UNIVERSIDADE DA BEIRA INTERIOR
Faculdade de Engenharia
Departamento de Informática

Sistema de controlo de uma estrutura híbrida e modular de revestimento exterior de edifícios

Luís Miguel Ferreira de Almeida

Submetido na Universidade da Beira Interior no âmbito da candidatura
ao grau de Mestre em Engenharia Informática

Orientador: Simão Melo de Sousa

Coorientador: João Castro Gomes

Departamento de Informática
Universidade da Beira Interior
Covilhã, Portugal
http://www.di.ubi.pt

Agradecimentos

Agradeço ao Prof. Simão Melo de Sousa a constante disponibilidade, os conselhos valiosos e orientação prestada ao longo da realização deste trabalho.

Ao Prof. João Paulo de Castro Gomes, do departamento de Engenharia Civil e Arquitetura da Universidade da Beira Interior, pela introdução ao sistema GEOGREEN, pelos conselhos e pela disponibilização do material necessário à concretização da prova de conceito realizada neste trabalho.

Ao Roberto Gama pela utilização do sistema GeogreenPlus no processo de validação experimental da sua tese de mestrado.

À Isabel, minha esposa, agradeço, entre muitos outros, a paciência e apoio durante este período.

Resumo

O sistema GEOGREEN define-se como sendo um sistema modular de revestimento exterior de edifícios, tanto paredes como telhados, que combina um material de construção obtido a partir de resíduos com material orgânico vivo. Esta conjugação permite a constituição de paredes/superfícies verdes horizontais e verticais que podem inclusivamente ser aproveitadas na forma de hortas urbanas. O sistema apresenta um grande potencial tendo em conta as suas características, como o reaproveitamento de resíduos minerais no seu fabrico, o facto de ser um sistema de módulos relativamente pequenos que permitem ser acoplados facilmente uns aos outros e ainda de se poderem criar fachadas verdes sem necessidade de acrescentar outros materiais entre a estrutura existente e os módulos GEOGREEN.

No entanto este potencial é meramente material, isto é, não há forma de monitorizar (medir e atuar, potenciar em conhecimento de causa) o seu desempenho. Esta tese propõe uma solução tecnológica para suprir essa necessidade: uma rede de sensores de captura de medidas de desempenho, mais um sistema de informação moderno, que permite a consulta dos dados recolhidos e tomada de decisões.

Esta solução tecnológica será capaz de capturar diversa informação do meio envolvente às placas modulares acrescentando assim potencial ao material existente resultando da simbiose entre tecnologia e placas GEOGREEN. A consequência de tal simbiose é ter um sistema de controlo eficiente e com o objetivo de ser autónomo (rega automática, cuidado da flora, etc…)

Palavras chave

GEOGREEN

Superfícies ajardinadas

Internet of Things

IoT

Arduino

Raspberry Pi

Sensores

Azure

Conteúdo

Lista de Imagens

Lista de Tabelas

Acrónimos

IoT: *Internet of Things*

API: *Application Programing Interface*

IDE: *Integrated Development Environment*

BD: Base de dados

WPF: *Windows Presentation Foundation*

MVVM: *Model-View-ViewModel*

GUI: *Graphical User Interface*

IIS: *Internet Information Services*

JSON : *JavaScript Object Notation*

SQL: *Structured Query Language*

PoC: *Proof of Concept* (Prova de Conceito)

1. Introdução

1.1. Enquadramento

Desde alguns anos para cá, a sociedade tem vindo a desenvolver uma importante consciência ecológica. Esta consciência visa, entre outros fatores, fazer com que os recursos do planeta sejam utilizados de forma consciente e reutilizá-los sempre que possível de forma a proteger ao máximo o meio ambiente.

A própria União Europeia, na sua estratégia de crescimento da economia denominada "Europa 2020", defende o crescimento sustentável como uma das suas prioridades [1].

No contexto do projeto de investigação GEOGREEN da unidade de investigação CMADE da Universidade da Beira Interior, foi criado um sistema modular de revestimento exterior de edifícios que combina um material de construção obtido de resíduos com material orgânico vivo (plantas, vegetação) [2]. A acoplagem de material de construção cuidadosamente escolhido (material de construção obtido a partir de resíduos com aglomerado negro de cortiça) com material vivo confere à estrutura propriedades extremamente interessantes e inovadoras no contexto da qualidade do ecossistema, sustentabilidade e eficiência energética dos edifícios e mais genericamente das cidades.

Uma estrutura desta natureza precisa de cuidados de manutenção diferentes dos que são necessários em materiais de construção clássicos. Outro aspeto relevante é que se espera conseguir estabelecer uma simbiose entre a vegetação usada e os edifícios, que permita melhorar o desempenho já medido em fases anteriores do projeto. Assim, na fase atual do projeto, entende-se que a tecnologia pode dar contributo a uma maior eficiência da estrutura ao permitir:

- Um maior controlo e potenciação das suas propriedades.
- Um melhor diagnóstico dos fenómenos/reações que ocorrem da relação entre o material vivo e o material estrutural.
- Uma ação focalizada à zona que necessita eventual intervenção.

1.2. Objetivos

O objetivo principal deste trabalho é o de tentar demonstrar que o sistema de placas modulares GEOGREEN podem beneficiar da introdução de um sistema constituído por uma rede de sensores, que recolhem dados do meio envolvente às placas

modulares, e por um *software* que, entre outros, guarde esses dados e permita retirar informação útil dos mesmos.

Essas melhorias encaixam-se em dois eixos:

- Recolher informação que possa permitir perceber como melhorar a própria constituição dos módulos GEOGREEN, ou demonstrar as mais-valias da estrutura de uns módulos em relação a outros construídos com materiais diferentes.
- Potenciar o sistema GEOGREEN atual dando-lhe capacidade de extrair informação sobre o seu meio envolvente, permitindo-lhe de forma autónoma atuar sobre a sua estrutura, quer no seu todo quer em zonas específicas.
  Uma consequência desse melhoramento será, por exemplo, permitir otimizar e poupar recursos aos utilizadores que tenham optado por instalar o sistema modular GEOGREEN nos seus edifícios.

Para concretizar o objetivo deste trabalho será necessário construir um sistema que seja capaz de ler e armazenar dados provenientes de diferentes tipos de sensores (temperatura, humidade...) e mostrar esses dados ao utilizador final para que ele possa ter uma ideia concisa do estado do sistema. O sistema deverá também ser capaz de tomar decisões automáticas, baseadas em regras definidas pelo utilizador, tendo em conta os dados obtidos do meio ambiente.

Será então desenvolvido/construído um protótipo de um sistema conjunto *hardware/software* que possa recolher dados do meio ambiente envolvente ao material de revestimento exterior dos edifícios e tratar esses dados para que o resultado final seja útil, podendo tomar-se decisões baseadas no mesmo.

Como resultado deste trabalho queremos ter uma validação experimental, um protótipo a funcionar, ao qual chamaremos GeogreenPlus, a partir do qual queremos ser capazes de retirar lições/conclusões.

1.3. Organização da dissertação

Esta dissertação está organizada em 7 capítulos distintos.

O capítulo 1 define o enquadramento deste trabalho, tenta clarificar os seus objetivos e apresenta a organização desta dissertação.

No capítulo 2 é feita una contextualização tecnológica dos assuntos ligados à temática deste trabalho.

O capítulo 3 apresenta a análise de requisitos efetuada para o desenvolvimento do protótipo. Nele são definidos os requisitos funcionais, não funcionais, casos de uso do sistema, diagramas de atividade e de sequência e ainda os requisitos mais específicos da interface com o utilizador.

No capítulo 4 está descrito todo o trabalho efetuado no desenvolvimento do protótipo pretendido e cujos requisitos estão definidos no capítulo 3. Começa-se por apresentar o material e tecnologias escolhidas, e as razões dessa escolha, para a realização do protótipo. Depois é apresentada a arquitetura geral do sistema a desenvolver e continua-se com uma descrição da configuração efetuada na rede de sensores e outros materiais relacionados. As seções finais do capítulo debruçam-se sobre o trabalho desenvolvido no desenvolvimento do *software*, começando pela Web API, seguindo com a aplicação de recolha de dados e terminando com a aplicação de interface com o utilizador.

O capítulo 5 descreve o sistema em funcionamento, começando por apresentar os passos para a instalação e configuração do protótipo e terminando com a descrição dos passos seguidos para validar que essa instalação foi efetuada com êxito e o protótipo está em condições de ser testado em condições reais.

Os casos de validação experimental são apresentados no capítulo 6, sendo o primeiro caso apresentado, a validação experimental das qualidades do material GEOGREEN modificado com materiais de mudança de fase. O segundo caso de teste mostra um caso em que a utilização conjunta das placas modulares GEOGREEN e do protótipo desenvolvido permite retirar vantagem dessa simbiose.

Por fim no capítulo 7 estão patentes as conclusões retiradas bem como ações possíveis num trabalho futuro.

2. Contextualização tecnológica

2.1. Introdução

Nesta contextualização tecnológica iremos focar-nos na importância de sistemas para a execução superfícies ajardinadas tais como o GEOGREEN e no estado atual das tecnologias (*hardware*/*software*) existentes que irão permitir desenvolver o protótipo a que se propõe este trabalho.

Assim a secção 2.2 irá descrever a importância das superfícies ajardinadas na atualidade, a secção 2.3 fará uma descrição genérica do conceito de *Internet of Things.* Na secção 2.4 abordar-se-á o conceito de rede de sensores e por fim, na secção 2.5 falar-se-á dos microcontroladores/placas de prototipagem existentes no mercado e que são mais relevantes para o protótipo que nos propomos construir.

2.2. Superfícies ajardinadas

A incorporação de superfícies ajardinadas nos edifícios é um tema que tem despertado, ao longo dos últimos anos, um interesse crescente chegando certos países, tais como a França, a começar a legislar no sentido de tornar a sua utilização obrigatória num futuro próximo [18]. Certos governos, como por exemplo o da República Checa, disponibilizam subsídios para a construções de edifícios com este tipo de superfícies [20].

Em 1997 foi criada a *European Federation of Green Roof and Wall Associations* (EFB)[1] pela Áustria, Alemanha e Suíça com o intuito de promover a utilização de telhados e fachadas verdes nos países europeus. Atualmente existem 16 associações membro da EFB incluindo Portugal[2].

Estas superfícies são importantes na medida de que, além de estenderem a superfície de espaços verdes das cidades, permitem melhorar a eficiência energética dos edifícios a que estão acopladas e ainda proteger a sua estrutura das condições atmosféricas adversas a que estão sujeitos. Numa visão mais ampla permitem tornar as cidades mais

[1] http://efb-greenroof.eu

[2] http://efb-greenroof.eu/cubeportfolio/associacao-nacional-de-coberturas-verde/

sustentáveis, mais saudáveis, aumentar a sua biodiversidade e têm ainda uma componente estética importante [39].

Figura 1 - Telhado verde intensivo - Brno, República Checa[3]

Em várias cidades de diferentes países existem atualmente projetos de construção de larga escala, uns em fase de estudo e outros já em fase de construção, que fazem das superfícies ajardinadas o seu ex-libris. Damos como exemplo os projetos de Lausana na Suíça e de Luizhou na China.

O projeto suíço é o de um arranha-céus de aproximadamente 103 metros de altura que terá aspeto de floresta vertical uma vez que será composto de placas de betão intercaladas com cedros [21].

Figura 2 - Plano "floresta vertical" em Lausana

[3] Origem da figura [20]

Na China o projeto em perspetiva é a construção de uma cidade 100% sustentável com o objetivo de atenuar e tentar controlar o grave problema de poluição no país. Este projeto comtempla a construção de dezenas de edifícios todos eles compostos por superfícies ajardinadas, verticais e horizontais, no seu exterior [22].

Figura 3 - Imagem ilustrativa da futura "cidade floresta" em Liuzhou

Existem diversas metodologias/técnicas para a construção de superfícies ajardinadas, mas a maioria destas não têm a valência de poderem ser implementadas tanto em superfícies horizontais como verticais [38]. A solução GEOGREEN tem certas vantagens, em relação às técnicas existentes, sendo as mais relevantes as seguintes (ver [38]):

- Apresenta uma estrutura modular, de reduzidas dimensões, o que permite a sua aplicação em superfícies horizontais, verticais, inclinadas ou curvas.
- É uma solução válida quer para edifícios novos ou edifícios reabilitados.
- Não necessita de qualquer camada extra entre o betão (ou outro material de construção utilizado nas superfícies dos edifícios) e as placas GEOGREEN reduzindo assim significativamente o peso sobre a estrutura do edifício.
- As peças modulares são constituídas por camadas de diferentes materiais, combinando características como a baixa densidade, porosidade, retenção de água, isolamento térmico, resistência, durabilidade e resistência ao fogo.
- Utiliza materiais reciclados provenientes de desperdícios da atividade de extração mineral minimizando o impacte ambiental.

Na Figura 4 pode observar-se o aspeto visual de um módulo GEOGREEN, as suas dimensões e um esquema das suas possíveis aplicações.

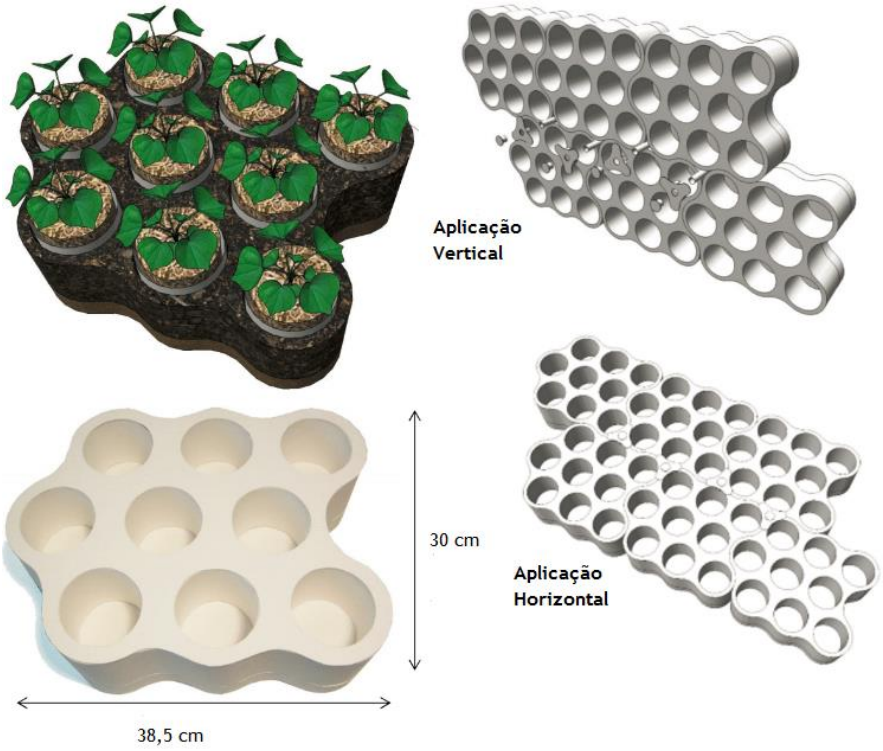

Figura 4 - Módulos GEOGREEN (retirada de [32])

2.3. Internet of Things – *IoT*

A cada ano que passa o termo *Internet of Things* tem-se ouvido com maior frequência sendo, no entanto, um conceito que não é propriamente novo. A nível empresarial já se utiliza este tipo de soluções há vários anos (M2M, SCADA, Telemetria...) [3], apenas se tem tornado mais popular nos últimos tempos devido a sua utilização por parte de utilizadores individuais.

Segundo a União Internacional de Telecomunicações (ITU – *International Telecommunication Union*) a IoT pode definir-se genericamente por "*A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies*" 85[4].

Mais genericamente pode dizer-se que a IoT é composta por dispositivos conectados entre si e que partilham/trocam informação [5]. Esta definição genérica remete-nos para uma grande amplitude de dispositivos interligados do dia-a-dia, sendo que na base do processo existe sempre algum tipo de aquisição de dados (muitas vezes retirados

de medições efetuadas com sensores), depois existe armazenamento desses dados, podendo existir algum pós-tratamento ou não, e por fim os dados são mostrados ao utilizador, ou servidos como ponto de entrada a algum tipo de processo, da forma mais útil possível.

2.4. Redes de sensores

Uma rede de sensores é composta, como o nome indica, por vários sensores que podem ser todos do mesmo tipo ou de vários tipos (temperatura, humidade, …) e que funcionam em paralelo permitindo recolher dados importantes sobre o funcionamento de uma determinada estrutura ou do seu meio envolvente [6]. Esta rede de sensores liga-se à internet, de modo a poder comunicar os valores lidos, para que estes sejam armazenados ou consumidos diretamente por aplicações terceiras.

Existem diversos contextos para a aplicação de redes de sensores, desde diferentes tipos de indústria como a aeronáutica, mecânica e petrolífera entre outras, passando por recolhas dados no contexto de cidades inteligentes até às casas de particulares que têm instalados sistemas de domótica, isto citando apenas alguns exemplos [37]. Para cada um destes contextos o tipo de rede de sensores pode variar desde terrestre, subterrâneo, subaquático, multimédia a móvel [35]. Cada contexto e tipo de rede diferente releva desafios próprios na hora de definir a sua arquitetura, tais como por exemplo, a fonte de energia para alimentar o sistema, a sua autonomia, o tipo de comunicação com os módulos capazes de se ligarem à internet e a forma de ligação à internet em si (Wifi, 3G/GPRS …) [36].

2.5. Microcontroladores/placas de prototipagem

No que respeita a sistemas de prototipagem para projetos IoT (*Internet of Things*) existem neste momento várias opções disponíveis no mercado. Poderíamos ter escolhido material de mais baixo nível (inclusive nesta revisão bibliográfica), mais específico, no entanto a escolha pelas opções apresentadas seguidamente espelha a nossa vontade inicial de ter um *proof of concept*. Sabemos, obviamente, que visar a produção real deste tipo sistema implicará uma reengenharia desta parte.

As placas retidas, depois de analisar as opções disponíveis no mercado, como potenciais candidatas à utilização na realização do protótipo são as seguintes:

- **Placas Arduino**[4] (vários tipos e configurações)
  Existem atualmente 21 placas diferentes disponíveis no mercado segundo o a página *web* oficial dos produtos Arduino. Estas placas permitem ligar a si, outro tipo de *hardware*, nomeadamente sensores, e fazer leituras dos valores recolhidos pelos mesmos. Uma desvantagem destas placas é que não permitem nativamente o armazenamento de dados.
- **Placas Raspberry Pi**[5]
  Estas placas são na realidade mini-computadores com características muito interessantes. Existem atualmente no mercado 6 modelos distintos. O sistema operativo suportado oficialmente para estas placas é o *Raspbian*, no entanto os modelos mais recentes (2 MODEL B e 2 MODEL B e 3 MODEL B) suportam a instalação do sistema operativo Windows 10 IoT Core.
- **Placas Netduino**[6]
  As placas Netduino são placas de prototipagem *open-source* baseadas na *framework* .NET Micro. Sendo baseadas na *framework* .NET podem ser programadas usando o *Visual Studio* ou o *Xamarin Studio*. Estas placas são compatíveis com as placas Arduino.
- **Módulo Intel Edison**[7]
  O módulo de computação Intel Edison não é por si só uma placa de prototipagem. Ele é normalmente utilizado em conjunto com uma *breakout board* (placa de componente seria a tradução mais aproximada) ou com uma placa de expansão do Arduino. Normalmente, compra-se esta placa como parte de um conjunto de material, que deverá ser montado pelo utilizador.
- **Placa Intel Galileo**[8]
  A placa *Intel Galileo* é uma placa baseada na arquitetura Intel x86, tendo sido certificada como compatível com as placas Arduino. O sistema operativo da placa é uma versão do Linux que incorpora bibliotecas do Arduino. Esta placa pode ser programada em ambientes OS X (Apple), Linux e Microsoft Windows.

[4] https://www.arduino.cc

[5] https://www.raspberrypi.org

[6] https://www.wildernesslabs.co/

[7] https://software.intel.com/en-us/get-started-edison-windows

[8] https://software.intel.com/en-us/get-started-galileo-windows

Todas as anteriores opções têm características próprias (muitas em comum), custos distintos e são configuradas (programadas) utilizando ferramentas e linguagens de programação distintas.

(NOTA: no momento da pesquisa de informação sobre as diferentes placas existentes no mercado a placas da *Intel*, *Edison* e *Galileo*, estavam disponíveis, no entanto, no momento de escrita desta tese descobriu-se que a Intel decidiu descontinuar as placas em 16 de Junho de 2017 [33])

2.6. Conclusão

Neste capítulo foi feita uma contextualização tecnológica das áreas relacionadas com o assunto desta tese.

Em 2.2 foi descrita a importância de sistemas similares ao GEOGREEN, bem como algumas vantagens deste em relação aos restantes. Em 2.3 uma descrição genérica do conceito de *Internet of Things* e em 2.4 introduziu-se o conceito de rede de sensores. Na seção 2.5 foi feita uma apresentação dos modelos de placas de prototipagem consideradas mais adequadas para a realização do protótipo pretendido.

3. Análise de requisitos

3.1. Introdução

Este capítulo vai focar-se na análise de requisitos do protótipo GeogreenPlus que se pretende desenvolver. Começaremos por definir o sistema na sua globalidade no ponto 3.2 e prosseguiremos para a definição dos requisitos funcionais em 3.3 e não funcionais em 3.4. Os pontos seguintes, 3.5, 3.6 e 3.7, irão apresentar respetivamente os diagramas de caso de uso, de atividade e de sequência resultantes da análise pormenorizada dos objetivos do protótipo. O ponto 3.8 irá detalhar os requisitos da interface com o utilizador.

3.2. Definição geral do sistema

O protótipo a desenvolver deverá ter como capacidade base ler diferentes tipos de dados do meio ambiente, circundante aos módulos GEOGREEN, e ter a capacidade de guardar esses dados para serem posteriormente consultados e analisados. Deverá ainda ser possível criar regras para que possam ser tomadas decisões baseadas em critérios definidos e dessas decisões resultarem ações práticas.

Sendo assim conseguem distinguir-se aqui três blocos diferentes que irão constituir o sistema: uma rede de sensores e placas de prototipagem capazes de interagir com eles, um sistema de informação que irá ser responsável por lidar com os dados (recolha, armazenamento e tomada de decisões) e ainda uma interface com o utilizador.

A recolha de dados e tomada de decisões deverá poder funcionar independentemente da interface com o utilizador, isto é, deverão existir várias "peças" que comuniquem entre si, mas que o correto funcionamento de umas não esteja dependente do correto funcionamento de outras.

Pretende-se ainda que seja, caso possível, privilegiada uma arquitetura do tipo nuvem, tirando assim partido das suas vantagens, tais como, a não necessidade de comprar máquinas físicas e não haver preocupações com a sua manutenção, a possibilidade de adicionar ou retirar recursos à medida das necessidades do sistema ou ainda a garantia da disponibilidade do serviço apresentada pelos fornecedores do serviço de nuvem.

A Figura 5 mostra a visão global do sistema pretendido e que deverá ser construído como resultado deste trabalho.

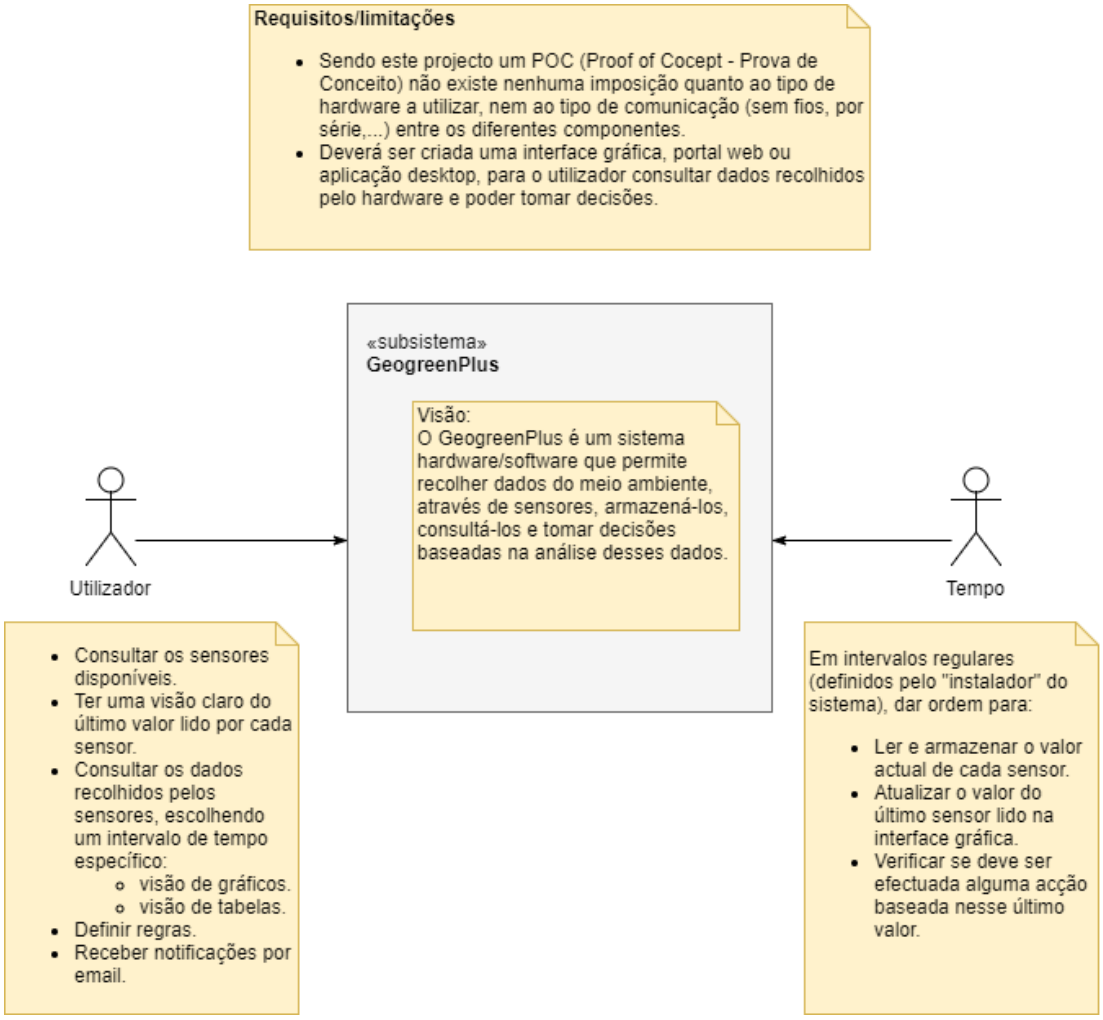

Figura 5 - Visão global do sistema

3.3. Requisitos funcionais

Os requisitos funcionais que se seguem ilustram as tarefas que o sistema deve fazer de forma automática e aquelas que deve permitir executar a um seu utilizador.

Tabela 1 - Lista dos requisitos funcionais

Identificador	Requisito	Ator
RF1	Obter lista dos sensores registados na BD	Sistema
RF2	Comunicar com os sensores	Sistema
RF3	Obter valor recolhido pelo sensor num determinado instante.	Sistema

	Medir (ler do meio envolvente às placas modulares): • Temperatura do ar; • Humidade do ar; • Humidade do solo; • Dióxido de carbono (CO2); • Luminosidade.	
RF4	Despoletar uma ação	Sistema
RF5	Guardar o valor recolhido pelo sensor na BD	Sistema
RF6	Consultar informação dos sensores registados na BD	Utilizador
RF7	Consultar os dados lidos pelos sensores registados na BD sob a forma de gráfico	Utilizador
RF8	Consultar os dados lidos pelos sensores registados na BD sob a forma de tabela	Utilizador
RF9	Definir regras e ações	Utilizador
RF10	Consultar regras e ações registadas na BD	Utilizador
RF11	Registar destinatários (*email*) de alertas	Utilizador
RF12	Consultar destinatários de alertas registados na BD	Utilizador

3.4. Requisitos não funcionais

A Tabela 2 lista os requisitos não funcionais a que o sistema deve obedecer. Estes requisitos focam-se em aspetos extra comportamentais do sistema, ou seja, não têm a ver com funcionalidades disponibilizadas pelo sistema em si, mas sim com questões relacionadas com o funcionamento do sistema.

Tabela 2 - Requisitos não funcionais

Identificador	Requisito
RNF1	O sistema deverá ser modular.
RNF2	O sistema deverá permitir a adição de sensores diferentes sem ter de alterar a estrutura existente.
RNF3	Escalabilidade
RNF4	Disponibilidade
RNF5	*Backup* de dados

3.5. Diagramas de casos de uso

Nesta secção irão ser descritos os diferentes casos de uso que o sistema deverá respeitar. Iremos começar por definir os casos de uso ligados à leitura e recolha de dados (rede de sensores) e posteriormente definir os casos de uso ligados à interface com o utilizador.

3.5.1. Casos de uso do sistema

Para recolher os valores do meio ambiente deverão ser usados diferentes tipos de sensores. Cada sensor terá de estar acoplado a uma placa de prototipagem, placa essa que terá a responsabilidade de, baseado num comando externo, pedir ao sensor o valor lido naquele instante, e enviar esse mesmo valor para ser armazenado. O sistema deverá ainda ser capaz de despoletar ações baseadas no valor recolhido, caso as haja configuradas, para o sensor em particular.

