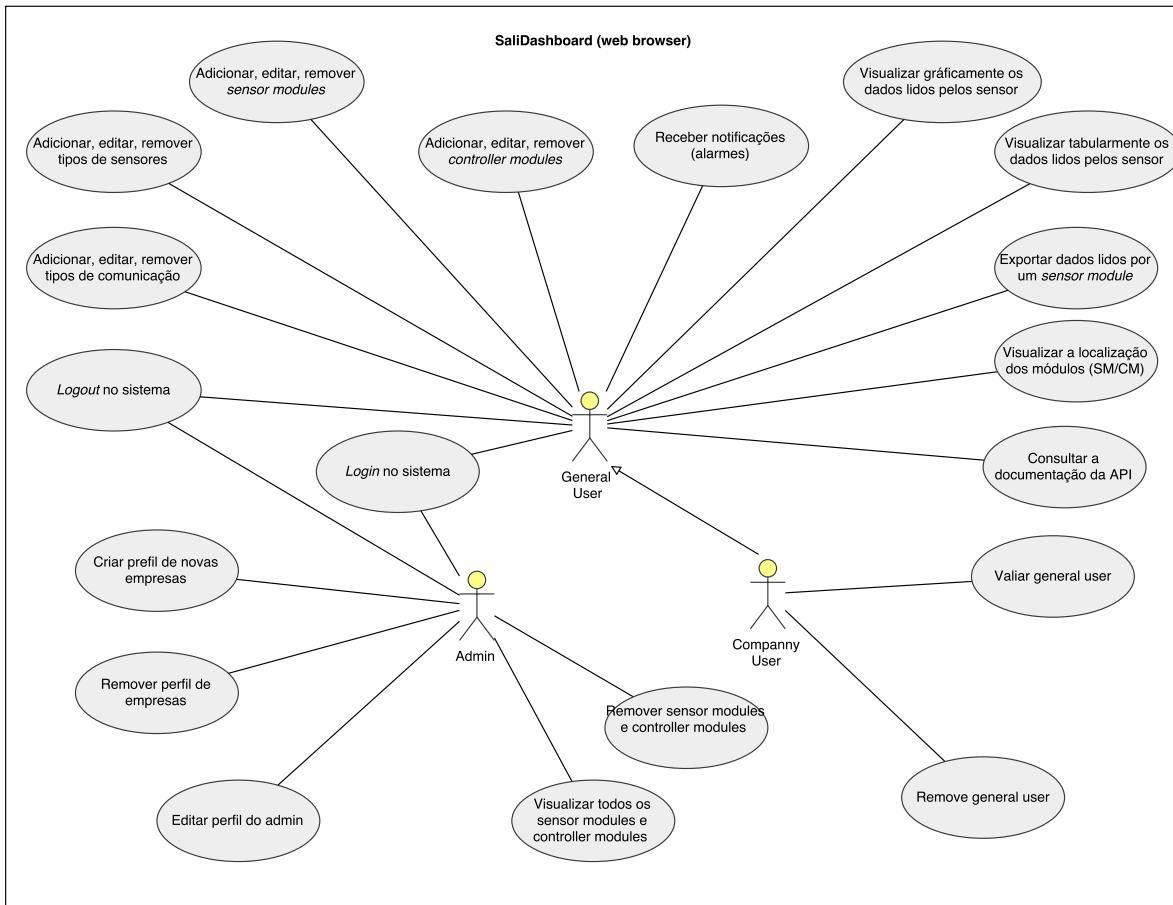


Figura 4.5: Casos de uso para a aplicação web (dashboard)

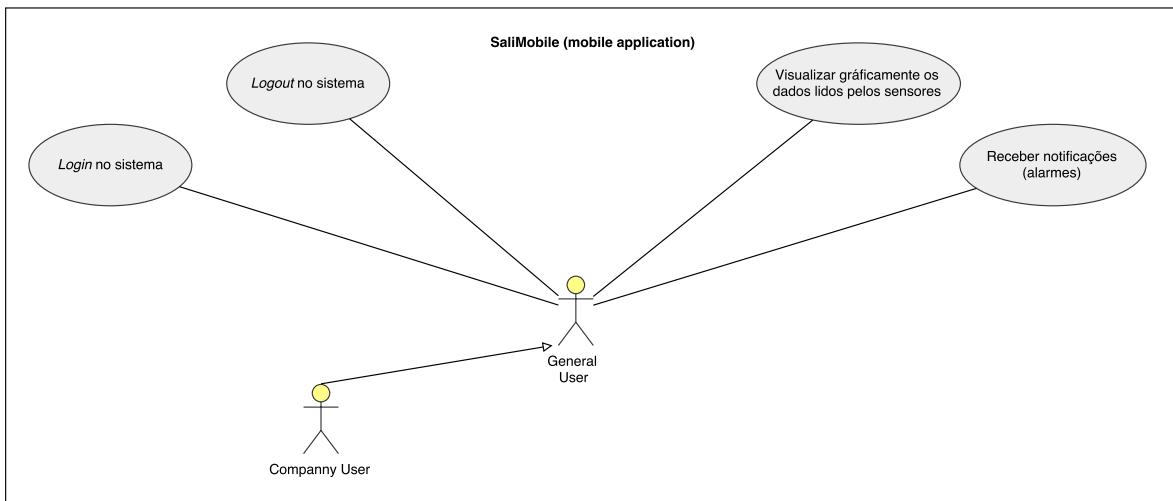


Figura 4.6: Casos de uso para a aplicação mobile

- **Login no sistema:** qualquer utilizador (general user, company user ou admin) poderá aceder ao sistema tendo para isso que possuir uma conta registada e validada no caso do general user. Após a validação das suas credenciais (username e password) o utilizador poderá aceder à pagina inicial da dashboard e a todo o seu conteúdo. Este caso de uso encontra-se disponível para os dois tipos de plataformas.
- **Logout no sistema:** qualquer utilizador (general user, company user ou admin) poderá sair da sistema tendo que para isso tenha que estar obrigatoriamente logado no sistema. Após o logout ser-lhe-á apresentado novamente a página de login. Este caso de uso encontra-se disponível para os dois tipos de plataformas.
- **Recuperar password:** qualquer utilizador (general user, company user ou admin) poderá recuperar a sua password de acesso ao sistema, para isso, terá que introduzir o seu email e posteriormente ser-lhe-á envia o acesso que possibilita a redefinição da mesma.
- **Adicionar, editar, remover tipos de comunicação:** qualquer general user ou company user poderá adicionar, editar ou remover os tipos de comunicação existentes no sistema. Ao adicionar, o utilizador terá que introduzir um nome, um caminho relevante para o tipo de comunicação e selecionar um icon ilustrativo. No caso da remoção, esta ação apenas é possível caso o tipo de comunicação não se encontre em utilização por nenhum CM ou SM.
- **Adicionar, editar, remover tipos de sensores:** qualquer general user ou company user poderá adicionar, editar ou remover os tipos de sensores existentes no sistema. Ao adicionar, o utilizador terá que introduzir um nome que o identifique, uma fonte de dados ou a escala dos dados e um icon que identifique o sensor. Para além disso, o utilizador terá que escolher uma cor que identifique o tipo de sensor na dashboard. No caso da remoção, esta ação apenas é possível caso o tipo de sensor não se encontre em utilização por nenhum SM.
- **Adicionar, editar, remover um controller module:** qualquer *general user* ou *company user* poderá adicionar, editar ou remover um controller module existente no sistema. Neste caso, pressupõe-se que o utilizador possua um micro-controlador com as seguintes características que podem ser adicionadas: determinado nome que o identifique, valor da memória RAM em MB, estado inicial (ativo ou desativo) e um modulo de comunicação que permita comunicar com o servidor web. Estas características são possíveis de alteração. Pretende-se que o controller module possa ser removido tendo SM a sí associado, sendo estes também removidos dos sistema.

- **Adicionar, editar, remover um sensor module:** qualquer *general user* ou *company user* poderá adicionar, editar ou remover um sensor module existente no sistema. Pressupõe-se o utilizador possui um determinado micro-controlador com um ou mais sensores. Para adicionar o SM à plataforma o utilizador terá que atribuir um nome que o identifique, definir qual o seu estado inicial (ativo ou desativo), escolher que tipos de comunicação possam estar presentes e definir o intervalo de tempo para o qual pretende receber os dados lidos pelos sensores. Para além disso, permite associar ao SM os sensores presentes e definir o valor máximo e mínimos para o qual são gerados alarmes e respectivas mensagens informativas. Todos os dados são possíveis de edição.
- **Receber notificações (gerar alarmes):** sempre que um valor lido por um determinado sensor sai fora do limite imposto (valor máximo e valor mínimo) é gerado um alarme para o utilizador de modo a que este possa intervir. Este caso de uso encontra-se disponível para os dois tipos de plataformas.
- **Visualizar gráficamente os dados lidos pelos sensores:**  
Este caso de uso encontra-se disponível para os dois tipos de plataformas.
- Visualizar tabularmente os dados lidos pelos sensor
- **Exportar dados lidos por um sensor module:** qualquer *general user* ou *company user* poderá exportar os dados lido pelos sensores para um ficheiro do tipo CSV para uma data especificada.
- **Visualizar a localização dos módulos (SM/CM):** qualquer *general user* ou *company user* ao aceder à dashboard ser-lhe-à apresentado um mapa para cada CM com a sua localização e dos seus respetivos SM.
- **Consultar a documentação da API:** qualquer *general user* ou *company user* poderá aceder à dashboard para consultar a documentação da API existente.
- **Consultar token de autenticação:** qualquer *general user* ou *company user* poderá aceder ao token de autenticação para utilização da API.
- **Validar general user:** qualquer company user tem a possibilidade de validar os general users que a si se associam. Sempre um novo general user é registado no sistema o company user é notificado via email. Posteriormente, caso o general user seja validado este também receberá um email de confirmação.
- **Remove general user:** qualquer company user tem a possibilidade de remove os general users que a si estão associados.

- **Remover sensor modules e controller modules** : o admin do sistema tem a possibilidade de remover os sensores modules e/ou controller modules existentes no sistema.
- **Visualizar todos os sensor modules e controller modules**: o admim do sistema tem a possibilidade de visualizar todos os sensores modules e controller modules existentes no sistema de modo a alertar os clientes em caso de anomalias.
- **Criar um novo company user**: o admim tem a possibilidade de adicionar novos company user ao sistema, sendo para isso necessário
- **Remover um company user**: o admim tem a possibilidade de remover qualquer company user registado no sistema.

#### 4.4.3 Modelo de dados

Depois da análise de requisitos desenhou-se um modelo de dados que permitisse modelar todo o sistema descrito, obtendo-se assim o seguinte esquema relacional apresentado na figura 4.7.

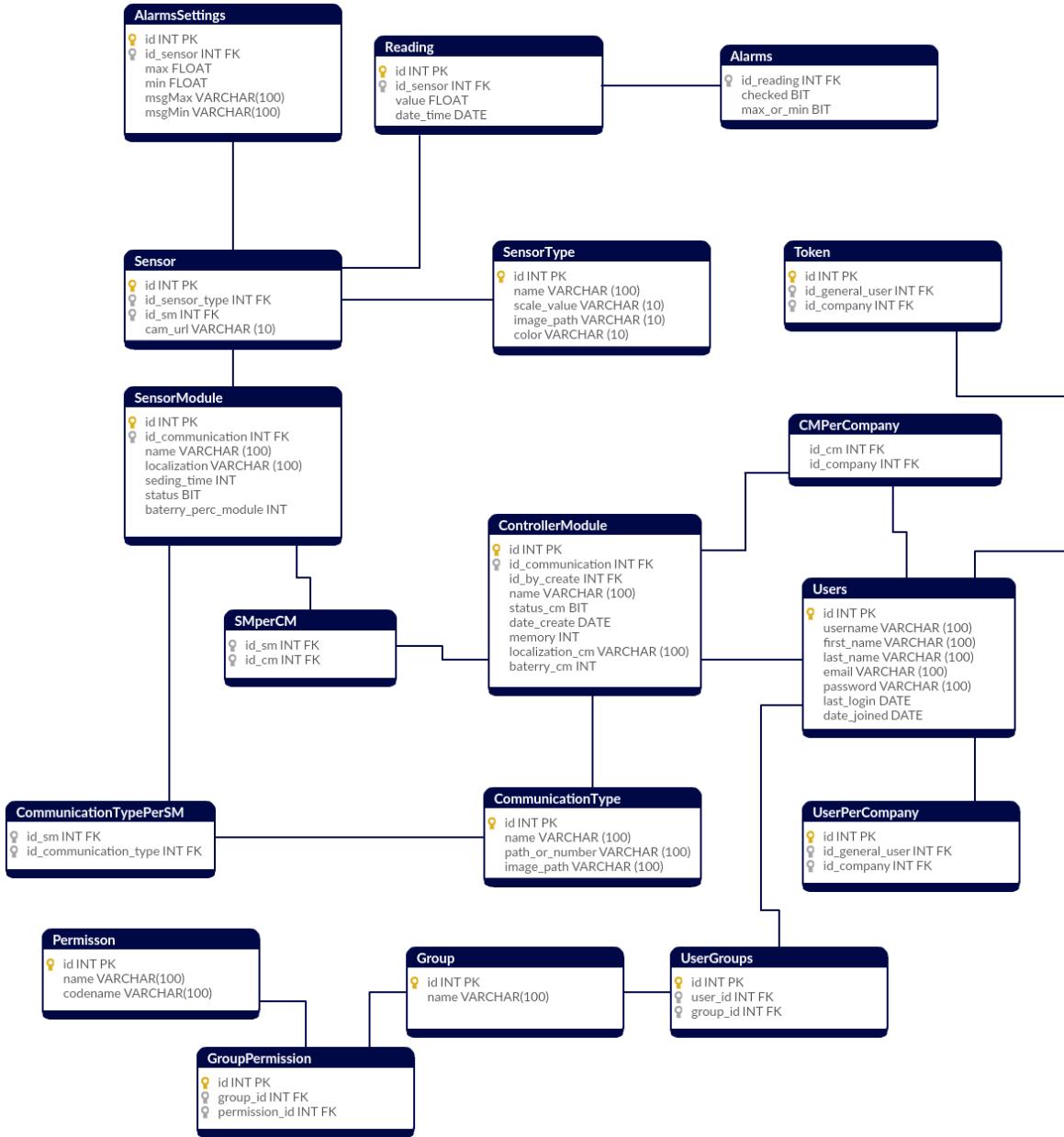


Figura 4.7: Esquema relacional da estrutura da base de dados

Nas tabelas 4.1 e 4.2 são descritos cada uma das entidades de dados existente neste sistema, evidenciando as chaves primárias e estrangeiras de cada uma.

| Nome da tabela    | Chave primária (PK)                              | Chave estrangeira (FK)     | Descrição   |
|-------------------|--|----------------------------|---|
| User              | id (auto-incrementado)                           | N/A                        | Identifica cada um dos usuários inseridos no sistema                  |
| Token             | auth_token_token_pkey<br>(character varying(40)) | user_id                    | Possui o token de autenticação do utilizador para a API               |
| Group             | id (auto-incrementado)                           | N/A                        | Possui todos os grupos existentes: general user, company user e admin |
| UserGroups        | id (auto-incrementado)                           | user_id group_id           | Permite associar um utilizador a um determinado grupo                 |
| Permissom         | id (auto-incrementado)                           | N/A                        | Possui todas as permissões existentes (escrita, leitura, delete...)   |
| GroupPermissom    | id (auto-incrementado)                           | group_id permission_id     | Associa a cada grupo determinadas permissões                          |
| UserPerCompany    | id (auto-incrementado)                           | company_id general_user_id | Associa cada general user a um company user                           |
| CommunicationType | id (auto-incrementado)                           | N/A                        | Identifica cada um dos tipos de comunicação inseridos no sistema      |
| ControllerModule  | id (auto-incrementado)                           | N/A                        | Identifica cada um dos controller module inseridos no sistema         |

Tabela 4.1: Especificação das tabelas existentes no sistema

| Nome da tabela                | Chave primária (PK)    | Chave estrangeira (FK)         | Descrição  |
|-------------------------------|------------------------|--------------------------------|--|
| <b>CMPerCompany</b>           | id (auto-incrementado) | cm_id<br>company_id            | Associa todos os controller module a um determinado company user         |
| <b>SensorModule</b>           | id (auto-incrementado) | N/A                            | Identifica cada um dos sensor module inseridos no sistema                |
| <b>SMperCM</b>                | id (auto-incrementado) | cm_id<br>sm_id                 | Associa os sensor modules a um determinado controller module             |
| <b>CommunicationTypePerSM</b> | id (auto-incrementado) | communication_type_id<br>sm_id | Permite associar a um sensor module um ou várias tipos de comunicação    |
| <b>SensorType</b>             | id (auto-incrementado) | N/A                            | Identifica cada um dos tipos de sensores inseridos no sistema            |
| <b>Sensor</b>                 | id (auto-incrementado) | sensor_type_id<br>sm_id        | Permite identificar um determinado sensor                                |
| <b>AlarmsSettings</b>         | id (auto-incrementado) | sensor_id                      | Permite guardar as configurações para um determinado sensor (max,min...) |
| <b>Reading</b>                | id (auto-incrementado) | sensor_id                      | Permite guardar as leituras de um determinado sensor                     |
| <b>Alarms</b>                 | id (auto-incrementado) | reading_id                     | Armazena os alarmes que são gerados                                      |

Tabela 4.2: Especificação das tabelas existentes no sistema (continuação)

## 4.5 Arquitetura lógica

Nesta secção é apresentada a arquitetura lógica do sistema, indicando as camadas que contém, especificando quais as relações de dependência que estas possuem entre si. Seguidamente, é explicado como implementar esta lógica com os respetivos componentes físicos.

