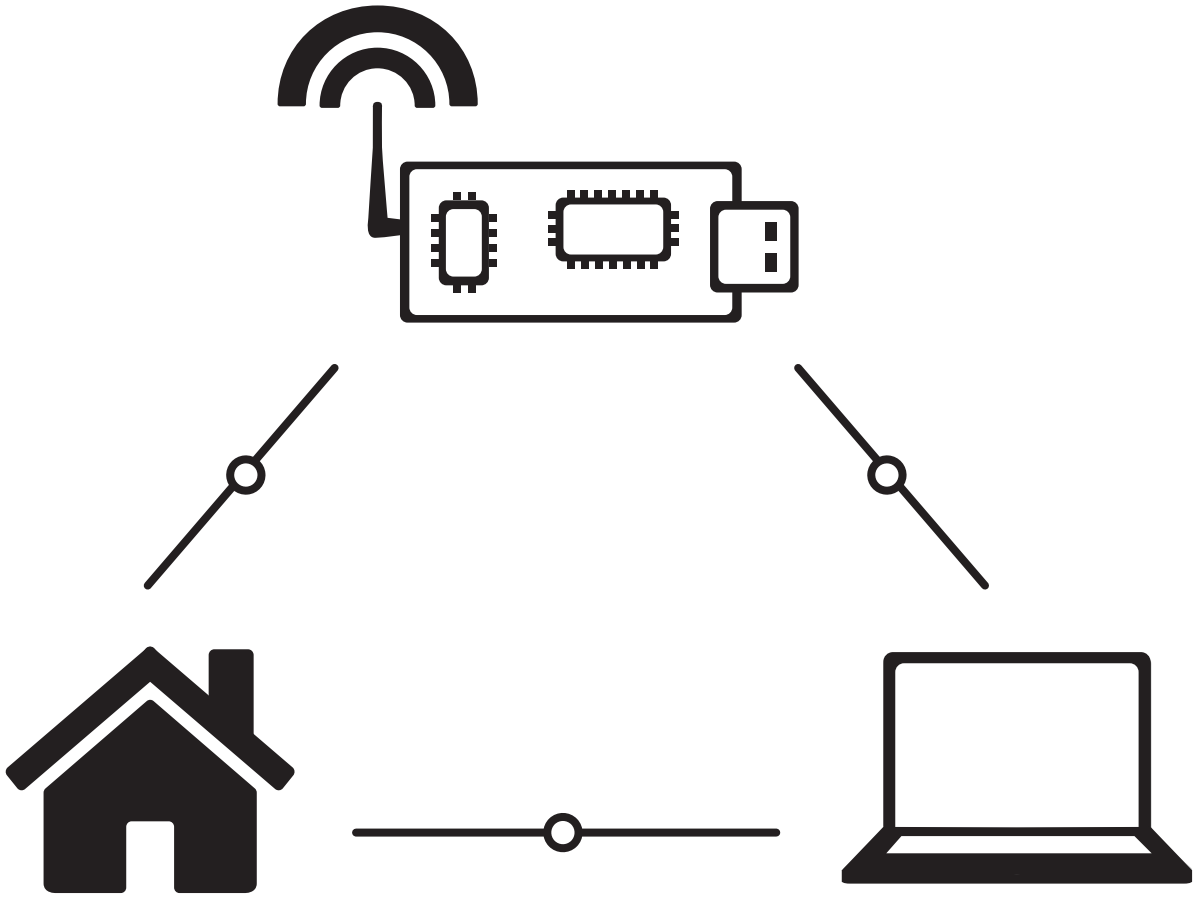

Hugo Fabião Pires da Ribeira

Desenvolvimento de uma rede de sensores para monitorização de idosos

Dissertação apresentada à Universidade de Coimbra para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Biomédica, orientada pelo Senhor Professor Doutor Mário Zenha-Rela e pelo Senhor Engenheiro Sérgio Santos.

Setembro de 2015

· U C ·

Universidade de Coimbra

Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Tese de Mestrado

Desenvolvimento de uma rede de sensores para monitorização de idosos

por
Hugo Ribeira

Sob a supervisão de

Prof. Mário Zenha Rela, Universidade de Coimbra
Eng. Sérgio Santos, Nourish Care

Tese submetida para obtenção do grau de mestre em Engenharia Biomédica pela Universidade de Coimbra.

14 de Outubro de 2015

Desenvolvida em colaboração com:

Nourish Care Systems — Bournemouth, United Kingdom

Abstract

First world population is aging at an astonishing pace, currently 5% of the european population is above their 80s and this number is expected to triple in the next 20 years. This aging will bring a new set of challenges in the elderly care sector, both social and financial. *Nourish Sensors* aims to bring a telemonitoring tool to aid the elderly maintain an independent and active life, as well as generate notifications when emergency situations occur. This work resulted in the integration of a sensor network installed in the habitation of the patient that transmits data over a 3G connection to a remote server. This data is then stored and analysed in order to infer a circadian cycle for the monitored person.

Keywords: *Sensors, life quality, senior, falls, telemonitoring, web technologies*

Resumo

A população mundial está a envelhecer a um ritmo alarmante. Na Europa, 5% da população tem mais de 80 anos e estima-se que este número venha a triplicar nos próximos 20 anos. Este envelhecimento populacional trará consigo um novo conjunto de desafios sociais e económicos para o panorama da prestação de cuidados. O *Nourish Sensors*, resultado deste projecto de tese de mestrado, constitui um sistema de tele-monitorização cujo o objectivo é auxiliar o idoso a manter uma vida activa e independente através da instalação de sensores na sua habitação. Estes sensores permitem inferir o ritmo circadiano do utente assim como coleccionar dados que farão parte de um sistema inteligente de geração de alarmes automático através da detecção de situações de emergência.

Palavras-Chave: *Sensores, qualidade de vida, idosos, senior, quedas, tele-monitorização, tecnologias web*

Agradecimentos

Esta tese representa o término da fase da minha vida responsável pela maior parte do que sou hoje.

Um obrigado às aleatoriedades que me traçaram um caminho repleto de pessoas e instituições, que determinantes na minha vida, seriam impossíveis de antever e calcular.

Obrigado à *Nourish Care* por me formar e empregar; e pelas pessoas que dela fazendo parte toldaram este último ano da minha vida.

Obrigado ao Prof. Doutor Mário Zenha-Rela pelas "achegas"oportunas e pelo incansável trabalho de orientação e revisão.

Obrigado ao João Diogo Costa, que mesmo não ciente, me deu força e "espicaçou"no momento mais definitivo. Que daqui a 20 anos continuemos assim!

À Lígia pelo apoio e carinho, pela contínua preocupação e, principalmente, pelo companheirismo. Obrigado.

A todos os meus amigos, estranhos, desconhecidos e momentos de sorte sem os quais não estaria aqui. As improbabilidades da vida são uma delícia que temo escassearem à medida que envelhecemos e caímos na rotina.

Ao imprevisível. Obrigado.

"Never run out of airspeed, altitude, and ideas all at the same time."

Desconhecido

Conteúdo

Abstract **v**

Resumo **vii**

Agradecimentos **ix**

Conteúdos **xiii**

Lista de Figuras **xvii**

Lista de Tabelas **xix**

Abreviaturas **xxi**

1 Introdução **1**
1.1 Projecto . . . 1
1.1.1 Motivação . . . 1
1.1.2 Objectivos . . . 2
1.1.3 Contextualização da Solução . . . 3
1.1.3.1 Fragilidades Motoras . . . 3
1.1.3.2 Fragilidades Cognitivas . . . 4
1.1.3.3 Função Renal . . . 5
1.2 Âmbito . . . 5
1.3 Organização do Documento . . . 5

2 Planeamento e Execução **7**
2.1 Planeamento e Gestão de Tarefas . . . 7
2.2 Repositório de Código Fonte . . . 8
2.3 *Test Driven Development* . . . 9
2.4 Configuração de Servidores . . . 9
2.5 Distribuição . . . 10

3 Estado da Arte **11**
3.1 Produtos Concorrentes . . . 12
3.2 Produtos Substitutos . . . 12
3.2.1 Botões de Alarme . . . 12
3.2.2 Prestação de Cuidados . . . 13

4 Solução **15**

4.1 Visão Geral . . . 15
4.2 Objectivos . . . 16
4.3 Descrição dos Componentes . . . 16
4.3.1 Sensores de Movimento PIR (Passive InfraRed) . . . 16
4.3.2 Sensores de Cama . . . 17
4.3.3 Hub . . . 17
4.3.4 Aplicação Web . . . 18
4.4 Métodos . . . 18
4.4.1 Monitorização Manual . . . 18
4.4.2 Monitorização Automática . . . 19
4.4.3 Gestão de Hubs . . . 19
4.4.4 Integração com o Sistema de Gestão *Nourish Care* . . . 20

5 Arquitectura **23**
5.1 Visão Geral . . . 23
5.2 Sensores . . . 24
5.2.1 Sensores de Movimento PIR . . . 24
5.2.2 Sensores de Cama . . . 25
5.3 Hub . . . 26
5.4 Servidor de Agregação de Dados . . . 27
5.4.1 Base de Dados . . . 27
5.4.2 IPA . . . 29
5.4.3 Aplicação Web . . . 30

6 Tecnologias **31**
6.1 Software da Plataforma de Agregação de Dados . . . 31
6.1.1 Base de Dados . . . 31
6.1.1.1 Bases de Dados Relacionais - SQL . . . 32
6.1.1.2 Bases de Dados Não-Relacionais - NoSQL . . . 32
6.1.2 Frameworks de Desenvolvimento Web . . . 33
6.1.3 Protocolos de Transmissão de Dados . . . 35
6.1.3.1 HTTP . . . 35
6.1.3.2 MQTT . . . 36
6.1.4 Formato de dados . . . 37
6.1.4.1 JSON . . . 37
6.1.4.2 XML . . . 38
6.1.4.3 Protocol Buffers . . . 38
6.1.5 Alojamento . . . 39
6.1.5.1 Alojamento de Servidor Físico . . . 39
6.1.5.2 Alojamento de Servidor Virtual . . . 40
6.1.5.3 Alojamento especializado . . . 40
6.2 Hardware para o Hub . . . 41
6.2.1 Feito à medida . . . 41
6.2.2 Micro-controladores . . . 42
6.2.3 Raspberry Pi e semelhantes . . . 42
6.2.4 Electric Imp . . . 43
6.3 Software do Hub . . . 44

6.3.1 C++ . . . 45
6.3.2 Python . . . 45

7 Resultados e Análise **47**
7.1 Simulador . . . 47
7.1.1 Consumo de dados . . . 47
7.1.1.1 Remoção de dados redundantes . . . 49
7.1.2 Envio de dados em lotes . . . 49
7.2 Nourish Sensors . . . 50
7.2.1 Consumo de Dados . . . 51
7.2.2 Sensores de Movimento . . . 51
7.2.3 Sensores de colchão . . . 53
7.2.4 Sensores de Porta . . . 55
7.2.5 Cruzamento de dados . . . 55
7.2.5.1 Validação . . . 56
7.3 Integração com a plataforma *Nourish Care* . . . 57

8 Conclusão **59**
8.1 Sistema de Monitorização . . . 59
8.2 Trabalho Futuro . . . 60
8.3 Considerações Finais . . . 61

Bibliografia **63**

Lista de Figuras

3.1 Comparação de botões de alarme 13

4.1 Funcionamento do sensor de movimento PIR 17
4.2 Visualização de Dados 19
4.3 Gestão de Hubs 20
4.4 Integração com *NourishCare* 21

5.1 Diagrama 24
5.2 Campo de visão do sensor de movimento PIR 25
5.3 Diagrama de classes do *firmware* do hub 26
5.4 Esquema de base de dados da aplicação segundo o modelo de *Bachman*. . 28

6.1 RaspberryPi 43
6.2 ElectricImp 43

7.1 Simulador 48
7.2 Inspector Web 48
7.3 Gráfico de Activação dos sensores de movimento 52
7.4 Gráfico de activação de 5 dias do sensor instalado na sala 52
7.5 Detalhe dos instantes seguintes à primeira detecção de movimento do dia 52
7.6 Resposta de um sensor de colchão à variação de pressão entre as suas extremidades 54
7.7 Aquisição do sensor de colchão durante a noite 54
7.8 Aquisição e normalização dos dados da totalidade dos sensores 56
7.9 Integração com *NourishCare* 58

Lista de Tabelas

1.1 Intervenientes no projecto 5

3.1 Produtos concorrentes 12
3.2 Produtos substitutos 14

5.1 Endpoints da IPA 29

6.1 Frameworks de Desenvolvimento Web 34
6.2 Formatos de transferência de dados 37

7.1 Envio de dados por lotes 50
7.2 Registo do utente 56

Abreviaturas

PIR	**P**assive **I**nfra**R**ed
REST	**RE**presentational **S**tate **T**ransfer
HTTP	**H**iper **T**ext**T**ransfer **P**rotocol
ACID	**A**tomicity **C**onsistency **I**solation and **D**urability
CRUD	**C**reate **R**ead **U**pdate **D**elete
SQL	**S**tructured **Q**uery **L**anguage
JSON	**J**ava**S**cript **O**bject **N**otation
XML	**X**tensible **M**arkup **L**anguage
VPS	**V**irtual **P**rivate **S**erver
SPI	**S**erial **P**eripheral **I**nterface
VPS	**V**irtual **P**rivate **S**erver
IPA	**I**nterface **P**rogramação **A**plicações

Capítulo 1

Introdução

Ao longo dos últimos 20 anos a esperança média de vida na Europa aumentou em cerca de 5 anos para uma média de 80,6 anos, sendo a percentagem da população acima desta mesma idade de 5%[1].

Espera-se que o envelhecimento gradual da população continue a avançar, exacerbando a pressão social e económica na manutenção da qualidade de vida desta faixa populacional. É de especial relevância a necessidade de encontrar mecanismos que sustentem a vida independente e activa dos idosos beneficiando a sua saúde e atrasando o recurso a serviços de cuidados continuados[2].

É neste contexto que o projecto desenvolvido no âmbito desta tese de mestrado se propôs a desenvolver um sistema de tele-monitorização com o objectivo de garantir a segurança e assistência de idosos na manutenção da sua autonomia diária, assim como na geração de informação útil para o acompanhamento do utente pela família.

1.1 Projecto

1.1.1 Motivação

A Nourish Care é uma empresa focada no desenvolvimento e comercialização de produtos de natureza tecnológica para apoio à prestação de cuidados sociais e de saúde. A empresa desenvolveu uma plataforma de integração de informação que suporta a avaliação das

carências de uma pessoa e permite planear os cuidados continuados, assim como recolher e gerir todos os registos dos serviços prestados por terceiros.