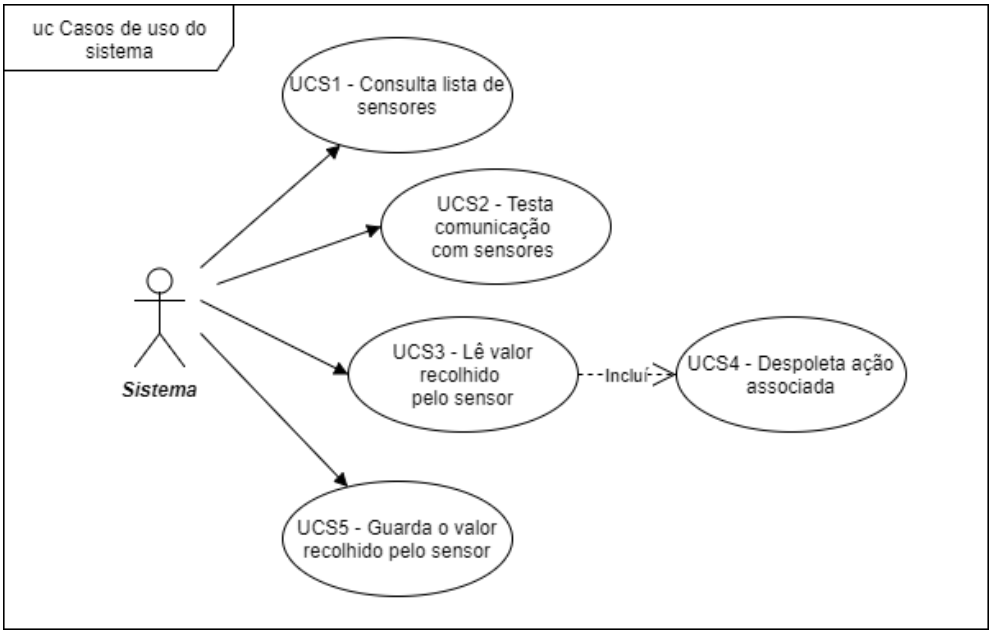

Figura 6 - Casos de uso da rede se sensores

Na Figura 6 podemos ver o diagrama de casos de uso da rede de sensores, no qual estão identificados 5 casos de usos distintos (UCS1 até UCS5):

- **UCS1 – Consulta lista de sensores**
  A primeira ação que o sistema deverá efetuar, depois de iniciar, será de consultar a lista de sensores disponíveis. Os sensores disponíveis estarão

registados numa base de dados e a sua consulta deverá ser efetuada através de um serviço web (a desenvolver no âmbito deste trabalho).

- **UCS2 – Testa comunicação com sensores**
  Depois de recuperar a lista de sensores disponíveis, o sistema deverá testar se consegue comunicar com cada um deles par que o processo de recolha de dados possa iniciar. Caso a comunicação seja bem sucedida, o sensor deverá ser instruído no sentido de começar a leitura de dados do meio ambiente.

- **UCS3 – Lê valor recolhido pelo sensor**
  O sistema deverá poder ler, em intervalos regulares, os valores recolhidos por cada um dos sensores.

- **UCS4 – Despoleta ação associada**
  A cada nova leitura do valor recolhido pelo sensor, o sistema deverá verificar se existe alguma regra de atuação associada a esse sensor e se essa ação deve ou não ser despoletada em concordância com o valor lido. As regras estarão guardadas na base de dados pelo que o sistema deverá utilizar o serviço web para efetuar a sua consulta.

- **UCS5 – Guarda o valor recolhido pelo sensor**
  O sistema deverá ser capaz de guardar os valores lidos na base de dados. O sistema não deverá ligar-se diretamente à base de dados pelo que deverá utilizar um método do serviço web para efetuar a operação.

3.5.2. Casos de uso do utilizador

Deverá ser disponibilizado ao utilizador uma interface gráfica que lhe permita tirar partido do sistema. Essa interface deverá permitir-lhe consultar a lista de sensores disponíveis, consultar os dados armazenados em formato de gráfico e de tabela e ainda definir regras de atuação do sistema.

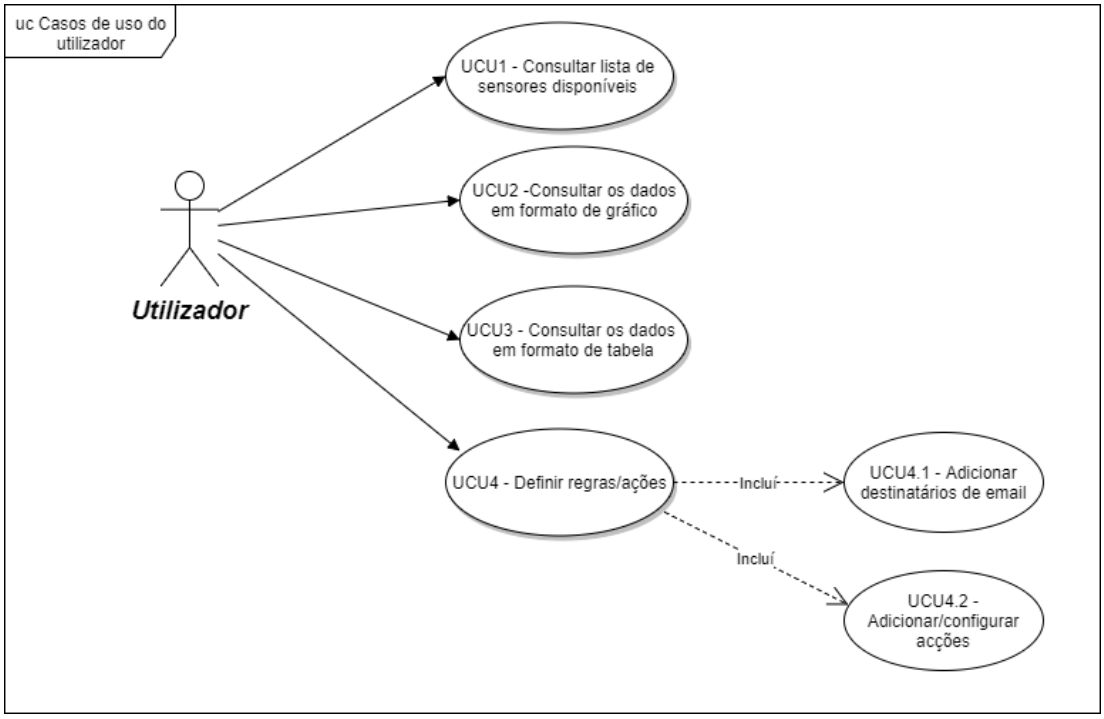

Figura 7 - Diagrama de casos de uso do utilizador

O diagrama de casos de uso representado na Figura 7 ilustra os 4 grupos de ações que deverão ser implementados, em vistas separadas, na interface gráfica. Segue uma descrição mais pormenorizada dos casos de uso em questão:

- **UCU1 - Consultar lista de sensores disponíveis**
  - O utilizador deverá ter, nesta vista, uma perceção global sobre a quantidade de sensores disponíveis no sistema.
  - Qual o estado atual de cada um dos sensores (ativo, inativo, defeituoso).
  - Qual a data do último valor lido e o respetivo valor.

- **UCU2 - Consultar os dados em formato de gráfico** (ver diagrama de atividade "act Consultar dados")
  - O utilizador deverá começar por selecionar o sensor (ou qual o tipo do valor do sensor) que pretende consultar.
  - Seguidamente escolher uma data de início e uma data de fim.
  - Obter os resultados para o intervalo temporal definido sob a forma de gráfico de barras.

- **UCU3 - Consultar os dados em formato de tabela** (ver diagrama de atividade "act Consultar dados"):

  - O utilizador deverá começar por selecionar o sensor (ou qual o tipo do valor do sensor) que pretende consultar.
  - Seguidamente escolher uma data de início e uma data de fim.
  - Obter os resultados para o intervalo temporal definido sob a forma de uma tabela com as seguintes colunas:
    - Data
    - Valor

- **UCU4 - Definir regras/ações**
  - Nesta vista o utilizador deverá poder definir certas regras e ações que o sistema deverá respeitar bem como adicionar destinatários de *email* para que estes recebam notificações:
    - **UCU4.1 – Adicionar destinatários de *email***
      - O utilizador deverá poder adicionar uma, ou várias, contas de *email* para receber notificações acerca do estado do sistema.
    - **UCU4.1 – Adicionar/configurar ações**
      - Esta funcionalidade deverá permitir ao utilizador definir várias ações que deverão ser executadas baseadas em regras definidas. Por exemplo, sempre que a temperatura exterior atingir certo valor, enviar *email* ao utilizador recomendando-lhe uma certa ação a tomar.

3.6. Diagramas de atividade

Nesta seção encontram-se os diagramas de atividade para os casos de uso referidos no ponto anterior. Estes diagramas ajudam a perceber o fluxo entre atividades sequenciais do sistema.

3.6.1. Atividade "ACT1 - Consultar dados"

A Figura 8 representa a atividade "Consultar dados" que vai permitir ao utilizador da aplicação consultar os dados recolhidos pelos diferentes sensores do sistema. Como representado no diagrama a atividade terá as seguintes etapas:

1. Escolher o tipo de dados a consultar (humidade do solo, temperatura, …).
2. Escolher o intervalo temporal que se pretende consultar:
   a. Escolher a data de início.
   b. Escolher a data de fim.
3. Carregar no botão que permite consultar os dados para o intervalo de tempo definido:

a. Este botão irá fazer uma chamada à API esperando como resultado uma lista com a informação pedida.

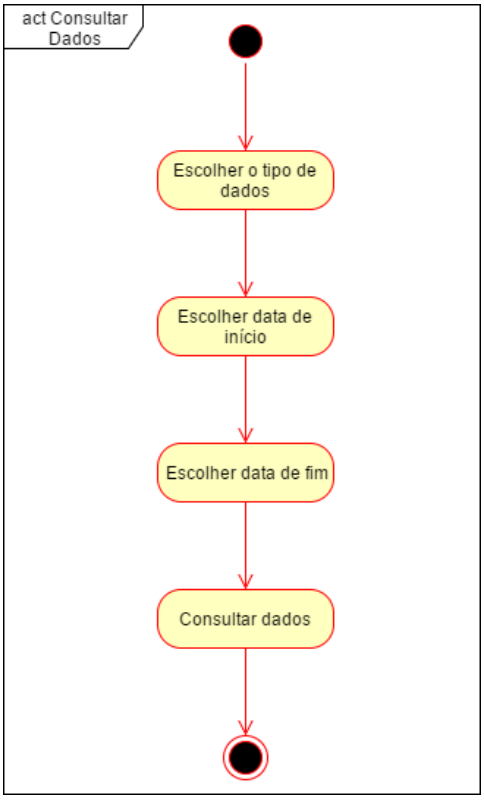

Figura 8 - Diagrama de atividade Consultar Dados

3.6.2. Atividade "ACT2 - Adicionar regra"

O diagrama representado na Figura 9 mostra os passos da atividade "Adicionar regra". Esta atividade permite ao utilizador criar uma nova regra que será depois utilizada pelo sistema para efetuar uma ação baseando-se numa dada condição. Os passos desta atividade são os seguintes:

1. Escolher o tipo de dados/sensor que a nova regra vai afetar
2. Escolher a condição:
    a. Neste passo o utilizador escolhe qual a condição que irá ser utilizada para, em conjunto com o valor da condição, despoletar a ação.
    b. Os valores possíveis para as condições são: igual, diferente, menor ou igual, maior ou igual.
3. Escolher o valor da condição:
    a. O valor da condição é um valor numérico.
4. Escolher a ação:

a. A ação escolhida irá ser executada sempre que o resultado da combinação entre a condição e o seu valor seja verdadeiro.
b. Os valores possíveis para as ações são: enviar *email*, adicionar água, adicionar nutriente.

5. Guardar a regra
    a. Para guardar a regra o utilizador irá carregar num botão que será responsável por invocar o método da API responsável por inserir uma nova regra na BD.

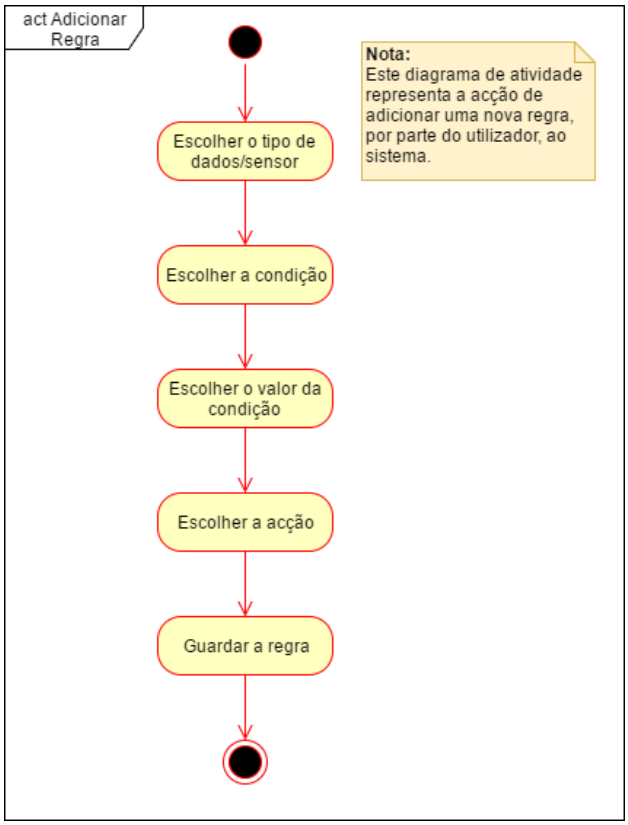

Figura 9 - Diagrama de atividade Adicionar Regra

3.6.3. Atividade "ACT3 - Adicionar email"

O utilizador pode adicionar destinatários de *email* de forma a que estes recebam notificações do sistema (ver "uc Casos de uso do utilizador – UCU4.1"). A Figura 10 ilustra essa atividade que deverá conter os seguintes passos:

1. Introduzir o nome do destinatário do *email*.
2. Introduzir um endereço de *email* válido para o destinatário.
3. Guardar os valores introduzidos.

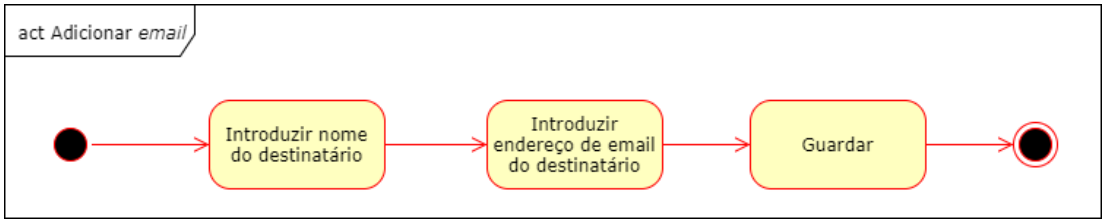

Figura 10 - Diagrama de atividade "Adicionar *email*"

3.7. Diagramas de sequência

Neste ponto estão definidos os diagramas de sequência que têm por objetivo ajudar a perceber a interação entre diferentes entidades do sistema bem como dar uma ideia da ordem ou sequência em que essa interação ocorre.

3.7.1. Sequência "SD1 - Controlo sensores"

A Figura 11 representa a sequência completa de acontecimentos que deverão decorrer no sistema que irá interagir diretamente com a rede de sensores. Nela podemos verificar que haverá 4 entidades envolvidas nesta sequência, sendo que ela é iniciada pela entidade "*:placa Prototipagem*" no momento do arranque da aplicação. Detalha-se seguidamente a sequência de acontecimentos:

- É feita uma chamada ao método da Web API que permite consultar os sensores existentes (registo guardados na BD):
    - O método da web API consulta a BD e devolve a lista de sensores existentes.
- Cada sensor devolvido pela API deverá ter um identificador único. É feita uma tentativa de comunicação com esse sensor e, em caso de sucesso:
    - O programa lê o intervalo de tempo que deve ser respeitado para consultar os dados lidos pelo sensor e configura um timer responsável por despoletar o evento que irá ler o valor lido pelo sensor naquele instante.
- A cada ocorrência do evento disparado pelo timer o programa pede o valor lido pelo sensor, a que está associado esse evento:
    - É então feita uma chamada ao método da Web API responsável por guardar o valor na BD.
    - É também feito um pedido ao método da Web API que devolve as regras existentes na BD que estejam associadas ao sensor em causa e,

caso exista(m) alguma(s) regra(s), o programa irá executar as ações definidas por esta(s).

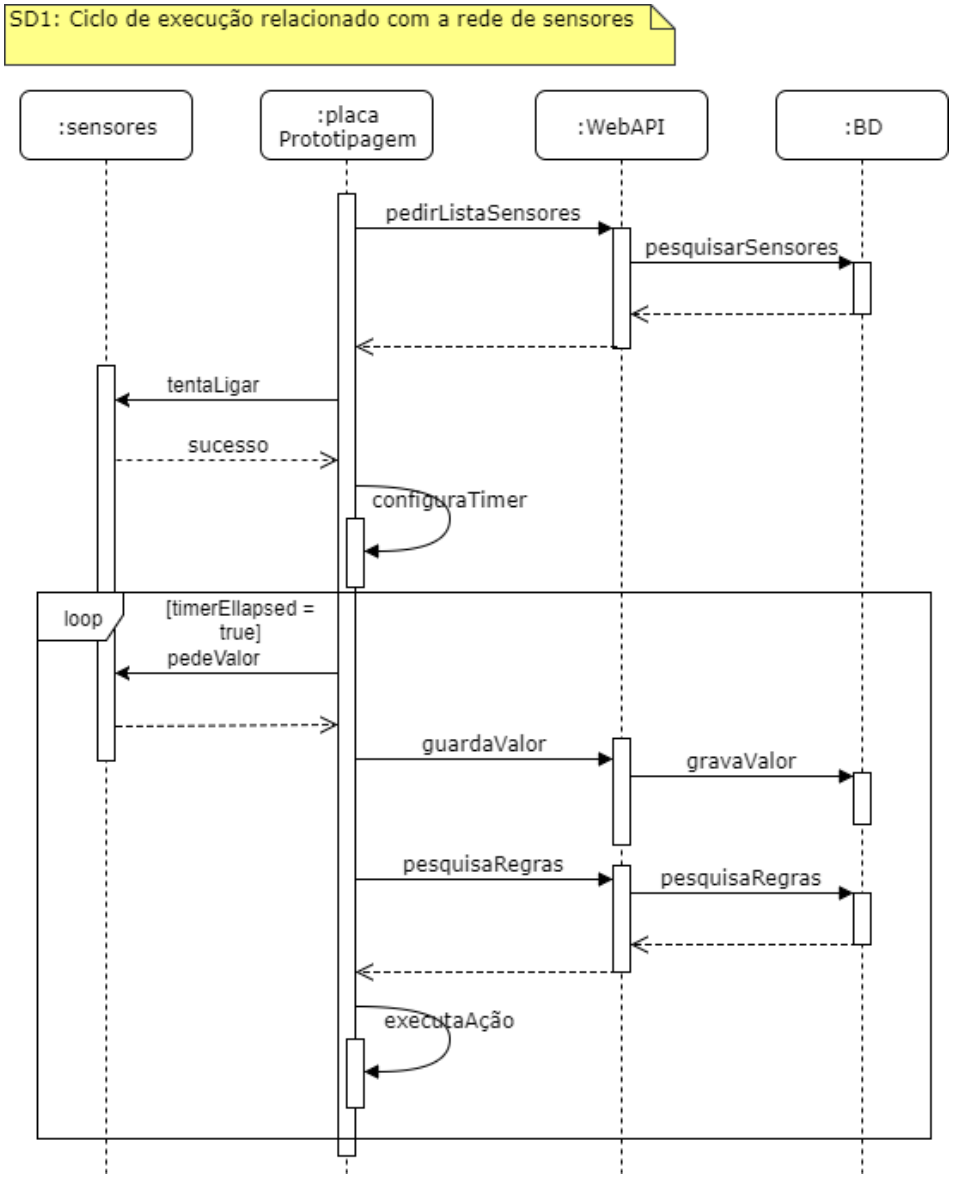

Figura 11 - Diagrama de sequência "Controlo sensores"

3.7.2. Sequência "SD2 - Consultar dados"

O diagrama representado na Figura 12 representa as ações efetuadas ao consultar os dados recolhidos pelos sensores. O diagrama mostra a existência de 4 entidades

diferentes nesta sequência, o utilizador, a interface gráfica do GeogreenPlus, a Web API e a base de dados.

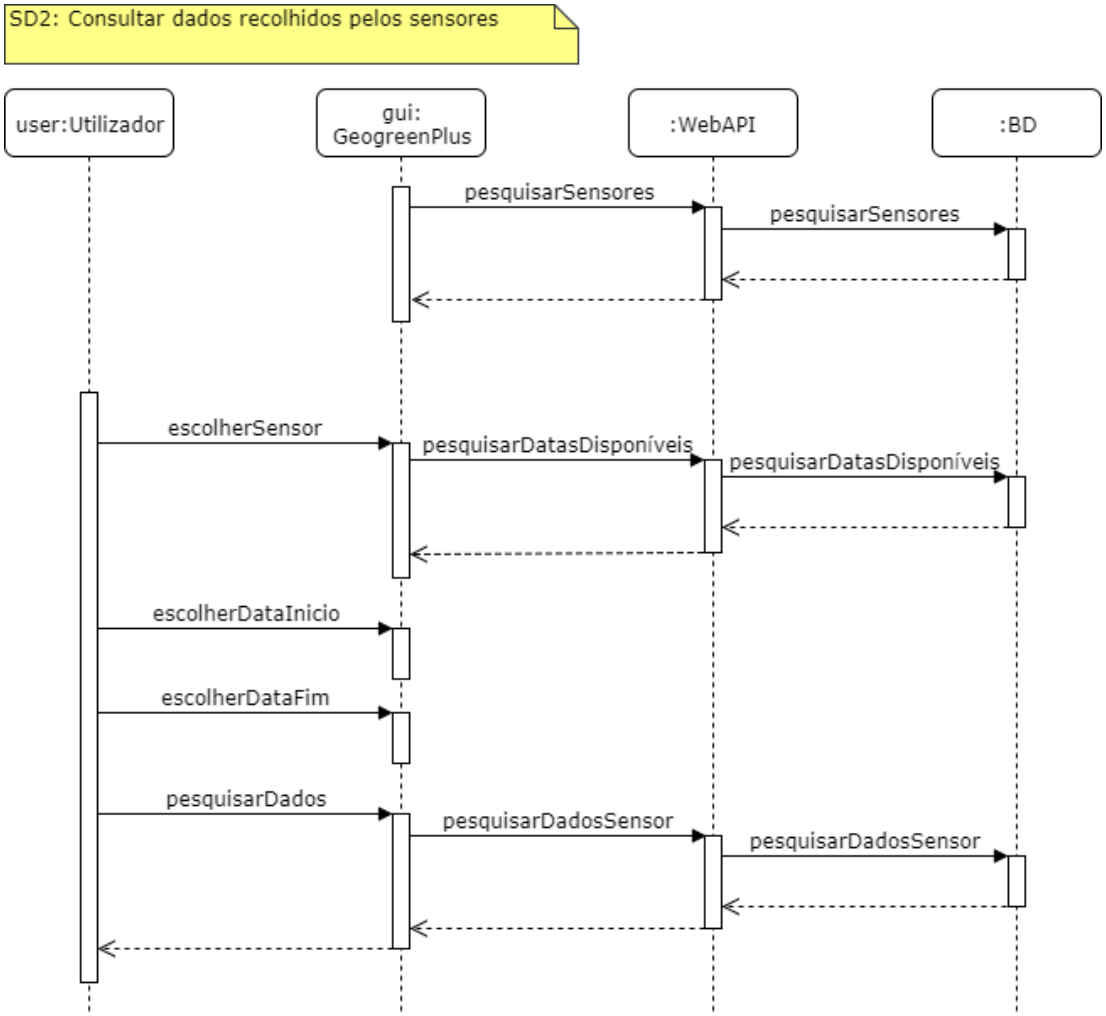

Figura 12 - Diagrama de sequência "Consultar dados"

A interface com o utilizador do GeogreenPlus deverá possibilitar ao utilizador consultar os dados recolhidos pela rede de sensores. Como referido anteriormente, na seção 3.5.2 (casos de uso UCU2 e UCU3), essa consulta poderá ser efetuada em formato de gráfico ou de tabela, pelo que deverão existir 2 "ecrãs" distintos de forma a respeitar esse requisito. A sequência de acontecimentos será, no entanto, igual para cada um dos casos e deverá ser a seguinte:

1. É feita uma chamada ao método da Web API que permite listar os sensores:
   1.1. O método da Web API consulta a BD e devolve a lista de sensores existentes.
   1.2. A interface disponibiliza a lista de sensores recebidos para que o utilizador possa selecionar um deles.

2. O utilizador escolhe um sensor:
   2.1. Essa ação despoleta uma chamada ao método da Web API que vai devolver as datas com dados disponíveis para o respetivo sensor.
   2.2. A Web API pesquisa os dados na BD e devolve o resultado.
   2.3. A interface disponibiliza a lista de datas.
3. O utilizador escolhe a data de início.
4. O utilizador escolhe a data de fim.
5. O utilizador carrega no botão pesquisar dados:
   5.1. É efetuada uma chamada ao método da web API que irá devolver os valores lidos pelo sensor no intervalo de datas definido.
   5.2. A Web API pede os valores à BD e devolve-os.
   5.3. A interface disponibiliza os pares (data, valor) para consulta. A interface poderá disponibilizar os dados em formato de gráfico de linhas ou de tabela.
6. O utilizador consulta os dados.

3.7.3. Sequência "SD3 - Adicionar regra"

A Figura 13 mostra o diagrama com a sequência de ações para a atividade de adicionar uma nova regra ao sistema. O diagrama mostra a existência de 4 entidades diferentes nesta sequência, o utilizador, a interface gráfica do GeogreenPlus, a Web API e a base de dados.

A interface gráfica do GeogreenPlus deverá permitir ao utilizador adicionar regras ao sistema. A sequência de acontecimentos para tal deverá ser a seguinte:

1. É feita uma chamada ao método da Web API que permite listar os sensores:
   1.1. O método da Web API consulta a BD e devolve a lista de sensores existentes.
   1.2. A interface disponibiliza a lista de sensores recebidos para que o utilizador possa selecionar um deles.
2. O utilizador escolhe uma condição.
3. O utilizador escolhe um valor.
4. O utilizador escolhe a ação que irá ser despoletada.
5. O utilizador carrega no botão para adicionar a regra:
   5.1. É efetuada uma chamada ao método da Web API responsável por criar a nova regra.
6. O método da Web API insere a nova regra na BD e devolve o estado (sucesso ou insucesso) da operação.

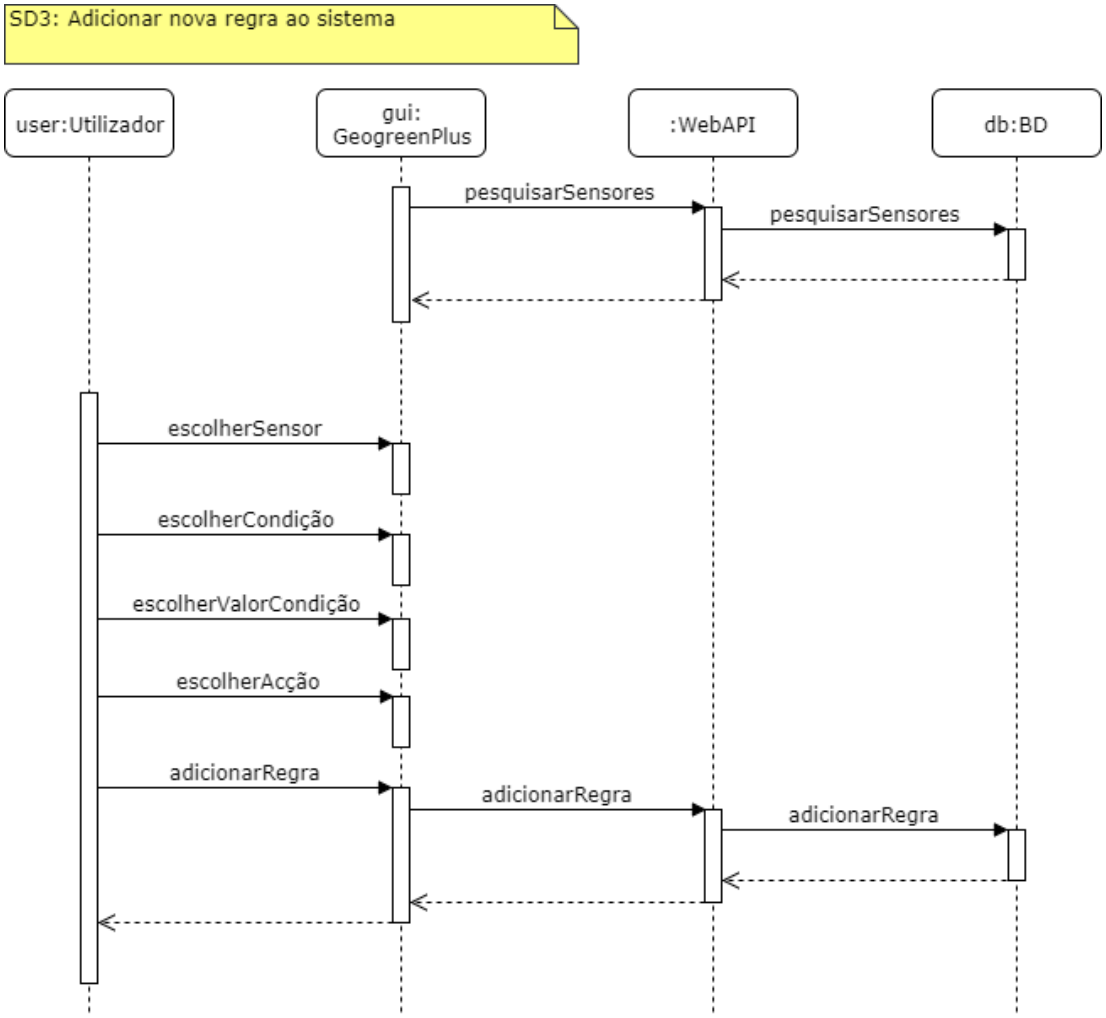

Figura 13 - Diagrama de sequência "Adicionar regra"

3.8. Interface com o utilizador

Após a recolha de dados, lidos pelos diversos sensores, é necessário disponibilizar esses mesmos dados ao utilizador para que ele possa compreender como se está a comportar o sistema e, se tal for o caso, realizar ações em concordância com a informação obtida.