Normalmente este tipo de arquitetura é composto por três camadas com intenções diferentes: camada de apresentação, camada de lógica de negócio e camada de acesso a dados. Pretende-se que com a descrição desta arquitetura seja facilitada a manutenção, a portabilidade e a escalabilidade, fatores importantes quando queremos partilhar funcionalidades e informação entre aplicações de diferentes tipos.

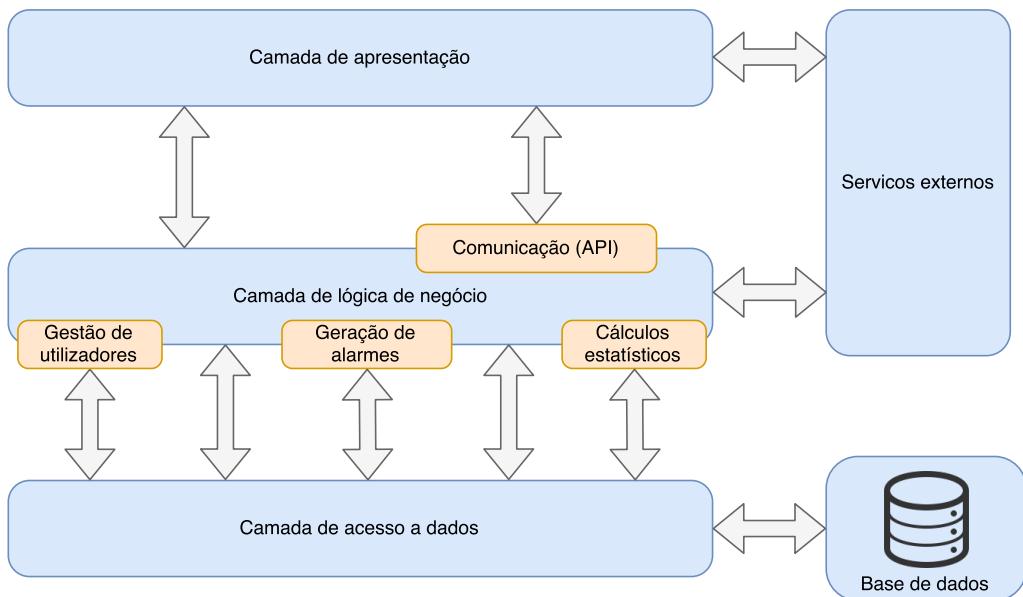


Figura 4.8: Arquitetura lógica

A camada de apresentação é responsável pela comunicação entre os utilizadores e a aplicação, sendo ela *web* ou *mobile*, exibindo informações aos utilizadores, abrangendo uma interface que permite solicitações ao sistema. Esta camada tem uma relação de dependência com a camada de lógica de negócio e tira partido do acesso a serviços de informação externos, que fornecem diversas funcionalidades. A interface do utilizador foi desenvolvida em HTML e CSS, fazendo uso de jQuery e Javascript.

A camada de lógica de negócio diz respeito à camada onde é realizado todo o processamento dos dados adquiridos pelos sensores ou introduzidos pelos utilizadores do sistema através da camada de apresentação. Tal como apresentado na figura 4.8, existem quatro sub-camadas principais, que enfatizam os principais conceitos existentes nesta camada, a seguir descritos.

- **Gestão de utilizadores:** todas as operações realizadas por cada utilizador devem ser medidas pelas permissões que estes possuem. Nesta camada é verificado se um determinado utilizador viola ou não essas permissões.
- **Geração de alarmes:** todos os alarmes são gerados conforme a verificação automática realizada a cada valor que chega ao sistema.
- **Cálculos estatísticos:** este componente da camada de lógica ganha especial relevância na altura em que se pretende determinar os valores médios, máximos e mínimos de um conjunto de medições para um determinado intervalo de tempo.
- **Comunicação via API:** este componente é fundamental para que se possa aceder, atualizar ou adicionar dados ao sistema de um modo simples.

Relativamente à camada de acesso a dados, deverão estar presentes as funcionalidades de criação, edição, remoção ou simples de visualização dos dados, sendo responsável pelas operações de persistência e consulta de dados solicitados pela camada de lógica de negócio.

## 4.6 Arquitetura física

Enquanto que a arquitetura lógica se concentra nos diferentes níveis de abstração do sistema a arquitetura física foca-se na estrutura de alto nível. Seguidamente são apresentados todos os componentes físicos, tanto a nível de *software* com de *hardware*, que são fundamentais para ter um maior percepção do real funcionamento de todo o sistema. A figura 4.9 representa genericamente os blocos principais do sistema proposto que seguidamente serão descritos.

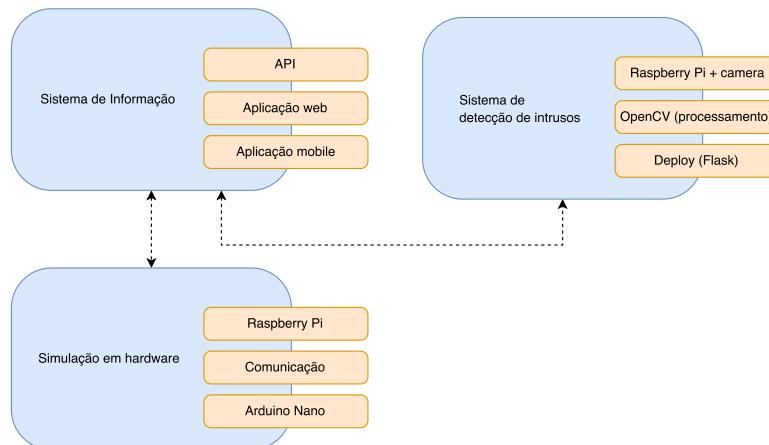


Figura 4.9: Arquitetura física (blocos)

#### 4.6.1 Sistema de informação

Segundo *Laudon et al.*[23] um sistema de informação define-se como sendo uma inter-relação de múltiplos componentes podendo estes ser equipamentos, telecomunicações, *software*, bases de dados e outras tecnologias de transformação de informação. Todos estes componentes permitem a recolha, processamento, armazenamento e distribuição de informação que possibilita a tomada de decisões e controlo para uma determinada organização, ou até mesmo para a sociedade, de modo a torná-la mais acessível e útil.

Dada a elevada complexidade de um sistema de informação, é possível identificar algumas funcionalidades comuns aos diversos sistemas existentes, são elas[24]:

- **Recolha de dados:** sequência de tarefas que possibilitam a adição de novos dados ao sistema.
- **Organização e armazenamento de dados:** é imprescindível uma boa organização na estrutura de dados possibilitando uma fácil e rápida localização.
- **Processamento de dados:** qualquer funcionalidade que permita a produção de resultados mais úteis do que os dados em bruto.
- **Distribuição de informação:** após o processamento de dados é fundamental a distribuição destes a quem precise deles.
- **Utilização da informação:** por si só, a informação não tem qualquer valor, a sua utilização em contexto adequado possibilita a extração de determinadas conclusões para que possam ser tomadas decisões.

A figura 4.10 ilustra a arquitetura do sistema de informação incluindo especificamente a aplicação web (*dashboard*), base de dados, API e respetiva implementação do sistema.

#### Aplicação web

A aplicação web é um componente essencial, enquadrando-se na camada de lógica de negócio como também na camada de apresentação, permitindo a interação por parte do utilizador (*frontend*) como também no processamento lógico (*backend*).

Tal como vimos na secção do Estado de Arte, a tecnologia para o desenvolvimento web recaiu sobre a *framework* Django, mais precisamente na versão 1.11 para python 2.7, sendo que como Integrated Development Environment (IDE) foi utilizada a verão 2016.3.3 do PyCharm. Dada a facilidade com que esta *framework* tem em manipular views e templates, optou-se que ambas as componentes (*frontend/backend*) fossem desenvolvidas recorrendo ao Django.

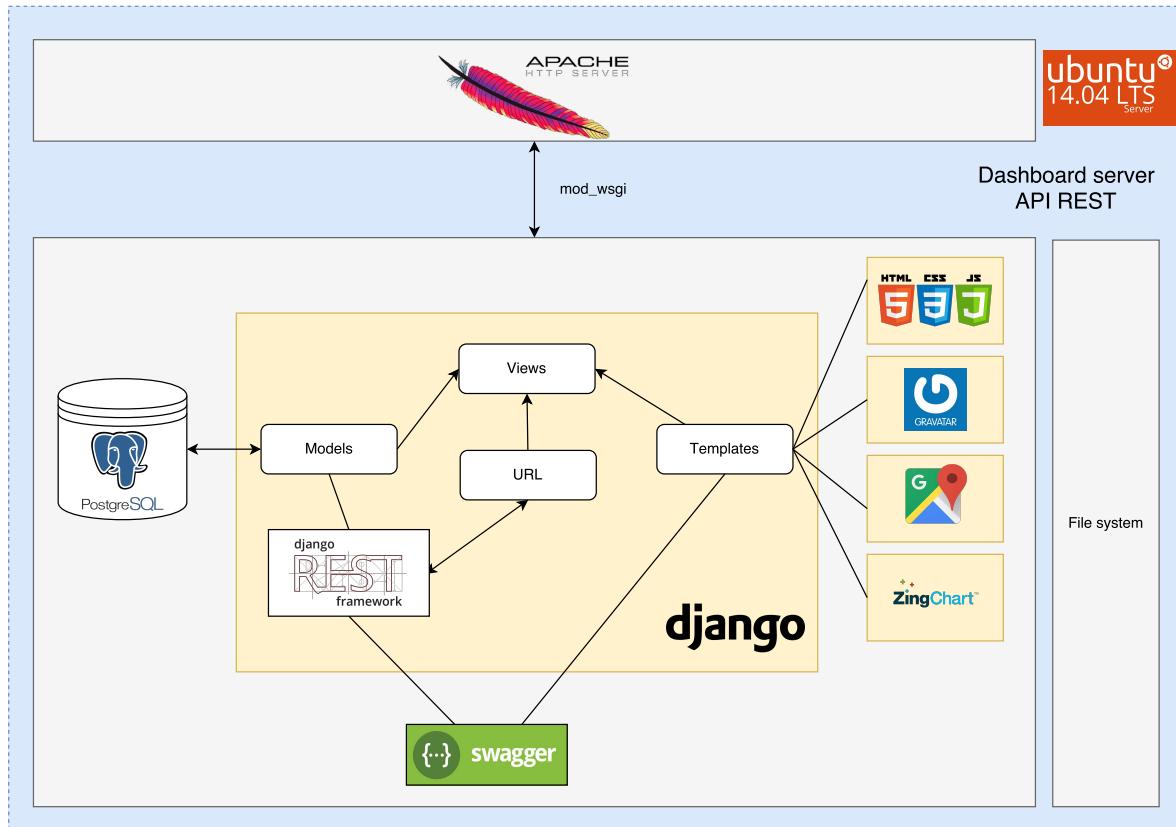


Figura 4.10: Arquitetura do sistema de informação (*dashboard*, base de dados e API)

Para além disso, optou-se por criar uma API REST que permitisse a manipulação dos dados existentes na base de dados, sendo esta também desenvolvida paralelamente com a aplicação web. Seguidamente será explicada a sua arquitetura.

Um dos pontos mais interessantes no Django é sem dúvida a existência de Object-Relational Mapping (ORM). Consiste numa técnica de desenvolvimento que permite modelar os dados através de classes em Python. Através dela, é possível gerar tabelas na base de dados e manipulá-las sem a necessidade de interagir diretamente com Structured Query Language (SQL) (embora também seja possível). A verificação dos valores lidos pelos sensores e possível geração de alarmes é averiguado por um *trigger* desenvolvido em SQL, no capítulo 5 será explicada a sua implementação.

Relativamente à escolha do SGBD recaiu sobre o PostgreSQL, mais concretamente na versão xxx. Como ferramenta gráfica para administração deste SGBD foi utilizado o PgAdmin III<sup>1</sup> na versão 1.22. Este software gráfico tem inúmeras funcionalidades desde a possibilidade de ligação a base de dados remotas até à adição, edição, remoção e consultas em tabelas. Esta ferramenta é *opensource* e encontra-se disponível para Windows e UNIX. Como vimos

<sup>1</sup>[www.pgadmin.org/](http://www.pgadmin.org/)

no capítulo 3, este SGBD permite uma grande facilidade de incorporação a um projeto em Django.

No que diz respeito à camada de apresentação na plataforma web foram utilizadas as seguintes bibliotecas/*frameworks*:

- **HTML, CSS e JS:** o ponto de partida para a criação da interface web assentou no template denominado por AdminLTE<sup>2</sup> sendo este baseado em Bootstrap 3<sup>3</sup>. Neste template prevalecem as seguintes características: design responsivo, interface leve e atraente, existência de múltiplos plugins, compatibilidade entre navegadores entre outros.
- **Gravatar:** serviço que disponibiliza um avatar que esteja associado a endereços de email registado. O Gravatar disponibiliza uma API que pode ser utilizada nas mais diversas linguagens de programação[25].
- **API Google Maps:** consiste num serviço de visualização de mapas e imagens de satélite, desenvolvido pela Google. É usado na visualização da localização dos módulos de sensores.
- **ZingChart:** biblioteca em JS que permite receber dados a apresentá-los em formato gráfico. Esta solução *opensource* disponibiliza mais de 35 tipos e módulos de gráficos.

## API REST

Os métodos desta API permitiram execuções funções do tipo REST. A tecnologia REST foi apresentada por Roy Fielding na Universidade de Califórnia no ano de 2000, cujo o título da sua dissertação era "Architectural Styles and the Design of Network-based Software Architectures". Roy estudou um conjunto de arquiteturas de software que usam a Web como uma plataforma para computação distribuída[26]. Esta tecnologia define um conjunto de princípios que possibilitam desenhar serviços web com base nos recursos do sistema, considerando a forma com os recursos são coordenados e transferidos através de HTTP para vários clientes nas mais diversificadas linguagens.

Os métodos da API permitem executar as funções REST usando métodos HTTP explicitamente. Assim, torna-se fundamental perceber estes métodos para ter um melhor conhecimento do funcionamento da API. Seguidamente são descritos os métodos mais importantes que dão suporte a cada uma das funções REST.

- **GET:** permite efectuar operações de leitura

---

<sup>2</sup><https://adminlte.io/>

<sup>3</sup><http://getbootstrap.com/>

- **POST**: permite realizar operações de escrita, permitindo criar novos recursos ao sistema.
- **PUT**: permite criar ou atualizar um novo objecto ao sistema
- **DELETE**: permite apagar objecto ou recurso ao sistema.

Como vimos no capítulo 3, a escolha da *framework* para construção desta API REST recaiu sobre o *Django REST framework*. Esta framework possui uma extensa documentação e um elevado apoio da comunidade que a utiliza, sendo uma das mais utilizadas e incorporadas em projetos Django.