O produto existente é bastante poderoso no sentido em que permite a coordenação do apoio a um idoso por cuidadores formais assim como pela família. Foi neste contexto que a empresa decidiu explorar a possibilidade de utilizar monitorização não intrusiva no ambiente para aferir a possibilidade de utilizar tecnologia para, por um lado, monitorizar que a pessoa se mantém activa durante períodos em que se encontra sozinha, e por outro, garantir que o desvio de um comportamento normal despoleta os mecanismos de prevenção necessários.

Neste âmbito, os objectivo da empresa é o de conseguir detectar sinais de presença e de movimento no contexto de uma habitação para aferir o ritmo circadiano do utente. Este ritmo será então utilizado para 1) assegurar o bem-estar do utente 2) despoletar alarmes quando há desvios relevantes da rotina normal do paciente.

A empresa apoiou o projecto tanto no que toca à coordenação, alocação de recursos e materiais, assim como na criação de contexto do sector dos serviços sociais que permitiu que este projecto se desenrolasse com um enquadramento de uma aplicação real próxima do mercado.

1.1.2 Objectivos

Este projecto tem como objectivo desenvolver uma rede de sensores a colocar na habitação de idosos que vivam sozinhos e de um servidor agregador que gere alertas em tempo real a partir dos dados enviados pela rede de sensores.

O projecto está dividido em quatro fases, das quais esta tese abordará apenas as três primeiras:

1. Integração do hardware da rede de sensores e hub agregador;
2. Desenvolvimento do servidor central responsável pela agregação de dados;
3. Sincronização de dados entre a rede de sensores e o servidor central;
4. Geração inteligente de alertas em tempo real (Quedas, Inanimação, etc).

Inicialmente, pretendem-se integrar dois tipos de sensores: Sensores de movimento PIR (Passive Infrared Sensors) e sensores de colchão. Responsáveis, respectivamente, por detectar a presença de pessoas em divisões da habitação e na cama ou sofá. Sendo importante projectar o sistema de forma a facilitar uma futura integração de outros tipos de sensores.

1.1.3 Contextualização da Solução

Cada vez mais pessoas vivem para além dos 80 ou 90 anos. No entanto, as pessoas nestas faixas etárias tendem a ter fragilidades tanto no que toca às funções cognitivas, como à motricidade e equilíbrio.

Exploram-se de seguida algumas fragilidades mais recorrentes nesta faixa etária no sentido de evidenciar a utilidade da monitorização passiva dos idosos na manutenção da sua autonomia e bem-estar.

1.1.3.1 Fragilidades Motoras

As fragilidades motoras levam a que haja uma elevada incidência de quedas no ambiente doméstico, representando um decréscimo súbito de qualidade vida e perdas de independência[3]. As quedas entre os idosos constituem a razão mais frequente para hospitalização, e contribuem também para uma despesa avultada.

No Reino Unido, país para o qual há melhor disponibilidade de dados de custos do sistema de saúde, as quedas entre pessoas com mais de 65 anos causam uma despesa de £4.6 milhões por dia, ou seja mais de 1600 milhões de libras por ano[4].

A aplicação do produto resultante deste projecto neste contexto é a de recolher dados concretos sobre mobilidade ao longo do tempo, assim como diminuir o tempo de resposta a situações de emergência, podendo algumas das seguintes métricas ser de interesse:

- Tempos de transição entre sensores: o aumento sistemático de tempos de transição pode mostrar perda de mobilidade;

- Tempo de presença na cama ou num sofá: o aumento sistemático de tempo passado na cama ou num sofá pode indicar aumento de sedentarismo;

- Tempo de ausência da habitação: esta informação pode dar uma indicação de sedentarismo da pessoa;

- Ausência de actividade (pessoa dentro da habitação): a ausência de actividade pode indicar que houve uma queda – a relação de compromisso entre a velocidade de detecção e a redução de falsos alarmes é determinada pela observabilidade dada pela cobertura do sistema de sensores – se uma rede de sensores permite cobertura total então a ausência de actividade é um sinal forte de que houve uma queda. Num caso real, a ausência de actividade é frequente quando a pessoa não está a interagir com sensores, assim, a detecção de quedas terá que tomar em consideração os padrões de interacção habituais para uma pessoa ao longo do dia;

- Ausência de actividade (pessoa fora da habitação): o produto pode detectar que a pessoa não voltou a casa à hora esperada ou hora habitual.

1.1.3.2 Fragilidades Cognitivas

A incidência de doenças que afectam a capacidade cognitiva está a aumentar significativamente no mundo ocidental. A *Alzheimer Europe* aponta para 8,7 milhões de casos diagnosticados nos 28 países da união europeia, sendo provável a duplicação deste valor nos próximos 20 anos[5].

Pessoas com perdas de memória têm dificuldade em lembrar-se de manter actividades regulares como a toma de medicação, refeições ou o consumo de líquidos. A utilização de sensores que detectem e classifiquem as actividades permite estabelecer um ritmo circadiano para a pessoa e ajudá-la lembrando-a das actividades que já fez e das que estão por fazer, assim como alertar a família em caso de desvio relevante.

1.1.3.3 Função Renal

A incidência de infecções do tracto urinário é muito elevada em pessoas idosas, e complicações resultantes destas podem levar a admissões hospitalares, perda de função renal ou sepsia em casos extremos.

A frequência da visita à casa de banho, assim como as alterações súbitas de mobilidade podem indicar a presença de uma infecção do tracto urinário.

1.2 Âmbito

Esta tese pretende documentar o projecto desenvolvido na disciplina de Projecto, no ano lectivo de 2014/2015. Projecto esse que será apresentado à Faculdade de Ciências e Tecnologias da Universidade de Coimbra para a obtenção do grau de mestre em Engenharia Biomédica.

Os intervenientes responsáveis pela concretização do projecto encontram-se descritos na tabela abaixo.

Nome	**Função no Projecto**	**Contacto**
Hugo Ribeira	Estudante que executou o projecto	hugoribeira@gmail.com
Prof. Mário Zenha Rela	Orientador do projecto na FCTUC	mzrela@dei.uc.pt
Eng. Sérgio Santos	Supervisor do projecto na Nourish Care	sergio@nourishcare.co.uk
Prof. Miguel Morgado	Responsável pela coordenação dos projectos do MIEB	miguel@fis.uc.pt

Tabela 1.1: Intervenientes no projecto

1.3 Organização do Documento

Esta tese de mestrado tem início no capítulo actual cujo o objectivo é expor o contexto e motivação do projecto, assim como os objectivos que o mestrando se propõe a alcançar.

O capítulo 2 aborda as metodologias e planeamento do projecto.

O capítulo 3 descreve o estado da arte relativo às soluções existentes no mercado que se destinam a resolver o mesmo problema que o projecto em execução, assim como possíveis alternativas ao mesmo.

A solução de monitorização é descrita em termos funcionais no capítulo 4, onde se abordam cada um dos componentes assim como os métodos de interacção que estes expõem.

O capítulo 5, aborda a arquitectura do sistema, isto é, a compartimentação do sistema e a forma como é realizada a comunicação entre componentes.

O capítulo 6 faz uma análise das tecnologias disponíveis à implementação do sistema e detalha as escolhas efectuadas para cada componente, assim como para o modelo de comunicação.

Uma pequena aquisição de dados e análise é efectuada no capítulo 7 com o objectivo de fazer uma validação preliminar da utilidade dos dados na reconstrução do ciclo circadiano do utente.

Por fim, o capítulo 8 tece as conclusões finais, assim como algumas considerações pessoais em relação à execução do projecto e ao trabalho a realizar no futuro.

Capítulo 2

Planeamento e Execução

Este projecto caracterizou-se por um processo de desenvolvimento exploratório e iterativo. Os requisitos foram sendo definidos e aprimorados de forma orgânica com o desenrolar da implementação e em função das dificuldades encontradas, muito ao estilo de uma abordagem ágil[6] com objectivos semanais e testes frequentes que permitiram o refinamento do protótipo final.

Esta secção aborda algumas das ferramentas e metodologias que permitiram o desenvolvimento do *Nourish Sensors*.

2.1 Planeamento e Gestão de Tarefas

No início de cada semana foram decididas as tarefas e objectivos a realizar, usando o *Trello*[7] - ferramenta de gestão flexível baseada em listas e cartões, recorrentemente utilizado para gerir o processo de desenvolvimento de software.

O quadro do *Trello* foi dividido em várias listas, sendo que a alocação de uma tarefa a uma destas listas permite saber o estado de realização da mesma:

- **Backlog** - Esta lista acomoda tarefas que são identificadas como de interesse, mas para as quais ainda não foi alocado tempo de implementação. Funciona por isso como um repositório de ideias e a fazeres.

- **Current Week** - Esta lista é um repositório de tarefas que devem ser implementadas na semana corrente mas cuja a implementação ainda não foi iniciada. No início de cada semana são transferidas tarefas da lista de *Backlog* de forma a planear os objectivos semanais.

- **In Progress** - As tarefas cuja a implementação se esteja a desenrolar são transferidas para esta lista, permitindo saber em qualquer momento qual é o trabalho que está a ser realizado.

- **Code Review** - Uma vez concluída a implementação de uma tarefa, esta é transferida para a lista *Code Review*, onde aguarda um processo de escrutínio ao código com o intuito de encontrar defeitos.

- **Deployed** - Depois de executada a revisão do código e assegurada uma correcta implementação, a nova versão do código é posta em produção (através de um sistema de integração contínua detalhado no capítulo 2). Esta lista permite saber qual é a versão de produção actual, assim como a data de introdução de uma determinada funcionalidade; este registo é importante na identificação de falhas introduzidas no *software* por relacionar a data de introdução do defeito com a data de introdução de novas funcionalidades.

Esta informação é imprescindível para planear com sucesso cada um dos *sprints*[8] semanais, assim como manter as macro-tarefas necessárias à implementação do sistema em foco.

2.2 Repositório de Código Fonte

O uso de um repositório com a capacidade de realizar o versionamento de versões é imprescindível no acto de prototipagem, uma vez que além de registar todas as alterações no código desde o primeiro momento, permite de forma muito simples manter várias versões paralelas de cada componente.

O código fonte do *Nourish Sensors* foi gerido através do sistema de controlo de versões *Git* e hospedado pelo serviço *Github*[9]. Tendo sido utilizado um repositório privado por

componente de forma a facilitar a manutenção de versões diferentes entre componentes, já que uma alteração num deles não tem necessáriamente impacto nos outros.

2.3 *Test Driven Development*

O processo de desenvolvimento iterativo é por definição bastante dinâmico o que por consequência aumenta a possibilidade das alterações efectuadas introduzirem novos defeitos na aplicação. Torna-se por isso importante garantir o correcto funcionamento do sistema a cada nova iteração.

Test Driven Development[10] é uma metodologia de desenvolvimento em que os testes automatizados responsáveis por exercitar e garantir o correcto funcionamento do programa são escritos antes da própria funcionalidade.

A implementação de testes automatizados é um factor importante para o sucesso de um projecto de software[10]. No escopo deste projecto identificam-se os seguintes benefícios:

- Diminuir o número de defeitos em versões de produção da aplicação;
- Garantir que uma vez implementada uma funcionalidade e assegurado o seu bom funcionamento, não existem regressões introduzidas por alterações subsequentes;
- Os testes exercitam os casos de uso, funcionando assim como documentação que está sempre actualizada. Uma funcionalidade só está completa quando os seus testes não detectam qualquer erro;
- Diminuir a probabilidade de inconsistências na interface entre componentes. Num sistema multi-componente, os testes de integração funcionam como um contracto de comunicação garantindo que as respostas da *IPA* (Interface de Programação de Aplicações) conformam com o estabelecido entre aplicações.

2.4 Configuração de Servidores

O processo de instalação e configuração de um servidor é um processo moroso e propenso a erros é por isso um excelente candidato a ser automatizado.

A capacidade de configurar um servidor a partir do zero através do clique de um botão é um factor importante na resposta a uma necessidade de escalar o serviço e na diminuição do tempo de resposta a uma falha catastrófica, reduzindo também a possibilidade de uma configuração inconsistente e poupando tempo ao programador.

A configuração de servidores foi automatizada usando a ferramenta *Ansible*[11].

2.5 Distribuição

No seguimento de uma metodologia ágil e da distribuição frequente de novas versões dos componentes faz sentido que este processo seja facilitado e, de preferência, automático.

A distribuição frequente de novas versões permite que as alterações sejam efectuadas de forma gradual expondo possíveis problemas mais cedo e facilitando a sua resolução - alterações mais pequenas e regulares tornam mais fácil isolar a alteração responsável por um determinado defeito.

No sentido de desdramatizar o processo de *deploy*, optou-se por integrar o serviço *Wercker*[12] com a plataforma de repositórios de código *Github*. A actualização da versão *master* do código no repositório despoleta um *script* escrito em *Ruby* que é responsável por actualizar o código das diversas aplicações e reiniciar os servidores.

Capítulo 3

Estado da Arte

O projecto desenvolvido no âmbito desta tese deu origem a um sistema de tele-monitorização para idosos conforme os requisitos da Nourish Care. Faz sentido no entanto que se analisem os sistemas existentes no mercado de forma a conhecer as suas vantagens, desvantagens e limitações e usar esse conhecimento para alavancar o processo de desenvolvimento.

Este capítulo começa assim com a análise de alguns produtos concorrentes e termina com a análise de possíveis produtos substitutos.

3.1 Produtos Concorrentes

Várias empresas comercializam soluções de tele-monitorização para idosos. Os líderes de mercado no espaço Europeu são empresas como a *Tunstall Healthcare*, a *Appello*, ou a *Just Checking Ltd*[13]. Todos estes fabricantes fornecem sistemas que monitorizam os movimentos do utente ao longo do dia na sua habitação.

A tabela 3.1 faz uma análise comparativa das soluções de *telecare* propostas por estas empresas.

	Sensores	**Visualização**	**Eventos**	**Alertas**	**Preço**
Tunstall[14]	Movimento, Presença na Cama, Botão de Pânico, Temperatura e outros	Sem informação disponível	Entradas e saídas, visitas	SMS, Email	Sem informação disponível
Appelo[15]	Movimento, Luz, Temperatura	Sem informação disponível	Entradas e saídas, visitas, anormalidade de parâmetros (luz, temperatura, visitas programadas que não ocorreram e outros)	SMS, Email	279£ + 15£/mês
Just Checking[16]	Movimento, Presença na Cama	Gráfico de activação dos sensores	Entradas e saídas, visitas, horários anormais	SMS	30£/mês

Tabela 3.1: Produtos concorrentes

No caso da *Tunstall* e da *Appello*, estas empresas fornecem o mercado dos lares e dos cuidados domiciliários, instalando sensores de movimento, abertura de portas, presença em sofá e em cama, como parte de um sistema que inclui a monitorização por parte de equipas de centros de atendimento, ou por parte de equipas de cuidados domiciliários. Os sensores fornecidos por estes fornecedores utilizam sistemas de comunicação proprietários, fazendo com que a expansão e adaptação de instalações existentes sejam caras, ou mesmo impossíveis.