O que se pretende para este protótipo é construir uma interface gráfica com as seguintes características:

- Comunicar com a base de dados de forma a poder enviar ou receber informação.
- Mostrar a informação recebida, os valores lidos pelos sensores, ao utilizador na forma de gráficos e tabelas.

- O utilizador deverá poder selecionar qual o intervalo temporal para o qual pretende consultar os valores.
- Permitir ao utilizador configurar o comportamento das recomendações e dos atuadores:
    - Caso o sistema possua atuadores, como por exemplo válvulas que possam ser abertas para regar, se o valor da humidade do solo for inferior a determinado valor, chamar o serviço web responsável por mandar abrir a válvula.
    - Caso contrário, se o valor da humidade do solo for inferior a determinado valor, recomendar ao utilizador regar as plantas através do envio de uma mensagem de *email*.

A interface gráfica deverá interagir com a base de dados através de uma Web API (REST). Essa API deverá ter métodos que permitam consultar os dados existentes na BD, atualizar alguns desses dados e inserir novos valores. Esses métodos deverão poder ser testados de forma simples e deverão ser documentados.

3.9. Conclusão

Neste capítulo começamos, no ponto 3.2, por definir o sistema de forma geral identificando os componentes e ações a implementar. Em 3.3 definiram-se os requisitos funcionais que o sistema deve respeitar e em 3.4 os requisitos não funcionais. Em 3.5 foram definidos os diagramas de caso de uso do sistema e do utilizador. Em 3.6 definiram-se os diagramas de atividade e em 3.7 os diagramas de sequência. A seção 3.8 descreve os requisitos a que a interface com o utilizador deve obedecer.

Estando os requisitos do protótipo pretendido definidos pode avançar-se para o desenvolvimento do mesmo. No próximo capítulo irá detalhar-se o processo de desenvolvimento.

4. Desenvolvimento do protótipo

4.1. Introdução

No capítulo 4 irão ser descritos os passos efetuados para o desenvolvimento do protótipo GeogreenPlus. Em 4.2 irão ser apresentadas as tecnologias e material escolhido para a realização do protótipo bem como a razão dessas escolhas. Na seção 4.3 irá ser apresentada a arquitetura definida para o sistema. Nos pontos 4.4 e 4.5 explica-se a o processo de configuração do material, nomeadamente os sensores, as placas Arduino e Raspberry Pi. Em 4.6 detalha-se a implementação da Web API, apresentando os métodos desenvolvidos e as suas características. Em 4.7 está patente o trabalho efetuado no desenvolvimento do *software* que irá correr nas placas Raspberry Pi, ou seja, o software que terá por responsabilidade comunicar com a rede de sensores para obter e armazenar os dados lido por esta. Além disso, este *software* será responsável por despoletar ações baseadas em regras definidas no sistema. Na seção 4.8 descreve-se o desenvolvimento da interface com o utilizador que irá permitir a consulta dos dados recolhidos pelos sensores e a definição de regras para o sistema.

4.2. Escolha do material e tecnologias a utilizar

Partindo dos resultados da análise de requisitos descritos no capítulo 3 temos de escolher quais as tecnologias, material (sensores e placas de prototipagem) e ferramentas a usar para o desenvolvimento do protótipo pretendido.

Do ponto de vista geral quisemos optar por um compromisso entre facilidade de programação e preço baixo do equipamento o que levou à escolha de material *embedded "multi-purpose"*. Para a construção de uma prova de conceito esta abordagem é válida, no entanto, no futuro, se houver passagem para uma fase de produção industrial a configuração deverá ser diferente. Por exemplo as comunicações entre componentes deverão utilizar tecnologias sem fios (*Bluetooth* ou outras) em vez de cabos USB.

O sistema irá ser constituído por vários módulos diferentes com componentes de *hardware* e *software* sendo que queremos uma boa integração entre eles. Pareceu-nos importante a maior uniformidade possível do contexto tecnológico (desde que saímos dos sensores em si, estamos num mundo tecnológico uniforme, embora estejam ainda envolvidos componentes bastantes diferentes, *tablet*/computador, servidor/nuvem, BD, análise de dados, etc.).

Tendo em conta estes pontos e somando a experiência do autor desta tese no ambiente tecnológico .NET decidimos avançar com tecnologias desta família na construção do protótipo GeogreenPlus. Além do fator da experiência prévia pareceu-nos pertinente escolher o .NET pela possibilidade de, no futuro, se procurarem parcerias com empresas e/ou iniciativas que evoluem no mundo tecnológico do .NET/Azure[9].

4.2.1. Placas de prototipagem

Para recolher os valores do meio ambiente serão usados diferentes tipos de sensores (definidos no ponto 4.2.2). Cada sensor terá de estar acoplado a uma placa de prototipagem, placa essa que terá a responsabilidade de, baseado num comando externo, pedir ao sensor o valor que está e recolher num determinado instante, e enviar esse mesmo valor para ser armazenado.

A placa Arduino Uno tem a vantagem de ser uma placa relativamente barata e de existirem no mercado diferentes tipos de sensores, que correspondem às necessidades deste projeto, compatíveis e com documentação para a sua correta utilização. Por estas razões escolhemos as placas Arduino Uno para servirem de "interface" aos sensores.

Para centralizar a recolha dos dados provenientes dos vários Arduinos a escolha recaiu sobre as placas Raspberry Pi 2, tendo sido uma das principais razões para esta escolha, a possibilidade de utilizar o sistema operativo Windows 10 IoT Core. Esta possibilidade vem de encontro ao desejo enunciado anteriormente de usar um contexto tecnológico uniforme e da escolha da tecnologia em si, o .NET. Outra razão para a escolha em relação às placas Netduino que suportam nativamente a programação em .NET foi o fator preço, sendo as placas Netduino mais caras do que as Raspberry Pi. As placas da família Intel seriam uma opção interessante, no entanto, em Novembro de 2015, a Microsoft deixou de suportar as placas *Galileo* [34] fazendo com que a programação num ambiente .NET fosse comprometido.

O Raspberry Pi não é apenas uma placa de prototipagem é na realidade um computador de dimensões reduzidas que pode correr *software* com variadas características. No caso concreto deste projeto, o Raspberry Pi será uma interface entre as diferentes placas Arduino e a aplicação gráfica para o cliente final. As suas responsabilidades serão as seguintes:

- Receber comandos provenientes de um serviço web.
- Baseado no comando recebido, enviar esse comando para o Arduino correto.
- Receber a mensagem do Arduino e comunicá-la através do serviço web.

[9] https://docs.microsoft.com/en-us/dotnet/azure/

A Figura 14 mostra uma visão geral do diagrama do sistema pretendido, focando-se mais na parte do *hardware*. Esta figura mostra a arquitetura mais tradicional de cliente-servidor.

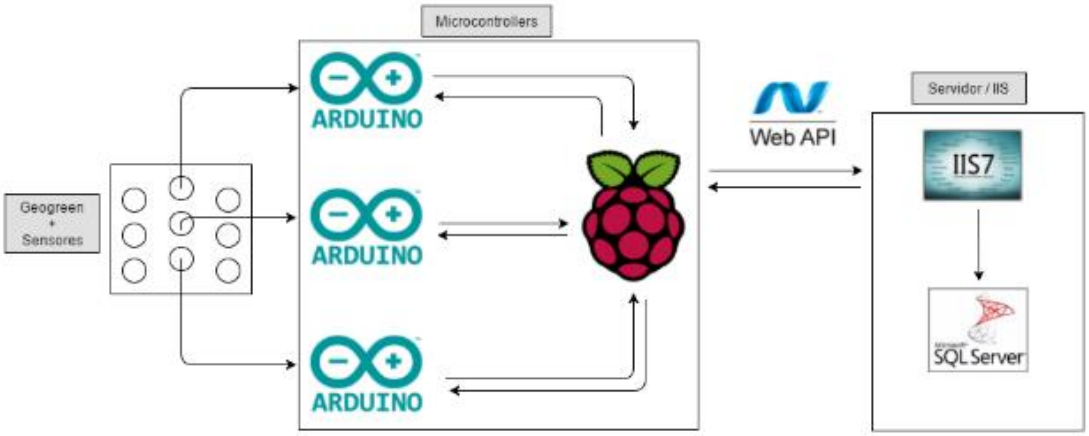

Figura 14 - Diagrama do *hardware* (arquitetura de cliente-servidor)

Na Figura 15 podemos ver o diagrama do sistema, mas com uma arquitetura de nuvem, neste caso utilizando o Microsoft Azure.

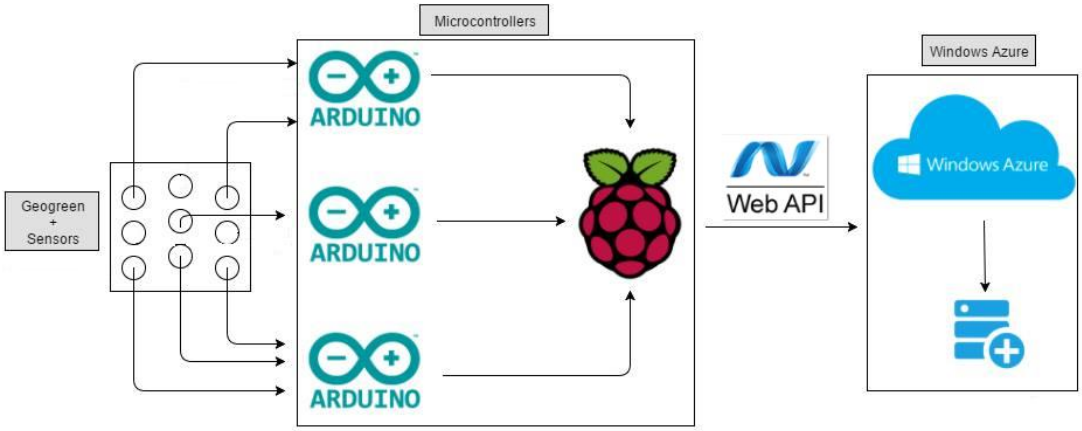

Figura 15 - Diagrama do *hardware* (arquitetura de nuvem)

4.2.2. Sensores

Existem no mercado sensores de vários fabricantes que permitem medir as propriedades descritas no requisito funcional RF3, e que são, normalmente, compatíveis com as diferentes placas enunciadas no ponto anterior. Para a realização deste protótipo a escolha dos sensores teve como principais critérios a compatibilidade com as placas Arduino e o seu baixo preço. Estes sensores poderão no futuro ser substituídos por outros mais precisos e mais resistentes.

Para esta versão beta foram escolhidos quatro sensores diferentes que leem cinco tipos de informação distinta do meio ambiente:

- **DHT22**: este sensor, relativamente barato, lê 2 tipos de informação, a temperatura e a humidade do ar.
- ***Soil Moisture*** (da *iTead Studios*): este sensor lê a humidade do solo.
- **MG-811**: lê os valores de dióxido de carbono (CO2) à sua volta.
- **TSL2561**: este sensor lê a luminosidade do meio envolvente.

Poderão ser adicionados mais sensores, aliás deverão ser adicionados mais sensores em trabalhos futuros que permitam ler valores dos nutrientes do solo (magnésio, potássio, …). Para este POC não há razões para utilizar sensores adicionais uma vez que eles não iriam alterar a solução apresentada. Sendo assim as razões por trás da escolha destes quatro sensores são, primeiro porque todos eles são sensores relativamente baratos e segundo porque são todos sensores suportados pelos Arduinos.

4.2.3. Linguagens de programação

- **C#**
  O C# [24], sendo uma linguagem orientada a objetos permite tirar partido das características deste paradigma tais como:
    - Herança – permite definir classes base com funcionalidades específicas e definir classes derivadas que podem herdar ou sobrepor (*override*) essas funcionalidades [27].
    - Polimorfismo – permite que várias classes derivadas de uma mesma classe base possam invocar métodos que têm a mesma assinatura, mas implementações distintas [28].
    - Encapsulamento – consiste em esconder os detalhes de implementação de uma classe, controlando assim o acesso aos seus atributos e métodos [30].

  Estas características permitem escrever código reutilizável, fácil de manter e fácil de estender. Além destas características, o C# é uma linguagem

extremamente versátil no ambiente .NET uma vez que pode ser utilizada, como linguagem principal ou de suporte, tanto na construção de *software* mais clássico como na construção de serviços web (WCF, ASP.Net Web API), interfaces com o utilizador (WinForms, WPF), aplicações universais no contexto tecnológico Microsoft (UWP) e páginas Web (ASP.NET, ASP.NET MVC).

- **WPF – *Windows Presentation Foundation***
  O WPF [25] é a tecnologia mais recente da Microsoft para a construção de aplicações cliente *desktop*. As principais características do WPF são a de permitirem uma separação clara entre a interface com o utilizador e a lógica de negócio, utilização da linguagem XAML para a definição da estrutura da interface gráfica, controlo total sobre o aspeto e comportamento dos componentes da interface gráfica e possibilidade de programar a lógica da aplicação (*Code-Behind*) com a linguagem C#.
  As aplicações escritas em WPF podem correr em máquinas com sistemas operativos Windows instalados, sejam eles computadores pessoais clássicos ou *tablets*. Esta última característica foi importante na escolha desta tecnologia uma vez que se pretende que o *software* de interação com o utilizador corra em *tablets Microsoft Surface* (ou qualquer outro *tablet* com sistema operativo *Windows* instalado).

- **MVVM (*Model-View-ViewModel*)**
  O MVVM [31] é um padrão de arquitetura para desenvolvimento de *software* que tem por base a separação de conceitos. Neste padrão a lógica é sempre independente da interface gráfica permitindo a *designers* e programadores trabalharem em simultâneo. Esta separação também permite que o código possa ser testado muito mais facilmente.

- **Entity Framework**
  O Entity Framework é um ORM (*Object-relational mapping*) e é a tecnologia recomendada pela Microsoft para acesso a dados [26]. Um ORM permite aos programadores trabalharem com dados relacionais (base de dados) como se fossem objetos de domínio do *software* que estão a desenvolver. Sendo assim o acesso a base de dados não é feito diretamente através de SQL, mas as suas tabelas são mapeadas para classes e são acedidas através de instâncias das mesmas.
  O Entity Framework pode ser utilizado em cenários de base de dados já existentes (*Database First*) mapeando as tabelas para classes ou então num cenário em que se pretende criar uma base de dados nova baseada no modelo de dados definido pelas classes do código (*Code First*). Iremos neste trabalho

utilizar esta segunda abordagem, definindo o modelo no código e gerar a base de dados baseada nesse modelo.

Refere-se ainda que o IDE escolhido para o desenvolvimento do *software* foi o *Microsoft Visual Studio 2015 Comunity*. O *Visual Studio* é o IDE de referência para a escrita de programas nas diferentes tecnologias do universo .NET.

4.2.4. ASP.Net Web API

Uma API (*Application Programming Interface*) é conjunto de comandos, protocolos e ferramentas utilizados para desenvolver *software*. As API's têm como objetivo facilitar a interação entre diferentes programas [8].

Uma Web API é um caso específico de uma API que, como nome indica, é uma API na web e pode ser acedida usando o protocolo HTTP. Estas API's podem ser escritas utilizando várias linguagens de programação.

Como foi referido varias vezes ao longo dos capítulos anteriores, existe a necessidade de desenvolver um serviço web que irá permitir a comunicação entre diferentes componentes do sistema.

O ASP.NET Web API é uma *framework* disponibilizada pela Microsoft com o objetivo de tornar mais simples o desenvolvimento de serviços HTTP que possam ser utilizados por diferentes clientes tais como dispositivos móveis e navegadores web [9][10].

Para efetuar a descrição, consulta e consumos dos métodos disponibilizados pela Web API irá ser usada a *framework* Swagger[10] (OpenAPI). Um dos objetivos do Swagger é fazer com que a documentação evolua à medida da implementação dos métodos uma vez que essa documentação vai ser gerada lendo anotações (comentários) do código.

4.2.5. Windows Remote Arduino

A Microsoft disponibiliza uma interface para controlar dispositivos compatíveis com Arduino através de aplicações Windows 10 *Universal Windows Application* [11]. *Windows Remote Arduino* é uma biblioteca *open source* que permite a qualquer dispositivo com sistema operativo Windows 10 controlar remotamente um Arduino

[10] https://swagger.io/

[15]. A biblioteca permite assim integrar sensores compatíveis com Arduino em projetos Windows.

O *Windows Remote Arduino* pode controlar as seguintes funções das placas Arduino:

- GPIO – I/O Analógico e digital
  - Escrita digital (*Digital write*)
  - Leitura digital (*Digital read*)
  - Escrita analógica (*Analog write*)
  - Leitura analógica (*Analog read*)
  - Configurar o modo do pin (*Setting pin mode*)
  - Receber eventos quando valores do pin são comunicados ou alterados
- Enviar e receber dados entre dispositivos através de I2C.
- Envio de comandos personalizados utilizando para isso *Firmata SysEx* [16].

A Microsoft tentou desenhar a API do *Windows Remote Arduino* de forma a que as diferenças com a API *Arduino Wiring* fossem as menores possíveis permitindo assim a utilizadores familiarizados com os métodos existentes no Arduino utilizarem facilmente a API da Microsoft. O protocolo utilizado para efetuar a comunicação entre o Windows 10 e as placas Arduino foi o protocolo Firmata[11], pelas razões de este ter implementações em várias linguagens incluído a API *Arduino Wiring*.

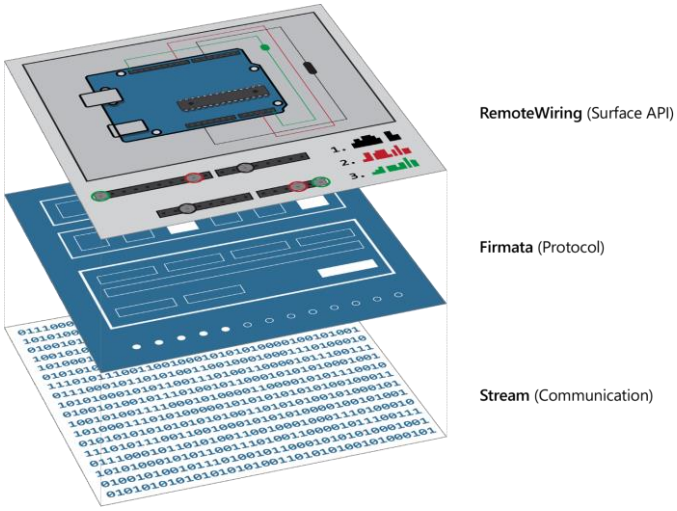

Figura 16 - Arquitetura do *Windows Remote Arduino*[12]

[11] http://www.firmata.org/wiki/Main_Page

[12] Imagem retirada de https://blogs.windows.com/buildingapps/2016/02/04/what-is-windows-remote-arduino-and-what-can-it-do/

4.3. Arquitetura do sistema

Depois de selecionados os componentes tecnológicos partiu-se para o desenvolvimento do protótipo. A base de partida para o desenvolvimento foi a análise de requisitos, apresentada no capítulo 3 deste documento. A partir dessa análise definiu-se a arquitetura do protótipo representada na Figura 17.

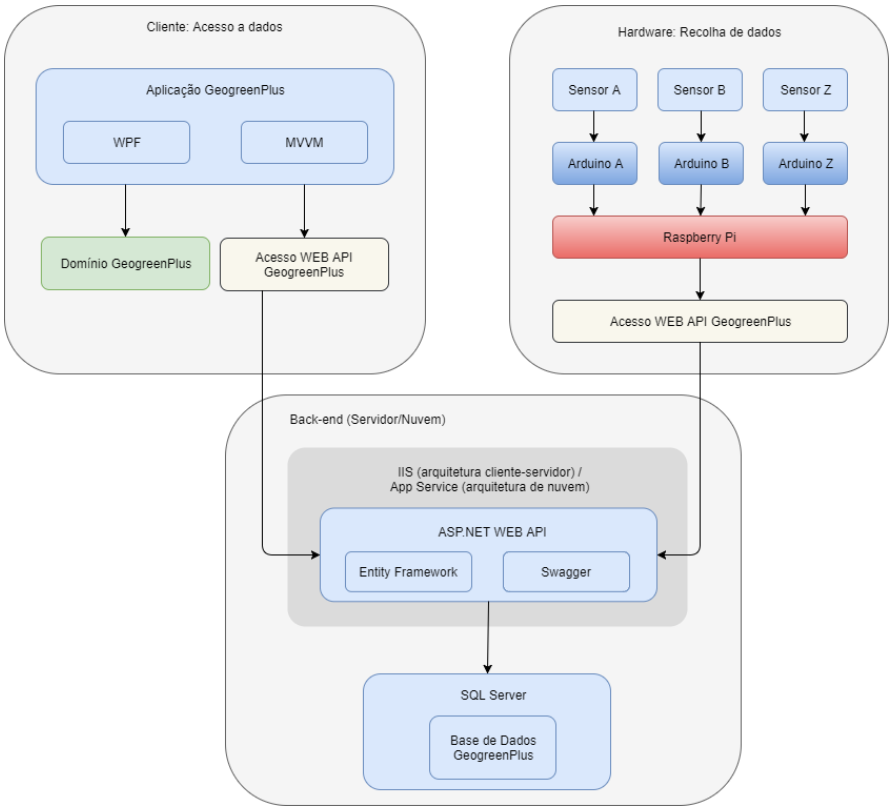

Figura 17 - Arquitetura do protótipo

Na Figura 17 podem observar-se os 4 blocos gerais que serão a base do protótipo:

- **Cliente: Acesso a dados**
  - Aplicação GeogreenPlus: GUI para a consulta de informações relacionadas com o sistema bem como para a definição de algumas regras. A tecnologia escolhida para a sua implementação foi o WPF (*Windows Presentation Foundation*) utilizando o padrão de desenvolvimento MVVM (*Model-View-ViewModel*).
  - Domínio GeogreenPlus: módulo com as classes de domínio da aplicação.
  - Acesso WEB API GeogreenPlus: módulo com as classes e métodos (leitura, escrita, atualização, remoção) de acesso à Web API que permite comunicar com a base de dados.

- ***Hardware*: Recolha de dados**
  - Sensores: diferentes tipos de sensores que vão recolher informação do meio ambiente.
  - Arduinos: as placas Arduino terão os sensores ligados a si e serão responsáveis por enviar os valores lidos por estes para a placa Raspberry Pi.
  - Raspberry Pi: a placa Raspberry Pi terá ligadas a si, diferentes placas Arduino. O Raspberry Pi terá como tarefa pedir os dados lidos, em intervalos de tempo regulares, definidos pelo utilizador, aos Arduinos e chamar o serviço web responsável por armazenar esses valores na base de dados:
  - Acesso WEB API GeogreenPlus: o *software* a desenvolver, que irá correr na placa Raspberry PI, deverá ter uma camada que fará o acesso aos serviços da Web API. Essa API terá métodos que serão acedidos tanto pelo *software* que corre no Raspberry Pi como pela aplicação GeogreenPlus.
- **Back-end (Servidor/Nuvem)**
  - **IIS/App Service**
    ASP.NET WEB API: o servidor web IIS, no caso da arquitetura cliente servidor, ou um serviço do tipo *App Service* no caso de arquitetura de nuvem, irá alojar a web API. Esta será escrita usando a tecnologia ASP.NET WEB API e conterá os métodos para guardar, alterar, consultar e apagar informação da base de dados GeogreenPlus.
  - **SQL Server**
    Base de dados GeogreenPlus: a base de dados SQL Server que irá alojar, entre outra, a informação recolhida pelos sensores. De notar que esta base de dados será gerada pelo Entity Framework, isto é, será utilizado um contexto de *Code First* e não *Database First*.

4.4. Configuração dos sensores e Arduinos

As placas Arduino terão ligadas a si os sensores necessários para a recolha de dados do meio circundante às placas modulares GEOGREEN. Para as necessidades deste protótipo iremos ligar apenas um sensor a cada placa Arduino, sendo assim teremos um total de 4 placas Arduino, cada uma ligada a um dos diferentes sensores listados no ponto 4.2.2.

A comunicação entre os Arduinos e os Raspberry Pi será efetuada por cabo USB. A solução ótima para a comunicação entre as placas seria uma comunicação sem fios, no entanto, as placas Raspberry Pi utilizadas neste protótipo não têm incorporado nem módulo de *Bluethoot* nem de *WiFi*.

Como referido anteriormente, cada um dos 4 sensores enumerados no ponto 4.2.2 foi ligado a uma placa Arduino seguindo os esquemas de ligação fornecidos pelos construtores ou distribuidores dos sensores. Depois de ligados os sensores foi necessário programar as placas Arduino para que pudessem ler os dados recolhidos pelos diferentes sensores e comunica-los à interface gráfica.

Sendo estes modelos de sensores bastantes populares entre a comunidade de *makers* o código fonte relativo a cada modelo está disponível de forma libre para ser usado pela comunidade, sendo assim, não foi desenvolvido nenhum código de raiz, mas sim reutilizado código escrito, ou pelos construtores dos sensores ou por algum membro da comunidade de *makers*.

As placas Arduino necessitam de ser programadas por forma a poderem receber os *inputs* dos sensores a si ligados e transmiti-los para os Raspberry PI. Cada um dos sensores utilizados neste projeto necessita de código específico. Esse código é enviado para a placa Arduino através da ferramenta **Arduino IDE**[13]. Este é um IDE *open-source* desenvolvido pelos criadores do projeto Arduino que incluí entre outros um editor de código (suporta as linguagens de programação C e C++) e um "*Serial monitor*" que permite ao programador ver o output do código executado pelo Arduino.

Como neste projeto temos por objetivo que as placas Arduinos possam comunicar através de USB com placas Raspberry PI (que terão o sistema operativo Windows 10 IoT Core instalado), iremos utilizar a biblioteca *Windows Remote Arduino* [11] disponibilizada pela Microsoft para lidar com as comunicações entre Arduino e Raspberry PI. Para poder utilizar esta interface deverá ser utilizado como base de código para as placas Arduino o ficheiro "*StandardFirmata.ino*" (também disponibilizado pela Microsoft) com alterações específicas para lidar com cada um dos sensores utilizados neste protótipo. Essas alterações serão o código específico de cada uns dos sensores listados nos pontos seguintes.

4.4.1. Arduino com sensor DHT22

O código fonte para a placa Arduino que tem o sensor DHT22 ligado foi recuperado *online*[14]. Nesta página está ainda descrito o modelo de ligação do sensor à placa Arduino.

Como referido anteriormente, o sensor DHT22, permite a leitura de 2 tipos de dados do meio ambiente, a temperatura e a humidade do ar. Sendo assim, o código fonte

[13] https://www.arduino.cc/en/Main/Software

[14] https://create.arduino.cc/projecthub/attari/temperature-monitoring-with-dht22-arduino-15b013

original, foi ligeiramente alterado de forma a permitir recuperar apenas o tipo de valor pretendido e não os dois de cada vez.

4.4.2. Arduino com Moisture Sensor

O sensor de humidade do solo é o mais simples de configurar uma vez que não necessita de nenhum código específico para funcionar. As instruções de utilização podem ser obtidas *online*[15].

O sensor pode ser utilizado usando os pins analógicos ou digitais, bastando para isso mover o botão presente no seu lado direito para a opção A ou D. O sensor pode ler informação específica da humidade do solo (pin analógico) ou informação de solo extra seco ou extra húmido de acordo com o limiar estabelecido (pin digital). O potenciômetro presente no sensor permite ajustar esse limiar.