Relativamente à autenticação desta API, optou-se por utilizar um esquema de autenticação fundamentado no mecanismo de autenticação HTTP baseado em *tokens*[27]. Neste mecanismo, o cliente primeiramente troca as suas credenciais (username e password) por um token, seguidamente, em vez de enviar essas credenciais a cada requisição, o cliente apenas enviará o token inicialmente recebido, permitindo assim o acesso aos conteúdos pretendidos, com respetiva autenticação e autorização dos mesmos (figura 4.11).

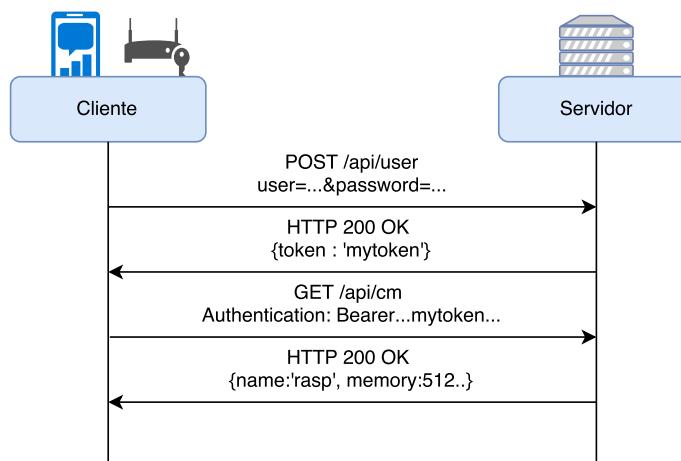


Figura 4.11: Processo de autenticação em HTTP através de token (adaptado de [1])

Resumidamente e com o objetivo de explicar o esquema da figura 4.11, são enumerados os passos de autenticação em HTTP via token.

1. O cliente envia as credenciais para um servidor.
2. O servidor valida e autentica essas credenciais gerando consequentemente um token.
3. O servidor envia o token para o cliente.
4. O cliente guarda o token e este é enviado sempre que existe uma requisição através do cabeçalho do protocolo HTTP.

5. O servidor, em cada requisição verifica se o token é válido ou não. Caso seja, o servidor aceita a requisição, caso contrário esta é rejeitada.
6. O servidor pode ter um endpoint que renova o token sempre que necessário.

Na tabela 4.3 encontram-se todos os endpoints e respectivas funções (POST/GET/PUT) a implementar. Seguidamente será descrito como foi implementada a documentação desta API.

| Endpoints da API REST                            | POST | GET | PUT | DELETE |
|--|------|-----|-----|--------|
| /api/user/                                       | ✓    | ✓   |     |        |
| /api/user/{pk_or_username}/                      |      | ✓   | ✓   | ✓      |
| /api/smpercm/                                    |      | ✓   |     |        |
| /api/smpercm/{pk_or_name_cm}                     | ✓    | ✓   |     |        |
| /api/sm/   | ✓    | ✓   |     |        |
| /api/sm/{pk_or_name}/                            |      | ✓   | ✓   | ✓      |
| /api/sensortype/                                 | ✓    | ✓   |     |        |
| /api/sensortype/{pk_or_name}                     |      | ✓   | ✓   | ✓      |
| /api/sensorpersrm/{id_sm_or_name_sm}             | ✓    | ✓   |     |        |
| /api/sensor/                                     |      | ✓   |     |        |
| /api/sensor/{pk_or_sensor_type}                  | ✓    | ✓   |     |        |
| /api/reading/{id_sensor}/{date_start}/{date_end} | ✓    | ✓   |     |        |
| /api/communication/{pk_or_name}/                 |      | ✓   | ✓   | ✓      |
| /api/cm/   | ✓    | ✓   |     |        |
| /api/cm/{pk_or_name}/                            |      | ✓   | ✓   | ✓      |
| /api/alarmssettings/{id_sensor}                  | ✓    | ✓   |     |        |
| /api/alarms_sensor/{id_sensor}                   | ✓    | ✓   |     |        |
| /api/alarms_reading/{id_reading}                 | ✓    | ✓   |     |        |

Tabela 4.3: Endpoints da API REST e respetivos métodos a implementar

## Documentação interativa

Para a geração automática da documentação da API utilizou-se o Swagger. Tal como descrito no site oficial desta framework[28], o Swagger é considerado a ferramenta de APIs mais popular e completa de todo o mundo permitindo facilmente o desenvolvimento do ciclo de vida de uma API, desde o design, documentação, testes e também implementação, tendo a grande vantagem de ser *opensource*. Neste contexto, apenas será utilizada como documentação, de modo a facilitar a interpretação da API criada. O Swagger possui uma interface apelativa e intuitiva, permitindo interagir com a API de modo que os seus utilizadores tenham uma ideia geral de como esta responde aos pedidos para diversos parâmetros e opções.

## Implementação do sistema

Para implementação do projeto anteriormente descrito e respetiva API, foi-me fornecida uma máquina com um distribuição Linux (Ubuntu 14.04.5) com as seguintes características:

- Central Processing Unit (CPU): Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
- Random Access Memory (RAM): 2 GB
- Disco: 10.7 GB

Para o processo de *deployment* deste projeto optou-se pela utilização do servidor Apache juntamente com o pacote mod\_wsgi<sup>4</sup>. Este pacote fornece uma Web Server Gateway Interface (WSGI) compatível para o alojamento de aplicações web em Python sob o servidor HTTP Apache. O Apache é um servidor web *opensource* mais utilizado em todo o mundo[29].

## Aplicação mobile

Após a análise de requisitos da aplicação mobile, optou-se por elaborar um protótipo da aplicação mobile antes de proceder à sua real implementação. O *mockup* da aplicação apresenta-se no Apêndice X.

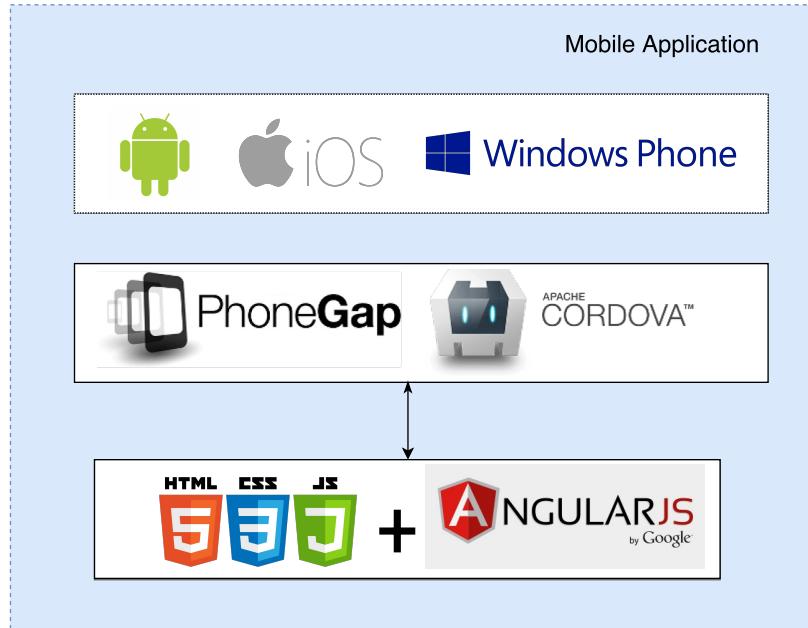


Figura 4.12: Arquitetura da aplicação mobile

<sup>4</sup><https://modwsgi.readthedocs.io/en/develop/>

Tal como descrito no capítulo 3, para o desenvolvimento do aplicativo mobile optou-se pela utilização de um paradigma multi-plataforma, mais concretamente a *framework* Phonegap. Esta framework utilizada a tecnologia Cordova da Apache que permite a integração com recursos nativos dos dispositivos. Através dela, é possível desenvolver aplicações móveis utilizando simplesmente HTML, CSS e JS sem a necessidade de depender de APIs específicas. De modo a facilitar a manipulação do JS optou-se por utilizar a *framework* AngularJS. Esta framework *opensource* mantida pela google desde 20XX

VAMOS USAR ANGULAR 2

Consumir a API REST

Consumir a API REST Consumir a API REST

#### 4.6.2 Simulação em *hardware*

Após a desenvolvimento da API, pretendeu-se simular o sistema num contexto real. Para tal, pretendeu-se encontrar hardware que encaixasse no contexto deste projeto. Foram utilizados dois micro-controladores (Arduino e Raspberry Pi 3) e alguns sensores. Para este cenário, assume-se que o Arduino é considerado um *Sensor Module* que possui um conjunto de sensores enquanto que o Raspberry Pi 3 é um *Controller Module* que recebe os dados provenientes do *Sensor Module* enviando-os para o servidor.

Seguidamente, são apresentados os sensores utilizados bem como os tipos de comunicação.

## Sensores utilizados

Nesta secção serão apresentados os sensores utilizados na simulação e as suas principais características. Todos os sensores foram escolhidos tendo em conta o seu enquadramento no projeto e a sua disponibilidade em laboratório, sendo que serão ligados ao Arduino Nano.

### Temperatura

Como sensor de temperatura foi utilizado um termíster do tipo Negative Temperature Coefficient (NTC). Como vimos no capítulo 3, um termíster é um semicondutor sensível à temperatura, ou seja, quando o coeficiente de variação da resistência com a temperatura é negativa, então a temperatura sobe e consequentemente a resistência diminui. Na figura 4.14 encontra-se o esquema de ligação deste componente e na tabela 4.4 as suas propriedades principais.



Figura 4.13: Sensor TTC 104 NTC

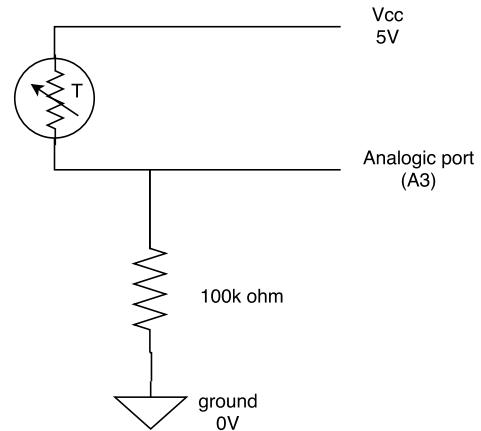


Figura 4.14: Esquema eletrotécnico da ligação do sensor de temperatura

|                    |                |
|--------------------|----------------|
| Dimensão           | 5mm            |
| Resistência        | 100 KΩ         |
| Valor máximo       | +125 °C        |
| Valor mínimo       | -30 °C         |
| Nível de confiança | + - 10%        |
| Preço              | 0.35 €/unidade |

Tabela 4.4: Características do sensor TTC 104 [30]

### Luminosidade

Para simular a luminosidade incidente foi utilizado um sensor do tipo foto-resistência. Este sensor, também conhecido como Light Dependent Resistor (LDR), não é mais do que uma resistência variável cujo o seu valor varia conforme a intensidade da luz que incide sobre ele,

isto é, à medida que a intensidade da luz aumenta, a sua resistência diminui. Este sensor tem múltiplas aplicações, entre as quais se destaca a monitorização solar, indicador da posição do sol (up/down), alarmes anti-roubo, alarme para abertura/fecho de portas entre outras. Como vimos no capítulo 3 é um sensor de baixo custo e bastante fácil de utilização. Na figura 4.16 encontra-se o esquema de ligação do componente e na tabela 4.5 são apresentadas as principais características do sensor utilizado.

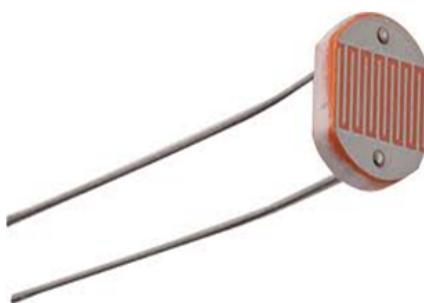


Figura 4.15: Sensor foto-resistência GL5528

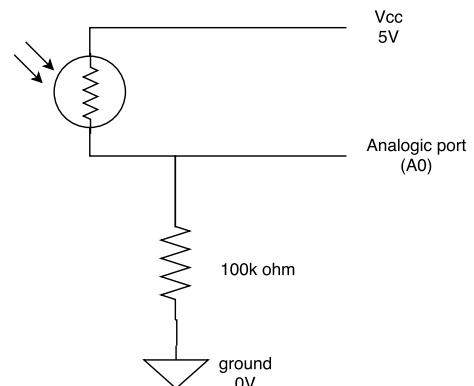


Figura 4.16: Esquema eletrotécnico da ligação do sensor de luminosidade

|                           |                       |
|---------------------------|-----------------------|
| Diâmetro                  | 5mm                   |
| Tensão máxima             | 150VDC                |
| Potência máxima           | 100mW                 |
| Tensão de operação        | -30 °C a 70 °C        |
| Espectro                  | 540nm                 |
| Comprimento com terminais | 32mm                  |
| Resistência no escuro     | 1 M (Lux 0)           |
| Resistência na luz        | 10-20 Komega (Lux 10) |
| Material                  | Carbono               |
| Preço                     | 0.22 €/unidade        |

Tabela 4.5: Características do sensor GL5528 [31]

#### Sensor para verificação do estado do nível de água

Este sensor não é mais do que um interruptor que é ativo sempre que um determinado líquido ultrapassa o mesmo, isto é, sempre que algum líquido atingir o pedaço de plástico este irá subir ativando assim o circuito. Na figura 4.18 encontra-se o esquema da ligação deste sensor.



Figura 4.17: Water Level Switch Liquid Level Sensor Plastic Ball Float



Figura 4.18: Esquema eletrotécnico da ligação do sensor de nível líquido

### Simulador de válvula para transferências de águas

Para a simulação de uma válvula que permitirá as transferências de água doce e/ou água salgada foi utilizado um Light Emitting Diode (LED). Este possibilita facilmente identificar através da sua activação se a válvula se encontra ativa ou não.



Figura 4.19: Led

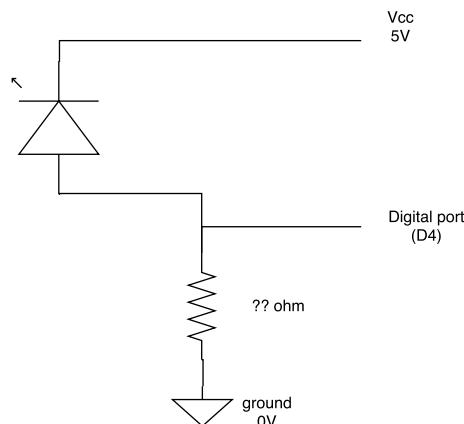


Figura 4.20: Esquema eletrotécnico da ligação do led

### Comunicação

Nesta secção, serão apresentados os tipos de comunicação para o cenário apresentado. Pretende-se que cada um dos módulo fique isolado entre si, o que implicou o estudo e respetiva escolha de algumas tecnologias de comunicações sem fio (secção XX do capítulo 3). Neste caso, foram escolhidas as seguintes:

- **Bluetooth:** utilizado para a comunicação entre o Arduino (*Sensor Module*) e o Raspberry Pi 3 (*Controller Module*). No Arduino foi utilizado um módulo Bluetooth HC-06

e no Raspberry Pi 3 o seu próprio módulo interno.

- **Wi-Fi:** utilizado para a comunicação entre o Raspberry Pi 3 (*Controller Module*) e o servidor web.

O esquema da figura 4.21 pretende ilustrar os tipos de comunicação envolvidos nesta simulação para cada um dos componentes.