3.2 Produtos Substitutos

3.2.1 Botões de Alarme

Os botões de alarme são dispositivos electrónicos com o formato de uma pulseira ou colar e que permitem gerar alarmes de forma manual pelo utente ou através de algoritmos de detecção de quedas.

A estes dispositivos está normalmente associada uma mensalidade que assegura a monitorização da geração de alarmes por um *Call Center* especializado em urgências médicas e prestação de apoio a idosos.

A aquisição destes aparelhos é normalmente efectuada por familiares do utente e segue-se à ocorrência de uma queda.[17]

A figura 3.1, retirada de um artigo[18] que compara e auxilia o utilizador na escolha do dispositivo certo para cada utente, ilustra as funcionalidades e custos associados aos botões de alarme mais vendidos do mercado.

Facts to consider	**Life Alert**	**LifeStation**	**Medical Alert**	**MobileHelp**	**Philips Lifeline**	**Rescue Alert**
Monthly service cost						
Landline/ Cellular	$50/ $60	$26/ $33	$30/ $35	NA/ $35	$30/ $42	$29/ $43
GPS mobile[1]	$70	$30 ($50 device fee)	$40	$38	$55 (landline); $65 (cell) ($149 device fee)	$45 ($90 device fee)
Features						
Range (in feet)	300	500	600	350 to 600	600	600
Mobile 911 phone[2]	Yes	Yes	Yes	No	No	Yes
Automatic fall detection[3]	No	No	Yes	Yes	Yes	No
Fees						
Minimum obligation	36 months[4]	30 days	90 days	None	None	None
Activation	$95	None	None	None	$0 to $50	None
Cancellation fee	Remainder of contract	None	None	None	None	$0 to $25[5]
Monitoring services						
In-house or outsourced	In-house	In-house	Outsourced	Outsourced	In-house	In-house
UL-listed (or comparable)	Yes	Yes	Yes	Yes	No	Yes[6]

FIGURA 3.1: Comparação de botões de alarme

A tabela apresenta-se em inglês para manter a coerência com o artigo original.

3.2.2 Prestação de Cuidados

Qualquer produto ou serviço de prestação de cuidados continuados a idosos é um possível substituto do Nourish Sensors. Estes serviços caracterizam-se pela presença constante

ou periódica de uma ou mais pessoas junto do idoso, ou pela sua institucionalização em lares e centros de dia.

A tabela abaixo analisa alguns destes serviços. Os preços médios apresentados são referentes ao Reino Unido e provenientes do relatório de mercado Laing & Buisson[19].

	Descrição	**Período do dia coberto**	**Preço Médio**
Apoio de Familiares	Um ou mais familiares fazem visitas ao idoso	Dependente da disponibilidade do familiar	N/A
Co-habitação com familiares	O idoso passa a residir com um ou mais familiares.	Dependente da disponibilidade do familiar	N/A
Serviço de apoio/acompanhamento ao domicílio	Um ou mais cuidadores visitam e/ou co-habitam com o idoso na sua própria casa.	0 a 24 horas / dia	13£ / hora
Centros de dia	A instituição presta cuidados durante o dia ou idoso que regressa à sua habitação para passar a noite	10£ / hora	
Residência	A instituição presta apoio total ao idoso em regime de internamento	24h / dia	525£ / semana

Tabela 3.2: Produtos substitutos

Dos serviços apresentados segregam-se dois grandes grupos: aqueles dependentes da disponibilidade familiar e os serviços profissionais de instituições de prestação de cuidados. Podem também dividir-se as várias soluções entre aquelas que obrigam o realojamento do idoso e as que lhe permitem manter a sua residência.

Capítulo 4

Solução

A *Nourish Sensors* é um esforço da *Nourish Care* no sentido de desenvolver uma solução de monitorização e alerta não invasiva para idosos com o objectivo de prolongar a sua vida autónoma e diminuir o tempo de resposta em situações de doença aguda ou acidente.

4.1 Visão Geral

A solução é composta por uma rede de sensores capaz de monitorizar a presença e movimento do utente na sua habitação e transmitir esta informação para um serviço remoto que permite a visualização e análise do bem-estar do utente em tempo-real.

A interacção do utente com o sistema é feita de forma passiva, isto é, à medida que o idoso se movimenta e realiza as tarefas do dia-a-dia na sua habitação, são armazenados os padrões de movimento entre divisões e/ou a presença em sofás e camas sem que este interaja directamente com o sistema.

Na habitação do utente, para além dos sensores, é instalado um *hub* cuja responsabilidade é centralizar a amostragem de cada um dos sensores e enviar de forma automática esta informação para um servidor remoto através de uma rede 3G.

O servidor remoto expõe uma interface *RESTful*[20] que permite à rede de sensores sincronizar os dados de movimento do utente e implementa uma aplicação *web* que permite gerir e visualizar os hubs e os seus respectivos dados.

Esta aplicação de gestão e visualização permite o acesso aos dados dos sensores instalados em casa do utente através de qualquer dispositivo com um navegador web e uma ligação à internet (computadores, telemóveis, tablets, etc), permitindo, por exemplo, a um cuidador lançar um alerta em caso de inactividade prolongada do utente.

4.2 Objectivos

1. Monitorizar em tempo-real o movimento e presença de um utente numa divisão da sua habitação.
2. Demonstrar a capacidade de inferir o bem-estar de um utente através da aquisição de dados provenientes de sensores de movimento e presença.
3. Demonstrar a capacidade de inferir eventos de doença aguda ou acidentes (e.g. quedas, ferimentos) através de sensores de movimento e presença.
4. Demonstrar que a monitorização em tempo-real do utente contribui para a manutenção da sua autonomia
5. Integração com o sistema de gestão da *Nourish Care* com o objectivo de geração de eventos em tempo-real (e.g. levante, presença/ausência na habitação).
6. Implementação de um sistema cuja base seja aproveitável no desenvolvimento de um mecanismo de classificação automática.

4.3 Descrição dos Componentes

A solução proposta pressupõe um meio de recolha de dados (os sensores), um meio de comunicação, um serviço de armazenamento e um meio de visualização que possibilite a análise dos dados recolhidos.

4.3.1 Sensores de Movimento PIR (Passive InfraRed)

Estes sensores medem alterações da dispersão de radiação infravermelha no seu raio de visão, funcionando efectivamente como detectores de movimento.

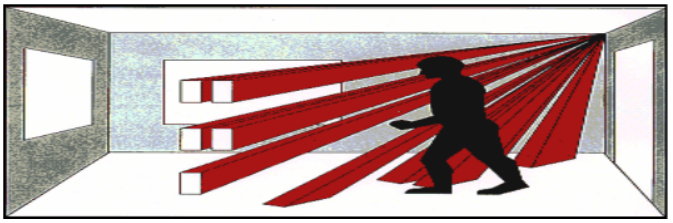

FIGURA 4.1: Funcionamento do sensor de movimento PIR

A detecção de movimento numa divisão é uma métrica eficaz da ocupação de uma divisão e também da actividade do utente dentro da mesma, uma vez que um indivíduo estático não faz o sensor mudar de estado.

A sua instalação em cada uma das divisões da habitação do utente permite mapear os movimentos (e a ausência de movimento) do utente durante o seu dia-a-dia, assim como padrões de inactividade anómalos.

4.3.2 Sensores de Cama

Os sensores de cama consistem numa banda resistiva rectangular de 20cm de altura e largura variável. Estes sensores registam deformações e, quando colocados num colchão ou sofá, permitem inferir a presença de um indivíduo assim como o seu posicionamento relativo.

No âmbito deste projecto, é utilizado com o fim de saber a ocupação de camas e sofás, assim como inferir indirectamente a agitação do utente durante a noite.

Em complemento aos sensores de movimento, é assim possível mapear tanto os períodos de actividade do utente como os períodos de sedentarismo.

4.3.3 Hub

O hub é responsável por fazer a interface entre os sensores e o servidor de agregação através de uma rede 3G e consiste num microcomputador capaz de correr o sistema operativo linux e com acesso a pinos digitais de leitura e escrita.

Este microcomputador oferece ainda conectividade USB, essencial na interface com módulos 3G necessários à comunicação com o servidor de agregação.

4.3.4 Aplicação Web

A aplicação web é uma instância de um servidor virtual alojada na nuvem, cuja função é armazenar e disponibilizar os dados provenientes dos sensores instalados em casa dos utentes.

Numa instância futura, esta aplicação deverá ter a capacidade de classificar eventos e gerar alertas de forma automática.

4.4 Métodos

A *Nourish Sensors* pretende diminuir o risco e as consequências de uma crescente população envelhecida a habitar de forma isolada e, por outro lado, implementar um ciclo de realimentação que potencie a confiança do utente na reabilitação para uma vida autónoma.

Podemos, por isso, distinguir um processo de prevenção e monitorização e um processo de resposta a acidentes e doenças agudas.

4.4.1 Monitorização Manual

Numa primeira instância, espera-se que um utilizador (e.g. cuidador ou familiar do utente) faça uma análise manual dos dados disponibilizados pela aplicação web de forma a caracterizar e correlacionar eventos com acontecimentos na vida do utente (mesmo que *a posteriori*).

Antevê-se que este sistema de monitorização seja capaz de revelar tendências através das alterações graduais dos padrões de movimento sejam elas positivas e em direcção a uma maior actividade física ou negativas e em direcção ao sedentarismo e inactividade do utente, assim como isolar eventos agudos através do registo de padrões de actividade ou inactividade irregulares - e.g. 1) o utente levanta-se a meio da noite e não volta à cama 2) o utente está há mais de duas horas na casa de banho.

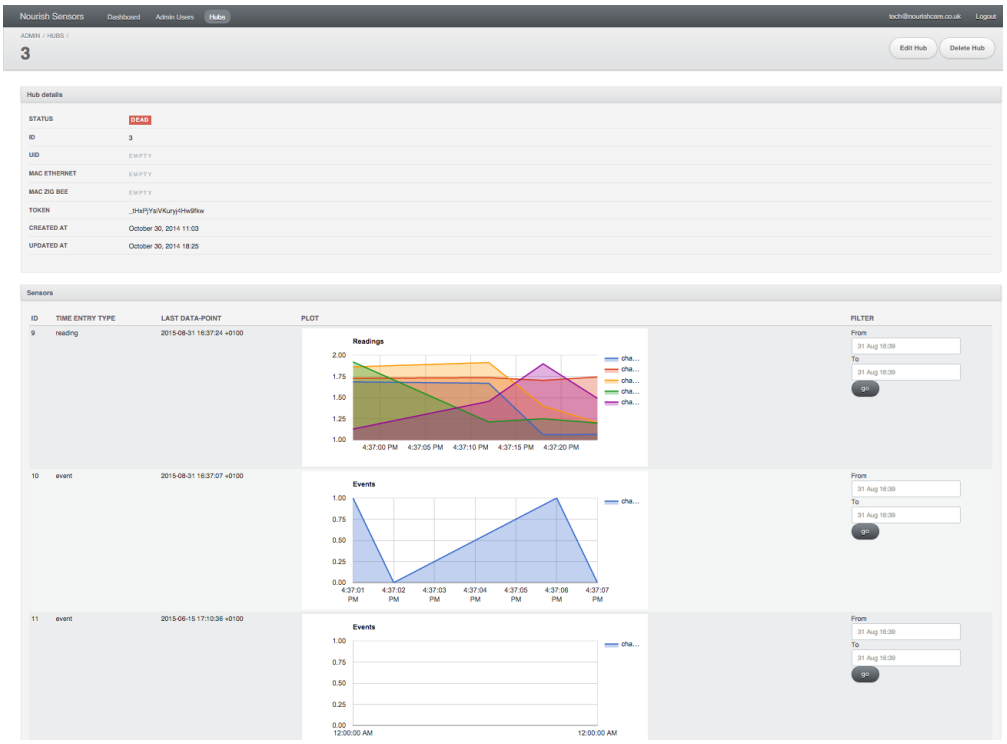

FIGURA 4.2: Visualização de Dados

4.4.2 Monitorização Automática

É relevante referir que o objectivo desta primeira implementação do sistema é tanto demonstrar o conceito e a viabilidade da aplicação como alicerçar a construção de um sistema de detecção e classificação automática.

A aquisição de dados e a sua classificação manual permitirá encontrar métricas relevantes no estudo dos utentes e desenhar soluções de inteligência artificial que permitam classificar e gerar alertas em tempo-real de forma automática.

A classificação automática de eventos está fora do escopo desta tese.

4.4.3 Gestão de Hubs

A gestão de hubs é feita através da aplicação web que permite a um administrador listar, registar e apagar hubs do sistema, assim como consultar os dados de cada um dos seus sensores.

Esta aplicação permite ainda identificar hubs com falhas de funcionamento.

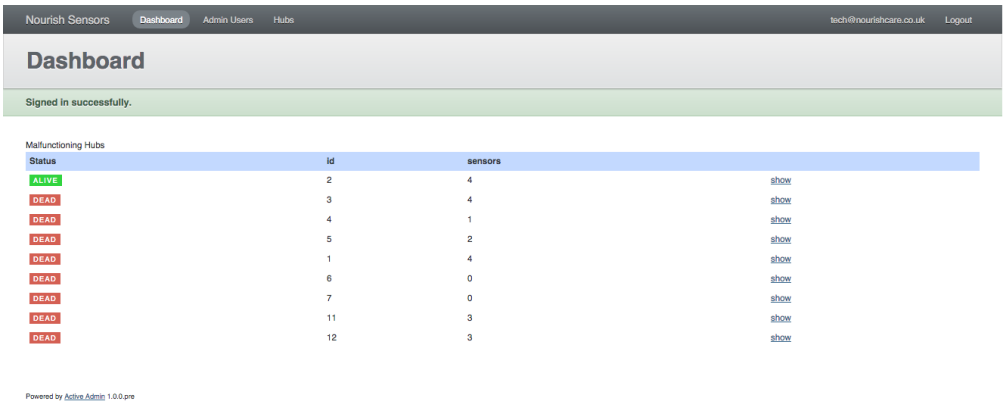

FIGURA 4.3: Gestão de Hubs

4.4.4 Integração com o Sistema de Gestão *Nourish Care*

A plataforma de gestão integrada da *Nourish Care* permite gerir e planear cuidados em lares e empresas de apoio ao domicílio. É de especial importância nestas últimas ter acesso a alguns dados em tempo-real do utente a quem se presta serviços como é o caso de 1) saber se este se encontra em casa 2) saber se está acordado.