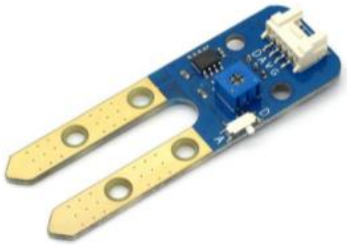

Figura 18 - Sensor de humidade do solo

Para este trabalho será usada a opção do pin analógico que permite obter o valor específico da humidade do solo.

4.4.3. Arduino com sensor TSL2561

O sensor de luminosidade TSL2561 é um sensor de luz digital que permite ler valores da luz circundante. As informações para a configuração do sensor de luminosidade podem ser encontradas *online*[16].

[15] https://www.itead.cc/wiki/Moisture_Sensor

[16] https://github.com/adafruit/TSL2561-Arduino-Library

4.4.4. Arduino com sensor MG811

O sensor MG811 permite medir os valores de dióxido de carbono no meio envolvente à sua localização. A informação para a sua configuração pode ser encontrada *online*[17].

4.5. Configuração do Raspberry Pi

O primeiro passo na configuração do Raspberry PI é a instalação do sistema operativo. Para este projeto irá ser utilizado o Windows 10 IoT Core [12] o que permitirá correr programas escritos em .NET.

A Microsoft disponibiliza um tutorial [13] e uma aplicação, *Windows IoT Core Dashboard* [14], para ajudar a efetuar o *download* do sistema operativo e posterior configuração do mesmo na placa Raspberry PI.

Os passos para a correta configuração da placa Raspberry Pi com o sistema operativo Windows 10 IoT Core são os seguintes (estes passos terão obrigatoriamente que ser feitos numa máquina com o Windows 10 instalado):

1. Descarregar a aplicação *Windows IoT Core Dashboard*.
2. Introduzir um cartão SD no leitor de cartões do computador, para onde irá ser descarregado o sistema operativo.
3. Abrir a aplicação e escolher a opção "*Set up a new device*" seguindo os passos de configuração até ao fim.
4. Retirar o cartão SD do computador e inseri-lo na placa Raspberry PI.
5. Ligar a placa à corrente, o que vai fazê-la iniciar o sistema operativo.

Depois de completar os passos anteriores é necessário ligar a placa à rede. A ligação vai ser feita por cabo e não por Wi-Fi pela razão de que, o modelo 2 da placa não tem módulo de Wi-Fi integrado. Depois de ligar o cabo de rede à placa e ao router, escolher a opção "*My devices*" na aplicação *Windows IoT Core Dashboard*. A placa deverá aparecer na lista e o utilizador poderá então visualizar a suas definições e alterá-las.

4.6. Desenvolvimento da Web API

O desenvolvimento da Web API é um ponto importante neste trabalho uma vez que esta irá ser utilizada tanto pelo *software* escrito para correr no Raspberry PI como pelo software que irá permitir ao utilizador consultar os dados recolhidos pelos sensores.

[17] http://sandboxelectronics.com/?p=147

Como referido anteriormente a tecnologia escolhida para o desenvolvimento desta API foi ASP.NET Web API [10].

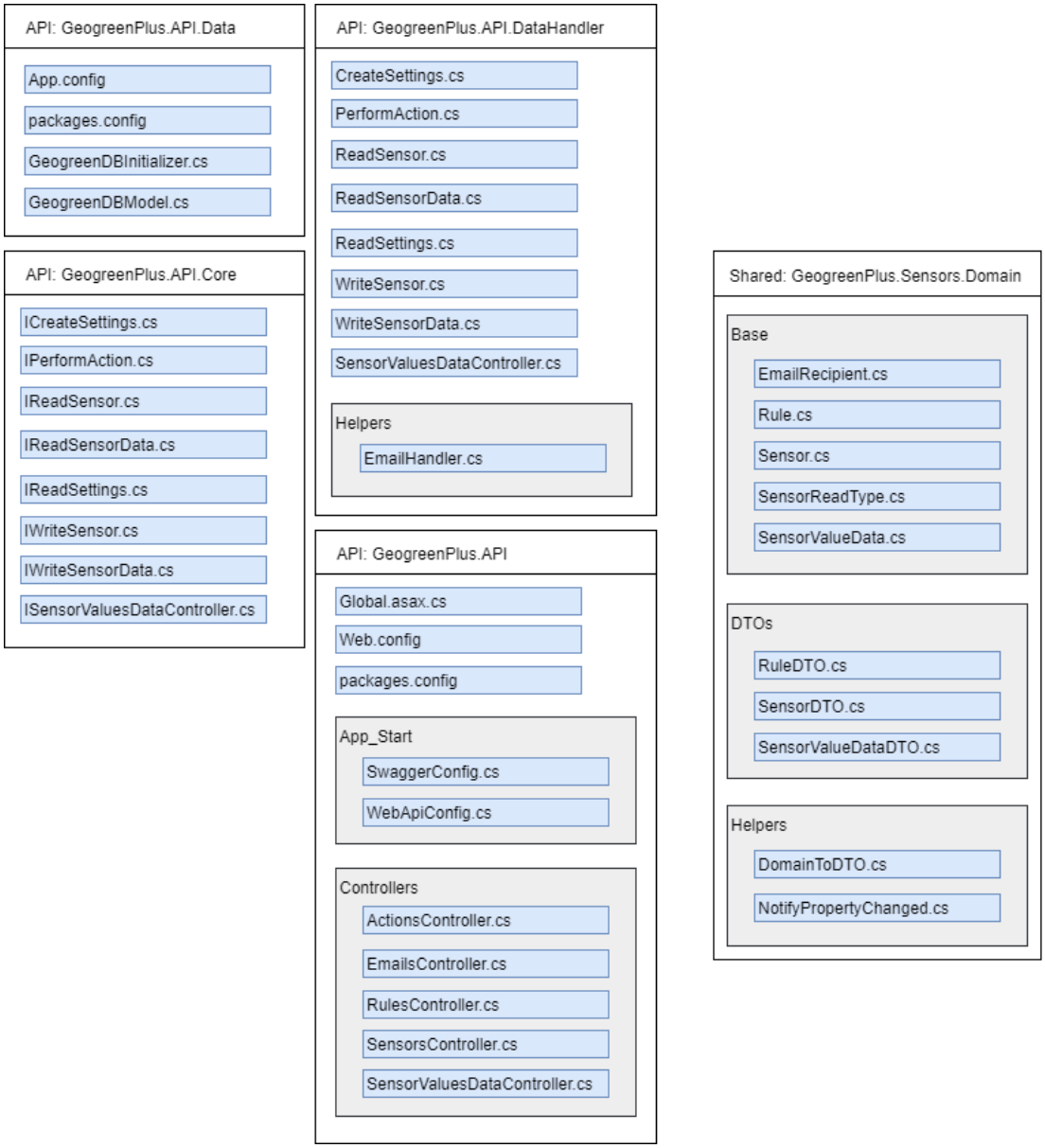

Figura 19 - Estrutura global dos projetos que compõem a Web API

Na Figura 19 podemos observar o conjunto de projetos criados para o desenvolvimento da Web API. Dos 5 blocos representados na imagem, 4 deles têm o título que começa com "*API:*" sendo estes específicos à API e o restante, com o título "*Shared: GeogreenPlus.Sensors.Domain*" é um projeto com código partilhado pelas três componentes da aplicação (Web API, aplicação *UWP* - Raspberry Pi e aplicação *WPF* - interface com o utilizador). Segue-se uma descrição de cada um dos projetos:

- **GeogreenPlus.Sensors.Domain**
  Este projeto contém as classes de domínio transversais a todas as aplicações da solução.
- **GeogreenPlus.API.Data**
  Este projeto tem como objetivo lidar com a criação da base de dados utilizando o Entity Framework. Além da criação é aqui definido o modelo da base de dados uma vez que iremos usar a opção *Code First*.
- **GeogreenPlus.API.Core**
  Este projeto apenas contém interfaces que deverão ser implementadas no projeto *Geogreen.Plus.API.DataHandler*. A razão deste projeto tem sobretudo a ver com uma perspetiva futura de escrita de testes unitários para os métodos da Web API.
- **GeogreenPlus.API.DataHandler**
  Este projeto contém as classes que vão permitir efetuar operações sobre a base de dados. Quando a Web API recebe um pedido HTTP para efetuar uma ação, o controlador que recebe o pedido não comunica diretamente com a base de dados, sendo essa comunicação efetuada na classe correspondente deste projeto. O objetivo desta implementação é a separação de responsabilidades do código e a facilidade em alterar blocos de código sem causar grandes impactos.
- **GeogreenPlus.API**
  Este projeto é a implementação da API em si e é este que irá lidar com os pedidos HTTP vindos do exterior. Os controladores presentes no projeto vão ser alvo de descrição aprofundada nos próximos parágrafos.

Para lidar com os pedidos HTTP é necessário criar controladores (*controllers*). Cada controlador terá métodos, correspondentes ao *endpoints* da API, que permitam receber pedidos HTTP (GET, PUT, POST, DELETE), executar a ação pretendida e devolver uma resposta.

Para as necessidades do protótipo GeogreenPlus foram criados no projeto *GeogreenPlus.API* quatro controladores diferentes:

- **SensorsController** (~/api/sensors)
  Este controlador tem os métodos que vão lidar com informação relacionada com os sensores:
  - Get() [HttpGet]: devolve a lista de todos os sensores existentes.
  - Get(long id) [HttpGet]: devolve o sensor correspondente ao identificador passado por parâmetro.
  - CreateSensor([FromBody]Sensor value) [HttpPost]: cria um novo sensor na BD e devolve o estado do sucesso da operação.
  - Put(long id, string sensorStatus) [HttpPut]: altera o valor do estado do sensor correspondente ao identificador passado por parâmetro.

- **SensorValuesDataController** (~/api/sensorvaluesdata)
  Este controlador contém os métodos que vão lidar com a os valores lidos pelos sensores:
    - GetSensorValues(long id, string readType, string startDate, string endDate) [HttpGet]: devolve uma lista de valores, lidos pelo sensor cujo identificador é passado por parâmetro, que estão dentro do intervalo de datas passadas por parâmetro.
    - GetSensorLastReadValue(long id, string readType) [HttpGet]: devolve o último valor lido pelo sensor cujo identificador e tipo de leitura são passados por parâmetro.
    - GetAvailableDates(long id, string readType) [HttpGet]: devolve todas as datas, para as quais existem dados disponíveis, relativas ao sensor cujo identificador e tipo de leitura são passados por parâmetro.
    - WriteSensorData(long id, [FromBody]SensorValueData value) [HttpPost]: adiciona à BD, o valor passado por parâmetro lido pelo sensor cujo identificador é também passado por parâmetro.
- **RulesController** (~/api/rules):
    - Get() [HttpGet]: devolve todas as regras existentes na BD.
    - GetById(long id) [HttpGet]: devolve a regra correspondente ao identificador passado por parâmetro.
    - CreateRule([FromBody]RuleDTO rule) [HttpPost]: cria uma nova regra na BD.
- **EmailsController** (~/api/emails):
    - Get() [HttpGet]: devolve todos os destinatários de *email* existentes na BD.
    - GetById(long id) [HttpGet]: devolve o destinatário de *email* cujo identificador corresponde ao valor passado por parâmetro.
    - CreateEmailRecipient([FromBody]EmailRecipient rule) [HttpPost]: cria um novo destinatário de email na BD.

Os métodos dos controladores acima listados permitem cumprir total ou parcialmente os requisitos funcionais apresentados em 3.3. A Tabela 3 mapeia os métodos dos controladores com o requisito funcional que estes permitem cumprir.

Tabela 3 - Relação entre Controlador da Web API e requisitos funcionais cumpridos

Controlador	Requisito Funcional
SensorsController	
Get()	RF1, RF6
SensorValuesDataController	

WriteSensorData(…)	RF5
GetSensorLastReadValue(…)	RF7 (parcialmente), RF8 (parcialmente)
GetSensorValues(…)	RF6
RulesController	
CreateRule(…)	RF9
Get()	RF10
EmailsController	
CreateEmailRecipient(…)	RF11
Get()	RF12

4.7. Desenvolvimento da aplicação de recolha de dados (Raspberry Pi)

A placa Raspberry Pi irá correr uma aplicação que será responsável por pedir às placas Arduino os valores lidos pelos sensores associados e enviá-los para serem guardados na base de dados. Com base na análise de requisitos efetuada no capítulo 3 mais propriamente nas seções 3.3 e 3.5.1 conseguimos definir um diagrama de fluxo para a aplicação que irá correr no Raspberry Pi.

A Figura 20 mostra esse diagrama e ajuda a perceber o fluxo de ações pretendido para a componente *hardware,* ou seja, a rede de sensores e placas de prototipagem, com as comunicações necessárias à API. A imagem mostra que o fluxo tem origem na aplicação que corre na placa Raspberry Pi. Uma vez iniciada, deverá ser feito um pedido à API para obter a lista de sensores disponíveis na rede. Depois de receber a resposta deverá se feita uma tentativa de ligação a cada placa Arduino ligada a cada um dos sensores. Caso a ligação seja bem sucedida, deverá ser criado um evento associado a um *timer*, que sempre que seja despoletado irá pedir o valor do sensor nesse momento e pedir à API que o guarde na BD. Deverá também nesse momento verificar se, baseado no valor lido, é necessário efetuar alguma ação automática no sistema e, em caso afirmativo, executar essa ação.

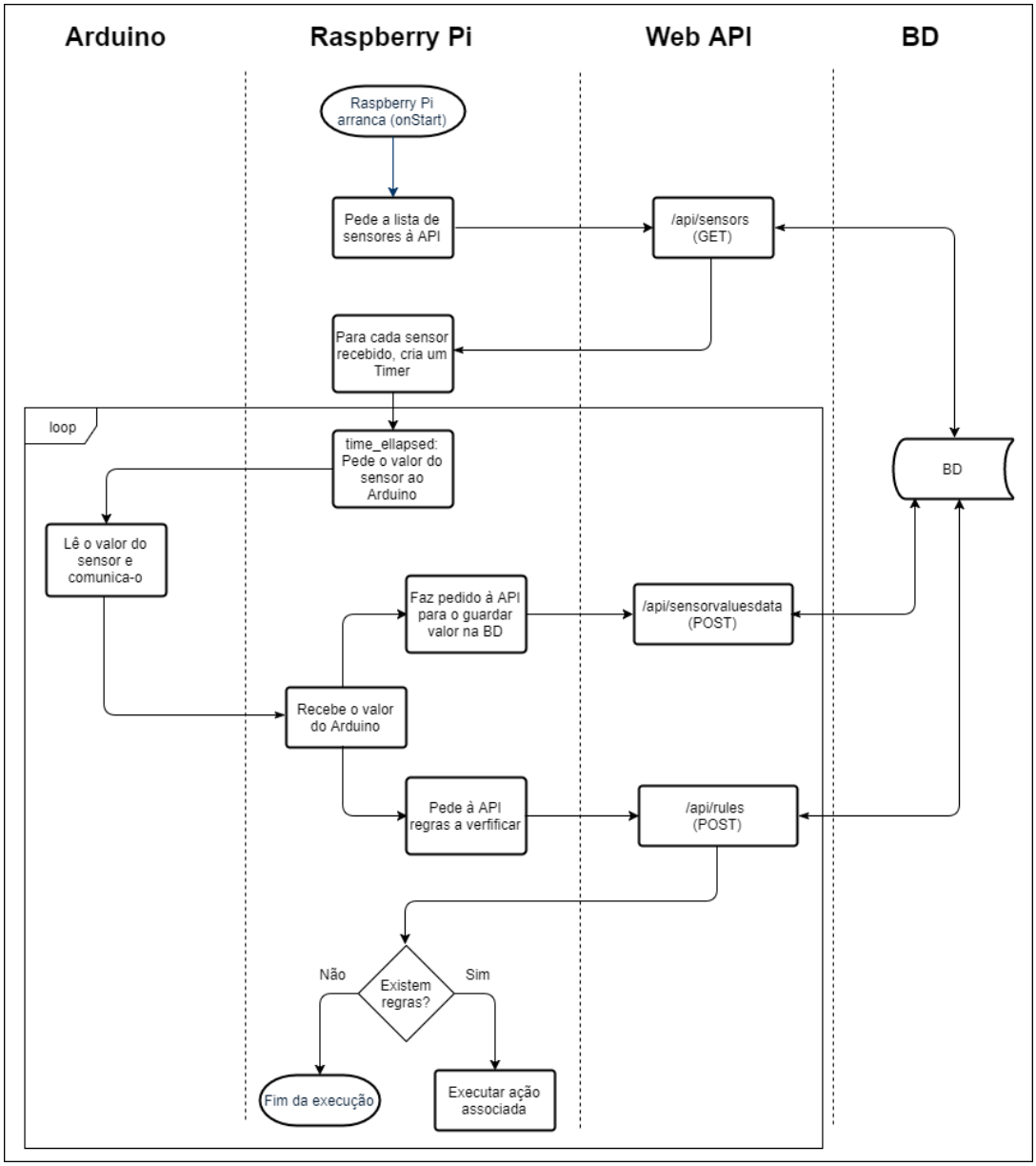

Figura 20 - Diagrama de fluxo do ciclo de execução ligado ao *hardware*

Para responder a estas necessidades foi desenvolvida uma aplicação do tipo "*Windows Universal - UWP*" para correr na(s) placa(s) Raspberry Pi que comunica com a base de dados utilizando a Web API desenvolvida (ver ponto 4.6).

Figura 21 - Estrutura da aplicação UWP

A Figura 21 mostra 2 módulos distintos que correspondem aos projetos da aplicação. O módulo "*UI: GeogreenPlus.Clients.UWPRaspberryClient*" representa o projeto *UWP* em si e o módulo "*Shared: GeogreenPlus.Sensors.Domain*", como já foi referido anteriormente, representa um projeto que contém classes de domínio transversais a todas as aplicações.

Nas imagens, cada retângulo cinzento representa uma das pastas da estrutura da aplicação desenvolvida. Cada pasta agrupa as classes (retângulos de tom azul) que contêm métodos para executar tarefas relacionadas diretamente com o nome dessa pasta.

Para desenvolver esta aplicação foi utilizado o padrão MVVM embora não haja nesta versão nenhuma interface com o utilizador (a que a parte *View* do padrão MVVM

estaria relacionada), tomou-se esta opção para que no futuro, caso haja necessidade, seja fácil de adicionar essa interface. Vamos analisar especificamente a estrutura do projeto *UWP*:

- **APIHandler**: esta pasta contém as classes cujos métodos permitem comunicar com a Web API.
- **Assets**: pasta criada automaticamente pelo *Visual Studio* no momento da criação do projeto e na qual não foi efetuada nenhuma alteração.
- **Helpers**: contém classes com funções ou constantes de apoio ao desenvolvimento de outros métodos.
- **Models**: esta pasta contém todas as classes que permitem interagir com cada um dos sensores utilizados no projeto.
- **ViewModels**: contém, no caso deste projeto, apenas uma classe que irá, no momento de início da aplicação no Raspberry PI, pedir a lista de sensores existentes na base de dados e fazer as configurações iniciais para dar início à leitura de dados.

Tabela 4 - Classes e respetivas funcionalidades da aplicação

Pasta - Classe	Funcionalidades
APIHandler	
GetSensorInfo.cs (cumpre requisito funcional RF1)	Esta classe contém métodos que permitem obter informação acerca dos sensores (lista de todos os sensores, sensor por identificador e por nome). Estes métodos fazem o pedido da informação pretendida à Web API, via a classe *HttpClient* nativa do .NET, e devolvem a informação recebida. Lida com os seguintes *endpoints* de tipo GET da API: • …/api/sensors • …/api/sensors/{sensorId} • …/api/sensors/{name}
PerformAction.cs (cumpre requisito funcional RF4)	Contém métodos que chamam o *endpoint* da Web API que permite efetuar ações cuja execução dependa da API. No caso concreto deste POC chama o *endpoint* "…/api/actions/{sensorId}/email" que irá enviar notificações por email aos utilizadores.
WriteSensorValue.cs (cumpre requisito funcional RF5)	Os métodos desta classe permitem chamar os *endpoints* de tipo POST, da Web API que são responsáveis por adicionar conteúdos à base de dados: • …/api/sensorvaluesdata/{sensorId}

ViewModels	
MainViewModel.cs	Quando a aplicação inicializa é executada uma instância da classe MainViewModel. Esta classe é responsável por chamar os métodos da API (APIHandler) que permitem recuperar a listagem de sensores existentes na BD (cumpre requisito funcional RF1). Para cada sensor dessa lista, vai tentar fazer uma comunicação com a placa Arduino correspondente e, caso tenha sucesso, subscrever um evento que irá ser despoletado a intervalos regulares (informação contida no sensor recebido) de forma a recuperar o valor lido pelo sensor.
Models	
ArduinoPi.cs (cumpre requisito funcional RF2)	Classe que representa virtualmente a placa Arduino. Haverá uma instância desta classe por cada placa Arduino existente no sistema. Esta classe é responsável por inicializar as comunicações entre o Raspberry Pi e as placas Arduino.
CO2SensorHandler.cs DHTSensorHandler.cs LuxSensorHandler.cs SoilMoistureSemsorHandler.cs (cumpre requisitos funcionais RF3 e RF4)	Como o nome das classes deixa a entender, cada uma delas será instanciada para representar um tipo de sensor diferente. É nestas classes que irá ser configurado o temporizador (*timer*) com o intervalo a que o valor deve ser lido e enviado para a BD. Além disso também é responsável por efetuar as ações que estiverem associadas ao sensor em questão.
Helpers	
Constants.cs	Esta classe, como o nome indica, contém valores constantes que podem ser usados em qualquer parte da aplicação.
NotifyPropertyChanged.cs	Implementa a interface *INotifyPropertyChanged* que permite notificar o Sistema quando as coleções são modificadas.

Podemos observar na Tabela 4 que a pasta "*Models*" contém, além da classe "*ArduinoPi.cs*", 4 outras classes que representam cada um dos tipos de sensores utilizados no protótipo. No futuro, caso seja necessário adicionar um (ou mais) novo sensor à rede deverá proceder-se da seguinte maneira:

1. Escrever o código específico do sensor novo sensor, utilizando como base o ficheiro "*StandardFirmata.ino*" utilizando o Arduino IDE.
2. Criar uma nova classe, com estrutura semelhante às atuais.
3. Adicionar o tipo do sensor ao enumerado "*SensorModel*" da classe "*Sensor.cs*".
4. No método "*Arduino_DeviceReady()*" da classe "*ArduinoPi.cs*" adicionar ao *switch* a opção para o novo tipo de sensor.
5. Adicionar o sensor/Arduino à base de dados.

Como vimos nos pontos anteriores, a única mudança no código previamente escrito será de adição e não de modificação, sendo assim a incorporação de novos sensores totalmente transparente cumprindo o requisito não funcional RF2.

4.8. Desenvolvimento da interface com o utilizador

Poder visualizar os dados recolhidos pelos sensores é um objetivo importante deste trabalho. Para isso foi desenvolvida uma aplicação, utilizando a tecnologia *Windows Presentation Foundation* (WPF), que permite ao utilizador consultar esses dados utilizando computadores ou *tablets* (com sistema operativo Windows e .NET Framework 3.0 ou superior instalados).

O desenvolvimento da aplicação respeitou as definições da fase de análise de requisitos descritos no capítulo 3 deste documento e a sua estrutura foi desenhada para ser o mais genérica possível para que desenvolvimentos futuros sejam integrados de maneira transparente.

Seguiu-se assim uma lógica de separação de conceitos, isto é, tentou organizar-se a estrutura do projeto em seções diferentes para que o código de cada seção seja responsável por um conceito distinto. A Figura 22 mostra essa separação, concretizada por diferentes pastas no *Visual Studio*, em que as classes inseridas em cada pasta tenham código diretamente relacionado com o nome desta. Na Figura 22, cada retângulo de cor cinzenta, representa uma pasta diferente e cada retângulo azul uma classe.

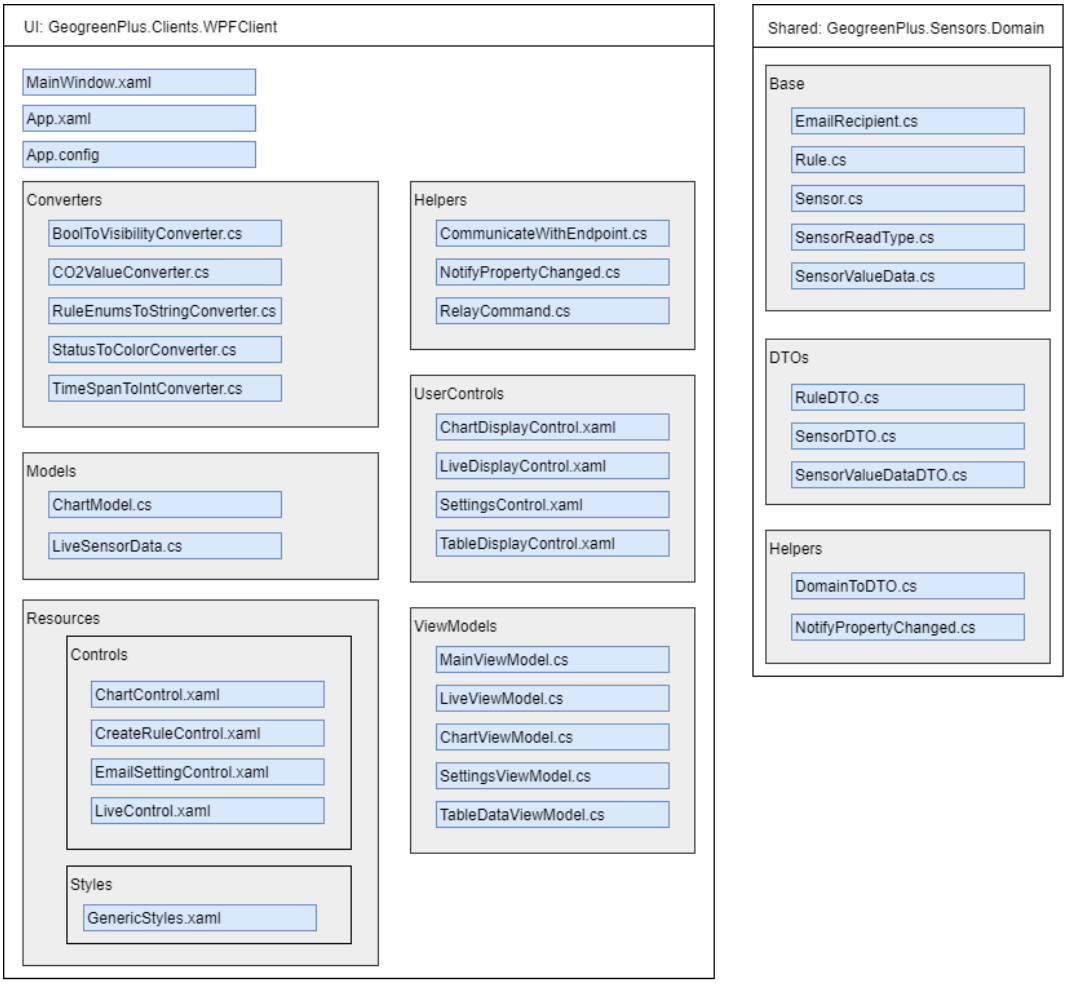

Figura 22 - Estrutura da aplicação WPF

A Figura 22 mostra a estrutura do projeto da interface gráfica. Na imagem podemos distinguir dois blocos principais, "*UI: GeogreenPlus.Clients.WPFClients*" e "*Shared: GeogreenPlus.Sensors.Domain*", que representam dois projetos diferentes na estrutura global da aplicação. O bloco "*Shared*" representa um projeto que contém partilhado pelas três componentes da aplicação (Web API, aplicação *UWP* - Raspberry Pi e aplicação WPF - interface com o utilizador). O bloco "*UI*" representa o projeto específico da interface com o utilizador e nele podemos distinguir os seguintes elementos:

- **Converters**: esta pasta contém todos os conversores necessários para o projeto. Estes conversores são classes que implementam a interface "*IValueConverter*" e que permitem fazer conversões de vários tipos, tendo em conta o tipo ou valor dos dados que recebem como parâmetro.
- **Helpers**: esta pasta contém classes com funções de apoio.

- **Models**: contém as classes que irão representar os objetos de domínio da aplicação.
- **Resources**:
    - **Controls**: pasta que contém controlos gráficos específicos e modulares que irão ser integrados nas "páginas" mais gerais. Por exemplo contém o controlo "*ChartControl.xaml*" que irá ser utilizado para mostrar ao utilizador os dados sobre a forma de gráfico na página de visualização dos dados nesse formato.
    - **Styles**: contém os estilos gerais que podem ser reutilizados por vários controlos WPF.
- **UserControls:** contém os controlos gráficos que serão as páginas da aplicação.
- **ViewModels:** esta pasta contém as classes que implementam a lógica da aplicação. Por cada classe na pasta "*UserControls*" existe uma classe correspondente nesta pasta.
- Podemos ainda constatar que 3 classes (*MainWindow.xaml*, *App.xaml* e *App.config*) não aparecem associadas a qualquer pasta. Isso deve-se a estarem situadas na pasta raiz do projeto.

A seguinte tabela apresenta uma breve descrição das funcionalidades das diferentes classes da aplicação. Mais uma vez se refere que foi seguida uma lógica de separação de conceitos no desenvolvimento da aplicação tentando criar código fácil de manter e de substituir.