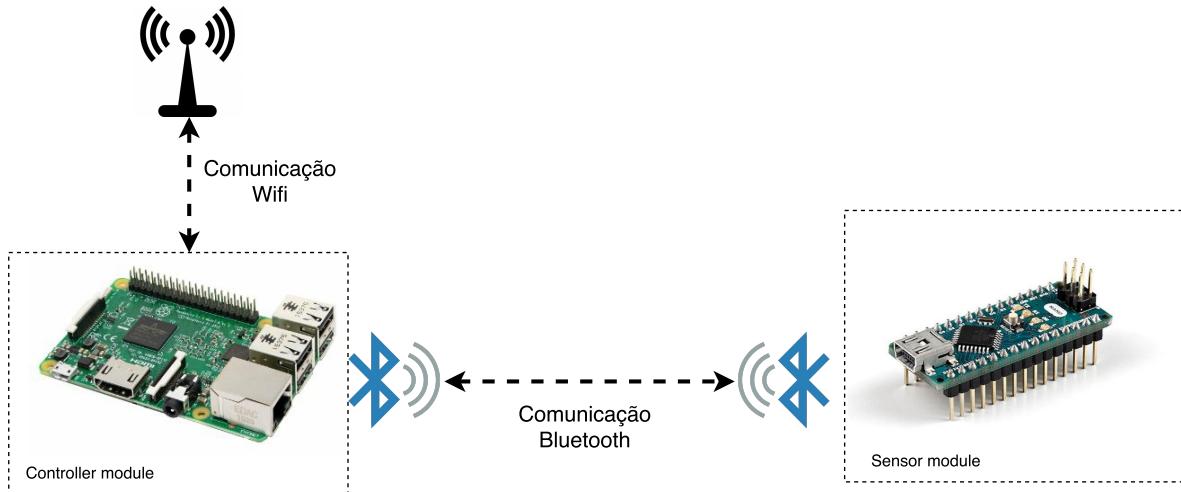


Figura 4.21: Comunicação entre componentes da simulação em hardware

### Módulo bluetooth HC-06

Este módulo bluetooth oferece uma forma simples e com custo reduzido para enviar e receber informações remotamente. Neste módulo existe um LED que indica se este se encontra emparelhado com algum dispositivo. Na figura 4.23 encontra-se o esquema de ligação deste módulo e na tabela 4.6 são apresentadas as suas principais características.



Figura 4.22: Módulo bluetooth HC-06

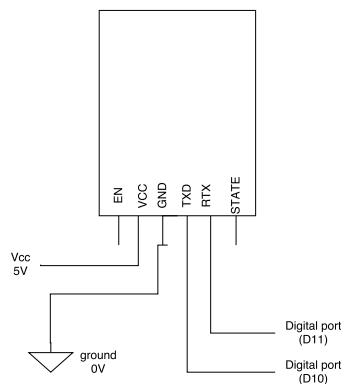


Figura 4.23: Esquema eletrotécnico da ligação do módulo bluetooth

|                             |  |
|-----------------------------|--|
| Versão bluetooth            | v2.0 com Enhanced Data Rate (EDR)                  |
| Frequência                  | 2,4GHz Banda Industrial, Scientific, Medical (ISM) |
| Segurança                   | Autentificação (PIN) e Encriptação                 |
| Tensão                      | Aconselhada 3,3v (2,7v - 4.2v)                     |
| Alcance                     | 10 metros  |
| Dimensões                   | 26,9 x 13 x 2,2mm                                  |
| Peso                        | 9,6g   |
| Temperatura (funcionamento) | -25C +75C  |
| Preço                       | 5.26 €/unidade                                     |

Tabela 4.6: Características do módulo bluetooth HC-06 [32]

#### 4.6.3 Sistema de deteção de intrusos

No contexto desta dissertação houve necessidade de implementar um sistema de video-stream que permitisse detetar intrusos, maioritariamente pessoas ou animais de grande porte, que possam invadir as quintas onde se produz Salicórnia. Esta necessidade prende-se essencialmente com elevado custo do *hardware* do sistema de monitorização e também de eventuais instrumentos de elevado custo necessários ao cultivo desta espécie, como por exemplo, geradores, máquinas elétricas para poda, entre outros.

Neste secção é descrita a tecnologia de processamento de imagem utilizada tal como o material necessário e respetiva arquitetura.

#### Biblioteca para processamento de imagem: OpenCV

O OpenCV, também conhecido por *Open Source Computer Vision Library*, é uma biblioteca de *software* de visão por computador de código *open-source* (figura 4.24).



Figura 4.24: Logótipo OpenCV

Esta biblioteca possui mais de 2500 algoritmos otimizados, que inclui um conjunto abrangente de algoritmos clássicos e avançados de visão computacional bem como algoritmos de *machine learning*. Esses algoritmos podem ser utilizados para os mais diversos fins, como por exemplo, para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, detetar movimentos numa câmara, seguir um objetos em movimento, produzir nuvens de pontos 3D de câmaras estéreo, entre outros. Esta biblioteca é amplamente utilizada em empresas e grupos de investigação, tendo interfaces nas mais diversas linguagens: C++, C,

Python, Java e MATLAB, embora seja nativamente escrita na linguagem C. OpenCV tem mais de 47 mil pessoas na sua comunidade e excede os 7 milhões de downloads, tendo suporte para Windows, Linux e Mac OS[33].

Desde logo, a escolha da tecnologia para processamento de imagem recaiu sobre o OpenCV não apenas por ser uma biblioteca bastante popular e possuir bastantes algoritmos implementados mas também por eu próprio possuir já algum *background* e projetos desenvolvidos nesta área. Pretendeu-se que este processamento fosse implementado em material já adquirido sem necessidade de gastos. Optou-se então por utilizar um *Raspberry Pi 3* que juntamente com um *Raspberry Pi camera module* (figura 4.25) permitirá a aquisição de imagem e servirá também como *Controller Module* ao sistema de aquisição de dados. Na tabela 4.7 apresentam-se algumas características deste módulo para o Raspberry Pi 3.

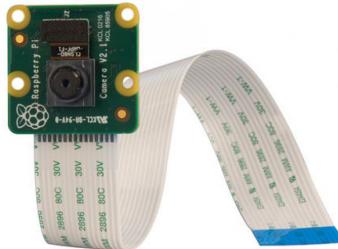


Figura 4.25: Raspberry Pi Camera Board V2 8MP 1080p

|                       |                                   |
|-----------------------|-----------------------------------|
| Sensor                | 8 megapixels Sony IMX219          |
| Resolução de exibição | 1080p, 720p e 640x480p para vídeo |
| Ligaçāo à placa       | Cabo fita curta (branca)          |
| Dimensões             | 25 x 23 x 9 mm                    |
| Peso                  | aproximadamente 3 g               |
| Compatibilidades      | última versão do Raspbian         |
| Preço                 | 26,67 €/unidade                   |

Tabela 4.7: Características do Raspberry Pi camera module

Inicialmente, pretendeu-se explorar um algoritmo disponibilizado pelo OpenCV para a deteção de pessoas (corpo inteiro). Este algoritmo, juntamente com técnicas de *machine learning* permitem a deteção de pessoas numa determinada imagem e/ou vídeo. O *machine learning* (em português aprendizagem automática) consiste num método de análise de dados que automatiza o desenvolvimento de modelos analíticos, sendo usados algoritmos que aprendem interativamente a partir de dados recolhidos à priori[34].

Após a realização de vários testes a este algoritmo, pretendeu-se implementar um servi-

dor de *streaming* que possa ser incorporado na *dashboard*. Para tal, optou-se por criar um aplicação em Flask (*framework* abordada no capítulo 3) que permita a aquisição de vídeo proveniente da *Raspberry Pi camera module* e realização do respetivo processamento. Esta aplicação web foi implementada num *Raspberry Pi 3* tendo sido necessário optar por um servidor web, neste caso o NGINX<sup>5</sup>. O NGINX é um servidor HTTP de alto desempenho e open-source. É conhecido pela sua alta performance, estabilidade, configuração simples e baixo consumo de recursos[35].

Para comunicação entre o servidor web com a aplicação em Flask foi necessária a utilização de uma Common Gateway Interface (CGI) que permita gerir páginas dinâmicas, possibilitando ao navegador passar parâmetros para uma aplicação existente num servidor web. Optou-se por utilizar o uWSGI<sup>6</sup>, sendo um dos mais populares. Na figura 4.26 encontra-se a arquitetura do sistema de deteção de intrusos e as respetivas tecnologias utilizadas.

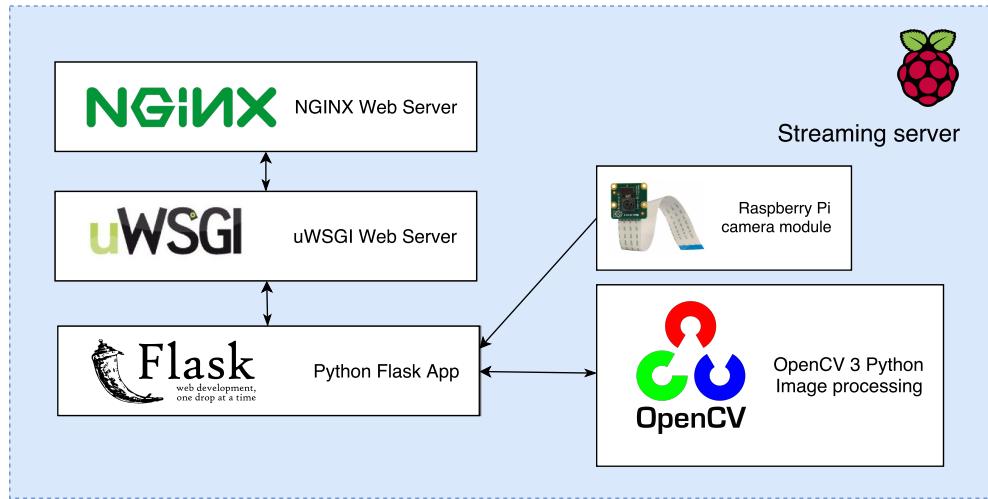


Figura 4.26: Arquitetura do sistema de video stream

<sup>5</sup><https://www.nginx.com/resources/wiki/>

<sup>6</sup><https://uwsgi-docs.readthedocs.io>

## 4.7 Diagrama de componentes

Para o cenário de simulação em *hardware* e respetivo sistema de informação, em que se inclui a *dashboard*, a API, aplicação mobile e sistema de deteção de intrusos, obtive o seguinte diagrama de componentes com as respetivas tecnologias utilizadas.

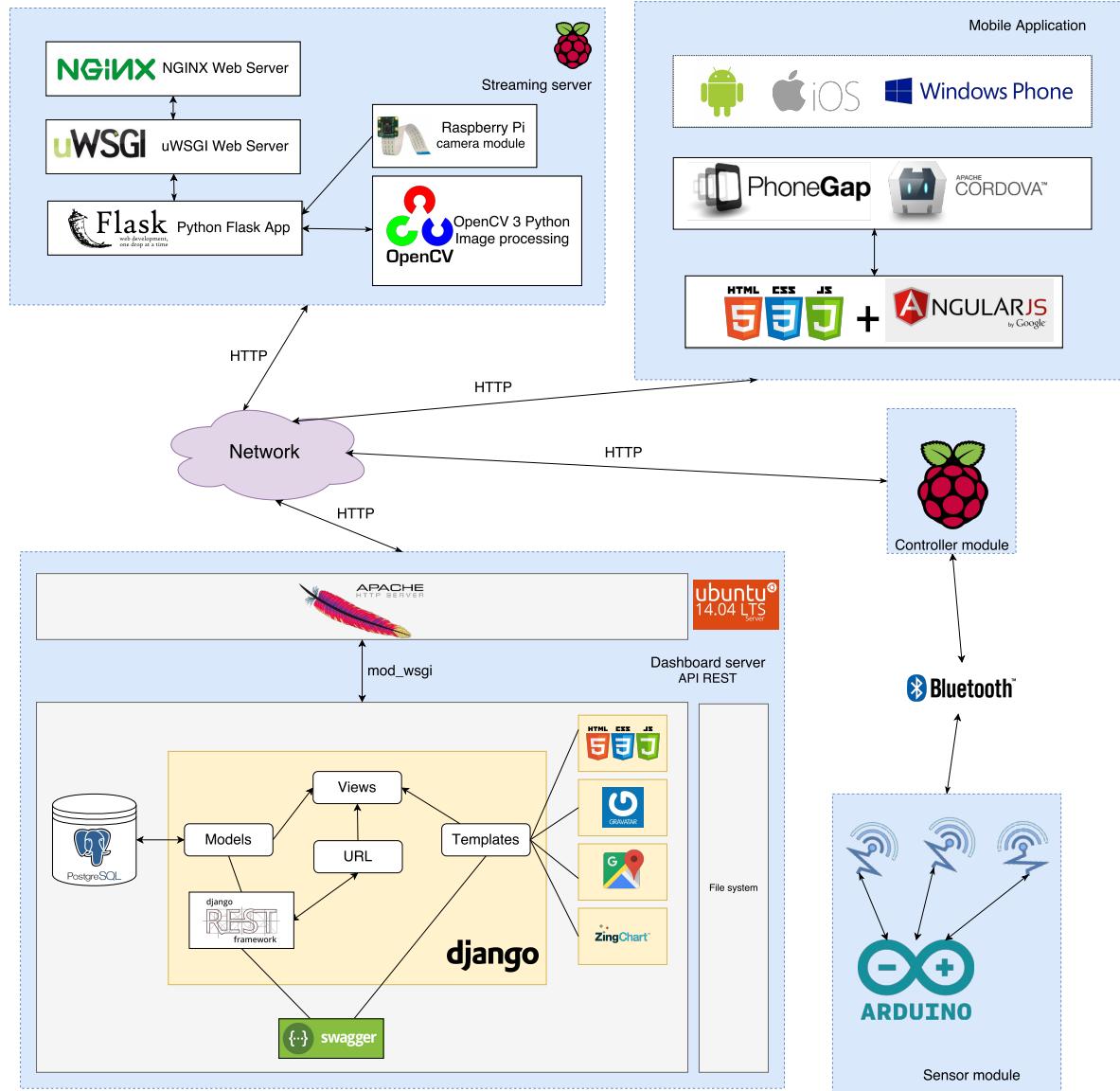


Figura 4.27: Diagrama final de componentes do sistema

## 4.8 Considerações finais

# Implementação

Neste capítulo, são explicados os detalhes da implementação deste projeto, incluindo especificamente o projeto Django que inclui o sistema de informação com a aplicação web (frontend/backend) e API REST e também a aplicação mobile em Phonegap. Para além disso, é descrita a implementação da simulação em *hardware* para o cenário descrito no capítulo anterior bem como o sistema de deteção de intrusos.

Segundo o ciclo de vida de um software, estabelecido por *Munish Kaur et al.*[3] que analisámos na secção X.X, este capítulo irá incluir, em parte, a fase de *designing* mas sobretudo a fase de *coding*.

## 5.1 Sistema de informação

Relativamente ao projeto Django, numa primeira fase procedeu-se à incorporação do SGBD PostgreSQL através do psycopg<sup>2</sup><sup>1</sup>. O psycopg2 consiste num adaptador entre o PostgreSQL com a linguagem de programação Python, que permite executar de forma eficiente qualquer *script* em SQL. É de notar que nesta fase inicial, encontram-se instalados algumas aplicações nativas do Django entre as quais, o `django.contrib.auth` e o `django.contrib.sessions`. Através delas é possível verificar o total funcionamento da base de dados uma vez que permitem criar automática das tabelas associadas ao utilizadores, grupos, permissões e respetivos conteúdos, administração, sessões entre outros. Estas tabelas são fundamentais ao bom funcionamento do sistema uma vez que são consideradas no modelo de dados descrito na secção XX.