De forma a demonstrar a utilidade deste sistema, procedeu-se à integração da *Nourish Sensors* com a *Nourish Care* utilizando para isso a *IPA* (Interface de Programação de Aplicações) *JSON* (Javascript Object Notation) disponibilizada pelo sistema de gestão.

Neste caso foi sincronizado o evento de presença na cama com o sistema de tarefas da plataforma de gestão, dando assim ao Cuidador a possibilidade de saber se o utente está acordado antes de se deslocar à sua habitação.

A figura 4.4 demonstra a integração dos dois sistemas.

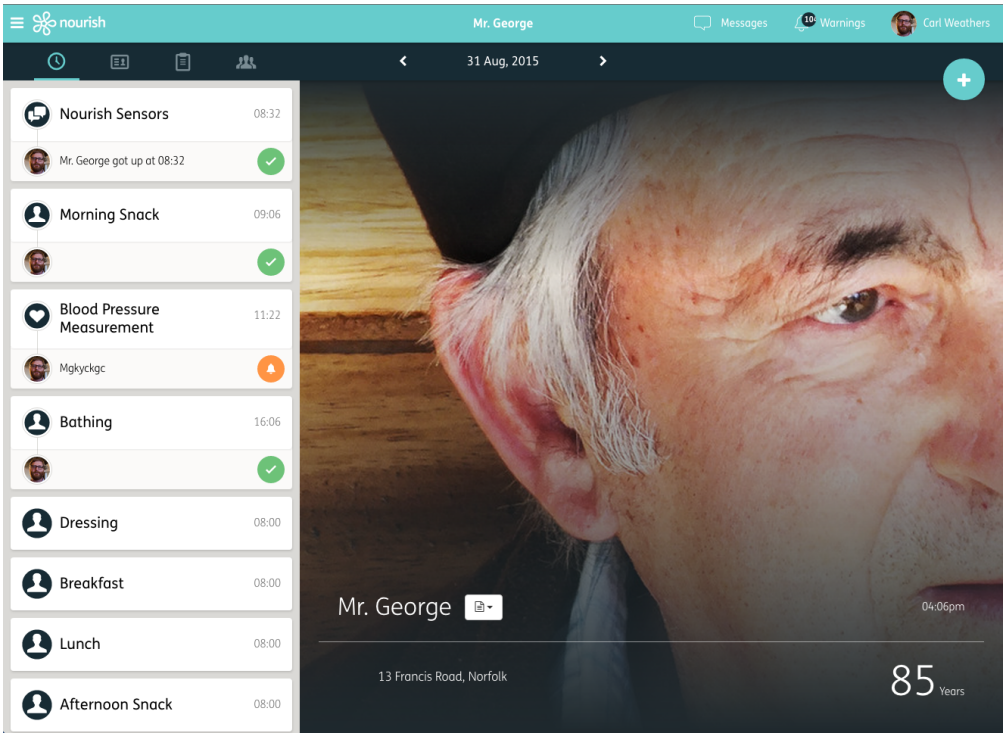

FIGURA 4.4: Integração com *NourishCare*

Capítulo 5

Arquitectura

Este capítulo descreve a arquitectura do sistema de aquisição e agregação de dados, assim como da plataforma web que lhe dá suporte.

De forma a tornar o capítulo mais breve e claro, considerações relativas a escolhas de tecnologia e implementação são detalhadas no capítulo 6, relativo ao design e implementação do sistema.

5.1 Visão Geral

As funcionalidades descritas no capítulo 4 deram origem ao desenho de um sistema com os seguintes componentes e responsabilidades:

1. **Sensores** - interface passiva entre o utente e o sistema informático. São responsáveis por digitalizar acontecimentos físicos e devem ser amostrados de forma a recolher os seus dados

2. **Hub** - interface entre os sensores e a plataforma web. É responsável por amostrar os sinais usando um ADC (Analog Digital Converter), armazená-los temporariamente e enviá-los através de uma rede 3G para o servidor de agregação de dados.

3. **Base de Dados** - Armazena e garante a integridade dos dados. São guardados dados relativos aos hubs, sensores e chaves de ligação aos utentes da plataforma *Nourish Care*.

4. **IPA** - interface HTTP[21] (HyperText Transfer Protocol) exposta pelo servidor de agregação de dados para que os hubs possam autenticar-se e enviar os seus dados.

5. **Aplicação Web** - interface com os cuidadores e administradores da plataforma. Permite a gestão dos hubs e visualização de dados.

Esta arquitectura segue o modelo cliente-servidor tanto nos seus macro-componentes (hub e navegador web são clientes que comunicam com o servidor de agregação de dados) como nos constituintes da plataforma web - a base de dados é um servidor ao qual a IPA e a aplicação web se ligam como clientes.

A comunicação entre o hub e a IPA é efectuada sobre HTTPS[21], usando o protocolo de dados JSON.

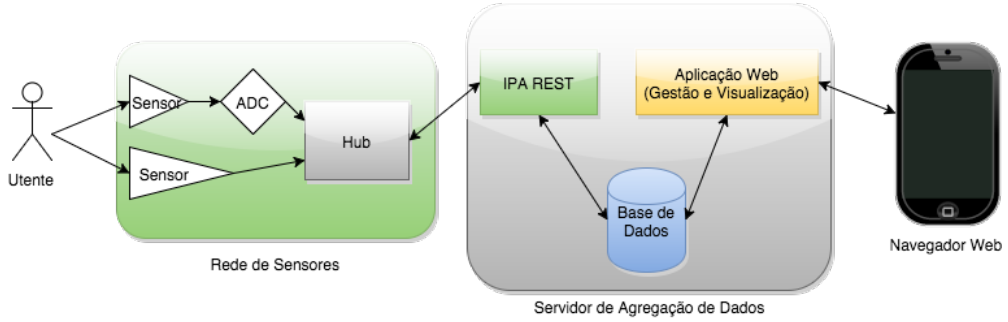

FIGURA 5.1: Diagrama

5.2 Sensores

No âmbito desta tese serão utilizados dois tipos de sensores: sensores de movimento PIR e sensores de presença na cama, os quais são apresentados nas secções abaixo.

5.2.1 Sensores de Movimento PIR

Os sensores de movimento PIR[22] foram introduzidos por volta de 1940, tendo sido usados originalmente para aplicações militares e científicas. Desde então, encontram-se aplicações da tecnologia em diversas áreas desde a domótica aos sistemas de intrusão [22].

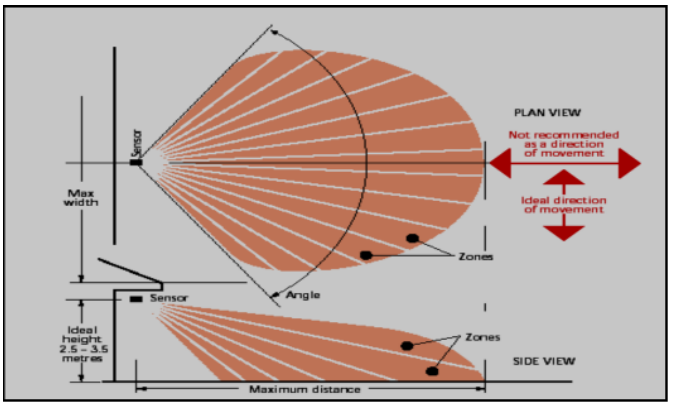

FIGURA 5.2: Campo de visão do sensor de movimento PIR

Estes sensores são interruptores electrónicos cujo estado se altera quando no seu campo de visão existem objectos em movimento. Este movimento é detectado através das alterações que causa na dispersão da radiação infravermelha que chega ao sensor[22].

O campo de visão destes sensores varia de fabricante para fabricante, de acordo com a temperatura ambiente e o tamanho da fonte de calor pelo que a escolha do sensor deve ter em conta a área a cobrir e a direcção principal espectável do movimento a detectar tal como representado na figura 5.2.

Nesta integração foram utilizados sensores da *DSC*[23] com um ângulo de detecção horizontal de 85º.

5.2.2 Sensores de Cama

Os sensores de cama consistem numa tira preenchida com bandas de resistência variável. Esta resistência varia de acordo com a deformação à qual o sensor está sujeito, correspondendo uma resistência infinita ao estado de não deformação e uma resistência muito pequena ao estado de deformação máxima (cerca de 100 Ω).

Este sensor é multi-canal, correspondendo o número de canais ao número de resistências que o sensor implementa. Os sensores de cama disponíveis para teste neste projecto possuem 5 canais.

Tratando-se de um sensor analógico a sua interface com o micro-controlador foi feita através de um *ADC*.

5.3 Hub

O hub é responsável por fazer a leitura de dados provenientes dos sensores e armazená-los até serem enviados com sucesso para o servidor de agregação de dados. Assim, podemos dividir as suas responsabilidades em dois focos distintos: Aquisição de dados e sincronização com o servidor remoto.

Tendo em conta os requisitos funcionais, é necessário que o hardware possua pinos de *GPIO* e alguma forma de comunicação de dados via rede.

A arquitectura de classes do *firmware* do hub foi criada de acordo com o príncipio de responsabilidade única, ou seja, cada classe deve ser responsável por uma e uma só tarefa.

A figura 5.3 representa as classes mais relevantes do programa.

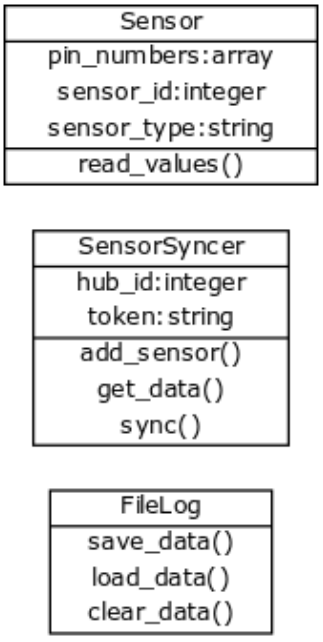

FIGURA 5.3: Diagrama de classes do *firmware* do hub

Sensor - Representa um sensor e é responsável por ler valores de um ou mais pinos do *ADC* ao qual estão ligados os sensores.

SensorSyncer - Representa um hub e é responsável por fazer a sincronização dos dados dos sensores para o servidor remoto via *API HTTP*. Em caso de falha de envio, os dados são persistidos utilizando o objecto *FileLog* para serem processados mais tarde.

FileLog - É responsável por abstrair os detalhes de persistência e consulta de dados gravados no disco. É utilizado pelo *SensorSyncer* para persistir os dados de sensores cujo o envio falhou.

5.4 Servidor de Agregação de Dados

O serviço web é composto pela base de dados, IPA e a aplicação web. A base de dados é partilhada entre ambos os sistemas, sendo que a IPA funciona como meio de introdução de dados dos sensores e a aplicação web como meio de configuração e visualização.

Ambos os componentes seguem uma arquitectura RESTful[20] sobre o protocolo HTTP(S). Esta arquitectura define uma estrutura standard para os *endpoints* da aplicação sendo no entanto agnóstica em relação ao formato dos dados em si - que no caso da IPA são JSON e no caso da aplicação web HTML.

5.4.1 Base de Dados

A base de dados é um componente fulcral do sistema, sendo responsável por armazenar e permitir a consulta de informação. A implementação de uma base de dados relacional inicia-se com a definição do esquema de dados.

O esquema de base de dados define a estrutura de dados, as ligações entre as várias entidades e as possíveis restrições que garantem a integridade dos dados (restringir por exemplo o valor de uma tensão a números positivos ou a obrigatoriedade de presença do campo de registo de hora de um evento). A tentativa de introdução de dados que não conformem com o esquema de base da dados resulta num erro, garantindo assim que apenas dados válidos são armazenados.

O diagrama da figura 5.4 representa o esquema da base de dados da aplicação.

AdminUsers - Esta tabela armazena os dados dos administradores da aplicação. Uma vez que as entradas desta tabela serão utilizadas para autenticar o utilizador, guardaremos o *email* e uma transformação unidireccional da *password*. A *password* não é guardada directamente de forma a mitigar possíveis ataques à base de dados, garantindo

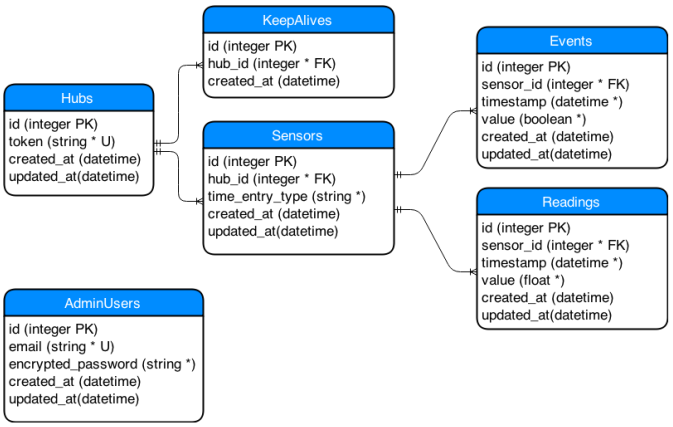

FIGURA 5.4: Esquema de base de dados da aplicação segundo o modelo de *Bachman*.

que mesmo que um utilizador externo mal intencionado consiga acesso à mesma não tenha acesso às palavras-passe dos utilizadores. Existem ainda algumas restrições como é o caso da presença obrigatória dos campos de *email* e *encrypted_password*, o *email* tem ainda uma restrição de unicidade que garante a sua utilização como identificador único aquando do processo de autenticação.

Hubs - Esta tabela armazena dados correspondentes aos hubs. A coluna *token* é utilizada como autenticação e incluída em cada pedido, este valor é gerado aquando da criação de uma nova sessão.

Um hub tem ainda duas relações de um-para-muitos: *KeepAlives* e *Sensors*. Isto é um hub pode ter zero ou mais sensores e zero ou mais *keep-alives*. Estes últimos são usados na aplicação como garantia de que o hub está a funcionar mesmo que não tenha dados a reportar.

KeepAlivee - Esta tabela da base de dados é responsável por armazenar *timestamps* referentes à última comunicação de cada hub.

Sensors - Cada linha desta tabela representa um sensor, sendo que para ser válido tem de estar relacionado com um *hub*. A coluna *time_entry_type*, que é obrigatória, especifica o tipo de sensor à qual a entrada se refere.

Os sensores relacionam-se de um para muitos com as tabelas de *events* e *readings*, sendo que a determinada altura um sensor deve apenas estar relacionado com *events* ou *readings*. Esta validação é feita por lógica de negócio da aplicação.