Tabela 5 - Classes e respetivas funcionalidades da aplicação WPF

Pasta - Classe	Funcionalidades
Raiz do projeto	
MainWindow.xaml	A classe *MainWindow.xaml* é do tipo *Window* e representa o *container* global da interface gráfica que vai alojar todas as páginas da aplicação.
App.xaml	Contém os recursos, ou referência a recursos, partilhados pelos elementos visuais da aplicação.
App.config	Ficheiro de configuração. É neste ficheiro que está definida a cadeia de ligação à Web API.
Converters	
BoolToVisibilityConverter.cs CO2ValueConverter.cs	O *WPF* possibilita a escrita de conversores quando o tipo dos dados de origem são diferentes do tipo dos dados de destino. Por exemplo, converter um número para uma cor, ou um booleano para a propriedade *Visibility* de um controlo, ou seja a

RuleEnumsToStringConverter.cs StatusToColorConverter.cs TimeSpanToIntConverter.cs	partir de um booleano esconder ou mostrar automaticamente um controlo. A pasta *Converters* contêm conversores para diferentes tipos de dados.
Models	
ChartModel.cs LiveSensorData.cs	Estas classes representam virtualmente os objetos de domínio da aplicação: • *ChartModel.cs* permite construir os objetos necessários para construir os gráficos. • *LiveSensorData.cs* permite construir os objetos para a página *Live*.
Helpers	
CommunicateWithEndpoint.cs	Esta classe contém os métodos necessários para comunicar com a Web API.
NotifyPropertyChanged.cs	Implementa a interface *INotifyPropertyChanged* que permite notificar o sistema quando as coleções são modificadas e assim atualizar automaticamente a interface gráfica.
RelayCommand.cs	Implementa a interface *ICommand* que permite utilizar comandos *WPF*.
UserControls	
ChartDisplayControl.xaml (cumpre RF7) LiveDisplayControl.xaml (cumpre RF6) SettingsControl.xaml (cumpre RF9, RF10, RF11 e RF12) TableDisplayControl.xaml (cumpre RF8)	A pasta *UserControls*, como o nome indica, contém os controlos gráficos do tipo *UserControl*, que correspondem às páginas disponíveis na aplicação.
Resources	
Controls: ChartControl.xaml CreateRuleControl.xaml EmailSettingControl.xaml	A pasta *Controls* contém controlos gráficos que irão ser utilizados por algumas páginas da pasta *UserControls*. A vantagem de definir controladores específicos, não acoplados diretamente à página "mãe" é o fato de se puder trocar os controlos da página de forma transparente sem causar nenhum conflito com a estrutura existente.

LiveControl.xaml	
Styles: GenericStyles.xaml	A classe *GenericStyles.xaml* contém a definição de estilos partilhados por diversos, componentes do mesmo tipo, em toda a aplicação. Este ficheiro deverá ser referenciado na classe *App.xaml* tornando assim a sua utilização possível por qualquer elemento gráfico da aplicação.
ViewModels	
MainViewModel.cs	Uma instância da classe *MainViewModel.cs* será o contexto global da aplicação (e especificamente da janela *MainWindow.xaml*) e o seu ponto de entrada. Esta classe contém uma propriedade para cada uma das classes da linha seguinte desta tabela que serão utilizadas para definir o contexto de cada uma das 4 páginas existentes na interface gráfica.
LiveViewModel.cs ChartViewModel.cs SettingsViewModel.cs TableDataViewModel.cs	Cada uma destas classes será o contexto de cada uma das páginas da interface gráfica (classes da pasta *UserControls*). Quando uma das páginas é mostrada ao utilizador, o seu contexto, ou seja, o seu conteúdo e comportamento, provém da classe *ViewModel* associada.

4.9. Conclusão

Neste capítulo começou-se, em 4.2, por se descrever o processo de escolha do material e tecnologias usadas para o desenvolvimento do protótipo. Essa escolha levou a reter placas Arduino Uno às quais serão ligados diretamente os sensores, depois essas placas serão ligadas a placas Raspberry Pi que terão a missão de comunicar com a Web API. O ambiente tecnológico para a programação foi o .NET, mais concretamente, C#, UWP, WPF e ASP.NET Web API. A base de dados será SQL Server e será utilizado Entity Framework para criar a BD e comunicar com ela.

Em 4.3 foi apresentada a arquitetura do sistema e em 4.4 as configurações do material físico, ou seja, os sensores e Arduinos e em 4.5 as placas Raspberry Pi.

Na seção 4.6 está detalhado o desenvolvimento da Web API, para o qual foi utilizada a tecnologia ASP.NET Web API, listando cada *endpoint* desenvolvido bem como a suas funcionalidades. Em 4.7 descreve-se o desenvolvimento da aplicação que irá correr na(s) placa(s) Raspberry Pi e para o qual foi utilizada a tecnologia UWP. Depois em 4.8 está descrito o processo de desenvolvimento da aplicação de interface com o utilizador, para o qual foi utilizada a tecnologia WPF.

Estando a fase de desenvolvimento concluída, o próximo capítulo irá incidir sobre os passos necessários para pôr o sistema em funcionamento.

5. Sistema em funcionamento

5.1. Introdução

Este capítulo incide sobre os passos de configuração e instalação do protótipo (5.2). Irão ser descritos os processos para a configuração do sistema utilizando uma arquitetura tradicional cliente-servidor (5.2.1) e utilizando uma arquitetura de nuvem (5.2.2).

5.2. Configuração e instalação do protótipo

Como visto anteriormente este sistema é constituído por diferentes componentes de *hardware* e de *software* sendo que o protótipo final deveria idealmente utilizar uma arquitetura de nuvem. Este ponto foi cumprido, no entanto, decidiu-se, como visto anteriormente também, utilizar também uma arquitetura tradicional cliente-servidor por forma a demonstrar a portabilidade e modularidade do sistema (e assim cumprir o requisito não funcional RNF1).

Sendo assim existem componentes que irão ser os mesmos em qualquer uma das arquiteturas e outros que irão ser específicos. Seguidamente listam-se aqueles que são comuns, em 5.2.1 serão apresentados os específicos à arquitetura cliente-servidor e em 5.2.2 os específicos à arquitetura de nuvem.

Os componentes comuns aos dois tipos de arquitetura são os seguintes:

- Duas placas modulares GEOGREEN (uma será monitorizada outra não)
- 4 Placas Arduino
- 2 Placas Raspberry PI
- 1 Sensor de humidade e temperatura do ar modelo DHT22
- 1 Sensor de CO2 modelo MG811
- 1 Sensor de humidade do solo da empresa *iTead Studios*
- 1 Sensor de luminosidade modelo TSL2561
- Um computador pessoal ou *tablet* com uma versão do sistema operativo Windows instalado.

Os módulos de *software* necessários para o sistema funcionar devem ser instalados nas respetivas "peças" de hardware. A seguinte tabela faz um mapeamento entre o componente *hardware* e o respetivo *software* que lhe está associado.

Tabela 6 - Correspondência *hardware - software*

Hardware	Software
Arduino com sensor DHT22	Arduino\DHT_22\StandardFirmata
Arduino com sensor CO2	Arduino\CO2_MG811\StandardFirmata
Arduino com sensor humidade solo	Arduino\SoilMoisture\StandardFirmata
Arduino com sensor luminosidade	Arduino\Lux_TSL2561\StandardFirmata
Raspberry PI	Clients\GeogreenPlus.Clients.UWPRaspberryClient
Microsoft Azure	API\GeogreenPlus.API (Nota: a base de dados estará igualmente alojada no Microsoft Azure e será gerada automaticamente)
Máquina virtual Windows Server	API\GeogreenPlus.API (Nota: a base de dados está alojada nesta máquina e será gerada automaticamente)
Computador pessoal/*tablet*	Clients\GeogreenPlus.Clients.WPFClient

(NOTA: as linhas com fundo azul representam as duas arquiteturas que foram implementadas, nuvem e cliente-servidor, e não serão utilizadas em paralelo.)

Cada um dos sensores foi ligado a uma placa Arduino independente e cada placa Arduino foi depois ligada, via cabo USB, a uma porta USB de uma das placas Raspberry PI. Esta foram por sua vez ligadas a um *router* via cabo de rede RJ45.

Os sensores foram dispostos junto às placas modulares GEOGREEN sendo que o sensor de humidade do solo foi inserido na terra dentro de uma das cavidadas da placa modular. A Figura 23 mostra a placa modular GEOGREEN juntamente com a rede de sensores e placas de prototipagem utilizadas no protótipo.

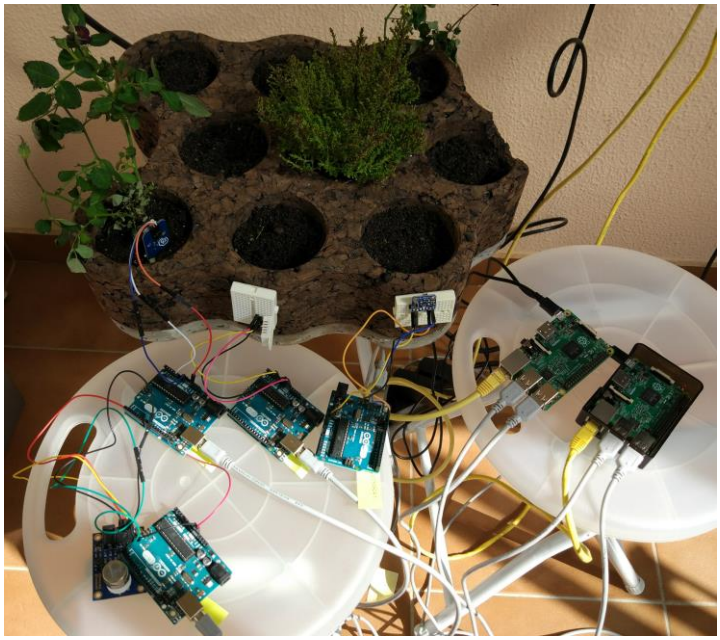

Figura 23 - Protótipo GeogreenPlus e placa modular GEOGREEN

Quando a placa Raspberry PI é ligada à corrente, o sistema operativo arranca e uma vez inicializado, a aplicação "GeogreenPlus.Clients.UWPRaspberryClient" começa a recolha de dados lidos pelos sensores.

A partir deste momento o utilizador poderá, através da interface gráfica "GeogreenPlus.Clients.WPFClient", consultar os dados lidos e guardados na base de dados.

5.2.1. Arquitetura cliente-servidor

Para o caso da arquitetura cliente-servidor, necessitou-se obviamente uma máquina para ser o "servidor", a qual irá alojar a Web API e a base de dados. Essa máquina está situada fisicamente no departamento de Engenharia Informática da Universidade da Beira Interior e tem as seguintes características:

- *Hardware e sistema operativo*:
  - Processador Intel Xeon CPU E7-4830 @ 2.13 GHz
  - Sistema operativo Windows Server 2012
  - 8 GB de RAM
- *Software*:
  - SQL Server 2016

- SQL Server 2016 Management Studio
- Internet Information Services 8 (IIS8)

Depois da instalação do SQL Server, do Management Studio e IIS8 é necessário publicar a Web API para que esta seja acessível a partir do Raspberry Pi e da interface com o utilizador. Para tal é preciso criar um pacote de publicação no *Visual Studio* da máquina de desenvolvimento, da seguinte maneira:

- Escolher o projeto "GeogreenPlus.API", clicar com o botão direito do rato e escolher a opção "*Publish*".
- Na janela que se abre escolher a opção "*Custom*" e introduzir um nome para o pacote de publicação.
- Na guia "*Connection*" escolher a opção "*File System*" para "*Publish method*". Escolher ainda a pasta de destino onde o pacote de publicação vai ser gerado.
- Carregar no botão "*Publish*" para acabar o processo.

Quando o pacote de publicação estiver gerado terá de ser copiado para a o servidor utilizando, por exemplo, o *Remote Desktop*. Uma vez copiado para o servidor terá de se criar uma aplicação no IIS8 que irá alojar a API e permitir que esta esteja acessível para fora.

Em relação à base de dados, como foi já referido, será o Entity Framework que lidará com a sua criação e isso acontecerá da primeira vez que qualquer um dos *endpoints* disponibilizados pela API seja acedido. Basta para tal que no ficheiro "*App.config*" incluído no pacote de publicação, esteja definida uma cadeia de ligação para a instância do SQL Server instalada no servidor.

5.2.2. Arquitetura de nuvem

A tecnologia de nuvem escolhida foi Microsoft Azure[18] tendo por base as justificações apresentadas na seção 4.2 e juntando a isso a possibilidade de utilizar uma conta livre, sem qualquer custo financeiro, devido aos protocolos que a Microsoft tem com múltiplas instituições de ensino superior do mundo inteiro.

Como foi referido no ponto 3.2, a utilização deste tipo de arquitetura tem vantagens em relação às arquiteturas tradicionais cliente-servidor. Algumas delas permitem cumprir parte dos requisitos não funcionais apresentados na seção 3.4. Com efeito,

[18] https://azure.microsoft.com/en-us/?v=17.14

para o serviço de nuvem Azure, a Microsoft garante[19] o acesso aos seus serviços 99.9% do tempo (RNF4) assim como no mínimo 99.9% de disponibilidade da funcionalidade de *backup* (RNF5) e restauro dos dados. Para além disto, o utilizador poderá pedir alterações à configuração do sistema (adicionar/remover memória, adicionar/remover capacidade de armazenamento, …) sempre que haja necessidade para tal (RNF3).

Seguidamente detalham-se os passos necessários para configurar os serviços Azure necessários para o bom funcionamento do protótipo. No futuro, para um protótipo industrial, será necessária uma conta de Azure paga, no entanto os passos para a configuração dos serviços serão idênticos.

O primeiro passo é aceder ao portal[20] que a Microsoft disponibiliza para a configuração dos serviços Azure. Para as necessidades do protótipo GeogreenPlus deverá ser configurado um serviço de base de dados (*SQL databases*) e um serviço para alojar a Web API (*App Services*).

O primeiro passo é configurar o serviço que irá permitir criar e utilizar uma base de dados do tipo SQL Server. Para isso deverão ser efetuados os passos seguintes na ordem apresentada (ver Figura 24, os números na figura correspondem ás entradas da lista seguinte):

1. Escolher "*SQL databases*" no painel esquerdo.
2. Escolher "*Add*" no painel que aparece ao meio.
3. Escolher um nome para a base de dados.
4. Para se ter uma base de dados disponível, deverá ter-se um servidor onde ela estará alojada. O primeiro passo para isso será criar um "*Resource group*".
5. Segue-se o pedido de criação do servidor escolhendo a opção "*Create a new server*". A criação e gestão desse servidor irá ser efetuada internamente pelo Azure.
6. Para o tipo de base de dados a criar escolher a opção "*Blank database*".
7. Finalmente carregar no botão "*Create*" e esperar pela notificação de que o serviço foi criado.

[19] https://azure.microsoft.com/en-us/support/legal/sla/summary/

[20] https://portal.azure.com/

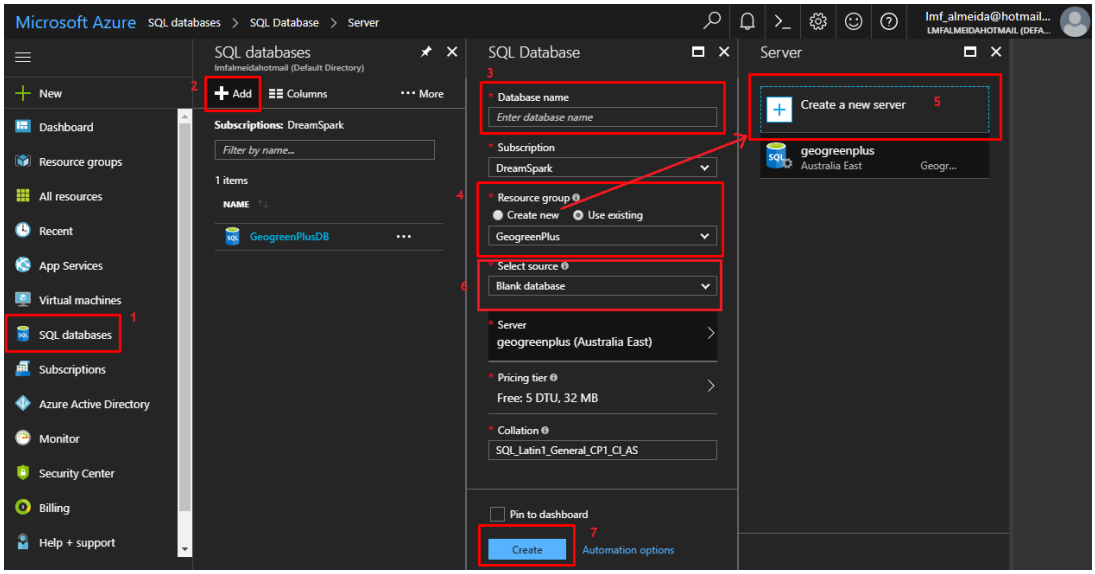

Figura 24 - Configuração do serviço de base de dados no Azure

O segundo passo é criar um serviço que irá alojar a Web API. Para tal deverão ser realizadas as seguintes ações, na ordem indicada (ver Figura 25, os números na figura correspondem ás entradas da lista seguinte):

1. Escolher a opção "*App Services*" no painel esquerdo.
2. Escolher "*Add*" no painel que aparece ao meio.
3. Escolher "*Web App*" na opção do serviço.
4. Carregar no botão "*Create*".
5. Introduzir um nome para a aplicação.
6. Selecionar o grupo de recursos criado quando se configurou o serviço de base de dados.
7. Finalmente carregar no botão "*Create*" e esperar pela notificação de que o serviço foi criado.

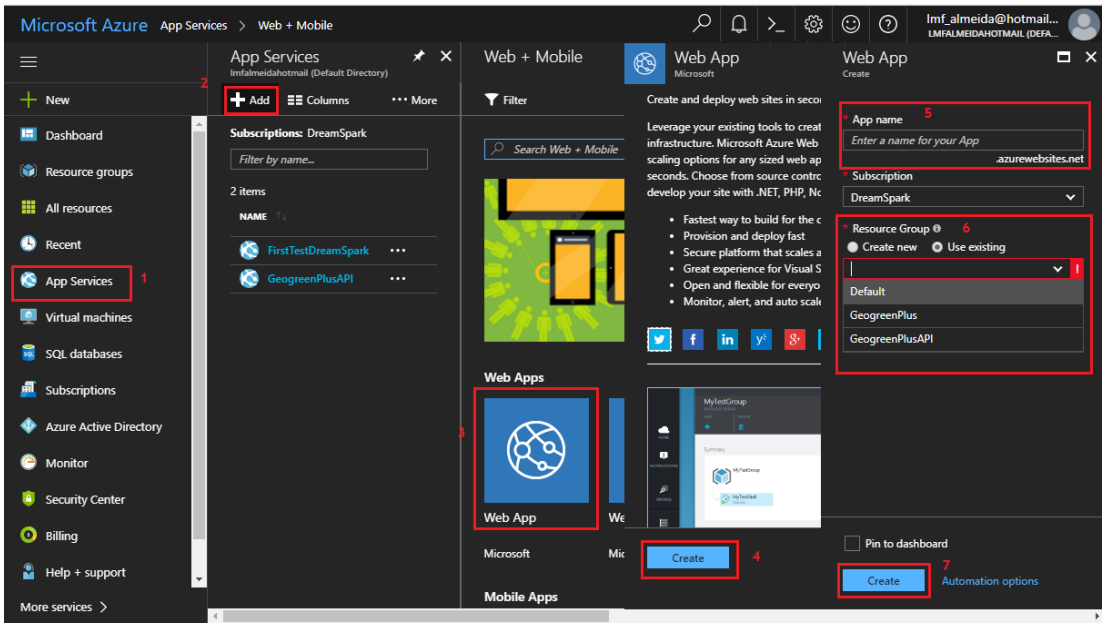

Figura 25 - Criar aplicação para alojar a Web API no Azure

Uma vez os serviços criados devem configurar-se para o caso específico do GeogreenPlus. Como referido anteriormente a base de dados será gerada automaticamente recorrendo ao Entity Framework pelo que o serviço de base de dados não necessita de nenhuma configuração extra, apenas de saber o seu endereço na nuvem para que a Web API possa comunicar com ele. Na primeira chamada feita através de um método da Web API, irá ser verificado se a BD existe, se sim o método é executado, se não a BD é criada primeiro e o método executado depois.

Em relação ao serviço que irá alojar a Web API é necessário recuperar o ficheiro de publicação específico e depois utilizá-lo para efetuar a publicação do serviço, diretamente a partir do *Visual Studio*. Para tal, no painel (*Dashboard*) que lista os serviços e aplicações existente no Azure, escolher a aplicação criada para alojar a API e depois escolher a opção "*Get publish profile*". Esta opção permite descarregar o ficheiro para o computador pessoal.

Uma vez o ficheiro descarregado é necessário publicar a Web API no Azure, através do *Visual Studio*. Para isso selecionar o projeto "GeogreenPlus.API", clicar com o botão direito do rato e escolher a opção "*Publish*". Na janela que se abre deve escolher-se a opção "*Import*" que irá permitir carregar o ficheiro de publicação previamente descarregado do Azure. De seguida deve carregar-se no botão "*Validate Connection*" para testar a ligação e, em caso, de sucesso carregar em "*Next*". Neste passo deve introduzir-se a cadeia de ligação para a base de dados no Azure. Carregar no botão "*Publish*" para terminar o processo de publicação.

5.3. Validação da instalação

Uma vez a fase de configuração terminada é necessário verificar que a Web API está acessível e adicionar à base de dados a informação dos sensores/Arduinos presentes na sistema. Para verificarmos se a API está acessível basta tentar aceder ao Swagger associado à Web API, bastando para tal abrir uma janela de um qualquer explorador de internet e escrever o caminho para o servidor/azure onde está alojada a API seguido de "/swagger". Se se obter uma página com a listagem dos controladores (*endpoints*) presentes na API é porque esta está disponível e funcional. No presente caso temos dois casos a testar que são:

- Cliente-servidor: http://193.136.66.21:1080/swagger
- Azure: http://geogreenplusapi.azurewebsites.net/swagger

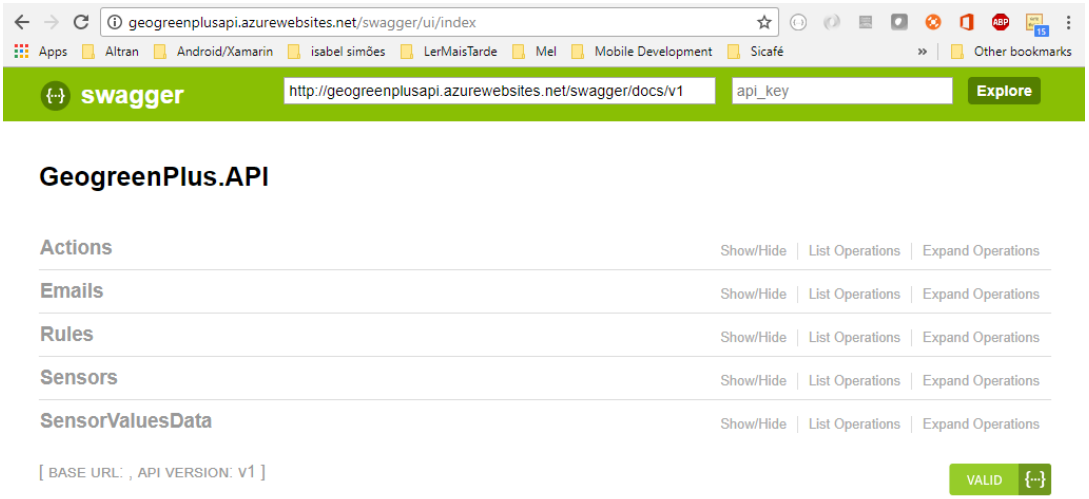

Figura 26 - Swagger associado à Web API no Azure

Depois de verificar a disponibilidade da Web API é preciso adicionar os sensores/Arduinos do sistema à base de dados. Para tal irá também ser usado o Swagger utilizando o *endpoint* de tipo POST "api/sensors", bastando para isso selecionar o controlador "Sensors" e depois o *endpoint* referido. No campo de texto "*value*" deverá ser introduzido o texto no formato JSON, de cada uma das linhas da Tabela 7, e carregar no botão "*Try it out!*".

Tabela 7 - Tabela com o JSON necessário para criar cada sensor/Arduino na BD

Sensor/Arduino	JSON
Arduino com sensor de temperatura e humidade do ar DHT22	{ "SensorTypes": [{ "SensorType": 0 }, { "SensorType": 1 }], "ArduinoId": "75439323735351909 1B1", "Name": "DHT22 Thesis", "Model": 0, "Status": 0, "ReadTimeInterval": "01:00:00" }
Arduino com sensor de CO2	{ "SensorTypes": [{ "SensorType": 3 }], "ArduinoId": "75431343334351E04021", "Name": "Outside CO2 Thesis", "Model": 2, "Status": 0, "ReadTimeInterval": "01:00:00" }
Arduino com sensor de humidade solo	{ "SensorTypes": [{ "SensorType": 2 }], "ArduinoId": "75431343334351E061E1", "Name": "Soil Moisture Thesis", "Model": 1, "Status": 0, "ReadTimeInterval": "01:00:00" }
Arduino com sensor de luminosidade	{ "SensorTypes": ["SensorType": 4 }], "ArduinoId": "75439323735351907071", "Name": "Luminosity Thesis", "Model": 3, "Status": 0, "ReadTimeInterval": "01:00:00" }

Depois de adicionar os sensores/Arduinos da solução à base de dados podemos ligar as placas Raspberry Pi à corrente e aceder ao "*Device Portal*" de cada uma das placas para executar a aplicação "GeogreenPlus.Clients.UWPRaspberryClient". Esta irá de imediato ligar-se à Web API e começar a pedir dados aos Arduinos. De notar que é preciso aceder ao "*Device Portal*" de cada placa apenas porque esta é a primeira ligação, depois deverá adicionar-se a aplicação UWP ao arranque automático da placa

e no futuro, mesmo que hajas falhas de energia elétrica, quando as placas reiniciarem, a aplicação irá iniciar automaticamente. Para aceder ao "*Device Portal*" deverá iniciar-se a aplicação "Windows 10 IoT Core Dashboard" (ver Figura 27) num computador que esteja na mesma rede das placas Raspberry Pi e, em cada uma das duas placas listadas, clicar com o botão direito do rato na opção "*Open in Device Portal*". Isto irá abrir numa janela do navegador internet padrão do utilizador, o portal de configuração da placa como ilustrado na Figura 28.

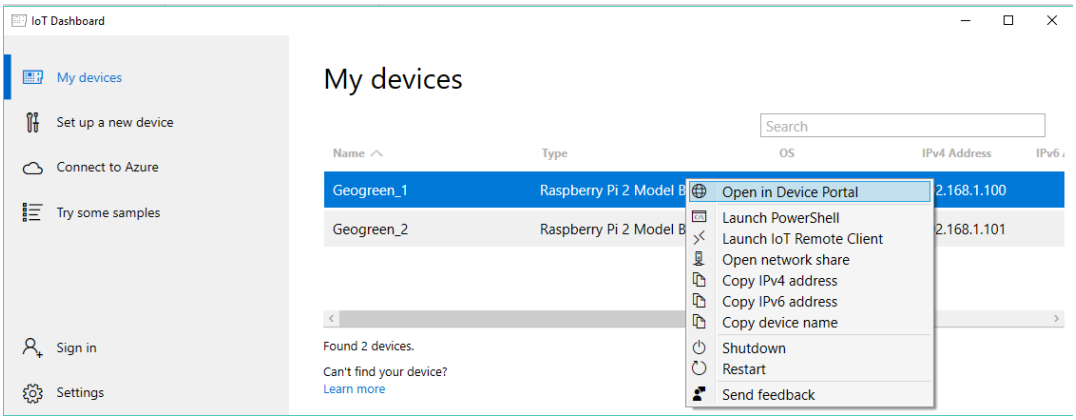

Figura 27 - Windows 10 IoT Core Dashboard

Nesse portal deverá escolher-se a opção "*Apps -> Apps manager*" no menu da esquerda e procurar-se na lista apresentada a aplicação "GeogreenPlus.Clients.UWPRaspberryClient". Uma vez localizada deverá selecionar-se a opção "*Startup*" e a ação "*Start*", o que fará a aplicação arrancar.