Posteriormente, procedeu-se à criação dos diferentes `Models` conforme a nomenclatura

---

<sup>1</sup><http://initd.org/psycopg/docs/>

apresentada nas tabelas X e X da secção XX. O excerto de código seguinte pretende ilustrar a criação do Model associado à tabela que representa a estrutura *Sensor Module*, com o respetivo identificador e atributos. Após a criação de cada Models procedeu-se à migração dos dados para o SGBD onde era possível verificar que as tabelas tinham sido criadas, através da utilização da ferramenta gráfica pgAdmin III. De modo a testar a estrutura criada, procedeu-se à introdução de dados através da zona administrativa do Django (figura 5.1). Através dos dados introduzidos e descrevendo um cenário realista foi possível validar a estrutura e proceder à implementação da aplicação web e criação da API REST.

```

1 class SensorModule(models.Model):
2     id = models.AutoField(primary_key=True)
3     name = models.CharField(max_length=128)
4     seding_time = models.IntegerField()
5     status_sm = models.BooleanField(default=True)
6     baterry_sm = models.IntegerField()
7     localization_sm = models.CharField(max_length=128)
8
9     def __str__(self):
10        return "#" + str(self.id) + " " + self.name

```

The screenshot shows the Django Admin interface. At the top, there's a dark blue header bar with the text 'Administração do Django'. Below it, a light blue bar says 'Administração do site'. The main area has several sections:

- AUTENTICAÇÃO E AUTORIZAÇÃO**: Contains links for 'Grupos', 'Permissões', and 'Utilizadores', each with 'Adicionar' and 'Modificar' buttons.
- AUTH TOKEN**: Contains a 'Tokens' section with 'Adicionar' and 'Modificar' buttons.
- SALIAPP**: Contains sections for 'Alarms settingss', 'Alarmss', 'Cm per companies', and 'Communication type per sms', each with 'Adicionar' and 'Modificar' buttons.
- Recent actions**: A sidebar on the right showing a list of recent actions:
  - SM: 812 Type: 2humidity | Valor lido:8.0 as 2017-05-31 09:59:20+00:00 Reading
  - SM: 812 Type: 2humidity | Valor lido:8.0 as 2017-05-31 09:50:33+00:00 Reading
  - SM: 812 Type: 2humidity | Valor lido:10.0 as 2017-05-31 09:56:25+00:00 Reading
  - 1modul1 Sensor module
  - 3cam Sensor module

Figura 5.1: Painel administrativo do Django

### 5.1.1 Sistema de registo e autenticação

Primeiramente, procedeu-se à adaptação do template AdminLTE de modo a criar as páginas de autenticação e registo dos utilizadores. Como vimos anteriormente irão existir

utilizadores distintos, uma vez que, de modo a identificá-los e a definir a sua permissões houve necessidade de criar grupos específicas. Foram então criados dois grupos: *company*, que identifica uma empresa e *general*, que identifica um utilizador comum. O administrador do sistema é um utilizador que tem o estado de *superuser* ativo, isto é, possui todas as permissões sem que estas lhe sejam atribuídas explicitamente.

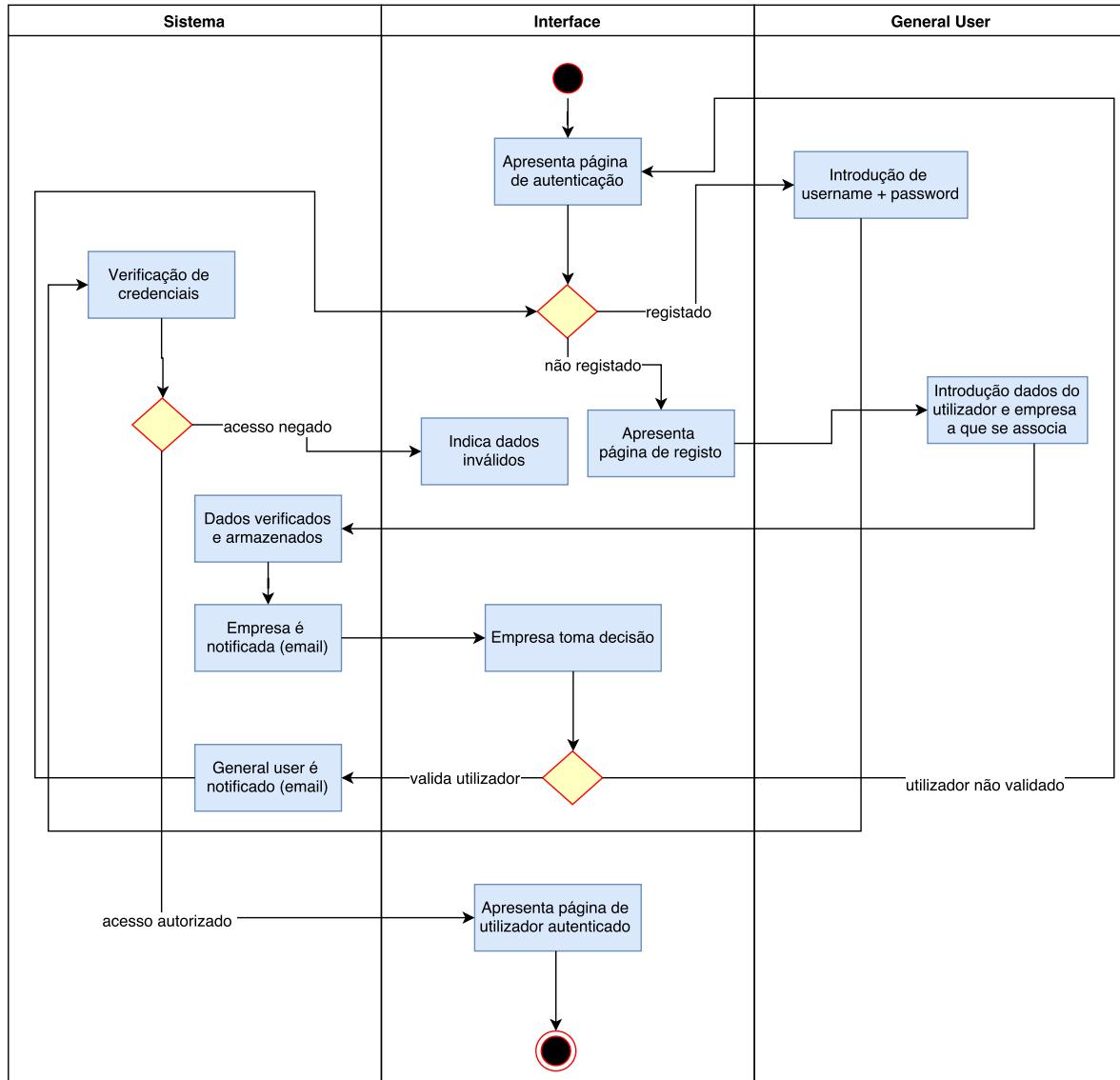


Figura 5.2: Diagrama de atividades do processo de registo e autenticação

Como vimos anteriormente, o *company user* apenas poderá ser adicionado ao sistema pelo administrador, tendo este também permissões de gerir todos os utilizadores registados. Por outro lado, os *company user* tem a possibilidade de validar os *general user* que se associam à sua empresa, tendo posteriormente acesso aos dados/conteúdos da empresa. Após o registo de

um novo *general user* e a validação por parte do *company user*, existem notificações que são enviadas via email. Estas mensagens são enviadas recorrendo ao método `send_mail`<sup>2</sup> existente nativamente no Django pelo pacote `django.core.mail`, sendo construído através do módulo `smtplib`<sup>3</sup>. Na figura 5.2 apresenta-se o diagrama de atividades que ilustra o processo de registo de um *general user* que envolve o utilizador responsável pela empresa (*company user*).

### 5.1.2 Geração de alarmes

No modelo de dados definido, cada sensor tem que ter obrigatoriamente associada uma entrada na tabela `AlarmsSettings` que permite definir o valor máximo e mínimo para o qual são gerados alarmes bem como as mensagens associadas que permitem informar o utilizador.

A geração de alarmes é condicionada pela comparação do valor lido pelos sensores com o valor máximo e mínimo definidos para o sensor em questão. Uma vez que é necessária esta verificação para cada leitura, decidiu-se criar um *trigger* em SQL que execute este procedimento, sendo este executado a nível da SGBD tornando o processo mais eficiente. Um *trigger* permite que uma determinada sequência de comandos sejam executadas sempre que um determinado evento ocorre, neste caso uma adição. No anexo X encontra-se um *stored procedure* (função) que implementa a sequência de comandos bem como a implementação do *trigger* na linguagem SQL. Na figura 5.3 encontra-se um diagrama de fluxo que permite ilustrar a lógica do *stored procedure*.

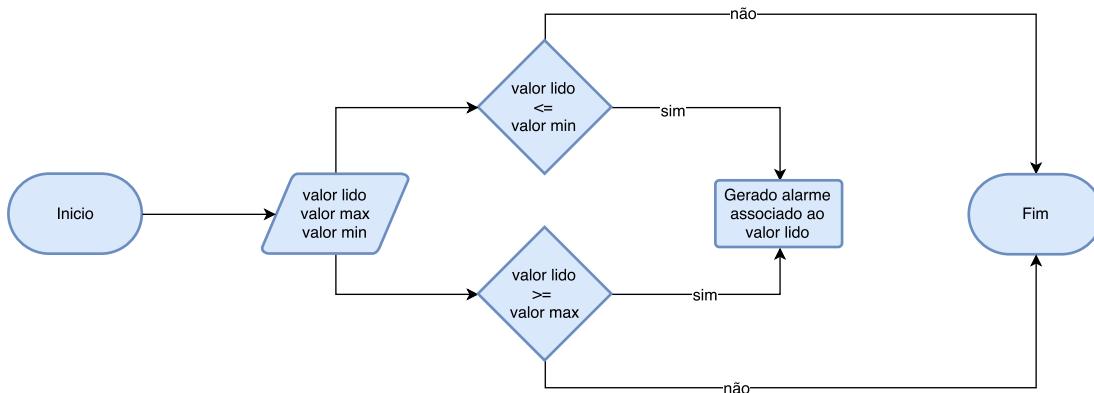


Figura 5.3: Diagrama de fluxo para geração de alarmes

Seguidamente são apresentadas algumas alternativas à implementação apresentada, pretendendo que estas sejam mais eficientes e igualmente competitivas às apresentadas pelas soluções apresentadas em XX.

<sup>2</sup><https://docs.djangoproject.com/en/1.11/topics/email/>

<sup>3</sup>Módulo que define objetos para sessões Simple Mail Transfer Protocol (SMTP) <https://docs.python.org/3/library/smtplib.html#module-smtplib>

- Análise do valor médio de cinco medições e respetivo desvio padrão
- Análise dos dados, também conhecida por data analist..... falta desenvolver data mining

O utilizador através da *dashboard* e da aplicação *mobile* poderá analisar os alarmes gerados bem como se o valor lido excede ou não atingiu o valor pretendido. Para além disso, é apresentada a mensagem pré-definida para que o utilizador possa tomar uma decisão.

### 5.1.3 Visualização dos dados e cálculos estatísticos

O cliente do sistema ao aceder à *dashboard* e pretender visualizar os detalhes de um determinado *Controller Module* ser-lhe-á apresentada uma lista de todos os *Sensor Modules* que este possui. Juntamente com esta lista são apresentadas as características principais de cada um, destacando os sensores existentes, o intervalo de tempo em que são enviadas dados, percentagem de bateria, coordenadas da localização do módulo e tipos de comunicação com o *Controller Module*. Complementarmente são apresentados quatro botões com icons sugestivos que permite realizar tarefas distintas.

- (editar): possibilita editar as características enumeradas anteriormente para cada *Sensor Module*.
- (remove): permite remover o *Sensor Module* do CM.
- (visualização gráfica): é possível visualizar graficamente os dados lidos pelos diferentes sensores existentes no *Sensor Module*.
- (visualização tabular): facilita em vista tabular uma visualização dos dados lidos pelos sensores bem como a respetiva exportação em ficheiro CSV.

Em ambas as visualizações anteriormente descritas é possível obter uma filtragem por data para os dados obtidos dos sensores de cada *Sensor Module*. Esta filtragem pode ser efetuada através de sete formas diferentes: do dia, do dia anterior, dos últimos sete dias, dos últimos trinta dias, deste mês, do ultimo mês e selecionando a data de inicio e de fim especificamente.

No que toca à visualização gráfica toda a representação foi concebida recorrendo à biblioteca ZingChart em JS, sendo que toda a manipulação de dados é realizada a nível de *backend* e posteriormente apresentada através dos *templates* do Django. A análise estatística para o intervalo de data considerado permitirá ao cliente obter o valor máximo lido, o valor mínimo lido bem como o valor médio, sendo também todo o processamento realizado a nível

de *backend*. Para além disso, esta página também possibilitará a ativação/desativação remota dos atuadores existentes no terreno.

Por fim, relativamente à visualização tabular os dados são apresentados em três campos distintas que representam o tipo de sensor, data e hora da leitura, valor lido e a escala. A tabela contém paginação sendo que é possível escolher o número de entradas que serão apresentadas (10, 25, 50 ou 100). Complementarmente, existe uma caixa de texto que permite ao cliente pesquisar pelos campos apresentados, para além disso é possível ordenar alfabeticamente ou numericamente esses. O cliente também poderá exportar os dados apresentados num ficheiro do tipo CSV com a estrutura apresentada na tabela 5.1. Para a implementação desta funcionalidade foram utilizados os métodos `writer` e `writerow()` disponíveis no pacote `csv` do Python.

| ID  | Sensor type | Scale | Date time                        | Value |
|-----|-------------|-------|----------------------------------|-------|
| 579 | Water valve | 0/1   | 2017-05-31 09:30:14.762099+00:00 | 1     |
| 580 | Temperature | C     | 2017-05-31 09:33:15.451236+00:00 | 25    |
| 581 | Water level | 0/1   | 2017-05-31 09:33:15.505198+00:00 | 1     |

Tabela 5.1: Estrutura do ficheiro do tipo CSV possível de exportação

#### 5.1.4 API

Tal como referido no capítulo anterior, para a implementação desta API do tipo REST foi utilizado o *Django REST framework*. Para tal, instalou-se a aplicação '`rest_framework`' no ficheiro de configuração do projeto Django, em `settings.py`. Para iniciar esta implementação procedeu-se à realização do `quickstart` disponível no site oficial da framework.

Primeiramente, procedeu-se à criação de um novo módulo `serializers.py` onde foram instanciadas todas as classes serializáveis<sup>4</sup> para cada `Model` existente bem como para os campos a considerar que possam ser usados na representação dos dados em formato JavaScript Object Notation (JSON). De seguida, desenvolveu-se o módulo `apiViews.py` onde são implementadas todas as classes baseadas na `APIView`. A classe `APIView` é uma subclasse da `View` (abordada no capítulo 3) tendo algumas diferenças:

- Na classe `View` são retornados objetos do tipo `HttpRequest` ou `HttpResponse`, enquanto que na `APIView` são objeto `Request` e `Response`.
- A classe `APIView` permite a implementação dos seguintes métodos HTTP: `get()`, `post()`, `put()` e `delete()`,

<sup>4</sup>Consiste num processo de tradução de uma estruturas de dados ou de um objeto que pode ser armazenado (num arquivo ou buffer de memória) e reconstruído posteriormente

- Através da classe `APIView` podem ser implementadas várias políticas da API

A título de exemplo, o excerto de código seguinte encontra-se a implementação do *endpoint* `api/cm/{pk_or_name}` com o método GET

```

1 #in serializers.py
2 class ControllerModuleSerializer(serializers.HyperlinkedModelSerializer):
3     id_communication = CommunicationTypeSerializer()
4     id_by_create = UserSerializer()
5
6     class Meta:
7         model = ControllerModule
8         fields = ('id', 'name', 'id_communication', 'id_by_create', 'battery_cm',
9                   'status_cm', 'date_create', 'memory', 'localization_cm')
10
11 #in apiViews.py
12 class ControllerModule_param(APIView):
13     def get_object(self, pk_or_name):
14         if pk_or_name.isdigit():
15             try:
16                 return ControllerModule.objects.get(pk=pk_or_name)
17             except ControllerModule.DoesNotExist:
18                 raise Http404
19         else:
20             try:
21                 return ControllerModule.objects.get(name__iexact=pk_or_name)
22             except ControllerModule.DoesNotExist:
23                 raise Http404
24
25     def get(self, request, pk_or_name, format=None):
26         cm = self.get_object(pk_or_name)
27         serializer = ControllerModuleSerializer(cm)
28         return Response(serializer.data)
29
30 #in url.py
31 url(r'^api/cm/(?P<pk_or_name>[-\w]+)/$', views.ControllerModule_param.as_view())

```

Como descrito anteriormente, a autenticação desta API usa um método de autenticação via token. Para tal, foi necessário a instalar a aplicação '`rest_framework.authtoken`' que fornece a estrutura `Token`, possibilitando a criação e modificação do token quando pretendido. Para que a autenticação seja possível é necessário incluir no módulo de configuração do Django o método pretendido, neste caso '`rest_framework.authentication.TokenAuthentication`' ■■■

No anexo A encontram-se as especificações de cada *endpoint* existente nesta API REST.