Events - Esta tabela representa cada evento gerado por um sensor de movimento, a noção de evento neste contexto está directamente ligada à mudança de estado do sensor, sendo o seu valor binário. Guardam-se dois campos de data e hora, um referente à data e hora de criação do registo na base de dados (*created_at*) e outro referente à hora de registo do evento (*timestamp*) que é enviado pelo hub.

Readings Esta tabela é em tudo semelhante à tabela de *Events*, alterando apenas o tipo de dados do valor para número decimal.

Em relação aos *Readings e Events* é pertinente discutir a escolha duas tabelas distintas ao invés de adicionar dois campos de valor, um binário e o outro um número decimal.

Em relação à complexidade e performance do modelo de dados esta escolha não é relevante, no entanto, em termos semânticos e de operações de dados, cada uma das tabelas vai sofrer operações diferentes pelo que faz sentido separá-las. É, além do mais, possível gerar um evento a partir de uma ou mais leituras (pense-se na transformação de um período de leituras do sensor de colchão em eventos de deitar/levantar) o que reforça a ideia de que os dados pertencem a classes (e por isso a tabelas) diferentes.

5.4.2 IPA

A IPA constitui uma camada no topo da base de dados e é responsável por estruturar, autenticar e permitir a introdução de dados por aplicações externas.

A arquitectura *RESTful* no desenho dos *endpoints* desta interface facilita a interoperabilidade com aplicações externas uma vez que a sua especificação é clara em relação à estrutura da IPA e da localização de recursos assim como da negociação do formato de dados.

Posto isto, a IPA expõe os seguintes *endpoints*:

Endpoint	**Acção HTTP**	**Descrição**
/hubs/:id/auth	POST	Criação de um token para autenticação
/hubs/:id/keep_alive	POST	Criação de um novo atestado de funcionamento
/hubs/:id/time_entries	POST	Introdução de novos dados

Tabela 5.1: Endpoints da IPA

5.4.3 Aplicação Web

Este módulo implementa funcionalidades que permitem a um administrador configurar e monitorizar a rede de hubs instalada em casa dos utentes. Expõe ainda os dados dos sensores por hub de forma visual.

Apresentam-se de seguida os casos de uso para este módulo:

Gerir hubs

- Listar os hubs existentes na base de dados
- Criar um novo hub
- Ver os dados de um hub em detalhe, incluindo os seus sensores
- Editar os dados de um hub
- Remover um hub existente

Gerir sensores de um hub

- Listar os sensores existentes na base de dados
- Criar um novo sensor
- Ver os dados de um sensor em detalhe
- Editar os dados de um sensor
- Remover um sensor existente

Visualizar dados de um hub

- Ver gráfico de activação de cada sensor de movimento
- Ver gráfico da deformação do sensor de colchão ao longo do tempo
- Consultar dados num intervalo de tempo específico

À excepção da visualização de dados em gráficos, os dois recursos que se pretende gerir (hubs e sensores) implementam as acções *CRUD* (Create, Remove, Update and Delete).

Capítulo 6

Tecnologias

Uma vez identificados os componentes do sistema e a forma como se interrelacionam - ou seja, a sua arquitectura - torna-se necessário tomar decisões em relação às tecnologias e componentes que vão integrar o sistema.

Este capítulo aborda, para cada componente do sistema, as várias alternativas e a motivação que levou a cada uma das escolhas.

6.1 Software da Plataforma de Agregação de Dados

Este serviço é constituído por uma camada de dados responsável pela persistência da informação enviada pelos hubs e uma camada funcional responsável por implementar e expor online uma *API REST* via *HTTP*.

Embora existam no mercado alguns serviços prontos *out-of-the-box* para persistência de dados provenientes de sensores (tais como *TempoIQ*[24] ou *OrchestrateIO*[25]) estes não serão tidos em conta por limitações impostas pela *Nourish Care* - apenas uma solução feita à medida permite garantir a resposta em tempo útil a novos requisitos e a uma possível necessidade de certificação.

6.1.1 Base de Dados

A base de dados implementa a camada de persistência de dados de uma aplicação e expõe uma interface que permite realizar operações *CRUD* em colecções destes dados.

De forma a reduzir custos e aumentar a flexibilidade da implementação a *Nourish Care* impôs como requisito o uso de software *open source*.

Existem duas grandes arquitecturas no que diz respeito a base de dados: A relacional e a não-relacional que serão abordadas nos parágrafos seguintes.

6.1.1.1 Bases de Dados Relacionais - SQL

Uma base de dados relacional, tal como proposto por *E.F. Codd* em 1970, organiza informação em uma ou mais tabelas nas quais uma linha representa uma entrada de dados identificada por uma chave única. O termo relacional aparece da possibilidade de existir uma coluna que representa o identificador único de uma linha de outra tabela, efectivamente relacionando[26] duas tabelas distintas.

Seguem-se algumas das características-chave deste modelo[27]:

1. O esquema de dados é rígido e permite garantir a integridade dos dados;
2. A capacidade de garantir transacções ACID [28] (Atomicidade, Consistência, Isolation, Durability);
3. O uso da linguagem SQL (Structured Query Language) para consulta e manutenção da base de dados;
4. A capacidade de executar consultas de dados complexas e que incluem simultaneamente várias tabelas.

O motor de base de dados *open source* mais utilizado dentro desta arquitectura é o *MySQL* seguido do *PostgreSQL*[29] que é também o motor utilizado nas restantes aplicações da Nourish Care.

6.1.1.2 Bases de Dados Não-Relacionais - NoSQL

O movimento *NoSQL* começou a ter mais expressão em meados dos anos 2009, no entanto a definição de *NoSQL* não é ainda universalmente aceite, oscilando entre *"Not Only SQL"* e *"Not Relational"*.

No contexto desta tese e de acordo com *Rick Cattel, Scalable SQL and NoSQL Data Stores*[30] tomamos algumas características-chave:

1. A capacidade de escalar horizontalmente a capacidade de resposta a operações CRUD entre vários servidores dentro da mesma colecção de dados;
2. A capacidade de adicionar novos atributos de forma dinâmica (não existe um esquema de base de dados);
3. A capacidade de replicar e distribuir dados para vários servidores;
4. A incapacidade de garantir transacções *ACID* - A propagação de dados não é instantânea e portanto as operações de leitura podem não ser consistentes.

Estes motores de bases de dados encontram aplicação em ambientes com dados simples e consultas simples; muitas das funcionalidades de pesquisa avançada presentes em motores relacionais não se encontram disponíveis em bases de dados NoSQL[31] - por exemplo, executar uma consulta com *multiple-table joins* é impossível num único comando e implica implementação no domínio da aplicação.

Por outro lado, esta arquitectura *schemaless* permite que a aplicação suporte modelos de dados dinâmicos sem a necessidade de migrar o esquema da base de dados, delegando a responsabilidade de manter a integridade do modelo de dados para a lógica da aplicação[32].

O motor de base de dados *open source* mais utilizado dentro desta arquitectura é o *MongoDB.*[29]

Aproveitando a infraestrutura e conhecimento da *Nourish Care* e sabendo que o esquema de dados em uso será relativamente estático, optou-se pelo motor de base de dados relacional *PostgreSQL*

6.1.2 Frameworks de Desenvolvimento Web

O uso de uma *framework* de desenvolvimento permite simplificar e reduzir o esforço de desenvolvimento e tem ainda um impacto positivo na qualidade do código produzido. Estas melhorias advêm de um grande reaproveitamento da lógica comum de todas as

aplicações *web* como o processamento de pedidos *HTTP(S)* ou a integração com a base de dados, substituíndo a necessidade de implementação por código gratuito de grande qualidade e testado em produção por milhões de utilizadores.

A utilização de uma *framework* traz ainda um conjunto de boas prácticas e recursos de outros programadores aos quais não se teria acesso implementando uma solução de raiz.

Existem no entanto também algumas desvantagens a ter em consideração: Há uma menor flexibilidade imposta pela arquitectura da *framework* e um possível impacto na performance ligado à modularidade e generecidade das soluções propostas.

No contexto desta tese, uma *framework* de desenvolvimento web deve incluir:

Funcionalidades Base - Incluir um ORM (Object-relational mapping), mecanismos de validação e integridade de dados, *routing*, geração de *templates* e gestão de *assets* (imagens, fontes, javascript, etc).

Gestão de dependências - Facilitar a sua extensão com bibliotecas de terceiros e incluir um mecanismo de gestão de extensões e dependências.

Documentação - Incluir boa documentação, tanto da API e arquitectura da *framework* como artigos e tutoriais de entusiastas.

Escalabilidade - Escalar uma aplicação é altamente dependente do domínio do problema e da arquitectura da solução, mas a *framework* utilizada não deve ser um *bottleneck* por si só.

Tendo em conta estas características, segue-se uma análise de três possíveis *frameworks*:

	Linguagem	**Funcionalidades Base**	**Gestão de dependências**	**Documentação**	**Escalabilidade**
Ruby on Rails	Ruby	Inclui por convenção: ORM, Sistema de *Templating*, Geração automática de Código, Gestão de Assets, etc	Sim, via *Ruby Gems*. Instala e gere dependências automaticamente	A documentação oficial é extensa e a comunidade é a mais activa das três	É a mais lenta das três, mas existem muitos artigos sobre como escalar aplicações escritas com Ruby on Rails
Express.js	Javascript (NodeJS)	Inclui sistema de templating e de routing por defeito, tudo o resto está disponível via *plugin*	Sim, via *NPM*. Instala e gere dependências automaticamente.	A documentação oficial é extensa e a comunidade activa mas muito segmentada	É a mais rápida das três, muito devido ao seu modelo de concorrência; alguns artigos sobre escalabilidade
Django	Python	Inclui por convenção ORM, Sistema de *templating*, geração automática de código, Gestão de Assets, etc	Sim, via utilitários do Python como *pip* e *easy_install*.	A documentação oficial é extensa e a comunidade activa	Existem vários artigos sobre escalabilidade em Django

Tabela 6.1: Frameworks de Desenvolvimento Web

Tanto o *Ruby on Rails*[33] como o *Django*[34] são *frameworks fullstack* completas (O seu domínio estende-se desde o nível da base de dados ao *HTML* final) e são encontradas em uso por *websites* de grande dimensão (e.g. Twitter, Washington Post, Github).

O *Express.js*, embora com uma performance superior, apresenta-se numa forma mais minimalista em que o programador é responsável por escolher os componentes que farão parte do sistema final[35], organização de código, etc. Esta filosofia oferece uma maior flexibilidade mas também um maior esforço inicial.

Tendo em conta a análise feita, e vários artigos [36–38] não se chega a vantagens claras para nenhuma das *frameworks*. Por este motivo e, usando a *Nourish Care Ruby on Rails* em todas as suas aplicações, optou-se também por esta *framework*.

6.1.3 Protocolos de Transmissão de Dados

O protocolo de comunicação especifica a estrutura do formato de dados que é trocado entre o *hub* e o serviço de agregação e tem por isso influência directa no tamanho do *payload* e na complexidade computacional da descodificação dos dados.

A comunicação de dados entre o *hub* e o servidor de agregação será eventualmente estabelecida através de uma rede móvel o que torna relevante o consumo de dados, tanto por questões económicas como pela baixa largura de banda disponível nos pacotes de dados mais económicos (GPRS - 35Kb/s a 171kb/s)[39].

Por outro lado, o cariz de protótipo deste projecto e a urgência em ter uma primeira implementação pronta, justifica o possível uso de uma tecnologia mais comum e com melhor suporte nas linguagens de programação escolhidas; diminuindo o esforço de implementação a custo de um maior consumo de dados.

6.1.3.1 HTTP

O protocolo HTTP é a base de comunicação para a *World Wide Web* e baseia-se num mecanismo de pedido-resposta. É baseado em texto estruturado e é constituído por duas partes: O *Header* e o *Body*.

O *Header* ou cabeçalho permite definir meta-dados que especificam, entre outros, o *endpoint*, tipo de dados que segue no *Body* e autenticação. O facto de não se manter estado entre pedidos obriga a que estes campos do *header* tenham de ser incluídos em cada um dos pedidos, contribuindo para o aumento do tamanho dos dados a enviar. Este *overhead* varia em média entre os 200 e os 2000 bytes por pedido.[40]

A utilização a larga escala deste protocolo tem a vantagem de ser suportado em todas as linguagens de programação tidas em consideração para este projecto. É também o meio de comunicação padrão utilizado no serviço de agregação (uma vez que é um serviço web) e facilita por isso a integração entre os vários componentes com um esforço de implementação muito reduzido.

6.1.3.2 MQTT

O MQTT é um protocolo inicialmente desenvolvido pela IBM baseado no modelo *publish/subscribe* com o intuito de implementar um protocolo de comunicação para telemetria de baixo *overhead.*[41]

Este protocolo apresenta algumas desvantagens, nomeadamente:

1. Implica um maior esforço de desenvolvimento do lado do serviço de agregação.
2. O número de bibliotecas e recursos disponíveis para trabalhar com este protocolo é reduzida.
3. O protocolo obriga a que se mantenha uma ligação aberta entre o hub e o servidor de agregação, aumentando o consumo de recursos e sendo afectado por ligações onde a perda de pacotes é frequente.[41]

Os problemas relacionados com ligações instáveis e perda de pacotes encontram-se resolvidos na implementação do protocolo MQTT-SN, que funciona sobre a *stack* UDP ao invés da *stack* TCP.

Os dois protocolos apresentam-se com pontos fortes e fracos quase opostos, o *HTTP* prima pela facilidade e universalidade de uso a custo de um maior *payload* típico de um protocolo baseado em texto. O MQTT-SN apresenta grande performance e baixo consumo de dados, no entanto a sua implementação é morosa, os recursos disponíveis escassos e não se encontram bibliotecas para as linguagens em uso cujo o uso seja elevado o suficiente para garantir estabilidade.

Tendo em conta esta realidade e em prol de uma maior velocidade de iteração, escolheu-se o *HTTP* como protocolo de comunicação entre o hub e o servidor de agregação. Esta

escolha tem a vantagem, de no futuro, ser facilitada e integração com outras aplicações externas, uma vez que as aplicações *REST HTTP* são o *standard* da web.

6.1.4 Formato de dados

Uma vez escolhida a camada de transporte, é importante estudar o formato que encapsulará os dados. Esta escolha terá impacto directo no tamanho final do *payload* assim como na complexidade computacional do processo de codificação e descodificação.

Esta secção pretende analisar três formatos de troca de dados: JSON, XML e Protocol Buffers. A tabela que se segue é baseada em dois artigos que comparam *JSON* com *XML*[42] e *XML* com *Protocol Buffers*[43]:

	Suporte nas linguagens em consideração	**Schema**	**Tamanho Relativo**	**Overhead Relativo**
JSON	Sim	Não	1	1
XML	Sim	Opcional	1.84	1.20
Protocol Buffers	Sim	Sim	0.18-0.62	0.06-0.