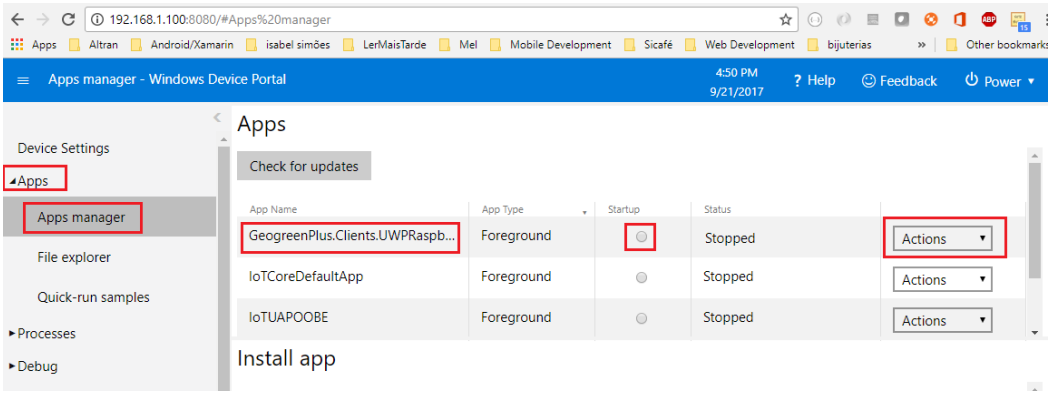

Figura 28 - "*Device Portal*": portal de configuração das placas Raspberry Pi

Para finalizar a validação da instalação do sistema e verificar o seu correto funcionamento, vamos inicializar a aplicação de interface com o utilizador. É de

esperar que apareçam listados os sensores adicionados previamente à base de dados e que já estejam disponíveis os primeiros valores lidos para cada um dos sensores (no Apêndice A deste documento estará disponível uma visita guiada à aplicação, detalhando cada "página" e os passos para a sua utilização). A Figura 29 mostra a página "Live" da interface gráfica e nela podemos ver a listagem dos sensores adicionados através do Swagger, o seu estado, o último valor lido pelos sensores e respetiva data de leitura. Pode, portanto, concluir-se que o protótipo foi instalado com sucesso.

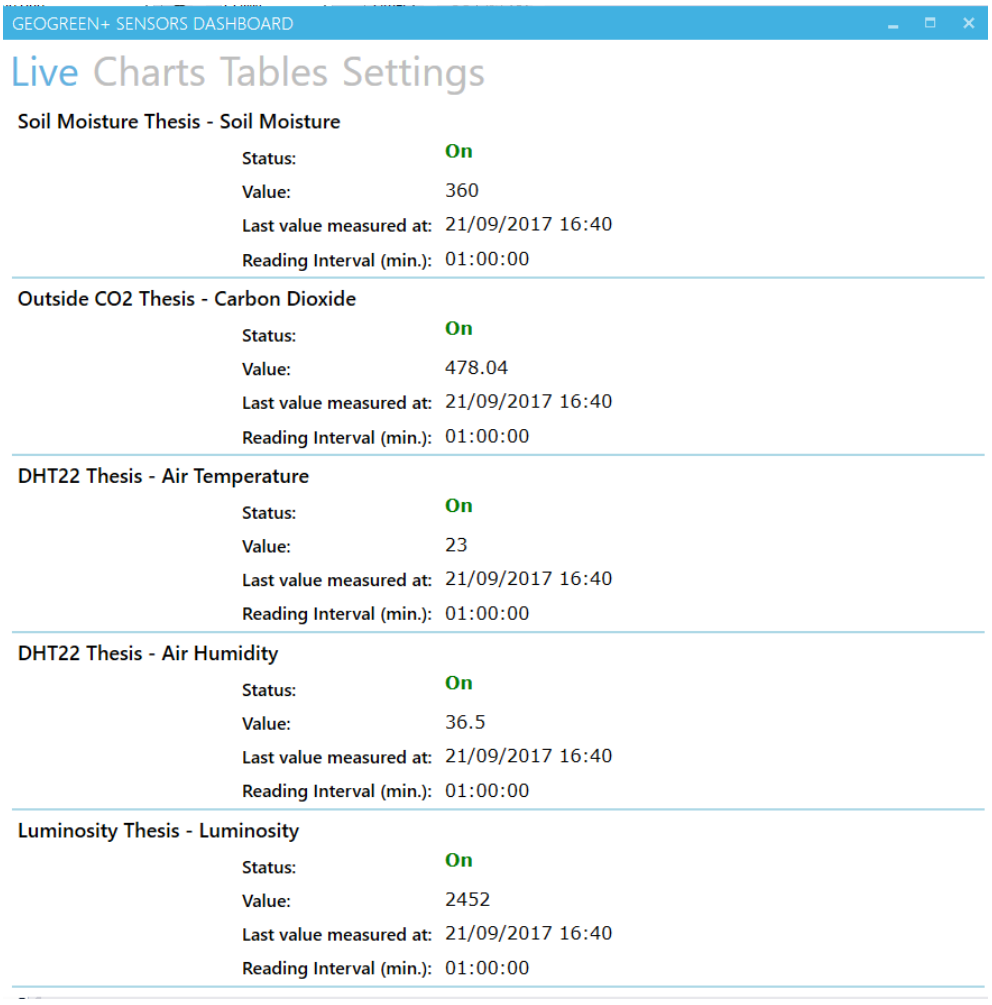

Figura 29 - Página "Live" da interface com o utilizador GeogreenPlus

5.4. Conclusão

Neste capítulo foram apresentados os passos necessários para a correta instalação e configuração do sistema GeogreenPlus. Foram apresentados dois modelos de arquitetura do sistema, em 5.2.1 a configuração específica para a arquitetura tradicional cliente-servidor e em 5.2.2 a configuração para a arquitetura de nuvem

utilizando a solução Microsoft Azure. Em 5.3 validou-se o sucesso dessa instalação, começando primeiro por testar o acesso à Web API, adicionando depois os dados relativos aos sensores/Arduinos do sistema à BD através do Swagger e acedendo aos dados através da aplicação de interface com o utilizador.

Após o sucesso da instalação do sistema pôde avançar-se para a validação experimental do protótipo desenvolvido. O capítulo 6 detalha os casos de teste efetuados e os resultados obtidos durante a fase de validação do sistema.

6. Validação experimental

6.1. Introdução

Foi efetuada uma validação experimental do sistema GeogreenPlus em dois contextos diferentes que serão explicados mais detalhadamente nos pontos 6.2 e 6.3. O primeiro, em 6.2, foi na validação prática das qualidades do material modificado dos módulos GEOGREEN, no âmbito da tese de mestrado intitulada "Comportamento térmico do sistema modular GEOGREEN, com incorporação de PCMs" [7] realizada no CMADE. Esta tese teve como objetivo principal estudar o comportamento térmico do sistema modular GEOGREEN melhorado com materiais de mudança de fase (PCM).

O segundo caso de teste, descrito em 6.3 focou-se mais no estudo da melhoria do sistema GEOGREEN como um todo, isto é, acoplando uma rede de sensores às placas modulares GEOGREEN obtendo assim dados sobre o meio envolvente, permitindo realizar ações cirúrgicas que levaram, por exemplo, à poupança de água na rega das plantas semeadas nas cavidades das placas modulares.

6.2. Caso 1 - Validação experimental das qualidades do material modificado

No âmbito da tese de mestrado "Comportamento térmico do sistema modular GEOGREEN, com incorporação de PCMs" [7] realizada pelo aluno Roberto Gama no CMADE foram criadas placas modulares com características diferentes das produzidas no âmbito do projeto GEOGREEN original [2]. Nesta tese pretendeu-se mostrar que a inclusão de materiais de mudança de fase na estrutura das placas iria melhorar o seu comportamento térmico.

De maneira a demonstrar que a introdução destes materiais melhora efetivamente o comportamento térmico das placas foi preciso fazer medições precisas desses valores. Para esse efeito foi usado o sistema GeogreenPlus.

6.2.1. Cenário de teste

Os testes foram realizados utilizando placas modulares GEOGREEN originais, placas modulares GEOGREEN com materiais de mudança de fase (PCM), vários sensores de temperatura e humidade do ar e um sensor de luminosidade.

Numa câmara de ensaios com dois compartimentos independentes entres si, com as mesmas dimensões e características térmicas, foram "construídas" duas paredes sobre as fachadas dos compartimentos, cada qual com o seu tipo de placas modulares. No exterior da câmara de ensaios foram colocados, um sensor de luminosidade (TSL2561) e um sensor de temperatura e humidades do ar (DHT-22). No interior de cada compartimento foi instalado um sensor de temperatura e humidade do ar.

Figura 30 - Vista exterior da câmara com os dois tipos de placas GEOGREEN[21]

O sistema GeogreenPlus foi responsável pela leitura e armazenamento dos dados provenientes destes sensores que foram recolhidos todos os 30 minutos. O teste foi realizado em duas fases, primeiro foram lidos os valores da temperatura e humidade dor ar nos 2 compartimentos com as paredes "nuas", isto é, sem a aplicação dos módulos GEOGREEN. Posteriormente a parede A foi coberta com módulos GEOGREEN clássicos e a parede B com os módulos GEOGREEN com PCM.

[21] Todas as figuras desta seção (6.2) foram retiradas da tese de mestrado do Roberto Gama [7]

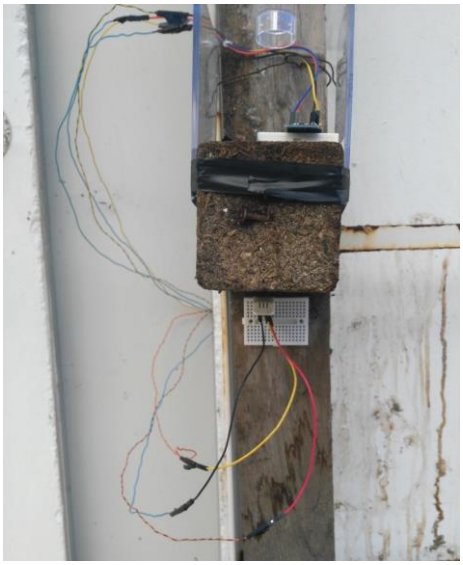

Figura 31 - Sensor de luminosidade (em cima) e sensor de temperatura e humidade do ar (em baixo)

6.2.2. Resultados do teste

NOTA: para uma consulta pormenorizada dos resultados do teste, consultar os capítulos 4.3 e 5 da tese de mestrado do Roberto Gama [7].

A análise dos dados recolhidos pelo GeogreenPlus permitiu concluir que para os mesmos períodos do dia, com condições de ambiente iguais, os valores da temperatura ambiente dos dois compartimentos independentes da câmara de ensaios eram diferentes. Verificou-se que a temperatura do compartimento com a parede revestida pelos módulos com o material de mudança de fase eram, mais baixos durante o período diurno e mais altos no período noturno, se comparados com os valores obtidos no compartimento revestido com os módulos originais.

Na primeira semana recolheram-se valores para a calibração dos resultados. Esperava-se que os valores lidos no interior das duas câmaras fossem muito semelhantes, no entanto o compartimento A apresentava valores um pouco diferentes do compartimento B. A razão mais provável para esta diferença é o fato do compartimento A ter mais exposição solar e/ou ter alguma abertura invisível ao olho nu que permitisse a entrada e saída de ar.

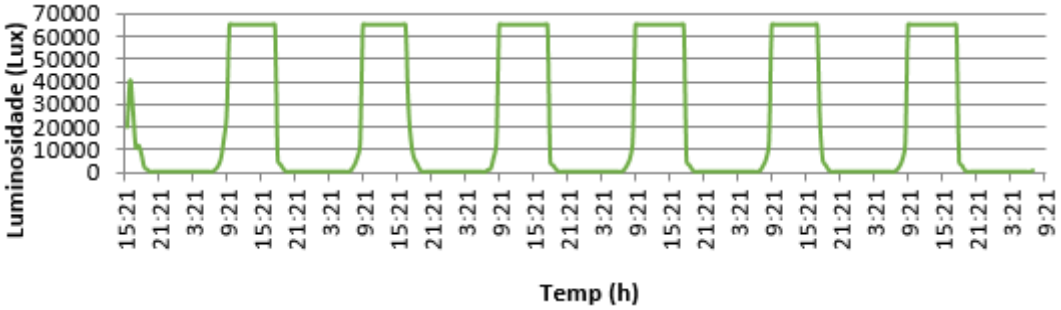

Figura 32 - Luminosidade (lux) da primeira semana

Na Figura 32 observam-se os ciclos diurnos e noturnos a que a câmara de ensaios esteve exposta durante a primeira semana. O valor 0 corresponde a ciclo da noite e os valores vão aumentando conforme o dia vai clareando. De referir que para os ciclos diurnos o valor máximo lido pelo sensor é de 63000 lux sendo essa a limitação do sensor utilizado, no entanto, para o caso de teste em questão essa limitação não tem qualquer impacto. Para se obterem valores mais precisos deverá utilizar-se um sensor de maior qualidade.

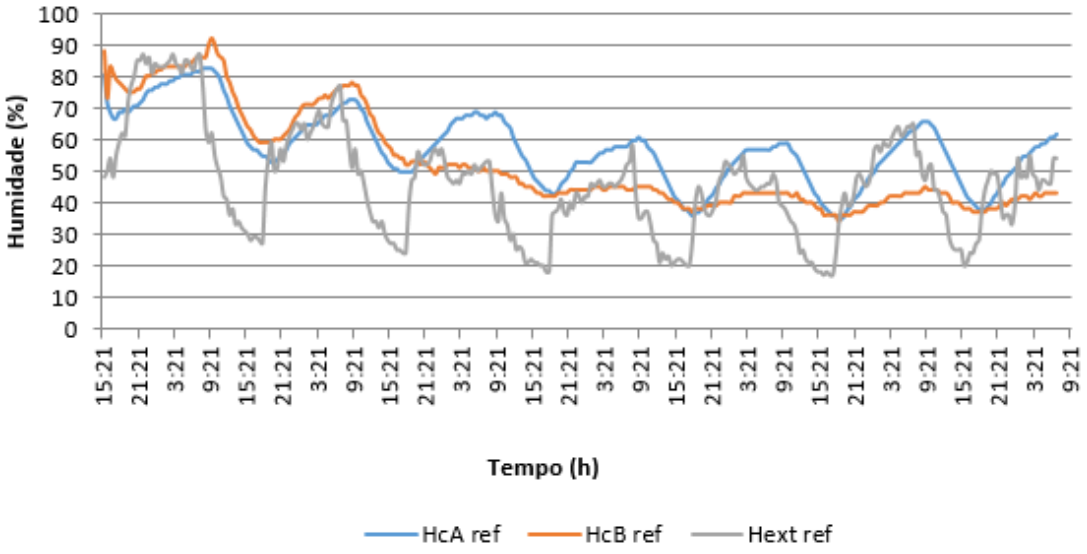

Figura 33 - Humidade exterior (Hext ref), humidade da câmara A (HcA ref), humidade da câmara B (HcB ref), da primeira semana

A Figura 33 mostra os valores da humidade durante a primeira semana para o exterior da câmara (Hext ref), o compartimento A (HCA ref) e compartimento B (HcB ref). A humidade no compartimento A continua a ser superior ao fim de 7 dias, mas ambas começam a ter valores constantes nos últimos 3 dias de testes.

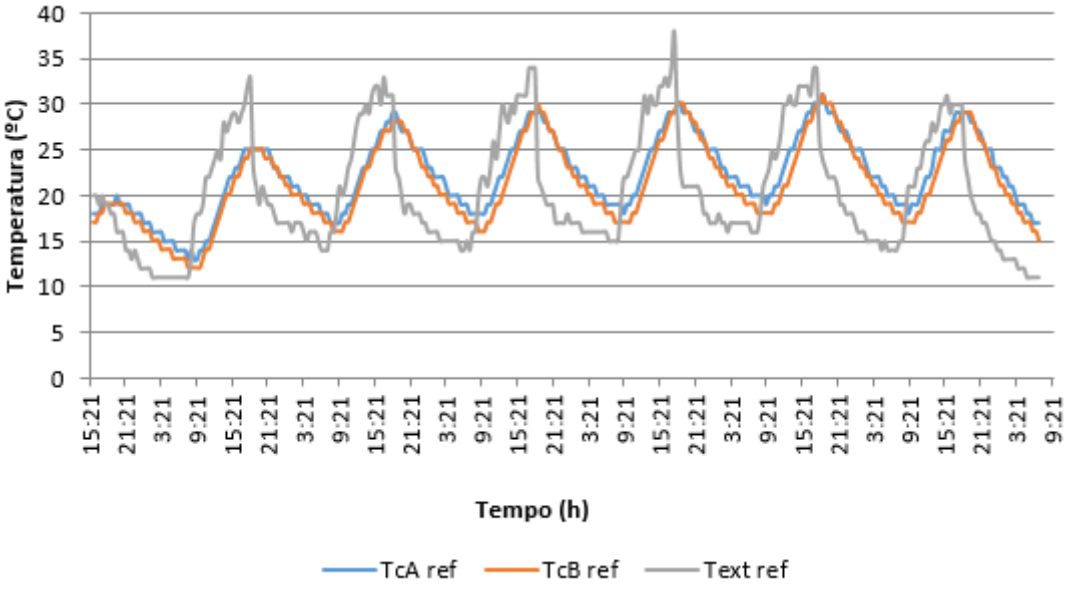

Figura 34 - Temperatura exterior (Text ref), temperatura da câmara A (TcA ref), temperatura da câmara B (TcB ref), da primeira semana

A Figura 34 mostra a evolução da temperatura durante a primeira semana e podemos constatar que a diferença entre as temperaturas das duas câmaras é mínima, sendo praticamente 0ºC para a máxima e entre 1ºC e 2ºC para a mínima.

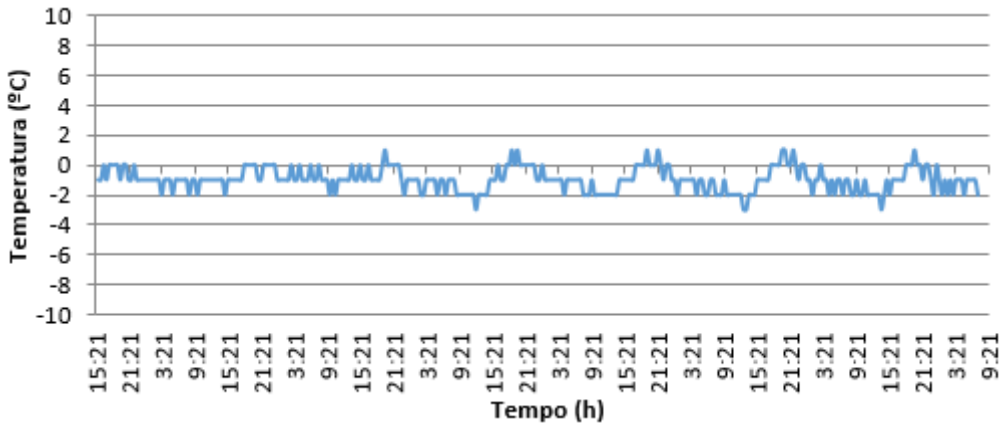

Figura 35 - Diferencial de temperatura entre a câmara B e a câmara A

Na Figura 35 pode observar-se que a diferença média de temperatura entre as câmaras B e A é de -1ºC, isto é, devido a menor exposição solar a câmara B está mais fria 1ºC.

Na segunda semana foram utilizados os mesmos sensores para recolher o mesmo tipo de informação, mas desta vez as paredes da câmara de ensaios foram recobertas com os módulos GEOGREEN. A parede da câmara A foi recoberta com as placas GEOGREEN clássicas e a parede da câmara B com as placas GEOGREEN e PCM.

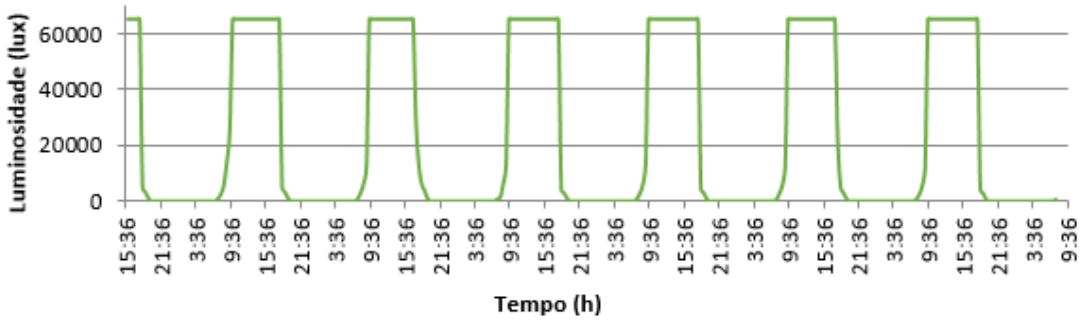

Figura 36 - Luminosidade (lux) da segunda semana

A Figura 36 mostra os ciclos noturnos e diurnos a que a câmara esteve exposta durante a segunda semana.

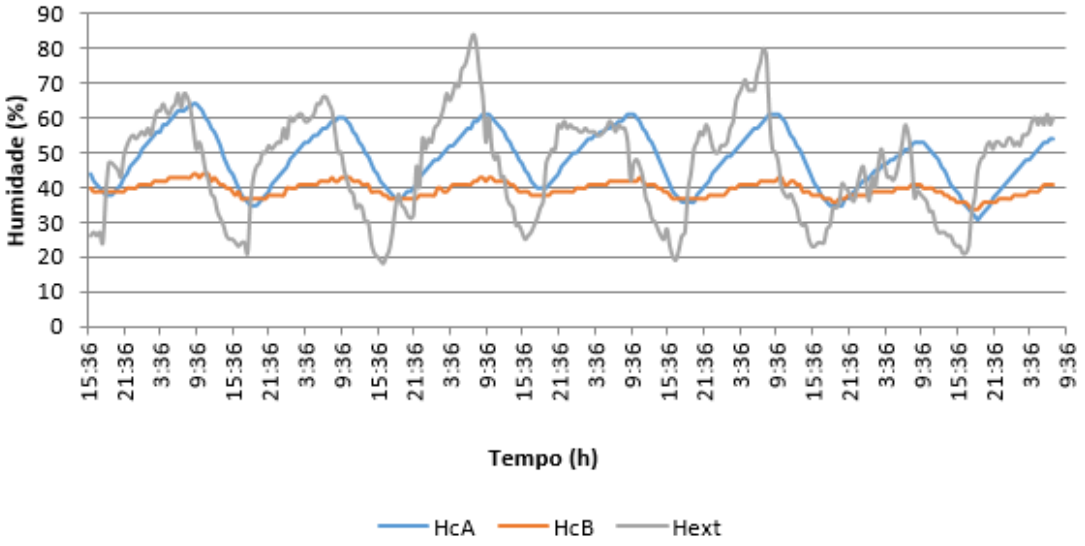

Figura 37 - Humidade exterior (Hext), humidade da câmara A (HcA) e humidade da câmara B (HcB) durante a segunda semana

Na Figura 37 pode verificar-se que a câmara A tem uma variação média de 20% durante os ciclos diurnos e a câmara B de 5% de humidade. Estas variações são praticamente constantes durante toda a semana.

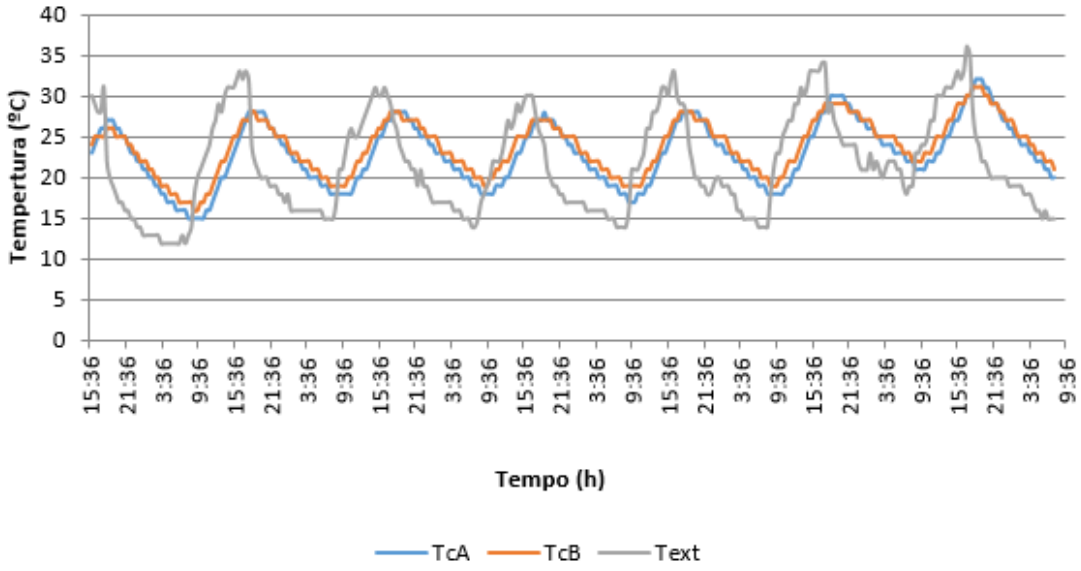

Figura 38 - Temperatura exterior (Text), temperatura da câmara A (TcA), temperatura da câmara B (TcB) durante a segunda semana

A Figura 38 mostra-nos que a câmara A, com sistema GEOGREEN simples, tem valores máximos de temperatura superiores ao da câmara B com sistema GEOGREEN e PCM. No que respeita a valores mínimos a câmara A tem valores inferiores aos da câmara B. Estes resultados mostram que durante o dia a câmara A aquece mais que a B e durante a noite arrefece mais.

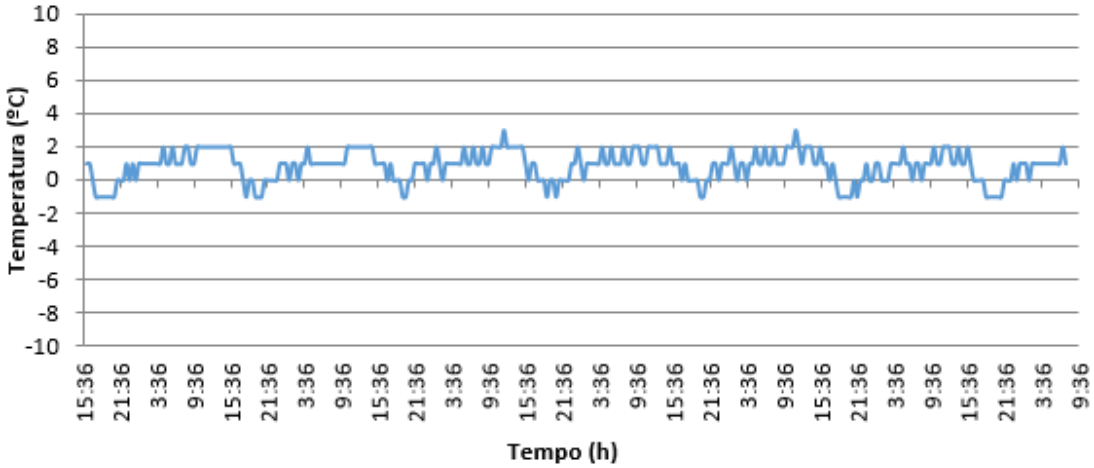

Figura 39 - Diferencial de temperaturas entre a câmara B e a câmara A na segunda semana

Podemos observar na Figura 39 que existe uma diferença média entre a câmara B e a câmara A de -1ºC durante o período diurno e +2ºC durante o período noturno.

Perante os resultados anteriores podemos concluir que a parede B, ou seja a parede com os módulos GEOGREEN e PCM, é mais eficiente que a parede A com o sistema modular GEOGREEN clássico.

6.3. Caso 2 – Sistema GeogreenPlus

Um dos objetivos deste trabalho é o de tentar demonstrar que a combinação das placas modulares GEOGREEN com uma rede de sensores e um sistema de informação se traduzem num produto com maiores potencialidades do que a utilização das placas modulares por si só.

Um dos ganhos potenciais seria a tomada de decisões automáticas por parte do sistema, baseadas na informação recolhida pelos sensores. Essas decisões levariam a tomada de ações práticas em zonas focalizadas das superfícies recobertas com os módulos GEOGREEN.

Neste caso de teste tentou demonstrar-se que tal cenário é possível e pode levar a ganhos quantificáveis potenciando assim o valor do sistema modular GEOGREEN.

NOTA: O teste foi realizado durante o mês de Setembro de 2017.

6.3.1. Descrição do caso de teste

Como referido anteriormente o sistema modular GEOGREEN é um sistema para a construção de superfícies ajardinadas. Estas superfícies necessitam de ser regadas por forma a manterem a vegetação viva. Neste caso de teste vamos tentar demonstrar a utilidade da acoplagem de uma rede de sensores ao sistema modular GEOGREEN nas situações de rega.

A necessidade de rega das superfícies ajardinadas pode definir-se pela quantidade de água no solo, ou seja, o valor da humidade do solo. Esses valores não são os mesmos para todas as espécies vegetais, logo numa situação ideal, cada espécie com necessidades de água semelhante deveria receber quantidades de água semelhante. De mais, a rega deveria apenas ser executada quando a quantidade de água no solo fosse inferior ao valor desejável para essa espécie. O presente caso de teste tentou responder a estas necessidades mostrando a utilidade do sistema GeogreenPlus no processo.

Sendo assim neste caso de teste foram plantadas ervas aromáticas em duas placas modulares GEOGREEN distintas, uma das quais recebeu um sensor de humidade do solo para monitorizar esse valor e a outra não teve qualquer monitorização. Foi

definido um intervalo regular para a leitura do valor da humidade do solo e definido um valor a partir do qual o sistema deveria efetuar uma ação. Essa ação foi a de enviar um *email* ao utilizador do sistema informando-o de que deve regar as plantas. Neste caso concreto a rega foi efetuada à mão utilizando a informação produzida pelo sistema, no entanto como objetivo futuro, o sistema de rega deverá ser integrado nos módulos GEOGREEN e a rega controlada pelo sistema GeogreenPlus.