### 5.1.5 Documentação da API

Para a utilização da documentação interativa da API REST foi utilizada a ferramenta Swagger. Para a sua utilização é necessário incluir a aplicação '`rest_framework_swagger`' no ficheiro `settings.py` do Django e seguindamente definir o URL que lhe permitirá o acesso. Na figura 5.4 encontra-se um exemplo ao consultar a documentação desta API.

The screenshot shows the Swagger UI interface for a REST API. At the top, there's a green header bar with the Swagger logo and links for 'Django Login' and 'Authorize'. Below the header, the main content area is titled 'API documentation'. It lists several API endpoints grouped by module:

- alarms\_reading**
  - GET /api/alarms\_reading/{id\_reading}
  - POST /api/alarms\_reading/{id\_reading}
- alarms\_sensor**
  - GET /api/alarms\_sensor/{id\_sensor}
  - POST /api/alarms\_sensor/{id\_sensor}
- alarmssettings**
  - Show/Hide | List Operations | Expand Operations
- cm**
  - GET /api/cm/
  - POST /api/cm/
  - DELETE /api/cm/{pk\_or\_name}/
  - GET /api/cm/{pk\_or\_name}/
  - PUT /api/cm/{pk\_or\_name}/
- cmperusers**
  - Show/Hide | List Operations | Expand Operations
- communication**
  - Show/Hide | List Operations | Expand Operations

Figura 5.4: Documentação da API REST com a ferramenta Swagger

### 5.1.6 Aplicação web

O utilizador da dashboard após autenticar-se ser-lhe-á apresentado todos os *Controller Modules* a que tem acesso bem como as seguintes informações para cada um:

- Mapa onde é possível localizar todos os módulos associados a um *Controller Module*;
- Alarmes gerados pelos sensores que os *Sensor Modules* possuem;
- Últimos valores lidos pelo sensores dos *Sensor Modules*;

Cada um dos mapas interativo permitirá localizar o *Controller Module* bem como os *Sensor Modules* a si associados, sendo estes identificados por marcadores<sup>5</sup> de diferentes cores, no caso do *Controller Module*, a vermelho e nos *Sensor Modules*, a azul. A implementação

<sup>5</sup>Um marcador identifica uma localização no mapa.

<https://developers.google.com/maps/documentation/javascript/markers?hl=pt-br>

destes mapas recorre à API do Google Maps em JS, permitindo receber a localização dos módulos em coordenadas GPS. Ao clicar sob um determinado marcador este irá direcionar o utilizador para a página de detalhes do módulo em questão.

Quando ocorre um alarmes, o utilizador poderá verificar o motivo pelo qual este foi gerado, consultar a mensagem resultante bem como valor lido, tendo ainda a possibilidade de verificar/validar o alarme para que este não seja mais apresentado em destaque. Para além disso, o utilizador ao aceder à dashboard ser-lhe-ão apresentados os últimos valores lidos por cada sensor distribuídos por cada *Sensor Module*. Adicionalmente, na página principal da plataforma são apresentados alguns valores estatísticos da mesma: numero de utilizadores registados, numero de *Sensor Modules*, numero de *Controller Modules* e numero de leituras que já foram recolhidas pelos diferentes sensores.

A interface gráfica da plataforma permite ainda que o utilizador comum possa:

- Aceder à página de perfil que permite alterar as suas informações básicas e a password mas também consultar o token de autenticação na API.
- Consultar, adicionar ou editar os detalhes dos *Controller Modules* e dos *Sensor Modules*
- Adicionar e respetiva consulta dos tipos de sensores e comunicação existentes
- Alterar o avatar existente através da ferramenta *Gravatar* disponível em <http://pt.gravatar.com/>

gravatar

Sempre que o utilizador executa uma determinada ação na plataforma este é notificado informando-o se a ação foi bem sucedida ou não. A implementação deste mecanismo recorre à estrutura de mensagens<sup>6</sup> do Django, também conhecida por mensagens de flash.

### 5.1.7 Deploy do projeto

Para implementar o projeto Django juntamente com um servidor web Apache 2.0 num Virtual Private Server (VPS) Linux (Ubuntu 14.X), foram seguidos os seguintes passos:

1. Instalação do Python na versão 2.7 bem como pip<sup>7</sup> e respetivo *upgrade* para a versão mais recente

```
sudo apt-get install python-pip python-dev build-essential  
sudo pip install --upgrade pip
```

<sup>6</sup><https://docs.djangoproject.com/en/1.11/ref/contrib/messages/>

<sup>7</sup>Sistema de gestão de pacotes usado para instalar e gerir pacotes de software escritos na linguagem Python.

2. Instalação do PostgreSQL na ultima versão e criação de uma base de dados com o nome salibd

```
sudo apt-get install postgresql postgresql-contrib
createdb salibd
```

3. Instalação do apache2 e do mod\_wsgi que fará a ligação do projeto Python com o servidor apache. Seguidamente dar reset ao apache2.

```
sudo apt-get install apache2
sudo apt-get install libapache2-mod-wsgi
```

4. Configuração do virtualenv<sup>8</sup>, que consiste numa ferramenta para criar ambientes Python isolados. Por outro lado o virtualenvwrapper<sup>9</sup> permite uma fácil gestão do virtualenv, sendo que ao instalá-lo através do pip o virtualenv será instalado automaticamente.

```
pip install virtualenvwrapper
```

5. Criar um ambiente virtual (virtualenv) para o projeto. Para sua criação foi utilizada a opção `--system-site-packages` uma vez que a nossa aplicação terá que aceder a recursos fora do ambiente virtual, neste caso o PostgreSQL.

```
mkvirtualenv exampleenv --system-site-packages
```

6. Adicionar projeto e instalar dependências. Primeiramente é necessário ativar o virtualenv. Seguidamente procedeu-se à instalação do Django e instalação dos requisitos da aplicação (através do ficheiro requirements.txt disponibilizado pelo IDE).

```
workon exampleenv
pip install Django
cd /var/www
git clone https://github.com/ruipoliveira/ThesisSalicornia-web.git
pip install -r requirements.txt
```

7. Criação do virtual host. Terá que estar no directório `/etc/apache2/sites-available/` e criar um ficheiro `.conf` com o seguinte conteúdo.

```
1 <VirtualHost *:80>
2 ServerAdmin webmaster@mydomain.com
3 ServerName 192.168.160.20
```

<sup>8</sup><https://virtualenv.pypa.io/en/stable/>

<sup>9</sup><https://virtualenvwrapper.readthedocs.io/en/latest/>

```

4  ServerAlias http://192.168.160.20
5  WSGIScriptAlias / /var/www/example.wsgi
6
7  Alias /static/ /var/www/ThesisSalicornia-web/saliDashboard/saliapp/
8      static/
9      <Location "/static/">
10     Options -Indexes
11     </Location>
12
13 <Directory /var/www/ThesisSalicornia-web/saliDashboard>
14 Order deny,allow
15 Allow from all
16 </Directory>
17 </VirtualHost>

```

Por fim, é necessário ativar esta configuração através do comando  
**a2ensite example.conf**

#### 8. Criação do ficheiro wsgi no directório `/var/www/`

```

1  #file name 'example.wsgi'
2  import os
3  import sys
4  import site
5  # Add the site-packages of the chosen virtualenv to work with
6  site.addsitedir('/var/www/.virtualenvs/exampleenv/local/lib/python2.7/
7      site-packages')
8  # Add the app's directory to the PYTHONPATH
9  sys.path.append('/var/www/ThesisSalicornia-web/saliDashboard')
10 sys.path.append('/var/www/ThesisSalicornia-web/saliDashboard/
11     saliDashboard')
12 os.environ['DJANGO_SETTINGS_MODULE'] = 'saliDashboard.settings'
13 # Activate your virtual env
14 activate_env=os.path.expanduser("/var/www/.virtualenvs/exampleenv/bin/
15     activate_this.py")
16 execfile(activate_env, dict(__file__=activate_env))
17 from django.core.wsgi import get_wsgi_application
18 application = get_wsgi_application()

```

#### 9. Sincronizar da base de dados através do comando

`python manage.py migrate`

##### 5.1.8 Aplicação mobile

## 5.2 Simulação em *hardware*

Nesta secção pretende-se explicar a implementação a nível de *software* no contexto desta simulação para cada um dos micro-controladores.

### 5.2.1 Arduino

No que diz respeito ao Arduino Nano (*Sensor Module*), numa fase inicial, procedeu-se à ligação dos diversos componentes apresentados no capítulo anterior a uma *breadboard* tal como se encontra apresentado no Anexo G. Para auxiliar o desenvolvimento de *software* foi utilizada a versão 1.8.1 do IDE do próprio Arduino<sup>10</sup>. Seguidamente apresenta-se a implementação necessária a nível de sensores e de comunicação.

### 5.2.2 Sensores

Foram desenvolvidos os seguintes métodos que permitem aceder aos valores lidos de cada um dos sensores. Para além disso, foi criado um método que permite alterar o estado da válvula para transferência de água.

- `int readTemperature(int port)`: é efetuada uma leitura no porto analógico e seguidamente realizada uma conversão para para °C (graus Celsius)
- `long readLuminosity(int port)`:
- `int readWaterValve(int port)`: é efetuada uma leitura no porto digital através do método `digitalRead`.
- `int readWaterLevel(int port)`: é efetuada uma leitura no porto digital através do método `digitalRead`.
- `void setWaterValve(int port, int state)`: se a variável `state` for 1 então o porto é colocado a HIGH (1) através do método `digitalWrite`, caso contrário é colocado a LOW (0)

Inicialmente procedeu-se à leitura de cada sensor de forma individual de modo a garantir o seu total funcionamento. Sempre que é feita um pedido de leitura dos sensores pelo CM os valores são enviados com o seguinte formato:

`<temperatura>;<nível_água>;<luminosidade>;<estado_válvula>` (5.1)

<sup>10</sup><https://www.arduino.cc/en/Main/Software>

### 5.2.3 Comunicação

Numa primeira fase procedeu-se à comunicação entre o SM e CM através de porta série. Seguidamente resolveu-se incorporar o módulo bluetooth de modo a tornar cada módulo independente. De módulo de interagir com o módulo bluetooth utilizou-se o package `SoftwareSerial.h` disponível no Arduino. Decidiu-se que caso o módulo bluetooth recebesse valores de 0 a 2 tinha diferentes comportamentos:

- **0:** ativação (ligar) da válvula;
- **1:** desativação (desligar) da válvula;
- **2:** recebe dados obtidos pelos sensores no formato definido em (5.1)

Antes de proceder à implementação de envio e receção de dados por bluetooth no Raspberry Pi 3 optou-se por testar este mecanismo através de uma aplicação Android existente na *Play Store* chamada de *Bluetooth Terminal HC-05*<sup>11</sup>

#### Raspberry Pi

##### Comunicação

Como é possível observar na figura 4.21, para a comunicação no Raspberry Pi (CM) entre o Arduino (SM) foi utilizado o modulo interno de bluetooth 4.1 que este incorpora no seu hardware. Para tal, foi desenvolvido um *script* em Python que permite o seguinte:

1. Verificar dispositivos bluetooth disponíveis
2. Estabelecer conexão com módulo HC-06 através de um socket para comunicação utilizando para isso o package `socket` do python.
3. Aceder à API para verificar estado da válvula de admissão de águas e enviá-lo através do socket utilizando o método `send()`
4. No caso se ser enviado o dígito 1 a válvula será aberta, enquanto que se for enviado o dígito 2 a válvula é fechada.
5. No caso de ser enviado o dígito 2, o socket ficará à espera de receber os dados lidos pelos sensores, utilizando para isso o método `recv()`
6. Após receber os dados lidos, é efetuado algum processamento para que os dados sejam enviados através da API.

---

<sup>11</sup><https://play.google.com/store/apps/details?id=project.bluetoothterminal>

7. Todos os pontos 3 a 6 são repetidos com um atraso igual ao seding time definido o sensor module na dashboard.

Para permitir o acesso aos recursos do sistema Bluetooth foi utilizada uma extensão (*package*) do Python denominado de *pybluez*<sup>12</sup>.

#### 5.2.4 Considerações finais

1 fase testar conexão arduino to rasp via porta serie; foi criado um script em python para processar info e enviar para o servidor através da API

2 fase : necessidade de tornar um módulo isolado sem necessidade de fio; foi testado um modulo wifi e bluetooth;

neste contexto modulo wifi não!... pretende-se que os sensor moduels sejam de baixo custo e low power. foi utilizado um modulo bluetooth; foi testada a conexão da comm bluetooth através de uma client disponivel na google play bluetooth terminal HC-05

porque não foi usado um sensor de salinidade? não havia orçamento..

---

<sup>12</sup><https://github.com/karulis/pybluez>

## 5.3 Sistema de deteção de intrusos

No que toca ao desenvolvimento do sistema de deteção de intrusos, optou-se por utilizar o package picamera. Este pacote fornece uma interface em Python (disponível para qualquer versão) para o módulo de câmara Raspberry Pi<sup>13</sup>, permitindo uma fácil interação entre a aquisição da imagem e respetivo processamento. Neste contexto optou-se obviamente por utilizar a interface Python da biblioteca do OpenCV.

### 5.3.1 Algoritmos de deteção de intrusos

De modo a estudar alguns algoritmos de deteção de pessoas foram estudados alguns artigos neste contexto.

Para a resolução deste problema foi efetuados

HOGDescriptor: classe que implementa um histograma de gradientes orientado ( [Dalal2005] ) detetor de objetos.

hog = cv2.HOGDescriptor() hog.setSVMClassifier(cv2.HOGDescriptor.getDefaultPeopleDetector())

Usado biblioteca do opencv que permite detectar HOGDescriptor

Deteção de intrusos:

<http://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/>

versão simplificada: <http://www.pyimagesearch.com/2015/02/16/faster-non-maximum-suppression-python/>

Servidor em flask

deploy <https://iotbytes.wordpress.com/python-flask-web-application-on-raspberry-pi-with-nginx-and-uwsgi/>

Dataset:

é usado um detector HOG juntamente com um classificador linear SVM  
parametros do método detectMultiScale do opencv

- **img**: parâmetro obrigatório.
- **hitThreshold**: parâmetro opcional.
- **winStride**: parâmetro opcional.
- **padding**: parâmetro opcional. Os valores típicos para preenchimento incluem (8, 8) , (16, 16) , (24, 24) , e (32, 32) .
- **scale**: parâmetro opcional.

<sup>13</sup><http://picamera.readthedocs.io/en/release-1.13/>

- `finalThreshold`: parâmetro opcional.
- `useMeanShiftGrouping`: parâmetro opcional.

Neste contexto apenas foram utilizados os seguintes parâmetros `winStride`, `scale`, `padding`.

### 5.3.2 Testes

Foram considerados 4 frames de imagens .... e no apêndice X

### 5.3.3 Implementação

Flask é considerada uma microframework web desenvolvida em Python e baseado nas bibliotecas WSGI Werkzeug e Jinja2. Escolhi esta microframework pois pretende-se que esta seja executada num microcontrolador com baixos recursos. Para além disso, considera-se ser de fácil aprendizagem relativamente ao Django (já abordado na capítulo XX) e com uma ótima documentação.