Tabela 6.2: Formatos de transferência de dados

Os valores de tamanho e *overhead* foram calculados para o esquema de dados do artigo [42] nos casos do JSON e XML, os valores para o *Protocol Buffers* são correspondentes ao pior e melhor caso, respectivamente.

As secções que se seguem abordam de forma sucinta cada um dos formatos de dados tidos em consideração.

6.1.4.1 JSON

JSON (JavaScript Object Notation) é um formato de troca de dados compacto baseado num subconjunto do standard da linguagem de programação JavaScript.

Este formato de dados tem vindo a tornar-se no *standard* para trocas de dados dentro e fora da web [42], tendo já representação, inclusivé, em motores de bases de dados (MongoDB e CouchDB) e sendo o formato padrão para IPAs Web.

A sua utilização ubíqua traz a vantagem de ser suportado oficialmente pela grande maioria das linguagens de programação, incluindo as bibliotecas disponíveis nos sistemas embebidos em consideração.

6.1.4.2 XML

XML (eXtensible Markup Language) é um formato de troca de dados simples baseado em texto para representação de informação estruturada.

É um formato redundante, o que permitindo encontrar de forma automática erros de estruturação, contribui também para o aumento da verbosidade do formato o que por sua vez resulta num maior *overhead* de comunicação.[44]

O formato aceita a adição de um ficheiro de esquema que permite validar a integridade da estrutura de dados.

A sua utilização é extensa, cobrindo desde software empresarial à representação de dados gráficos (SVG) e é por isso largamente suportado pela maior parte das linguagens de programação.

6.1.4.3 Protocol Buffers

Protocol Buffers é um formato de troca de dados binário desenvolvido pela Google e que recorre a um ficheiro de *schema* para validar e definir a estrutura dos dados.

A sua implementação fornece um formato de dados muito compacto e rápido de processar, sendo entre 3 a 10 vezes mais pequeno que o correspondente em XML e 20 a 100 vezes mais rápido de processar.[43]

Existem implementações do protocolo para as linguagens tidas em consideração no escopo deste projecto e, inclusivé, geração de código automático para encapsular as mensagens em objectos nativos de cada uma destas linguagens, no entanto a rigidez do protocolo pode ser uma desvantagem no processo iterativo de desenvolvimento.

Desta selecção de protocolos distingue-se o *JSON* e o *XML* do *Protocol Buffers*, sendo os primeiros baseados em texto e o último um formato binário com um esquema de dados associado.

Devido à incerteza inicial sobre o esquema de dados, excluiu-se o *Protocol Buffers*, já que a necessidade de definição de um esquema de dados e consequente ajuste do código do cliente e servidor atrasariam o processo de iteração necessário para encontrar uma boa solução.

A escolha entre *JSON* e *XML* é de baixa consequência, até porque uma arquitectura *RESTful* prevê a utilização de mais do que um formato de dados por endpoint.

Em função de uma mais fácil leitura e do menor tamanho relativo optou-se por utilizar o formato *JSON*.

6.1.5 Alojamento

É de relevância discutir o alojamento de uma aplicação web (neste caso do servidor de agregação de dados), uma vez que esta escolha influenciará a disponibilidade do serviço, tempo de resposta e esforço de manutenção da infraestrutura.

A escolha de alojamento está directamente ligada à flexibilidade, preço alvo e, neste caso, também pela antevisão da necessidade de certificação do serviço no Reino Unido. Um dos pontos chave para a certificação deste tipo de aplicações é o alojamento dos dados no próprio país, limitando assim a escolha de alojamento a fornecedores com localização física no Reino Unido.

Esta limitação impede o uso de alguns serviços como a *Amazon AWS* e outros que utilizam a sua infrastrutura (e.g. *Heroku, Engine Yard*) por não possuírem localização física no Reino Unido.

Existem no mercado variadas alternativas no que diz respeito a alojamento, dividindo-se em três tipos diferentes de serviços: Alojamento de servidores físicos, virtuais e serviços de alojamento especializado.

6.1.5.1 Alojamento de Servidor Físico

Este serviço consiste na montagem e hospedagem de um servidor especificado pela equipa de desenvolvimento. Neste caso a flexibilidade é total e apresenta vantagens mais significativas quando o software requer hardware específico.

O custo do serviço é elevado uma vez que aos custos de manuntenção e operação é preciso somar o investimento de aquisição inicial.

6.1.5.2 Alojamento de Servidor Virtual

O alojamento de VPS (Virtual Private Server) é um serviço que disponibiliza acesso total a uma máquina como se de um servidor dedicado se tratasse. Esta máquina é virtual e não mapeia directamente para uma máquina física, isto é, um servidor físico pode na realidade alojar (e virtualizar) várias VPS que partilham os mesmos recursos de hardware.

Este serviço tem um custo muito menor que o alojamento de servidores físicos (não há investimento inicial) e permite uma maior flexibilidade na altura de escalar os recursos, uma vez que sendo uma máquina virtual podem ser alocados novos recursos dinamicamente.

À altura os maiores serviços de alojamento virtual são a DigitalOcean [45] e a *Amazon AWS.*[46] O plano mais barato destas plataformas, constituído por um VPS (Virtual Private Server) dotado de uma unidade de processamento, 512MB de memória ram e 32GB de disco de estado sólido tem um custo mensal aproximado de 5 dólares e é suficiente para correr a aplicação durante a fase de desenvolvimento, sendo em qualquer altura possível alterar a configuração do servidor sem perder dados.

6.1.5.3 Alojamento especializado

Existem ainda no mercado soluções de alojamento especializadas que fornecem alojamento à medida para *frameworks* específicas.

Estes serviços oferecem um reduzido esforço de manutenção e instalação ao custo de uma menor flexibilidade na gestão de dependências e um maior custo. O utilizador não gere máquinas mas sim uma abstracção do ambiente no qual a aplicação corre, sendo possível alocar e desalocar recursos à medida do necessário.

À altura os maiores serviços de alojamento especializado de aplicações são as plataformas *Heroku* e *RackSpace*[46–48]. Fornecendo alojamento para aplicações escritas em *Ruby on Rails*, *NodeJS* e *Django*, entre outras.

A configuração mínima necessária para correr o servidor de agregação de dados numa plataforma deste género teria um custo mensal de cerca de 30 dólares [48], sendo ainda necessário alojar o servidor de base de dados de forma separada.

Antevendo-se uma possível necessidade de certificação do produto no Reino Unido e a necessidade de reduzir custos, optou-se por alojar o servidor de agregação de dados na plataforma *Digital Ocean*[45].

Esta escolha resulta num custo mensal francamente mais baixo que uma opção de alojamento especializado, aumentando no entanto o esforço de configuração e manutenção do servidor. Este esforço é de alguma forma reduzido pelo reaproveitamento de código de configuração de infraestrutura da *Nourish Care*, que é em tudo semelhante à utilizada neste projecto.

6.2 Hardware para o Hub

O hardware do Hub deverá permitir a interface com os sensores e com redes de dados *Ethernet ou WiFi* e/ou *3g*.

Para suportar os sensores, o hub deve ter acesso programático a pinos GPIO (General Purpose Input/Ouput). Estes pinos permitirão integrar directamente sensores com estado binário, sensores inteligentes que comuniquem via protocolo série e módulos de aquisição, nomeadamente ADCs (Analog Digital Converters), necessários para integrar os sensores de presença na cama.

O hub será ainda responsável por enviar dados para o servidor de agregação pelo que o suporte a uma interface de rede é obrigatório.

Esta secção apresentará possíveis escolhas de hardware para o módulo Hub.

6.2.1 Feito à medida

A opção mais flexível é sempre criar o hardware de raiz, permitindo a escolha de cada um dos componentes e da sua integração de acordo com o objectivo do projecto.

No entanto e tendo em conta o cariz de protótipo do projecto não se justifica o custo e o dispêndio de tempo em causa. Este processo é adequado uma vez que o projecto atinja uma fase de maturidade tal que seja necessário um grande número de unidades e a sua arquitectura esteja estabilizada, uma vez que alterações de hardware são complicadas e morosas.

6.2.2 Micro-controladores

Micro-controladores como *Basic STAMP, Atmel e TexasTI* oferecem grande flexibilidade a um custo reduzido (um micro-controlador da *Texas* ou *Atmel* tem preços na ordem dos cêntimos).[49]

Esta flexibilidade advém do acesso a muito baixo nível a todas as funcionalidades do CPU, incluindo acesso aos seus pinos de leitura e escrita e ao facto do *firmware* correr directamente sobre o CPU ao invés de um ambiente controlado por um sistema operativo. Por outro lado, esta flexibilidade traz consigo a responsabilidade de controlar a execução de código e interface com outros dispositivos de forma manual, incorrendo por isso num maior esforço de desenvolvimento.

Para perceber esta dicotomia, dá-se o exemplo da integração de um módulo 3G: Num ambiente gerido por um sistema operativo, instalando os drivers que acompanham o módulo este passa a gerir as comunicações com a rede de forma automática bastando ao utilizador utilizar uma biblioteca responsável pelas trocas de informação. Na programação de *firmware* nativo, este processo implicaria comunicar com o módulo 3G, normalmente através de comandos *AT (attention commands)*, no sentido de o inicializar e configurar e só então se pode começar a enviar dados (também através de comandos AT).[50]

Neste paradigma, trocar de módulo 3G ou até de meio de comunicação (e.g. *ethernet*) implicaria rescrever o *firmware* do dispositivo.

Usando um ambiente gerido por sistema operativo, esta troca resultaria no mais simples dos casos em nenhuma alteração (uma vez que os próprios sistemas operativos já incluem *drivers* para diversos periféricos) e no pior dos casos na instalação de um novo *driver* de dispositivo.

Custo médio: US$0.2-15[49]

6.2.3 Raspberry Pi e semelhantes

O *Raspberry Pi*[51] é um pequeno computador de custo reduzido capaz de correr o sistema operativo *Linux* e com acesso a 17 pinos de GPIO. Dotado de duas portas USB

e uma porta Ethernet permite a integração com uma grande quantidade de periféricos como é o caso dos módulos 3g e ADCs.

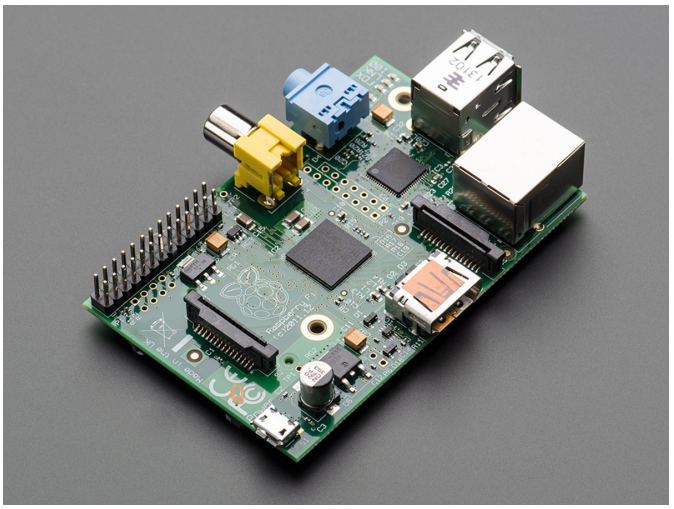

FIGURA 6.1: RaspberryPi

O seu projecto é *open-source*, facilitando por isso e caso haja necessidade, a modificação ou integração numa placa de circuito integrado.

A comunidade é muito activa, existindo muita informação sobre a integração de sensores e componentes electrónicos vários.[52]

Custo médio: 35$[49]

6.2.4 Electric Imp

O *Electric Imp* é um microcomputador e uma plataforma de desenvolvimento acoplado a um módulo de rede WiFi com o formato de um cartão *SD* (Secure Digital).[53]

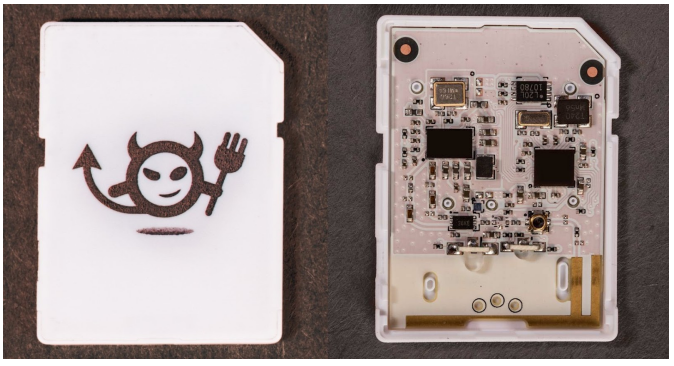

FIGURA 6.2: ElectricImp

Esta plataforma obriga a que todos os pedidos de rede passem pelos servidores da empresa que produz o aparelho, o que levou a descartar esta hipótese por preocupações ao nível da privacidade dos utentes. O aparelho é também de fonte-fechada o que leva a que modificações fossem difíceis e impossíveis de certificar.

Custo médio: 30$ + mensalidade [53]

Dando uma maior importância à flexibilidade e facilidade de prototipagem foi utilizado um *Raspberry Pi* como base de *hardware* para o hub. Esta escolha permite o acesso a um sistema operativo com uma base de utilização extensa e a abstracções ao nível do *hardware* de rede que não seriam possíveis com o uso de um microcomputador.

Por outro lado, esta escolha permite-nos ser agnósticos em relação ao *hardware*, sendo muito fácil portar a solução de *software* implementada para qualquer outro microcomputador capaz de correr o mesmo sistema operativo.

6.3 Software do Hub

Uma vez escolhido o hardware que será a base de implementação do hub, é importante também definir a linguagem de programação em que o *firmware* será escrito. Esta escolha está obviamente limitada pelas linguagens suportadas pelas ferramentas de compilação/interpretação do hardware escolhido e pelos requisitos funcionais do próprio *firmware*.

Estando a escolha do hardware limitada ao *Raspberry Pi* e sabendo que os requisitos mínimos da linguagem a utilizar são:

- Acesso aos pinos de GPIO;
- Acesso à camada de comunicação via socket do sistema operativo;
- Acesso à camada de leitura e escrita de ficheiros do sistema operativo.

Surgem então dois possíveis candidatos: *Python* e *C++*.

6.3.1 C++

C++ é uma linguagem compilada que permite acesso a instruções de baixo nível e a algumas abstrações de alto nível com a programação orientada a objectos e sistemas de meta-programação. O acesso a instruções de baixo nível permite escrever programas extremamente compactos e de baixo *overhead*.