Um dos objetivos do teste é demonstrar que através da monitorização, com o sistema GeogreenPlus, a água utilizada no processo de rega é administrada de forma mais rigorosa evitando assim desperdícios e ultimamente permitir ao utilizador reduzir os gastos com esse processo. Para poder efetuar essa comparação, foi pedido a um jardineiro amador que tomasse conta do módulo GEOGREEN não monitorizado, no contexto da sua atividade regular e que medisse a quantidade de água utilizada quando efetuasse a rega das plantas.

6.3.2. Valores de referência

O sensor utilizado foi, como indicado no capítulo 5, secção 5.2.2, foi o sensor de humidade do solo, modelo *Soil Moisture* da empresa *iTead Studios*. A tabela seguinte apresenta os valores de referência lidos pelo sensor.

Tabela 8 - Valores de referência do sensor de humidade do solo

Tipo	Valor
Água	< 170
Substrato vegetal fertilizado (sem adição de água)	635 - 660
Ar	> 695

No dia 5 de Setembro de 2017 foram compradas as plantas e transferidas para o módulo GEOGREEN. O primeiro valor lido pelos sensores foi 252 às 18:08 desse mesmo dia, alguns momentos depois das plantas terem sido transferidas para as cavidades do módulo GEOGREEN. Para a calibração do sistema e obter valores de referência da humidade do solo, foram recolhidos valores durante 2 dias seguidos, entre 05/09/2017 às 18:08 até 07/09 às 18:57, sendo que inicialmente foram recolhidos dados todos os 10 minutos e a partir das 7:43 do dia 06/08 passaram a recolher-se dados todos os 30 minutos.

Na Figura 40 podemos ver os valores lidos pelo sensor de humidade do solo durante o período indicado. Como referido no parágrafo anterior, o primeiro valor lido pelo sensor foi 252, tendo o valor mais baixo (maior humidade) registado sendo 169 e o valor mais alto (menor humidade) 282. Podemos verificar uma quebra acentuada entre as 21:28 e as 22:28 do dia 5 resultando de uma rega, cujo objetivo foi o de perceber o comportamento do sensor nessa situação. Cada cavidade individual do módulo GEOGREEN recebeu 90 ml de água, sendo que a placa tem 9 compartimentos distintos, foi usado um total de 810 ml de água.

O sensor teve o comportamento esperado, ou seja, os valores lidos pelos sensores foram baixando com o passar dos minutos e depois voltando a subir gradualmente à medida que o substrato vegetal ia ficando mais seco.

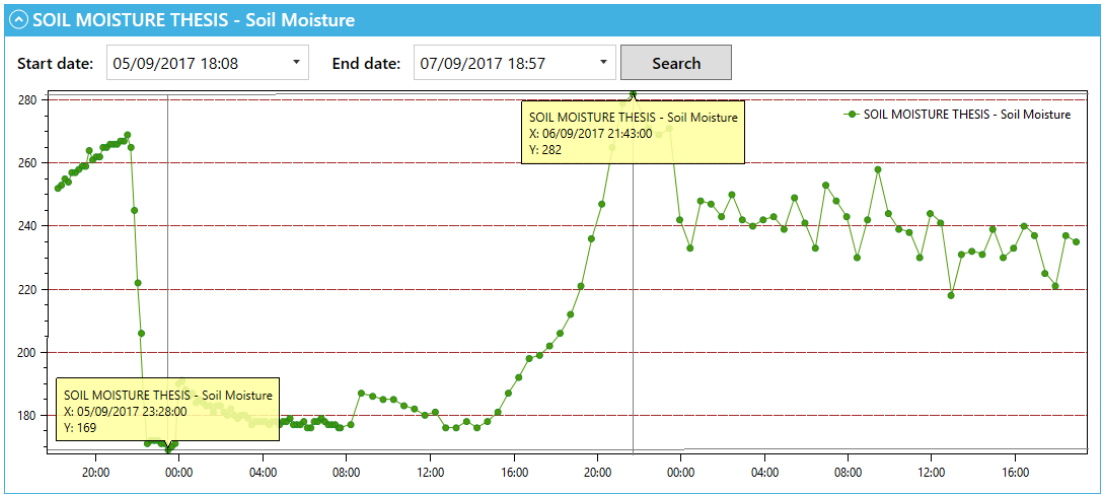

Figura 40 - Dados recolhidos pelo sensor de humidade do solo no período de calibração

Para definir o valor a partir de qual o utilizador deve ser notificado para regar as plantas, procedeu-se da seguinte maneira. Tendo em conta o primeiro valor lido pelo sensor depois da mudança das plantas, dos vasos originais para o módulo GEOGREEN ter sido 252, decidiu dar-se uma margem de 50% no valor máximo de falta de água (252 + (252*0.5) = 378) obtendo assim um valor máximo de 378 que passamos a 380 para ter um valor arredondado às dezenas. Sendo assim sempre que o sensor leia um valor superior a 380, deve enviar um *email* ao utilizador, ou utilizadores registados para este evento, indicando que deve efetuar uma ação devido a esta situação.

6.3.3. Ensaio em condições reais

O ensaio decorreu entre os dias 8 de Setembro de 2017 e 22 de Setembro de 2017. Entre os dias 9 e 12 foram efetuadas leituras todos os 30 minutos, no entanto a partir do início da manhã do dia 12 alterou-se o intervalo para 60 minutos, uma vez que 30 minutos se revelou ser um intervalo demasiado curto, não havendo necessidade de guardar tantos dados para tomar decisões úteis sobre o sistema.

Além dos valores da humidade do solo foram ainda recolhidos os valores da temperatura e humidade do ar, da luminosidade e do CO2 do meio envolvente aos módulos GEOGREEN. Estes valores não são relevantes para o teste em questão mas foram recolhidos numa ótica de demonstrar a capacidade do sistema em obter dados de diferentes tipos deixando em aberto a possibilidade de se adicionarem mais sensores conforme necessidades futuras[22].

É importante ainda referir que foram utilizadas as duas arquiteturas anteriormente referidas, cliente-servidor e nuvem, para a realização deste ensaio. Entre os dias 08/09/2017 às 10:57 e 18/09/2017 às 20:41 utilizou-se a arquitetura cliente-servidor, e nos restantes dias, entre 18/09/2017 às 23:40 e 22/09/2017 às 13:40, utilizou-se a arquitetura de nuvem. Essa mudança foi completamente transparente não afetando em nada o sistema ou o teste em si, uma vez que a única modificação necessária foi a alteração do caminho para a Web API nos ficheiros de configuração.

A Figura 41[23] mostra o gráfico da humidade do solo para a semana de 8 a 14 de Setembro. Nele podemos observar que os valores da humidade vão evoluindo gradualmente para valores mais altos até terem quedas abruptas, sendo esses os momentos em que as plantas, do módulo com monitorização, foram regadas.

[22] No Apêndice B deste documento podem ser consultados os resultados da leitura dos sensores referidos

[23] A Figura 41 foi retirada diretamente da página Charts da interface com o utilizador do sistema GeogreenPlus.

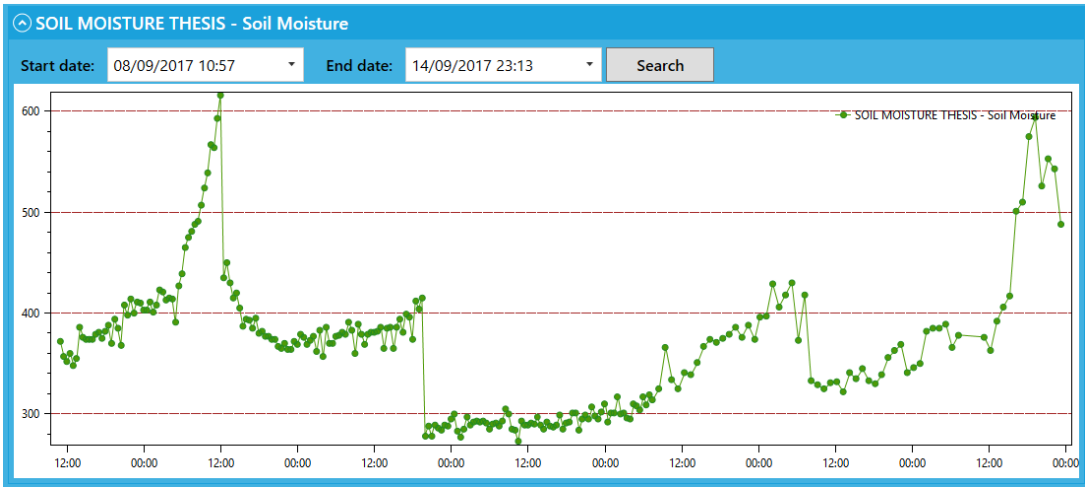

Figura 41 - Valores da humidade do solo na semana de 8 a 15 de Setembro

Vamos analisar um espaço temporal mais curto, de 08/09/2017 às 10:57 até 14/09/2017 às 23:57, para percebermos melhor o comportamento do sistema. A Figura 42 mostra-nos 3 períodos distintos:

- P1 - de 08/09/2017 às 10:57 até 09/09/2017 às 11:57;
- P2 - de 09/09/2017 às 12:27 até 10/09/2017 às 19:27;
- P3 – de 10/09/2017 às 19:57 até 11/09/2017 às 07:57.

Todos os períodos mostram valores com tendência crescente desde a data limite inferior até à data limite superior, momento em que há uma descida repentina dos valores e começa um novo período. Essa quebra representa o momento em que as plantas foram regadas. Como dito anteriormente foi configurada uma regra para que o utilizador fosse alertado sempre que os valores da humidade do solo ultrapassassem os 380, logo no período P1 o sistema enviou um *email* ao destinatário, alertando-o para a necessidade de regar as plantas, às 13:57 do dia 08/09/2017. Num sistema ótimo (ver 7.2) este deveria regar automaticamente as plantas com a quantidade de água definida pelo sistema, no entanto, não estando essa funcionalidade disponível no protótipo terá de ser o utilizador a fazer a rega manualmente usando medidas precisas de água.

Como podemos ver no gráfico a rega ocorreu entre as 11:57 e as 12:27 do dia seguinte uma vez que o utilizador não teve disponibilidade para o efetuar mais cedo.

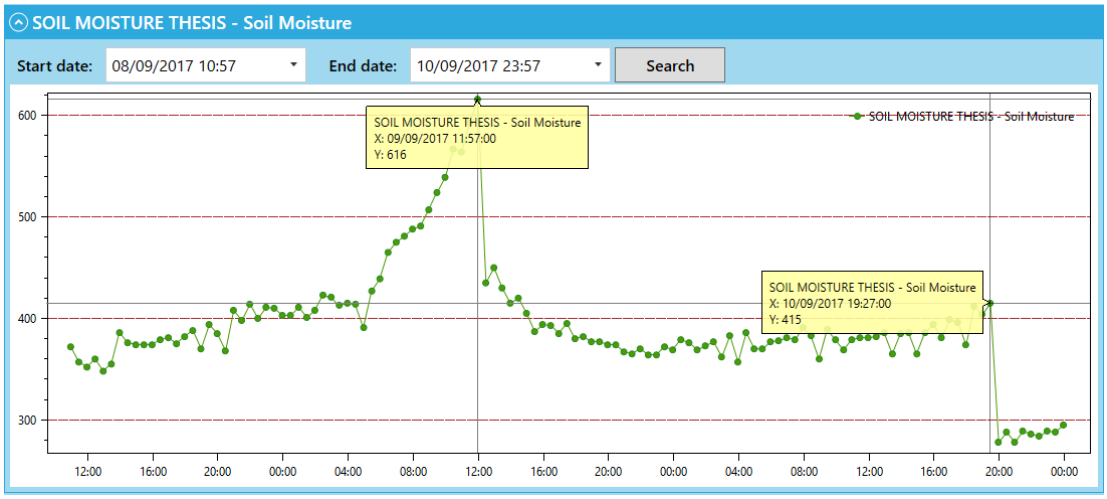

Figura 42 - Evolução da humidade do solo entre 08/09/2017 às 10:57 até 10/09/2017 às 23:57

As figuras Figura 43 e Figura 44 mostram os valores da humidade do solo lidos pelo sensor para a totalidade do período de teste. A Figura 43 representa o período em que foi utilizada a arquitetura cliente-servidor e nela podemos observar um período em que há um desvio acentuado no padrão de leitura dos dados. Esse desvio deveu-se à avaria do sensor tendo este sido substituído por outro, do mesmo modelo, no dia 17/09/2017 às 19:30, tendo imediatamente a leitura voltado aos valores esperados.

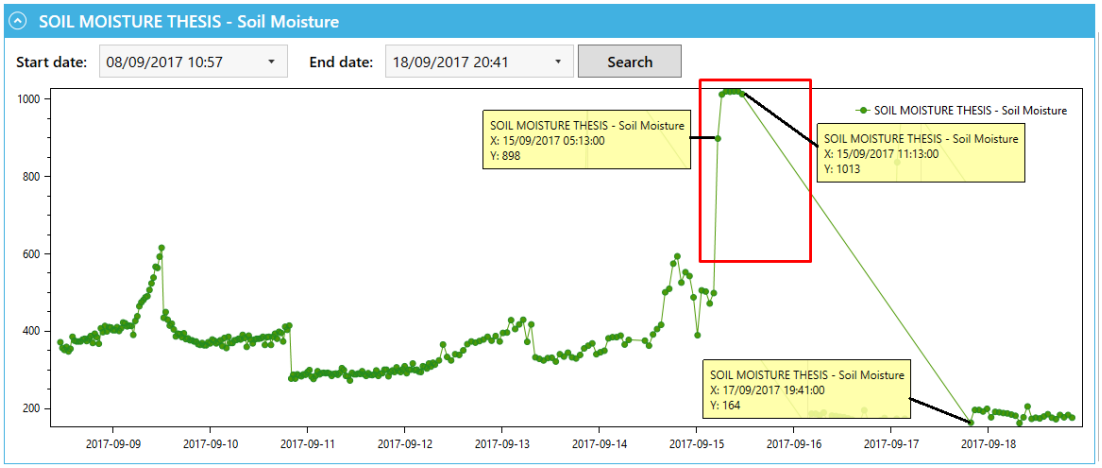

Figura 43 - Evolução da humidade do solo no período da arquitetura cliente-servidor

A Figura 44 mostra a evolução da humidade do solo no período em que foi utilizada a arquitetura de nuvem. Podemos observar que a quebra há uma única quebra acentuada

nos valores lidos. Essa quebra corresponde à única rega efetuado no período entre 18/09/2017 às 23:40 e 22/09/2017 às 13:40.

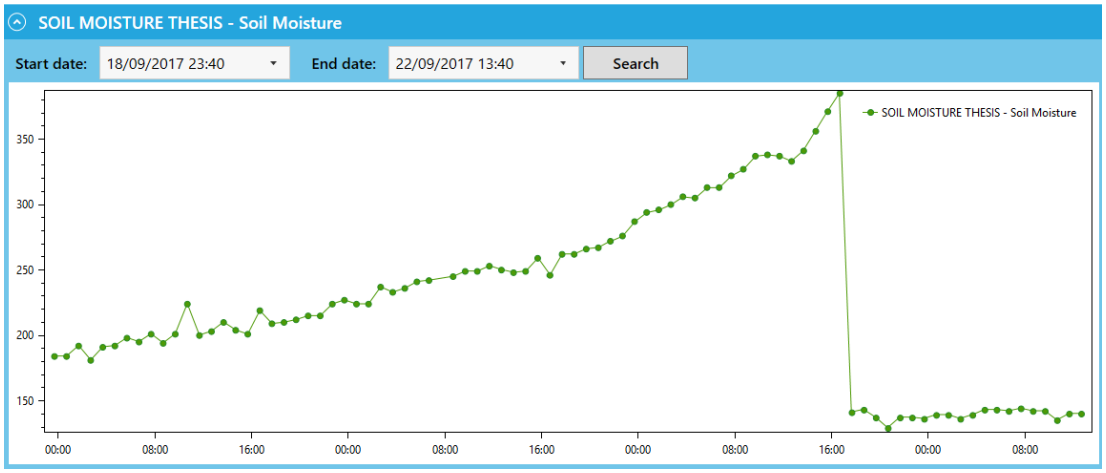

Figura 44 - Evolução da humidade do solo no período da arquitetura de nuvem

A Tabela 9 mostra detalhadamente o momento das regas e quantidades de água utilizadas, tanto no caso do módulo com monitorização como no módulo sem monitorização.

Tabela 9 - Horários das regas e quantidade de água usada

Módulo monitorizado		Módulo NÃO monitorizado	
Data	**Quantidade (mililitros)**	**Data**	**Quantidade (mililitro)**
09/09/2017 12:25	1080	08/09/2017 19:50	1120
10/09/2017 19:40	810	10/09/2017 18:30	1095
13/09/2017 07:50	810	12/09/2017 18:50	1110
14/09/2017 19:35	1080	15/09/2017 12:35	1050
21/09/2017 17:05	810	17/09/2017 20:10	1150
		20/09/2017 08:55	910
Total (ml)	4590	**Total (ml)**	6435

Como era expetável as datas das regas e número de vezes que aconteceram são diferentes uma vez que foram efetuadas por pessoas diferentes e, porque no caso do

módulo monitorizado, o utilizador tentou sempre efetuar a rega no espaço de tempo mais curto possível, depois de receber o alerta via *email* para efetuar essa ação. Por outro lado o jardineiro amador, que regou o módulo não monitorizado, fê-lo de maneira tradicional, ou seja, sem recurso a nenhum tipo de informação sobre o estado do solo, apenas recorrendo à sua experiência.

6.3.4. Análise dos resultados e conclusões

Partindo dos valores apresentados na Tabela 9 podemos tirar ilações quanto às vantagens de ter os módulos GEOGREEN monitorizados com uma rede de sensores. Olhando para o gráfico representado na Figura 45 podemos facilmente perceber que a quantidade de água utilizada no módulo sem monitorização é constantemente superior à utilizada na rega do módulo monitorizado.

Durante os 14 dias de duração do teste o jardineiro amador regou as plantas 6 vezes tendo gasto um total de 6435 mililitros de água, enquanto o utilizador do sistema GeogreenPlus regou as plantas 5 vezes tendo gasto 4590 mililitros de água. Estes resultados mostram que o sistema GeogreenPlus permitiu poupar um total de 1845 mililitros de água para o mesmo período de tempo e condições climatéricas iguais.

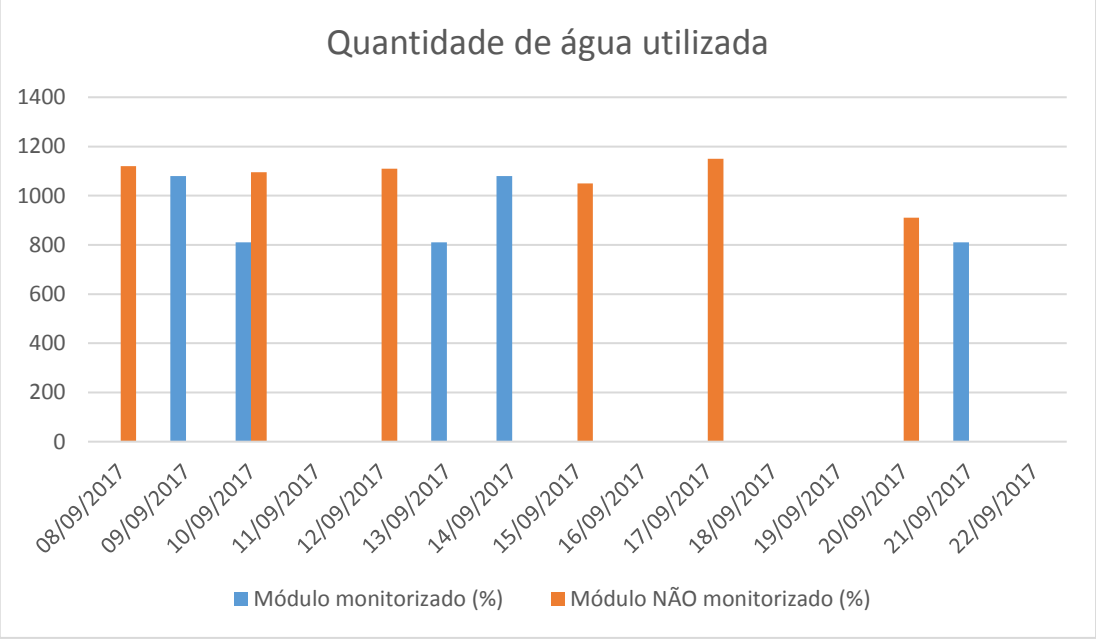

Figura 45 - Comparação entre consumos de água dos módulos monitorizado e não monitorizado

6.4. Conclusão

Neste capítulo foram apresentados os casos de validação experimental do sistema GeogreenPlus.

Em 6.2 apresentou-se o caso da comparação entre placas modulares GEOGREEN clássicas e placas modulares GEOGREEN com PCM. O principal objetivo deste caso de teste era demonstrar que o material alterado com PCM permite um melhor desempenho térmico. A medição dos resultados que permitiram retirar conclusões dessa comparação foi efetuada pelo sistema GeogreenPlus. Logo foi aqui atingido um dos objetivos desta tese enumerado no capítulo 1.2 que tinha por base permitir medir o desempenho de diferentes tipos (constituição do seu material) de placas modulares.

O caso de teste apresentado em 6.3 analisou os benefícios do acoplamento do sistema GeogreenPlus com as placas modulares GEOGREEN com as respetivas cavidades preenchidas com espécies vegetais. O objetivo do teste era o de demonstrar que placas GEOGREEN monitorizadas com a ajuda de sensores permitiam ter um maior controlo sobre a quantidade de água necessária para o processo de rega bem como efetuar esse processo apenas quando fosse verdadeiramente necessário. Durante 15 dias foi monitorizada uma placa modular GEOGREEN sendo enviado automaticamente um alerta via *email* ao utilizador do sistema, sempre que a humidade no solo descesse mais do que um limite fixado, permitindo assim ao utilizador regar as plantas apenas quando necessário. Durante os mesmos 15 dias foi pedido a um jardineiro amador que se ocupasse de outro módulo GEOGREEN, tratando da flora como se fosse um recipiente comum para plantas, pedindo-lhe que anotasse a quantidade de água despendida nos momentos de rega.

Foi feito o diferencial entre as quantidades de água gastas na rega do módulo monitorizado com o sistema GeogreenPlus e no módulo não monitorizado pelo sistema, podendo concluir-se que o primeiro permitiu uma poupança de 1845 mililitros no mesmo período. Este resultado permitiu confirmar a utilidade e mais-valia do sistema GeogreenPlus combinado com os módulos GEOGREEN.

7. Conclusões

7.1. Resumo

Com a realização deste trabalho pretendeu-se criar um sistema inovador, composto por uma rede de sensores e um sistema de informação baseado preferencialmente na nuvem, que permitisse adicionar valor às placas modulares GEOGREEN, desenvolvidas no CMADE, destinadas a serem utilizadas na criação de superfícies ajardinadas.

Foi construído o protótipo a que nos propusemos utilizando vários tipos de sensores e placas Arduino, ligados a placas Raspberry Pi que centralizam o processo de recolha e armazenamento de dados comunicando com uma Web API, e foi ainda desenvolvida uma aplicação de interface com o utilizador que permite a consulta dos dados guardados e a definição de regras para o sistema.

Para a componente de *back-end* foram implementados dois tipos de arquitetura, cliente-servidor e nuvem, mais precisamente utilizando tecnologia Microsoft Azure.

Foi feita uma validação experimental do protótipo desenvolvido em dois cenários de teste diferentes para responder aos objetivos propostos inicialmente. No primeiro caso foi utilizado o sistema GeogreenPlus para validar as qualidades das placas modulares GEOGREEN com PCM em relação às placas clássicas. No segundo caso demonstrou-se que a rede de sensores acoplados aos módulos GEOGREEN, em conjunto com o sistema de informação desenvolvido, permitem ter um controlo mais preciso das ações a efetuar sobre a estrutura, verificando-se ainda a mais-valia de permitir poupança de recursos aos proprietários desse sistema.

7.2. Trabalho futuro

Durante o desenvolvimento deste protótipo fomo-nos deparando com situações que nos levaram a pensar em abordagens diferentes no caso de se vir a avançar para um produto comercial. Uma dessas situações é a forma como os componentes de *hardware* comunicam entre si, utilizando cabos, tornando impraticável a sua instalação em estruturas de médias ou grandes dimensões. Para trabalho futuro deverão encontra-se soluções de comunicação sem fios entre os componentes, tais como por exemplo o *Bluetooth*.

Outro eixo de melhoria será a adição de sensores de nutrientes existentes no solo tais como o potássio, o fósforo e mais que sejam relevantes para que as espécies vegetais presentes nos módulos sejam nutridas convenientemente. Além dos sensores de

nutrientes deverá instalar-se um mecanismo de atuadores, ou seja, permitir automatizar as ações sobre o sistema, tais como a rega automática baseada em parâmetros precisos definidos na aplicação e a administração automática de nutrientes.

No que diz respeito à alimentação energética da rede de sensores poderão também introduzir-se melhorias, como por exemplo a utilização de módulos de energia renováveis (células fotovoltaicas) acopladas a baterias, como principal fonte de energia e utilizando a alimentação clássica apenas como recurso de emergência.

Em relação à interface com o utilizador poderá desenvolver-se uma aplicação mobile que possa correr em dispositivos móveis outros que *tablets* com sistema operativo Windows instalado.

Parte das melhorias propostas estão na realidade já em fase de execução (havendo já alguns resultados) mas cujo âmbito sai fora do trabalho desta tese.

Referencias

[1] Comissão Europeia, COMUNICAÇÃO DA COMISSÃO, EUROPA 2020, Estratégia para um crescimento inteligente, sustentável e inclusivo, Bruxelas, 2010/03/03

[2] Castro Gomes, J. P., Project PTDC/ECM/113992/2009: GEOGREEN – Waste geopolymeric binder-based natural vegetated panels for energy-efficient building green roofs and facades. Final Report: Description of the work undertaken and main results achieved, University of Beira Interior, 2014

[3] Mullaghan S. (2014, 7, 31), What is the 'Internet of Things' and where does Microsoft sit?. Available: https://blogs.technet.microsoft.com/uktechnet/2014/07/31/what-is-the-internet-of-things-and-where-does-microsoft-sit

[4] ITU-T, Recommendation ITU-T Y.2060 Overview of the Internet of things, 2012

[5] Burgess M. (2017, 04, 21), What is the Internet of Things? WIRED explains. Available: http://www.wired.co.uk/article/internet-of-things-what-is-explained-iot

[6] Schultz B. (2010), What is a sensor network?. Available: http://searchenterprisewan.techtarget.com/What-is-a-sensor-network

[7] Gama Marques R., "Comportamento térmico do sistema modular GEOGREEN, com incorporação de PCMs", MSc, Universidade da Beira Interior, Covilhã, Portugal, 2016

[8] Kang, J. (2011, 7), What is an API?. Available: https://www.quora.com/What-is-an-API-4

[9] Microsoft Developer Network, ASP.NET Web API. Available: https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx

[10] Microsoft, Learn About ASP.NET Web API. Available: https://www.asp.net/web-api

[11] Microsoft, Windows Remote Arduino. Available: https://github.com/ms-iot/remote-wiring

[12] Microsoft, Windows IoT Core. Available: https://developer.microsoft.com/en-us/windows/iot

[13] Microsoft, Get Started. Available: https://developer.microsoft.com/en-us/windows/iot/GetStarted

[14] Microsoft, Windows IoT Core Downloads and Tools. Available: https://developer.microsoft.com/en-us/windows/iot/Downloads.htm

[15] Windows Apps Team (2016, 02, 04), What is Windows Remote Arduino ans WHat Can It Do?. Available: https://blogs.windows.com/buildingapps/2016/02/04/what-is-windows-remote-arduino-and-what-can-it-do

[16] Microsoft (2015, 09, 04), ms-iot / remote-wiring advanced. Available: https://github.com/ms-iot/remote-wiring/blob/master/advanced.md

[17] SQLite. Available: https://www.sqlite.org/

[18] Bidault M. (2015, 04, 01), Une loi pour couvrir la France de toitures végétalisées. Available : https://www.lesechos.fr/01/04/2015/LesEchos/21910-351-ECH_une-loi-pour-couvrir-la-france-de-toitures-vegetalisees.htm

[19] Livingroofs.org, Czech buildings to be given financial support to implement green roofs, Available: https://livingroofs.org/czech-buildings-finance-green-roofs/

[20] Czech Landscape Gardening Association (2016, 12, 07), "New green savings program will support the construction of green roofs", Press release of the Czech Green Roof Association

[21] Mills C. (2015, 13, 11), We are all doomed to live in vertical forests now. Available: http://gizmodo.com/we-are-all-doomed-to-live-in-vertical-forests-now-1742295615

[22] Phillips T. (2017, 02, 17) 'Forest cities': the radical plan to save China from air polution. Available: https://www.theguardian.com/cities/2017/feb/17/forest-cities-radical-plan-china-air-pollution-stefano-boeri

[23] Microsoft (2017, 05, 02), .NET Guide. Available: https://docs.microsoft.com/en-us/dotnet/standard/

[24] Microsoft (2017, 08, 23), C# Guide. Available: https://docs.microsoft.com/en-us/dotnet/csharp/

[25] Microsoft (2017, 03, 30), Windows Presentation Foundation. Available: https://docs.microsoft.com/en-us/dotnet/framework/wpf/

[26] Microsoft (2016, 10, 23), Introduction to Entity Framework. Available: https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx

[27] Microsoft (2017, 08, 16), Inheritance in C# and .NET. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/inheritance

[28] Microsoft (2017), Polymorphism (C# Programming Guide). Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/polymorphism

[29] Microsoft (2017), Object-Oriented Programming (C#). Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/object-oriented-programming

[30] Thompson A. (2011, 06, 26), Encapsulation – An Introduction. Available: https://blogs.msdn.microsoft.com/alfredth/2011/07/26/encapsulationan-introduction

[31] Microsoft (2012, 02, 10), The MVVM Pattern. Available: https://msdn.microsoft.com/en-us/library/hh848246.aspx

[32] Manso M., "Modular system design for vegetated surfaces with alkaline activated materials", *in Proceedings of 2012 International Workshop On Environment And Alternative Energy*, Greenbelt-Maryland, United States, 2012/12/4-7

[33] Cnxsoft (2017, 06, 19), Intel Issues End-of-Life Notices for Galileo / Galileo 2, Edison and Joule Boards & Modules. Available: https://www.cnx-software.com/2017/06/19/intel-issues-end-of-life-notices-for-galileo-galileo-2-edison-and-joule-boards-modules

[34] Shah A. (2015, 11, 18), Microsoft pulls Windows 10 support from Intel's Galileo boards. Available: https://www.infoworld.com/article/3005584/hardware/microsoft-pulls-windows-10-support-from-intels-galileo-boards.html

[35] Agarwal T. (2015), Wireless Sensor Networks and their Applications. Available: https://www.elprocus.com/introduction-to-wireless-sensor-networks-types-and-applications/

[36] International Electrotechnical Commission, *"Internet of Things: Wireless Sensor Networks*", ISBN 978-28322-1834-1, Geneva, Switzerland, 2014

[37] Pellegrino B., "NDT Meets the Internet of Things (IOT)", *in Proceedings of NDTMA 2016 Annual Conference*, Las Vegas, United States, 2016/02/16-18

[38] Manso M., Virtudes A. L. & Castro-Gomes J. P., "As superfícies ajardinadas como sistema diferenciador na habitação", *in Proceedings of CIHEL - Conference Congresso Internacional da Habitação no Espaço Lusófono*, Lisboa, Portugal, 2013/03/13-15

[39] Pérez-Urrestarazu L., Fernández-Cañero R., Franco-Salas A. & Egea G. (2016), "Vertical Greening Systems and Sustainable Cities, Journal of Urban Technology", DOI: 10.1080/10630732.2015.1073900

Apêndices

Apêndice A – Guia de utilização da interface com o utilizador

Neste apêndice irá ser feita uma descrição das funcionalidades que o utilizador tem à sua disposição na interface gráfica

1. Layout geral

Cada página da interface gráfica é constituída por duas zonas distintas, a barra de navegação (quadrado azul na imagem) e a área de visualização/inserção de dados (quadrado verde na imagem), como se pode ver na imagem seguinte. A página ativa tem o seu título realçado com uma cor azul clara enquanto os títulos das restantes páginas permanecem com uma cor cinzenta.