## 5.4 Considerações finais

# 6

# Resultados

## 6.1 Interface web

A Web-based User Interface was created, which fully demonstrates the MDM capabilities and system management. The interface displays at all moments the alias and address of the current logged user. Through the UI it is possible to:

## 6.2 Interface mobile

## 6.3 Testes com API REST

## 6.4 Simulação em hardware

## 6.5 Sistema de deteção de intrusos



# 7

## **Conclusão e trabalho futuro**

### **7.1 Conclusão**

### **7.2 Trabalho futuro**

utilizar um sensor de salinidade para o cenário e implementar geração de alarmes não ser com base se um determinado valor sai do range mas sim por existir

### **7.3 Considerações finais**



# Bibliografia

- [1] Ado Kukic, “Cookies vs Tokens: The Definitive Guide,” 2016. [Online]. Available: <https://auth0.com/blog/cookies-vs-tokens-definitive-guide> [Accessed: 2017-07-03]
- [2] V. Isca, A. Seca, D. Pinto, and A. Silva, *An overview of Salicornia genus: the phytochemical and pharmacological profile*, natural pr ed., V. Gupta, Ed. Daya Publishing House, New Delhi, 2014.
- [3] P. T. Hiep, H. Noi, V. Nam, N. H. Hoang, H. Noi, and V. Nam, “A Review of Open Source Software Development Life Cycle Models,” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 5, pp. 391–402, 2014. [Online]. Available: <http://dx.doi.org/10.14257/ijseia.2014.8.3.38>
- [4] João Silva, “Sal verde, National Geographic.” [Online]. Available: <https://nationalgeographic.sapo.pt/23-arquivo/as-nossas-historias/298-sal-verde> [Accessed: 2017-02-01]
- [5] S. Beer and O. Demina, “A new species of Salicornia (Chenopodiaceae) from European Russia,” pp. 253–257, 2005.
- [6] M. Ferri and N. Menezes, *Glossário Ilustrado de Botânica*, 1st ed., Livraria Nobel, Ed., Brasil, 1981.
- [7] M. H. A. Silva, “Aspectos morfológicos e ecofisiológicos de algumas halófitas do sapal da Ria de Aveiro,” Ph.D. dissertation, Universidade de Aveiro, 2000. [Online]. Available: <http://ria.ua.pt/handle/10773/925>
- [8] E. Figueroa, J. Jimenez-Nieva, J. Carranza, and C. Gonzalez Vilches, “Distribucion y Nutricion Mineral de Salicornia ramosissima J. Woods, Salicornia europaea L. y Salicornia dolichostachya Moss. en el estuario de los rios Odiel y Tinto (Huelva, SO España),” *Limnetica*, vol. 3, no. 2, pp. 307–310, 1987.

- [9] R. Pinto, “Expresso — A planta que é uma alternativa ao sal: antes era uma praga, agora é uma erva gourmet,” 2015. [Online]. Available: <http://bit.ly/1PR7KAG> [Accessed: 2017-02-01]
- [10] A. J. Davy, G. F. Bishop, and C. S. B. Costa, “*Salicornia L.* (*Salicornia pusilla* J. Woods, *S. ramosissima* J. Woods, *S. europaea* L., *S. obscura* P.W. Ball & Tutin, *S. nitens* P.W. Ball & Tutin, *S. fragilis* P.W. Ball & Tutin and *S. dolichostachya* Moss),” *Journal of Ecology*, vol. 89, no. 4, pp. 681–707, 2001.
- [11] H. Silva, G. Caldeira, and H. Freitas, “*Salicornia ramosissima* population dynamics and tolerance of salinity,” *Ecological Research*, vol. 22, no. 1, pp. 125–134, 2007.
- [12] A. Rubio-Casal, J. Castillo, C. Luque, and M. Figueroa, “Influence of salinity on germination and seeds viability of two primary colonizers of Mediterranean salt pans,” *Journal of Arid Environments*, vol. 53, no. 2, pp. 145–154, feb 2003. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0140196302910426>
- [13] M. Filomena, D. J. Raposo, R. Manuel, S. Costa, A. Maria, and M. Bernardo, “Controlled atmosphere storage for the preservation of *Salicornia ramosissima*,” no. October 2016, 2009.
- [14] Y. Ventura, W. A. Wuddineh, M. Myrzabayeva, Z. Alikulov, I. Khozin-Goldberg, M. Shpigel, T. M. Samocha, and M. Sagi, “Effect of seawater concentration on the productivity and nutritional value of annual *Salicornia* and perennial *Sarcocornia* halophytes as leafy vegetable crops,” *Scientia Horticulturae*, vol. 128, no. 3, pp. 189–196, apr 2011. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0304423811000537>
- [15] Q. Z. Wang, X. F. Liu, Y. Shan, F. Q. Guan, Y. Chen, X. Y. Wang, M. Wang, and X. Feng, “Two new nortriterpenoid saponins from *Salicornia bigelovii* Torr. and their cytotoxic activity,” *Fitoterapia*, vol. 83, no. 4, pp. 742–749, jun 2012. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/22414316> <http://linkinghub.elsevier.com/retrieve/pii/S0367326X12000640>
- [16] D. Evans, “A Internet das Coisas Como a próxima evolução da Internet está mudando tudo,” pp. 5–7, 2011.
- [17] B. Getting, “Basic Definitions: Web 1.0, Web 2.0, Web 3.0 — Practical Ecommerce.” [Online]. Available: <http://www.practicalecommerce.com/articles/464-Basic-Definitions-Web-1-0-Web-2-0-Web-3-0> [Accessed: 2017-02-20]

- [18] J. Lovato, “Google’s evolution in 10 years,” 2014. [Online]. Available: <http://www.mediavisioninteractive.com/blog/search-enginenews/looking-back-moving-forward-google-evolution/> [Accessed: 2017-02-20]
- [19] T. Our, “Resume : Context Aware Computing for The Internet of Things : A Survey Article 2013,” pp. 1–5, 2013.
- [20] J. Rowley, “The wisdom hierarchy: representations of the DIKW hierarchy,” *Journal of Information Science*, vol. 33, no. 2, pp. 163–180, 2007. [Online]. Available: <http://alturl.com/7qike>
- [21] The PostgreSQL Global Development Group, “PostgreSQL: About,” 2012. [Online]. Available: <https://www.postgresql.org/about/> [Accessed: 2017-05-29]
- [22] A. S. H. G. Abreu, “Sistema de Monitorização de Estufas Agrícolas,” Ph.D. dissertation, University of Aveiro, 2012. [Online]. Available: <http://ria.ua.pt/handle/10773/10269>
- [23] Laudon, C. Kenneth, Laudon, and P. Jane, *Management Information Systems New Approaches to Organization & Technology*. Prentice Hall, 1998.
- [24] E. Turban, *Information technology for management : improving quality and productivity*. Wiley, 1996. [Online]. Available: [https://books.google.pt/books?id=FqxzQgAACAAJ&redir\\_esc=y&hl=pt-PT](https://books.google.pt/books?id=FqxzQgAACAAJ&redir_esc=y&hl=pt-PT)
- [25] “Gravatar - Globally Recognized Avatars.” [Online]. Available: <http://pt.gravatar.com/site/implementhttps://secure.gravatar.com/> [Accessed: 2017-06-10]
- [26] A. Rodriguez, “RESTful Web services: The basics,” no. February, pp. 1–11, 2015. [Online]. Available: <https://pdfs.semanticscholar.org/ca66/561d3602f65aef1301145e4e2689681b1967.pdf> <http://www.ibm.com/developerworks/webservices/library/ws-restful/>
- [27] T. Christie, “Django REST framework TokenAuthentication,” 2016. [Online]. Available: <http://www.django-rest-framework.org> [Accessed: 2017-07-03]
- [28] SmartBear Software, “Swagger – The World’s Most Popular Framework for APIs.” 2017. [Online]. Available: <http://swagger.io/> [Accessed: 2017-06-07]
- [29] The Apache Software Foundation, “Foundation Project,” 2016. [Online]. Available: <https://www.apache.org/foundation/http://apache.org/foundation/> [Accessed: 2017-06-12]

- [30] “Datasheet, NTC Thermistor TTC05 Series, Disc Type for Temperature Sensing/Compensation.” [Online]. Available: <http://extra-parts.com/datasheets/TTC.pdf>
- [31] L. LIDA OPTICAL&ELECTRONIC CO., “Datasheet, CdS Photoconductive cells, GL5528,” p. 1. [Online]. Available: <https://pi.gate.ac.uk/pages/airpi-files/PD0001.pdf> [Accessed: 2017-05-24]
- [32] L. Guangzhou HC Information Technology Co ., “HC06 Datasheet,” no. 13, pp. 1–17, 2011.
- [33] Itseez, “About - OpenCV library.” [Online]. Available: <http://opencv.org/about.html> [Accessed: 2017-05-19]
- [34] H. A. Razavi and T. R. Kurfess, “Detection of Wheel and Workpiece Contact/Release in Reciprocating Surface Grinding,” *Journal of Manufacturing Science and Engineering*, vol. 125, no. 2, p. 394, 2003. [Online]. Available: <http://manufacturingscience.asmedigitalcollection.asme.org/article.aspx?articleid=1447160>
- [35] Nginx, “Welcome to NGINX Wiki! — NGINX,” 2017. [Online]. Available: <https://www.nginx.com/resources/wiki/> [Accessed: 2017-07-04]



A

# Application Programming Interface

## REST

- /api/user/
  - Métodos disponíveis:
  - Descrição:
- /api/user/
  - Métodos disponíveis:
  - Descrição:
- /api/user/{pk\_or\_username}/
  - Métodos disponíveis:
  - Descrição:
- /api/smpercm/
  - Métodos disponíveis:
  - Descrição:
- /api/smpercm/{pk\_or\_name\_cm}
  - Métodos disponíveis:
  - Descrição:

- /api/sm/
  - Métodos disponíveis:
  - Descrição:
- /api/sm/{pk\_or\_name}/
  - Métodos disponíveis:
  - Descrição:
- /api/sensortype/
  - Métodos disponíveis:
  - Descrição:
- /api/sensortype/{pk\_or\_name}
  - Métodos disponíveis:
  - Descrição:
- /api/sensorpersm/{id\_sm\_or\_name\_sm}
  - Métodos disponíveis:
  - Descrição:
- /api/sensor/
  - Métodos disponíveis:
  - Descrição:
- /api/sensor/{pk\_or\_sensor\_type}
  - Métodos disponíveis:
  - Descrição:
- /api/reading/id\_sensor/{date\_start}/{date\_end}
  - Métodos disponíveis:
  - Descrição:
- /api/communication/{pk\_or\_name}
  - Métodos disponíveis:

- Descrição:
  - /api/cm/
    - Métodos disponíveis:
    - Descrição:
  - /api/cm/{pk\_or\_name}
    - Métodos disponíveis:
    - Descrição:
  - /api/alarmssettings/{id\_sensor}
    - Métodos disponíveis:
    - Descrição:
  - /api/alarms\_sensor/{id\_sensor}
    - Métodos disponíveis:
    - Descrição:
  - /api/alarms.reading/{id\_reading}
    - Métodos disponíveis:
    - Descrição:



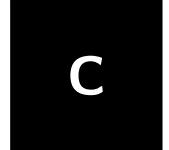
# B

## Mockups da aplicação mobile

Tabela B.1: Um nome qualquer

| Posição | País      | IDH  |
|---------|-----------|------|
| 1       | Noruega   | .955 |
| 2       | Austrália | .938 |
| 3       | EUA       | .937 |
| 4       | Holanda   | .921 |
| 5       | Alemanha  | .920 |



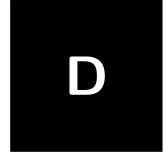


C

## Trigger SQL

```
1 CREATE OR REPLACE FUNCTION alarm_occurred() returns trigger as $alarm$  
2 DECLARE  
3 varmax FLOAT;  
4 varmin FLOAT;  
5 BEGIN  
6  
7 varmax := (select max from saliapp_alarmssettings where id_sensor_id= new.  
     id_sensor_id);  
8 varmin := (select min from saliapp_alarmssettings where id_sensor_id= new.  
     id_sensor_id);  
9  
10 IF (new.value >= varmax) THEN  
11     insert into saliapp_alarms (id_reading_id, checked, max_or_min) VALUES (new.id  
     , 'f' , 't');  
12 return new;  
13 END IF;  
14 IF (new.value <= varmin) THEN  
15     insert into saliapp_alarms (id_reading_id, checked, max_or_min) VALUES (new.id  
     , 'f' , 'f');  
16 return new;  
17 END IF;  
18  
19 RETURN NULL;  
20 END  
21 $alarm$  
22 LANGUAGE plpgsql;  
23  
24 create trigger trigger_alarm_occurred after insert on saliapp_reading
```

```
25 for each row execute procedure alarm_occurred();
26
27 DROP FUNCTION alarm_occurred();
28
29 DROP TRIGGER trigger_alarm_occurred ON saliapp_reading;
```



D

# Resultados processamento de imagem

Características do computador

- CPU: Intel Core i7-3630QM CPU @ 2.40GHz x 8
- SO version: Ubuntu 16.04.2 LTS
- Intel Corporation 3rd Gen Core processor Graphics Controller (rev 09) NVIDIA Corporation GF108M [GeForce GT 635M] (rev a1)

## D.1 Frame 1



Figura D.1: Pirâmide do conhecimento: modelo DIKW

### Características:

- Dimensões (px):
- Tamanho (MB):
- Número de pessoas existentes:

| <b>winStride</b> | <b>padding</b> | <b>scale</b> | <b>detection (number)</b> | <b>execution time (seg)</b> |
|------------------|----------------|--------------|---------------------------|-----------------------------|
| (2, 2)           | (8, 8)         | 0.5          | 4                         | 0.184819936752              |
| (4, 4)           | (8, 8)         | 0.5          | 3                         | 0.0488700866699             |
| (8, 8)           | (8, 8)         | 0.5          | 1                         | 0.0153889656067             |
| (2, 2)           | (8, 8)         | 1.0          | 4                         | 0.17699098587               |
| (4, 4)           | (8, 8)         | 1.0          | 3                         | 0.0484340190887             |
| (8, 8)           | (8, 8)         | 1.0          | 1                         | 0.0148591995239             |
| (2, 2)           | (8, 8)         | 1.5          | 6                         | 0.177606105804              |
| (4, 4)           | (8, 8)         | 1.5          | 5                         | 0.0484080314636             |
| (8, 8)           | (8, 8)         | 1.5          | 2                         | 0.0160319805145             |
| (2, 2)           | (16, 16)       | 0.5          | 4                         | 0.193215847015              |
| (4, 4)           | (16, 16)       | 0.5          | 3                         | 0.0518131256104             |
| (8, 8)           | (16, 16)       | 0.5          | 1                         | 0.0164451599121             |
| (2, 2)           | (16, 16)       | 1.0          | 4                         | 0.193369865417              |
| (4, 4)           | (16, 16)       | 1.0          | 3                         | 0.05233502388               |
| (8, 8)           | (16, 16)       | 1.0          | 1                         | 0.0161139965057             |

|        |          |     |   |                 |
|--------|----------|-----|---|-----------------|
| (2, 2) | (16, 16) | 1.5 | 6 | 0.193920850754  |
| (4, 4) | (16, 16) | 1.5 | 5 | 0.0550818443298 |
| (8, 8) | (16, 16) | 1.5 | 2 | 0.0162160396576 |
| (2, 2) | (24, 24) | 0.5 | 4 | 0.203732967377  |
| (4, 4) | (24, 24) | 0.5 | 3 | 0.0558068752289 |
| (8, 8) | (24, 24) | 0.5 | 1 | 0.0173289775848 |
| (2, 2) | (24, 24) | 1.0 | 4 | 0.203326940536  |
| (4, 4) | (24, 24) | 1.0 | 3 | 0.0569319725037 |
| (8, 8) | (24, 24) | 1.0 | 1 | 0.0179741382599 |
| (2, 2) | (24, 24) | 1.5 | 6 | 0.20330619812   |
| (4, 4) | (24, 24) | 1.5 | 5 | 0.0555651187897 |
| (8, 8) | (24, 24) | 1.5 | 2 | 0.0173530578613 |