Sendo uma das linguagens mais utilizadas[54] e estando em uso desde 1983 possui uma vasta colecção de ferramentas, bibliotecas e recursos de aprendizagem *open-source* disponíveis.

6.3.2 Python

Python é uma linguagem de programação interpretada de tipagem dinâmica que permite escrever e iterar programas de forma rápida e concisa (tipicamente entre 5 a 10 vezes mais curtos que o equivalente em C++[55]).

Esta linguagem é aconselhada pela própria *Raspeberry Pi Foundation* para uso com o Rasperry Pi, sendo uma boa escolha para problemas que não necessitem de uma utilização intensiva do CPU[55].

Sendo também uma linguagem com um grande número de utilizadores[54], é fácil encontrar recursos e bibliotecas de uso e fonte livre.

Em prol de uma maior velocidade de prototipagem e desenvolvimento e tendo em conta que não se espera que a capacidade de processamento seja um factor limitante, escolheu-se *Python* como linguagem de programação do hub.

Capítulo 7

Resultados e Análise

7.1 Simulador

De forma a testar e validar o modelo de comunicação e em virtude do atraso na entrega do material para a implementação da rede de sensores, optou-se por construir um módulo de simulação. Este módulo, construído com tecnologias web, permite simular e enviar dados dos sensores para o servidor de agregação e, embora a sua implementação não estivesse delineada inicialmente, este mostrou-se como uma mais-valia permitindo uma iteração rápida sobre a quantidade e formato de dados a enviar, assim como a demonstração da integração entre o *Nourish Sensors* e a *Nourish Care* a possíveis clientes.

O simulador (ver figura 7.1) consiste numa planta virtual de uma habitação na qual o utilizador pode accionar separadamente cada um dos sensores presentes por divisão. A interface detalha ainda o *payload* e resposta do servidor.

7.1.1 Consumo de dados

Pela facilidade de alteração e reprodução de variáveis, o simulador foi imprescindível na optimização do modelo de comunicação entre o servidor e o *hub*. É ainda relevante notar que a arquitectura escolhida para a implementação do servidor de agregação de dados permitiu as alterações e optimizações discutidas nesta secção sem a necessidade de alterações ao seu código de aplicação.

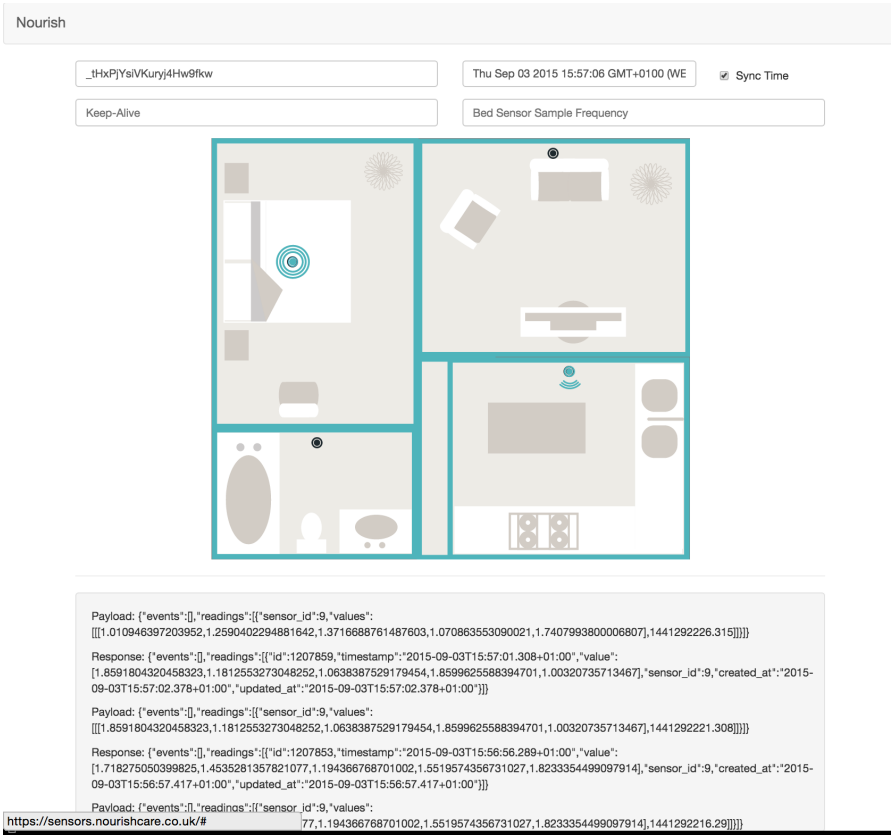

FIGURA 7.1: Simulador

De forma a quantificar o consumo de dados foi utilizada a ferramenta *iftop* (figura 7.2) e simulada uma instalação com 3 sensores de movimento e um sensor de colchão.

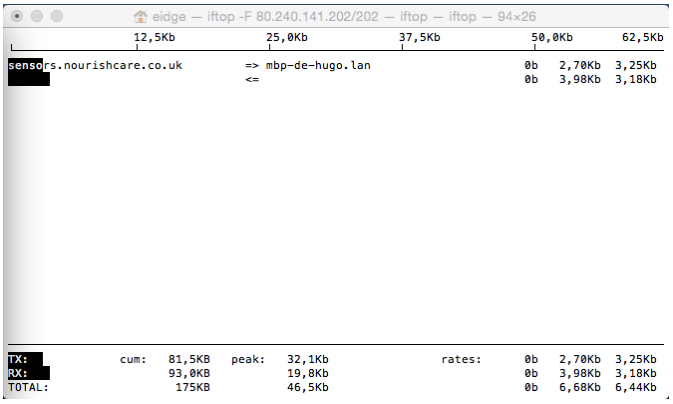

FIGURA 7.2: Inspector Web

Inicialmente, os sensores eram amostrados a cada segundo e os seus dados enviados sem compressão para o servidor. Com um tamanho médio por pedido de 1.355 ± 0.127 KB acumular-se-iam 3.58 ± 0.340 GB de tráfego por mês (excluindo os pedidos relativos ao

keep-alive), um quantidade de dados que impossibilitaria o uso dos pacotes de dados mais baratos das operadoras 3G restringidos a 1GB/mês.

As secções que se seguem descrevem os resultados da optimização do modelo de comunicação.

7.1.1.1 Remoção de dados redundantes

Não havendo activação dos sensores é redundante enviar o estado de inactividade, sendo a ausência de uma activação suficiente para descrever o estado do sensor. Embora esta alteração não tenha impacto significativo no tamanho de cada pedido, permite reduzir o número total de pedidos, uma vez que estes apenas são realizados aquando da activação de pelo menos um sensor.

Durante os períodos em que o utente não se encontra a activar nenhum dos sensores de colchão e sabendo que a taxa de activação dos sensores de movimento não excede as 3 vezes por minuto, esta optimização resulta numa redução potencial de 20 vezes a quantidade dos pedidos efectuados.

Assumindo que o utente activa um dos sensores de colchão durante 18 horas por dia, esta alteração pouparia 0.890 ± 0.032 GB de tráfego por mês.

7.1.2 Envio de dados em lotes

Do pedido inicial com uma média de 1.4KB, apenas 20 a 200 bytes são ocupados pelas leituras dos sensores, ou seja, o *overhead* tem um impacto de 86 a 98% no tamanho total do pedido.

O envio de dados por lotes permite reduzir o efeito do *overhead* causado pelo protocolo *HTTP(S)*, uma vez que o mesmo *overhead* é distribuído por uma maior quantidade de dados.

O tamanho do lote deve ser função do tempo de resposta máximo pretendido, isto é, o tempo máximo aceitável entre a geração dos dados e o seu envio para o servidor de agregação.

Uma vez que um instante temporal dos dados provenientes dos sensores é pouco relevante - é do contexto criado ao longo do tempo pelos sensores que podemos inferir acontecimentos - o sistema suporta um atraso de alguns minutos sem consequência temporal significativa na interpretação dos dados.

A tabela 7.1 traduz os ganhos em função do tempo de armazenamento do lote.

Tempo de Armazenamento (s)	Tamanho por Pedido (KB)	Tamanho por Datum (KB)	Projecção Tráfego Mensal (GB)	Optimização (%)
0	1.400	1.400	3.580	0
60	2.890	0.048	0.123	96.6
300	8.600	0.029	0.073	98

Tabela 7.1: Envio de dados por lotes

O envio de dados por lotes resulta numa diminuição significativa do tráfego total registado. Introduzindo um minuto de latência entre o registo e o envio dos dados é possível diminuir o tráfego mensal em 96.6%, possibilitando o uso dos pacotes de dados 3G mais baratos das operadoras móveis.

Durante a fase de desenvolvimento e tendo em conta que o uso de dados estimado é muito inferior ao limite mensal, optou-se por acumular dados durante apenas um minuto.

7.2 Nourish Sensors

Com o objectivo de monitorizar um utente residente no concelho de Coimbra, foram instalados 4 sensores de movimento (Quarto, Casa de Banho, Cozinha e Sala) e dois sensores de colchão (cama e sofá).

A aquisição de dados teve a duração de uma semana e permitiu inferir alguns padrões de comportamento do utente. De forma a suportar e corroborar a análise dos dados, foi pedido ao paciente que anotasse o seu comportamento durante a totalidade da experiência.

Esta secção detalha os resultados da análise destes mesmos dados.

7.2.1 Consumo de Dados

A fase de simulação que precedeu a implementação e teste do *hub* permitiu estimar o consumo de dados, no entanto, apenas um teste sobre a implementação final permitiria a sua correcta medição.

Em função dos valores simulados, e utilizando um tempo de armazenamento de 1 minuto, estimar-se-ia um consumo de 27.8MB durante os 7 dias de aquisição de dados. O valor final, tal como reportado pelo operador móvel para esse período, foi de 42.3MB.

Embora discrepantes, os consumos simulados e reais são da mesma ordem de grandeza, o que valida a utilidade da simulação inicial na prevenção da reimplementação final por excessivo consumo de dados. Esta diferença pode ser explicada pela simulação não ter em conta os pedidos de *keep-alive*, autenticação e sessões de *login* remoto utilizadas para *debugging*.

Em suma, extrapolando o consumo de 7 para 30 dias, percebeu-se ser possível a contratação dos pacotes mais básicos de dados 3G com bastante margem para um possível aumento de consumo.

7.2.2 Sensores de Movimento

De forma a facilitar a análise dos dados provenientes dos sensores de movimento procedeu-se à sua exportação para o *Microsoft Excel*.

Apresenta-se de seguida a análise de 24 horas de aquisição de dados através de um gráfico cartesiano da activação dos sensores.

Na figura 7.3, cada ponto representa a activação de um dos sensores sendo o estado de inactividade representado pela ausência de dados. O valor representado no eixo dos yy é uma classe arbitrária cujo o único objectivo é facilitar a visualização dos dados.

Embora o gráfico de activação de um único sensor tenha uma utilidade reduzida (figura 7.4), a conjugação dos dados dos vários sensores de movimento instalados na habitação do utente permitem reconstruir os seus movimentos ao longo do dia e inferir acerca do seu ritmo circadiano.

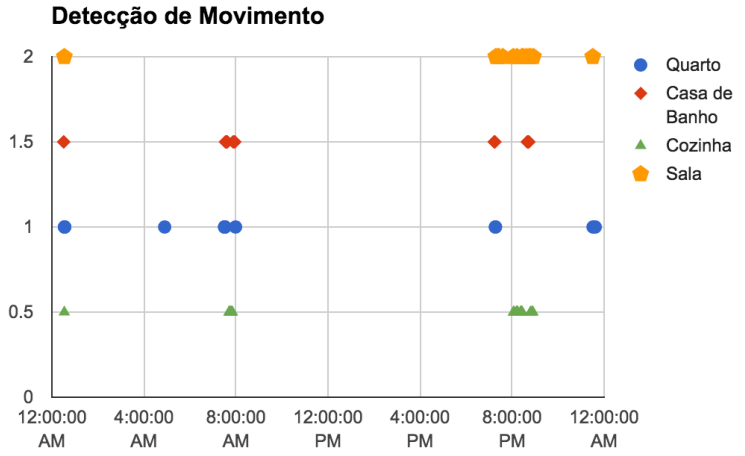

FIGURA 7.3: Gráfico de Activação dos sensores de movimento

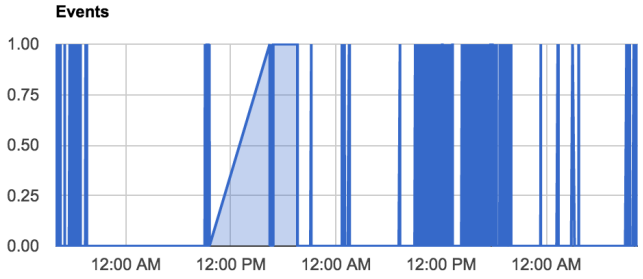

FIGURA 7.4: Gráfico de activação de 5 dias do sensor instalado na sala

A figura 7.3 é referente ao primeiro dia de aquisição de dados. O utente chega a casa à 00:30, deslocando-se entre a casa de banho, cozinha e sala antes de entrar no quarto (figura 7.5).

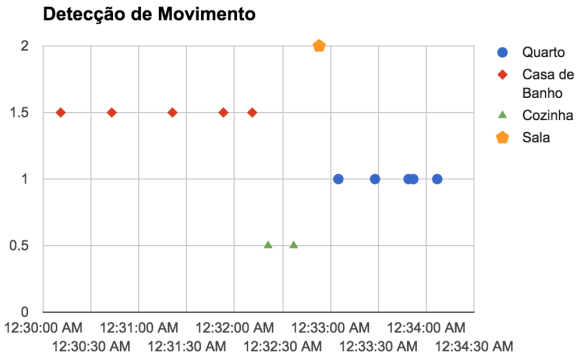

FIGURA 7.5: Detalhe dos instantes seguintes à primeira detecção de movimento do dia

Entre a 00:36 e as 07:30 da manhã, registam-se apenas algumas activações do sensor do quarto por volta das 04:55. É fácil imaginar que o utente se encontrasse a dormir e, por isso, o sensor não detectasse movimento, no entanto, e sem outros dados que o confirmem podemos apenas supor ao invés de inferir.

Entre as 08:04 e as 18:43, não há qualquer detecção de movimento. Supõe-se que o utente não se encontra em casa.

A detecção de movimento reinicia-se às 19:16 quando o utente visita a casa-de-banho, permanecendo de seguida na sala entre as 19:21 e as 20:03.

As deslocações entre a sala e a cozinha entre as 20:05 e as 20:55, sugerem o consumo de uma refeição seguidas de um período de descanso entre as 20:55 e as 23:32 na sala.