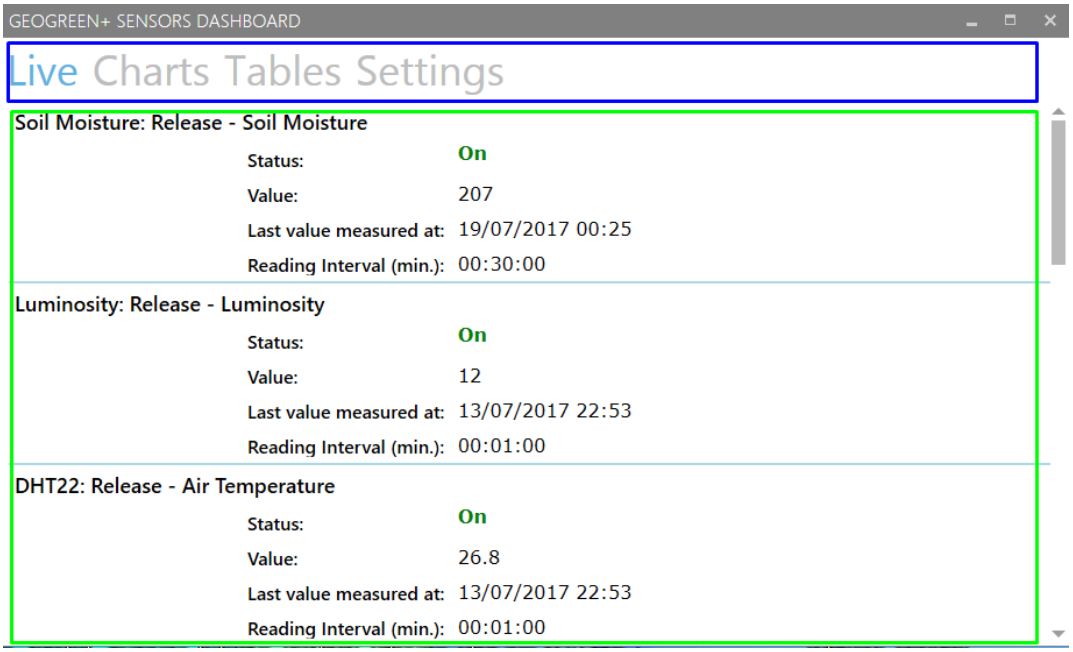

Figura 46 - Layout da interface

O utilizador pode visualizar uma página diferente clicando com o rato, ou com carregando com o dedo em interfaces táteis, sobre o título da página pretendida.

2. Página Live

A página ***Live*** (Figura 46 - Layout da interface) é uma página exclusivamente de visualização e mostra informação todos os sensores que estão guardados na base de dados. Visualmente cada sensor (ou tipo de dados diferentes para um mesmo sensor) é separado por uma linha horizontal.

A informação mostrada ao utilizador é:

- Nome do sensor
- O estado atual do sensor (linha *Status*:)
- O último valor lido pelo sensor (linha *Value*:)
- A data e hora em que o último valor foi lido (linha *Last value measured at:*)
- O intervalo de leitura dos dados (linha *Reading Interval (min:)*)

3. Página *Charts*

A página ***Charts*** permite consultar os valores lidos pelos diferentes sensores em formato de gráfico. Quando a página é mostrada pela primeira vez é apresentada uma lista de controlos com o nome de cada um dos sensores. Para poder consultar os dados o utilizador terá primeiro de expandir o controlo referente aos dados que pretende consultar.

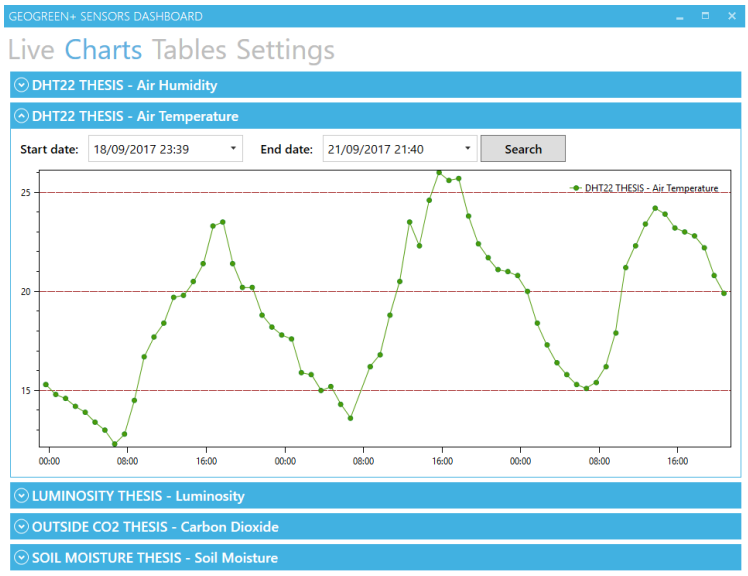

Figura 47 - Página *Charts*

Por exemplo se pretender consultar os dados relativos a "*DHT22 THESIS – Air Temperature*" o utilizador deverá efetuar os seguintes passos:

- Carregar na seta branca ao lado do título
- Selecionar a data de início (lista *Start date:*)
- Selecionar a data de fim (lista *End date:*)
- Carregar no botão *Search*.

Ao carregar no botão *Search* é enviado um pedido ao servidor que irá devolver os valores guardados na BD para o intervalo definido, sendo esses valores utilizados para contruir o gráfico mostrado ao utilizador.

4. Página *Tables*

A página ***Tables*** permite ao utilizador consultar os dados lidos pelos sensores em formato de tabela. Para poder consultar esses dados o utilizador deverá proceder da seguinte maneira:

- Selecionar o sensor para o qual pretende consultar os dados (lista *Select Sensor:*)
- Selecionar o tipo de dados do sensor escolhido (lista *Select Data Type:*). Este parâmetro permite lidar com os sensores que permitem ler tipos diferentes de dados como é o cado do DHT22
- Selecionar a data de início (lista *Start date:*)
- Selecionar a data de fim (lista *End date:*)
- Carregar no botão *Search* que lança a pesquisa.

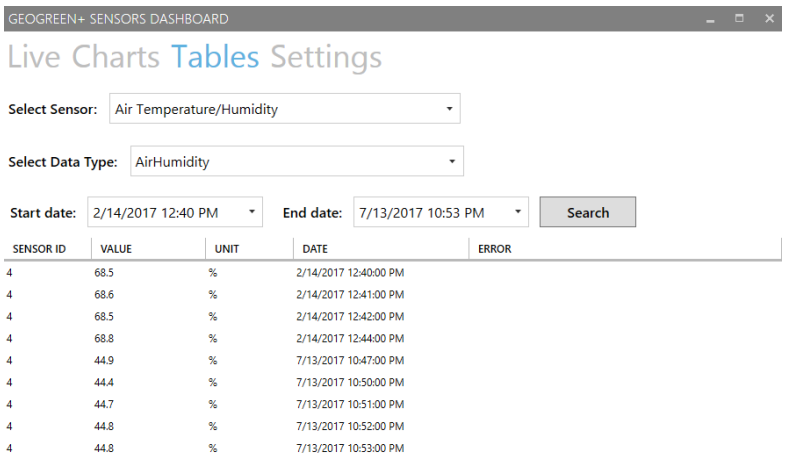

Figura 48 - Página *Tables*

5. Página Settings

A página ***Settings*** permite criar novas regras a aplicar ao sistema, consultar as regras existentes, adicionar novos destinatários de *email* e consultar os destinatários já existentes na base de dados. Esta página contém duas secções distintas, ***RULES*** e ***EMAIL SETTINGS***.

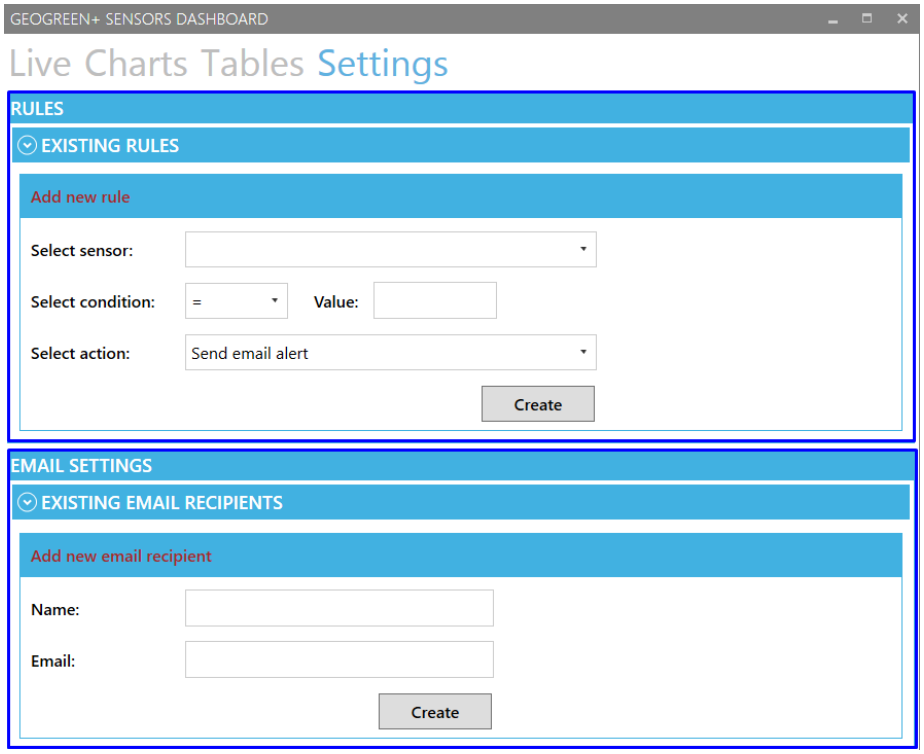

Figura 49 - Página *Settings*

Vamos ver cada uma das secções separadamente para uma melhor clarificação das suas funcionalidades.

- ***RULES***
  - **Consulta**
    Esta secção permite consultar as regras que existem na BD e criar regras novas. Para consultar as regras existentes o utilizador tem apenas de clicar na seta branca situada à esquerda de EXISTING RULES e o controlo irá expandir apresentando a lista pretendida.

  - **Criação**
    Para criar novas regras o utilizador tem de efetuar os passos seguintes:
    - Selecionar o sensor pretendido (lista *Select sensor:*)
    - Selecionar a condição (lista *Select condition:*)
    - Introduzir o valor numérico para a condição (caixa *Value:*)
    - Escolher a ação que irá ser concretizada (lista *Select action:*)

- Carregar no botão *Create* que irá guardar a nova regra na BD.

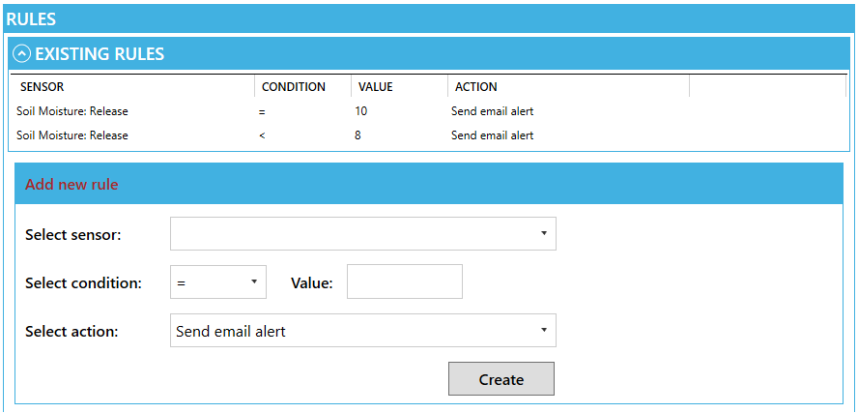

Figura 50 - Página *Settings*, secção *Rules*

- ***EMAIL SETTINGS***
  - **Consulta**

    Para consultar os destinatários de *email* que foram previamente adicionados, o utilizador deverá carregar na seta branca situado à esquerda de EXISTING EMAIL RECIPIENTS. Esta ação irá expandir o controlo e a lista de recipientes será mostrada.

  - **Criação**

    Para adicionar novos destinatários de *email* o utilizador deverá efetuar os seguintes passos:
    - Introduzir o nome do destinatário (caixa *Name:*)
    - Introduzir o endereço *email* do destinatário (caixa *Email:*)
    - Carregar no botão *Create* que irá permitir a inserção de um novo registo na BD.

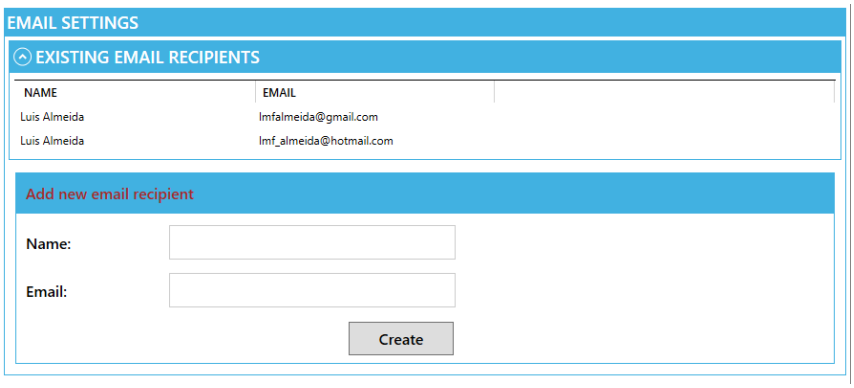

Figura 51 - Página *Settings*, seccção *Email Settings*

Apêndice B – Valores lidos pelos sensores

Este apêndice contém os gráficos com os resultados dos valores lidos pelos sensores utilizados no protótipo exceto os valores da humidade do solo, uma vez que estes estão presentes no capítulo 6.

São apresentados os valores para o período entre os dias 08/09/2017 às 10:57 e 22/09/2017 às 13:40. Durante o período entre 08/09/2017 às 10:57 e 18/09/2017 às 20:41 utilizou-se a arquitetura cliente-servidor, e nos restantes dias, entre 18/09/2017 às 23:40 e 22/09/2017 às 13:40 a arquitetura de nuvem.

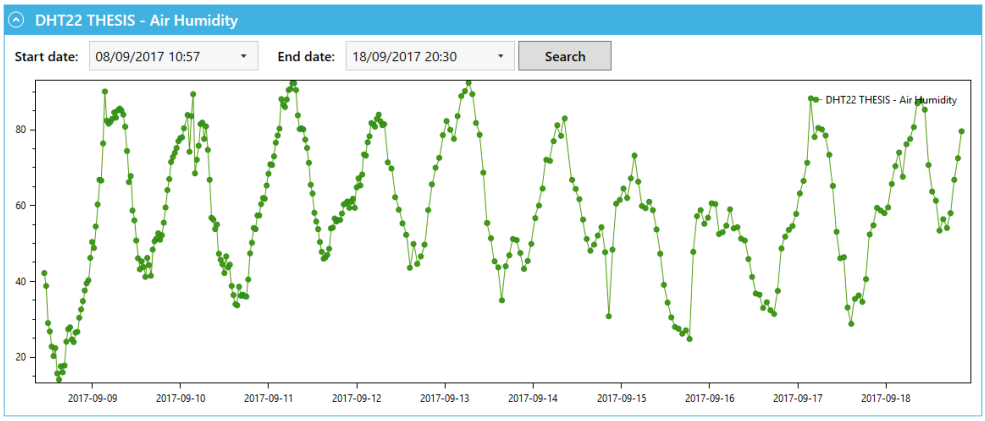

Figura 52 - Variação da humidade do ar (%) no período arquitetura cliente-servidor

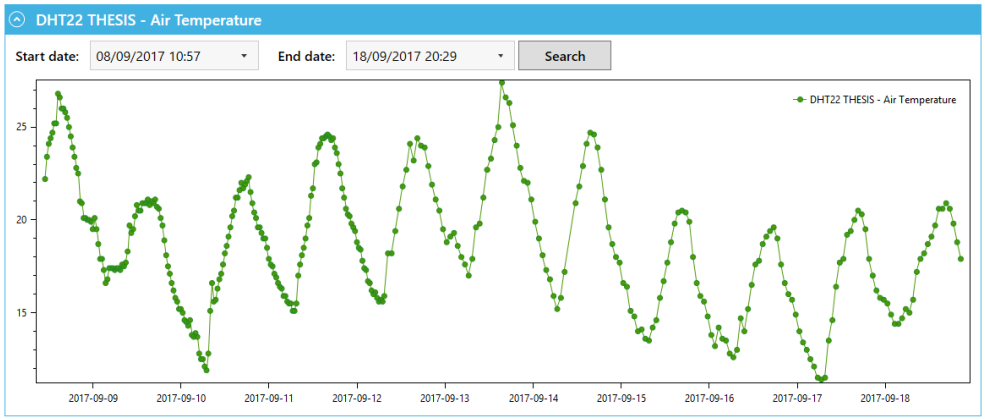

Figura 53 - Variação da temperatura (ºC) no período arquitetura cliente-servidor

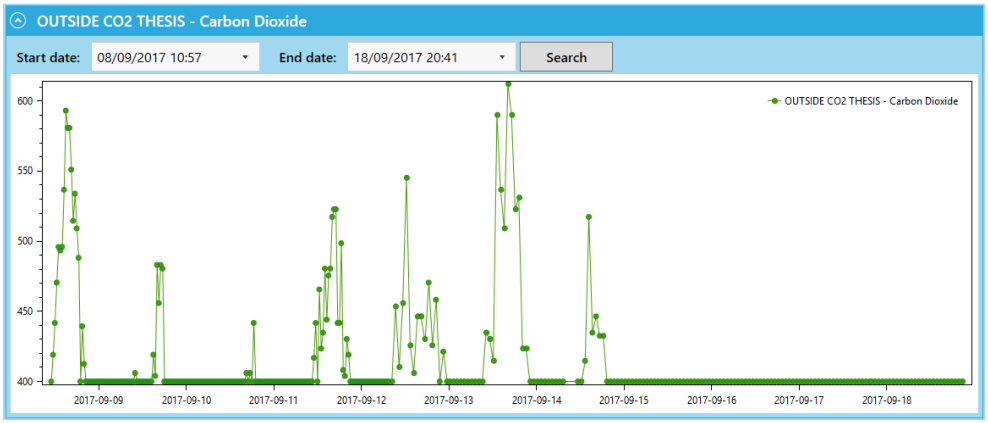

Figura 54 - Variação do C02 (ppm) no período arquitetura cliente-servidor

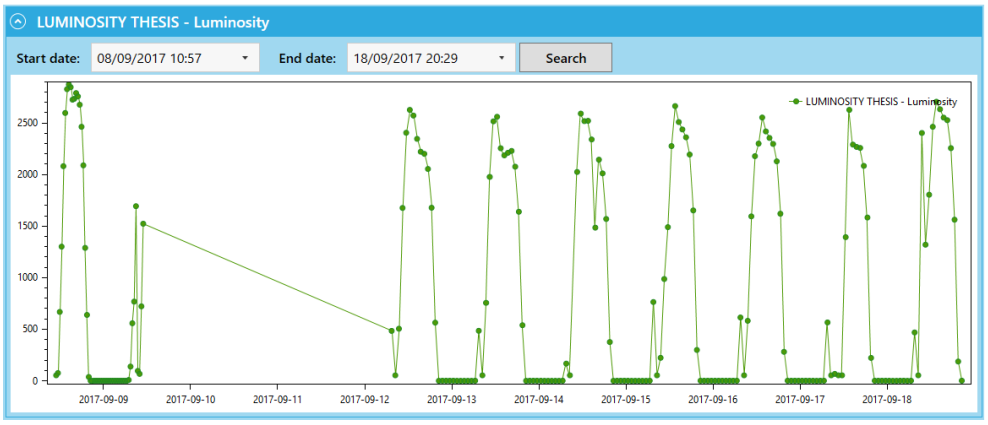

Figura 55 - Variação da luminosidade (lux) no período arquitetura cliente-servidor

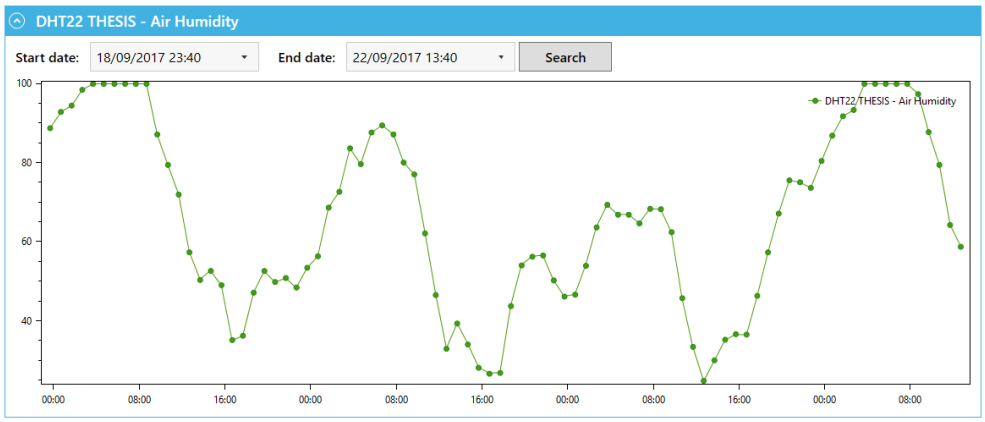

Figura 56 - Variação da humidade do ar (%) no período arquitetura de nuvem

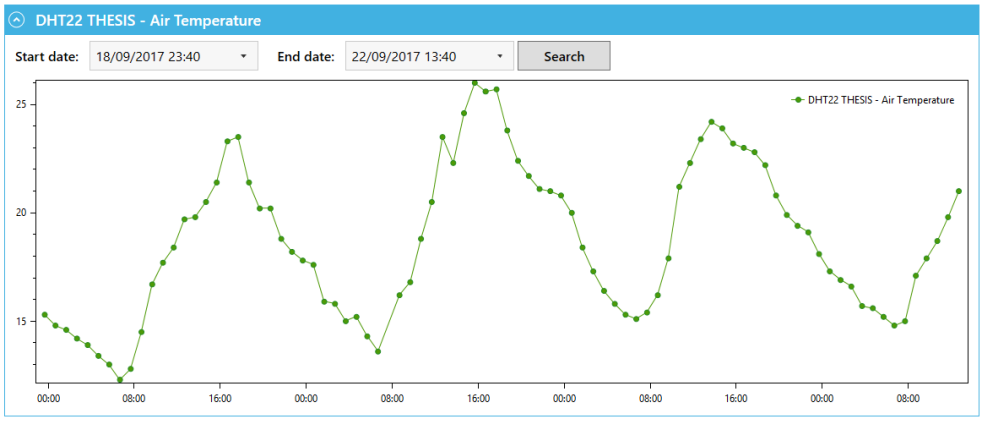

Figura 57 - Variação da temperatura (ºC) no período arquitetura de nuvem

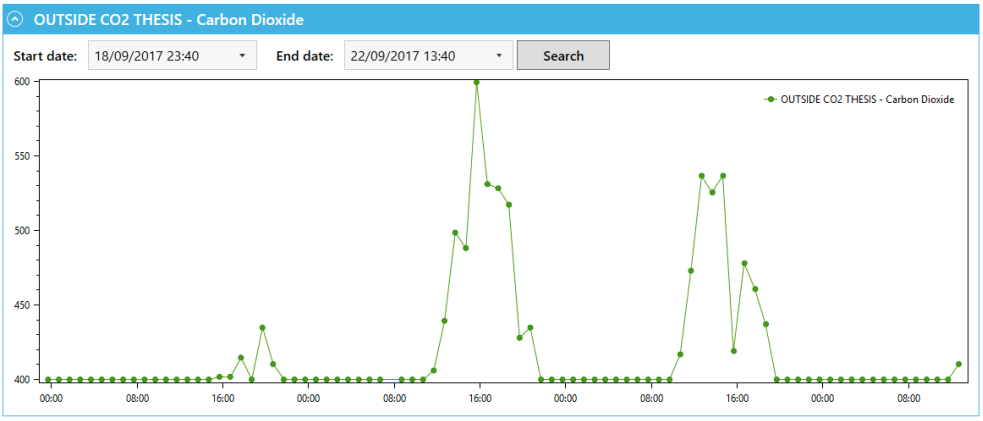

Figura 58 - Variação do CO2 (ppm) no período arquitetura de nuvem

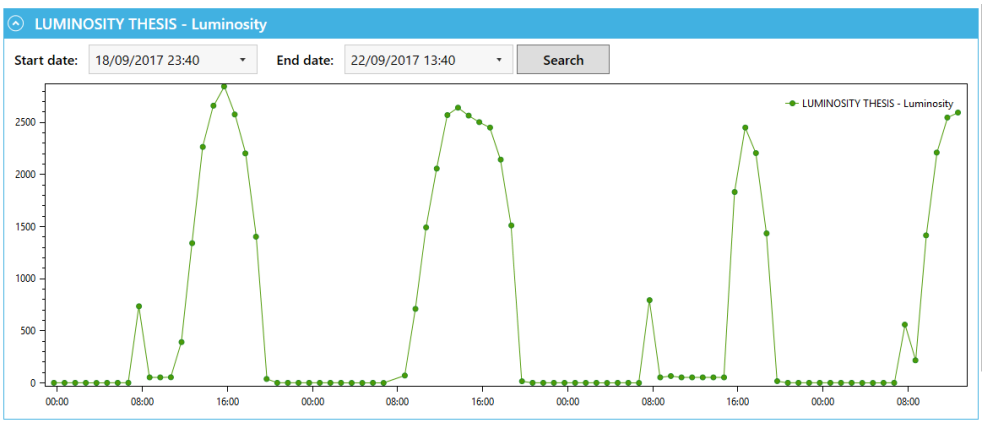

Figura 59 - Variação da luminosidade (lux) no período arquitetura de nuvem