Tabela D.1: Your caption here

## D.2 Frame 2



Figura D.2: Pirâmide do conhecimento: modelo DIKW

| <b>winStride</b> | <b>padding</b> | <b>scale</b> | <b>detection (number)</b> | <b>execution time (seg)</b> |
|------------------|----------------|--------------|---------------------------|-----------------------------|
| (2, 2)           | (8, 8)         | 0.5          | 11                        | 0.335342168808              |
| (4, 4)           | (8, 8)         | 0.5          | 4                         | 0.0799450874329             |
| (8, 8)           | (8, 8)         | 0.5          | 0                         | 0.0238499641418             |
| (2, 2)           | (8, 8)         | 1.0          | 11                        | 0.293792009354              |
| (4, 4)           | (8, 8)         | 1.0          | 4                         | 0.0808959007263             |
| (8, 8)           | (8, 8)         | 1.0          | 0                         | 0.024552822113              |
| (2, 2)           | (8, 8)         | 1.5          | 10                        | 0.310877084732              |
| (4, 4)           | (8, 8)         | 1.5          | 6                         | 0.0828230381012             |
| (8, 8)           | (8, 8)         | 1.5          | 1                         | 0.031553030014              |
| (2, 2)           | (16, 16)       | 0.5          | 11                        | 0.356366157532              |
| (4, 4)           | (16, 16)       | 0.5          | 5                         | 0.0858371257782             |
| (8, 8)           | (16, 16)       | 0.5          | 0                         | 0.0261859893799             |
| (2, 2)           | (16, 16)       | 1.0          | 11                        | 0.324184179306              |
| (4, 4)           | (16, 16)       | 1.0          | 5                         | 0.0870020389557             |
| (8, 8)           | (16, 16)       | 1.0          | 0                         | 0.0258660316467             |
| (2, 2)           | (16, 16)       | 1.5          | 10                        | 0.321846008301              |
| (4, 4)           | (16, 16)       | 1.5          | 7                         | 0.0916659832001             |
| (8, 8)           | (16, 16)       | 1.5          | 1                         | 0.0345950126648             |
| (2, 2)           | (24, 24)       | 0.5          | 11                        | 0.343872070312              |
| (4, 4)           | (24, 24)       | 0.5          | 5                         | 0.0918598175049             |
| (8, 8)           | (24, 24)       | 0.5          | 0                         | 0.0270938873291             |

|        |          |     |    |                 |
|--------|----------|-----|----|-----------------|
| (2, 2) | (24, 24) | 1.0 | 11 | 0.344779968262  |
| (4, 4) | (24, 24) | 1.0 | 5  | 0.090653181076  |
| (8, 8) | (24, 24) | 1.0 | 0  | 0.0263440608978 |
| (2, 2) | (24, 24) | 1.5 | 10 | 0.355221986771  |
| (4, 4) | (24, 24) | 1.5 | 7  | 0.0967049598694 |
| (8, 8) | (24, 24) | 1.5 | 1  | 0.0326068401337 |

Tabela D.2: Your caption here

### D.3 Frame 3



Figura D.3: Pirâmide do conhecimento: modelo DIKW

| <b>winStride</b> | <b>padding</b> | <b>scale</b> | <b>detection (number)</b> | <b>execution time (seg)</b> |
|------------------|----------------|--------------|---------------------------|-----------------------------|
| (2, 2)           | (8, 8)         | 0.5          | 8                         | 1.25844407082               |
| (4, 4)           | (8, 8)         | 0.5          | 5                         | 0.359390974045              |
| (8, 8)           | (8, 8)         | 0.5          | 1                         | 0.131782054901              |
| (2, 2)           | (8, 8)         | 1.0          | 8                         | 1.27126002312               |
| (4, 4)           | (8, 8)         | 1.0          | 5                         | 0.355902910233              |
| (8, 8)           | (8, 8)         | 1.0          | 1                         | 0.131030082703              |
| (2, 2)           | (8, 8)         | 1.5          | 16                        | 1.26964783669               |
| (4, 4)           | (8, 8)         | 1.5          | 12                        | 0.364797115326              |
| (8, 8)           | (8, 8)         | 1.5          | 1                         | 0.197186946869              |
| (2, 2)           | (16, 16)       | 0.5          | 8                         | 1.3578350544                |
| (4, 4)           | (16, 16)       | 0.5          | 5                         | 0.357763051987              |
| (8, 8)           | (16, 16)       | 0.5          | 1                         | 0.132702112198              |
| (2, 2)           | (16, 16)       | 1.0          | 8                         | 1.27961397171               |
| (4, 4)           | (16, 16)       | 1.0          | 5                         | 0.367429971695              |
| (8, 8)           | (16, 16)       | 1.0          | 1                         | 0.132242918015              |
| (2, 2)           | (16, 16)       | 1.5          | 17                        | 1.28247308731               |
| (4, 4)           | (16, 16)       | 1.5          | 12                        | 0.403631925583              |
| (8, 8)           | (16, 16)       | 1.5          | 1                         | 0.207641839981              |
| (2, 2)           | (24, 24)       | 0.5          | 8                         | 1.43096494675               |
| (4, 4)           | (24, 24)       | 0.5          | 5                         | 0.369131088257              |
| (8, 8)           | (24, 24)       | 0.5          | 1                         | 0.134386062622              |
| (2, 2)           | (24, 24)       | 1.0          | 8                         | 1.34318900108               |
| (4, 4)           | (24, 24)       | 1.0          | 5                         | 0.371593952179              |

|        |          |     |    |                |
|--------|----------|-----|----|----------------|
| (8, 8) | (24, 24) | 1.0 | 1  | 0.134378194809 |
| (2, 2) | (24, 24) | 1.5 | 17 | 1.39831089973  |
| (4, 4) | (24, 24) | 1.5 | 13 | 0.444314002991 |
| (8, 8) | (24, 24) | 1.5 | 1  | 0.137616872787 |

Tabela D.3: Your caption here

## D.4 Frame 4



Figura D.4: Pirâmide do conhecimento: modelo DIKW

| <b>winStride</b> | <b>padding</b> | <b>scale</b> | <b>detection (number)</b> | <b>execution time (seg)</b> |
|------------------|----------------|--------------|---------------------------|-----------------------------|
| (2, 2)           | (8, 8)         | 0.5          | 7                         | 1.3150138855                |
| (4, 4)           | (8, 8)         | 0.5          | 2                         | 0.36035490036               |
| (8, 8)           | (8, 8)         | 0.5          | 0                         | 0.129312992096              |
| (2, 2)           | (8, 8)         | 1.0          | 7                         | 1.24681711197               |
| (4, 4)           | (8, 8)         | 1.0          | 2                         | 0.358268976212              |
| (8, 8)           | (8, 8)         | 1.0          | 0                         | 0.130249023438              |
| (2, 2)           | (8, 8)         | 1.5          | 17                        | 1.58746790886               |
| (4, 4)           | (8, 8)         | 1.5          | 12                        | 0.513493061066              |
| (8, 8)           | (8, 8)         | 1.5          | 1                         | 0.197572946548              |
| (2, 2)           | (16, 16)       | 0.5          | 7                         | 1.36501693726               |
| (4, 4)           | (16, 16)       | 0.5          | 2                         | 0.363034009933              |
| (8, 8)           | (16, 16)       | 0.5          | 0                         | 0.132270812988              |
| (2, 2)           | (16, 16)       | 1.0          | 7                         | 1.29145503044               |
| (4, 4)           | (16, 16)       | 1.0          | 2                         | 0.359399080276              |
| (8, 8)           | (16, 16)       | 1.0          | 0                         | 0.132076025009              |
| (2, 2)           | (16, 16)       | 1.5          | 19                        | 1.61724209785               |
| (4, 4)           | (16, 16)       | 1.5          | 13                        | 0.467741012573              |
| (8, 8)           | (16, 16)       | 1.5          | 1                         | 0.170053005219              |
| (2, 2)           | (24, 24)       | 0.5          | 7                         | 1.33659911156               |
| (4, 4)           | (24, 24)       | 0.5          | 2                         | 0.365787982941              |
| (8, 8)           | (24, 24)       | 0.5          | 0                         | 0.133852005005              |
| (2, 2)           | (24, 24)       | 1.0          | 7                         | 1.29908204079               |
| (4, 4)           | (24, 24)       | 1.0          | 2                         | 0.377649784088              |

|        |          |     |    |               |
|--------|----------|-----|----|---------------|
| (8, 8) | (24, 24) | 1.0 | 0  | 0.13329410553 |
| (2, 2) | (24, 24) | 1.5 | 19 | 1.32506895065 |
| (4, 4) | (24, 24) | 1.5 | 13 | 0.38186788559 |
| (8, 8) | (24, 24) | 1.5 | 1  | 0.1673848629  |

Tabela D.4: Your caption here



E

## Interface gráfica

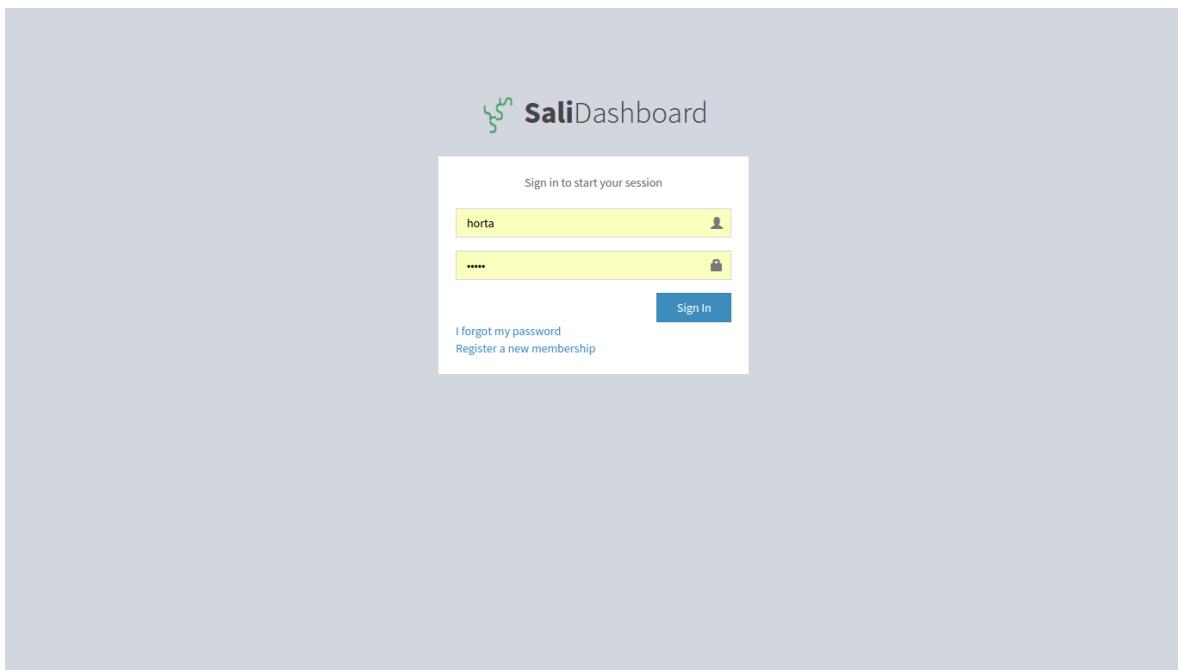


Figura E.1: Pirâmide do conhecimento: modelo DIKW

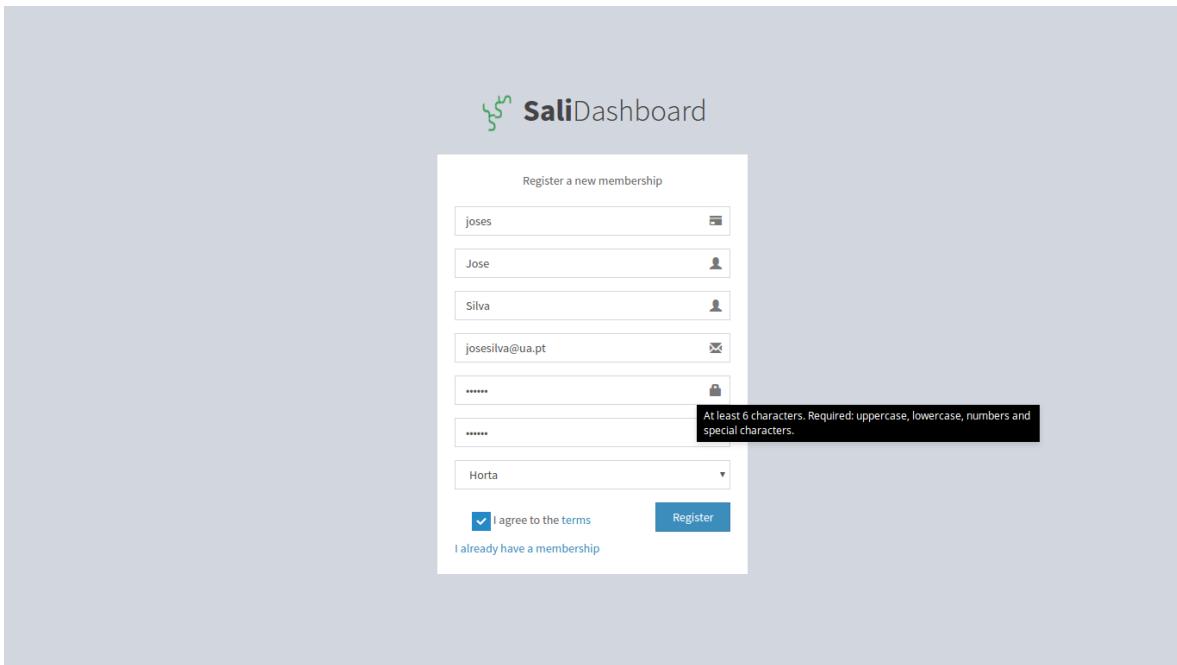


Figura E.2: Pirâmide do conhecimento: modelo DIKW

F

## **Descrição formal dos casos de uso gerais**



G

# Interligação de componentes

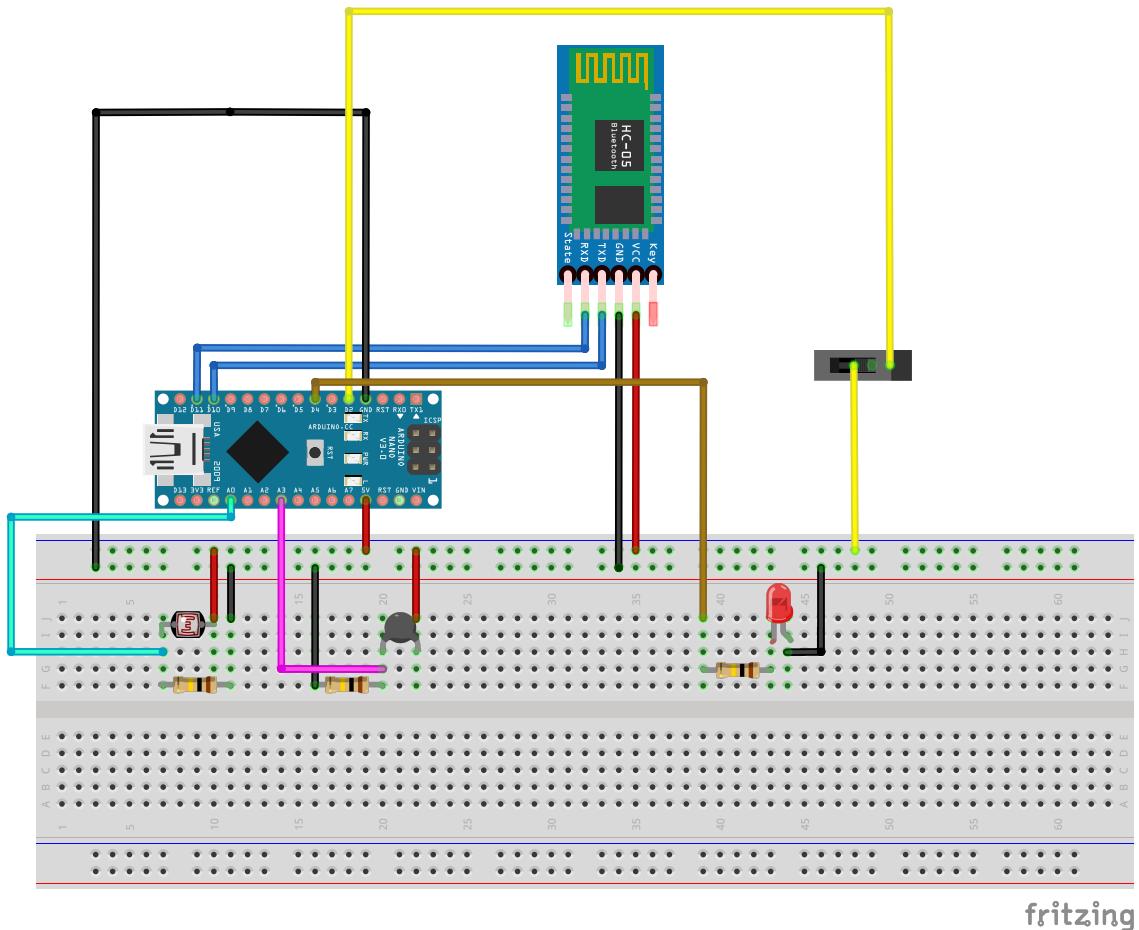


Figura G.1: Pirâmide do conhecimento: modelo DIKW