O utente desloca-se para o quarto às 23:33, onde os dados da madrugada seguinte sugerem um período de sono semelhante ao registado na figura 7.3.

Em suma, os sensores de movimento permitem inferir de forma grosseira o ritmo circadiano do utente, no entanto, os períodos de inactividade são difíceis de caracterizar - sendo por exemplo impossível distinguir uma queda do utente e a consequente ausência de movimento de um período de ócio no sofá ou cama.

Os sensores utilizados possuem um mecanismo que previne a detecção de falsos positivos causados por animais de estimação, pressupõe-se que este mesmo mecanismo diminua também a detecção de pequenos movimentos do utente, uma vez que é necessária uma maior alteração da quantidade de radiação infravermelha que chega ao sensor para o activar.

7.2.3 Sensores de colchão

Os sensores de colchão permitem complementar os dados dos sensores de movimento durante os períodos de descanso do utente.

Para caracterizar a resposta dos sensores à presença de um corpo, foi instalado um sensor de colchão numa cama de teste com o objectivo de descrever a sua resposta à movimentação de um sujeito entre as suas extremidades. A figura 7.6 representa a aquisição de dados durante um destes testes.

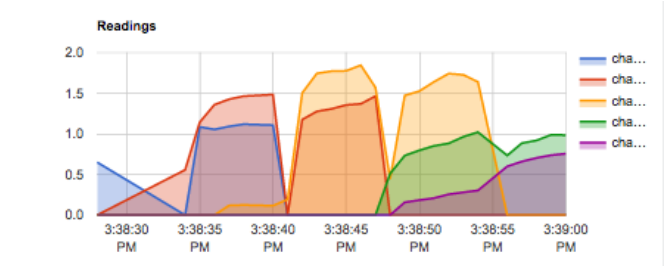

FIGURA 7.6: Resposta de um sensor de colchão à variação de pressão entre as suas extremidades

Do canal 1 (azul) ao canal 5 (roxo) fazem-se corresponder respectivamente as resistências na extremidade mais à esquerda e direita do sensor.

No gráfico da figura 7.6, é possível observar a transferência de pressão entre as extremidades do sensor. O gráfico tem início num valor diferente de zero para o sensor do canal 1 uma vez que há um limiar mínimo (físico e caracterizado pela banda resistiva presente no sensor) para activação do sensor. O mesmo acontece no término da aquisição com os canais 4 e 5.

Esta alteração na amplitude de aquisição do sinal de acordo com a localização do peso sobre o sensor permite detectar movimentos e agitações do utente que seriam de outra forma imperceptíveis através do sistema de monitorização.

Na prática, estas alterações permitem complementar e explicar a informação proveniente dos sensores de movimento. A figura 7.7 é relativa ao período da noite representado na figura 7.3 e onde se pode ver a activação do sensor de movimento do quarto do utente por volta das 04:50, a conjugação dos dados de ambos os sensores permite perceber que a activação do sensor é resultante da movimentação do utente sobre a cama sem que este se tenha levantado.

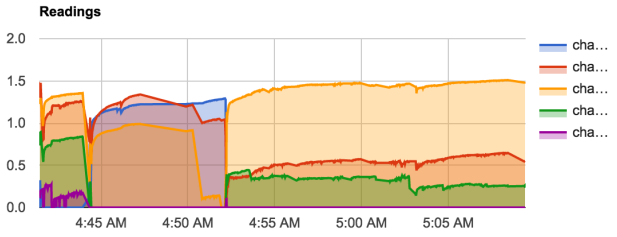

FIGURA 7.7: Aquisição do sensor de colchão durante a noite

7.2.4 Sensores de Porta

Durante a fase preliminar de testes chegou-se à conclusão que a ausência de dados dos sensores de movimento não constituía um indicador fiável da ausência do utente na sua habitação - um utente inconsciente seria interpretado como não estando em casa o que resultaria num erro não admissível no funcionamento do sistema.

Para colmatar esta falha do sistema optou-se por instalar um sensor de porta, este sensor funciona como um interruptor magnético cujo circuito é interrompido quando a porta se encontra fechada. Ou seja o método de aquisição é em tudo semelhante ao já implementado para os sensores de movimento, não tendo sido por isso necessário efectuar qualquer alteração ao sistema para suportar este novo tipo de sensor.

A complementação dos dados dos sensores de movimento com os dados deste sensor permitem inferir com algum grau de certeza os eventos de entrada e saída de casa.

7.2.5 Cruzamento de dados

Os sensores de movimento permitem estimar a posição do utente em relação à sua habitação, no entanto e por definição, oferecem informação reduzida durante os períodos de inactividade ou ausência. Os sensores de colchão permitem complementar esta informação com a presença do utente sobre uma cama ou sofá, enquanto os sensores de porta permitem distinguir uma possível situação de emergência da ausência do utente na sua habitação.

De forma a possibilitar a análise dos sensores de movimento e colchão lado-a-lado, optou-se por binarizar os dados destes últimos (a presença de dados corresponde a um estado de activação e a ausência a um estado de inactividade).

A figura 7.8, por comparação à figura 7.3, torna óbvia a utilidade da informação proveniente destes sensores durante os períodos da noite, descanso e ausência da habitação no mapeamento do ciclo circadiano do utente.

No período da meia-noite às 07:30, o sensor de colchão instalado na cama permite validar a presença do utente na cama, assim como explicar a detecção de movimento originada por volta das 04:50. Por outro lado, o sensor de porta permite validar a ausência do utente.

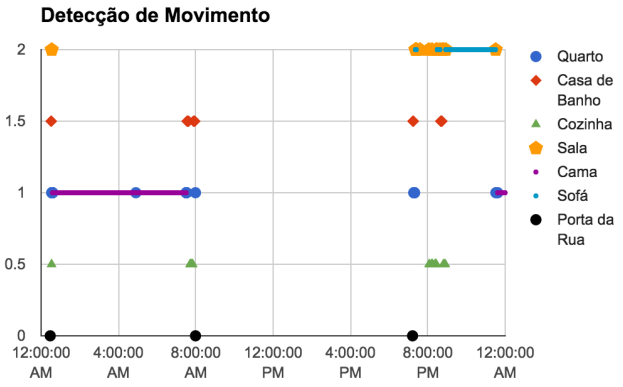

FIGURA 7.8: Aquisição e normalização dos dados da totalidade dos sensores

Sem a informação disponibilizada pelo cruzamento de dados estas duas situações poderiam facilmente ser mal interpretadas - seria impossível saber se o utente se encontrava na cama ou inconsciente no chão do quarto, da mesma forma que poderíamos assumir que a ausência de dados entre o período das 08:00 às 19:15 era uma situação anómala, quando na realidade é explicada pela ausência do utente.

7.2.5.1 Validação

A tabela 7.2 contém o registo mantido pelo utente e sobrepõe-se na perfeição ao registo capturado pelos sensores, validando a premissa da utilidade dos sensores no mapeamento do ciclo circadiano do utente.

Hora do Registo	**Nota**
00:30	Chegou a Casa
00:35	Deitou-se
07:30	Levantou-se (despertador)
07:30	Ida à casa de banho
07:40	Pequeno Almoço
08:00	Saiu de Casa
19:15	Chegou a Casa
19:15	Ida à casa-de-banho
20:15	Jantar
20:40	Ida à casa-de-banho
23:30	Deitou-se

TABELA 7.2: Registo do utente

7.3 Integração com a plataforma *Nourish Care*

De forma a demonstrar o potencial da integração do *Nourish Sensors* com a plataforma da *Nourish Care*, foi escrito um pequeno programa que corre em segundo plano no servidor de agregação de dados. Este programa é responsável por monitorizar os dados de um sensor de colchão e enviar um evento quando o utente se deita ou levanta da cama para a plataforma de gestão.

Para fazer a classificação destes eventos, faz-se correr uma janela de 30 segundos sobre a aquisição de dados do sensor de colchão instalado na cama. Inicialmente o estado de cada utente é *acordado*, sendo que a detecção de dados nessa mesma janela faz alterar o estado para *deitado* e despoleta a criação de um novo registo na plataforma. Se o estado do utente é *deitado* e uma janela de 30 segundos não detecta qualquer movimento, então o estado é revertido novamente.

A duração da janela é função do tempo de resposta e probabilidade de detecção de falsos negativos. Sendo que uma janela maior resulta numa menor probabilidade de falsos positivos mas também numa maior latência do sistema na criação dos registos. Durante a fase de testes chegou-se à conclusão que uma janela de 30 segundos reduz o número de falsos positivos virtualmente a zero e constitui uma latência satisfatória na classificação do estado do utente.

Na figura 7.9, retirada do ecrã de gestão de cuidados diários da plataforma *Nourish Care*, é possível ver a integração dos dados preenchidos manualmente pelo cuidador com a introdução de tarefas realizadas pelo *Nourish Sensors* - neste caso, a informação às 08:32 de que o utente se teria levantado.

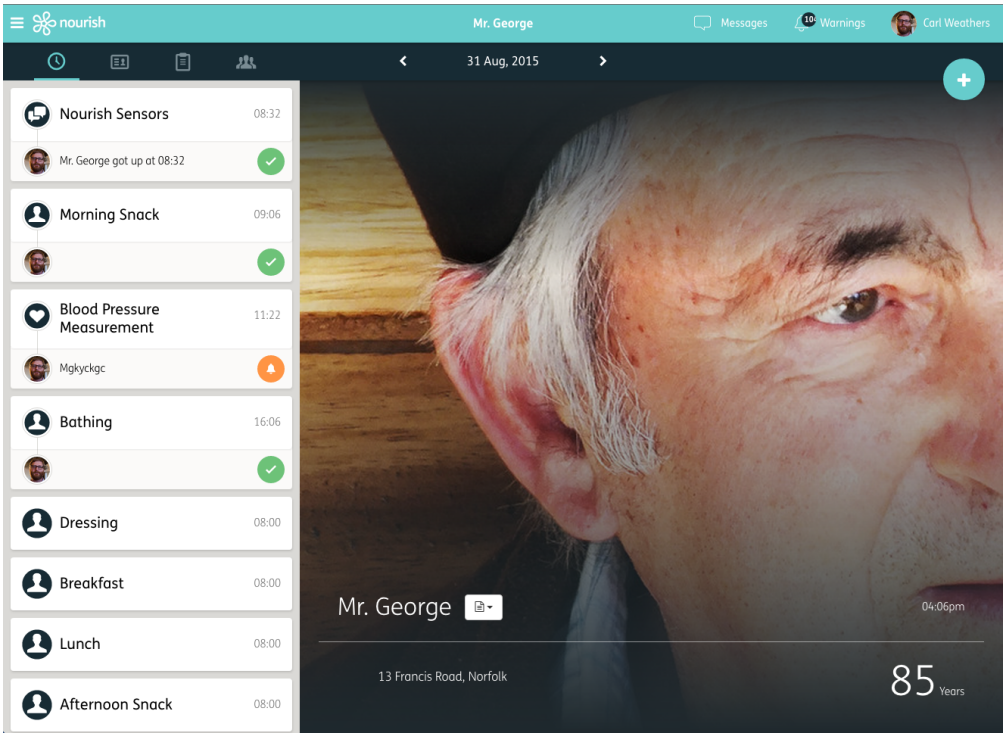

FIGURA 7.9: Integração com *NourishCare*

Capítulo 8

Conclusão

8.1 Sistema de Monitorização

O *Nourish Sensors* tinha como objectivo a integração de um sistema telemétrico para a monitorização de idosos com recurso a sensores de movimento e presença. Este objectivo foi alcançado através da implementação de um sistema de aquisição a instalar em casa dos utentes e de um servidor de agregação *online* que permite gravar e visualizar os dados recolhidos através de uma aplicação *web*.

O sistema faz uso de tecnologias *web* e padrões de industria, assim como de uma arquitectura simples e comum [56]. Estas escolhas mostraram-se essenciais durante a fase de testes e ajustes do sistema, permitindo a alteração de cada componente de forma isolada e sem a necessidade de alterar os restantes.

Não conhecendo à partida a quantidade e cadência de dados necessária à descrição de eventos e ciclo circadiano do utente, o protocolo de comunicação de dados foi especialmente difícil de definir. Esta escolha tinha impacto significativo na relação entre o esforço de desenvolvimento e a quantidade de tráfego de dados necessário ao funcionamento do sistema.

Optou-se por estabelecer a comunicação sobre o protocolo *HTTP* e o formato de dados *JSON*, esta escolha permitiu uma boa flexibilidade e facilidade de alteração mas resultou numa quantidade de tráfego inicialmente inaceitável. No entanto, através do envio de

dados por lotes foi possível diminuir significativamente o consumo de dados, respeitando os requisitos iniciais e sem um impacto significativo sobre o tempo de resposta do sistema.

Uma das dúvidas iniciais era a da utilidade dos dados na descrição do ritmo circadiano do utente. A análise preliminar de um conjunto reduzido de dados permitiu demonstrar a capacidade do conjunto de dados recolhidos reconstruir os movimentos do utente na sua habitação, assim como os períodos de ausência do mesmo.

Este sistema apresenta vantagens em relação aos tradicionais botões/pulseiras/colares de alarme e detecção de quedas uma vez que não implica o uso diário de um dispositivo que precisa de ser carregado e pode não estar a ser usado pelo utente durante a ocorrência de um queda ou outra situação de emergência.

A passividade do uso do sistema permite assegurar uma monitorização eficaz do utente sem que este tenha de alterar os seus hábitos diários, não sendo por isso vulnerável a esquecimentos ou falhas de bateria.

8.2 Trabalho Futuro

Este projecto de tese resultou na produção de um primeiro protótipo do *Nourish Sensors* cujo o objectivo é demonstrar a utilidade do produto na inferência do ciclo circadiano do utente, assim como alicerçar uma plataforma capaz de recolher e analisar de forma automática anomalias no dia-a-dia do utente.

Desta forma, antecipam-se as seguintes necessidades:

- Planear e levar a cabo a instalação da rede de sensores em diversas habitações com o intuito de recolher dados em volume e qualidade que permitam o desenvolvimento e teste de algoritmos de inteligência artificial para a classificação automática de eventos;

- Validar e quantificar a eficácia do sistema na detecção de quedas e outras ocorrências;

- Desenhar um módulo de aquisição capaz de ser produzido em grandes quantidades (o protótipo actual foi desenhado e construído pelo mestrando através da soldagem

manual de cada componente);

- Integrar sensores sem fios. O processo de instalação actual implica a passagem de cabo através da habitação do utente para cada uma das divisões a monitorizar, este processo é moroso e constitui entraves por parte dos utilizadores.

Além do desenvolvimento de novas funcionalidades é também de salientar a constante necessidade de monitorizar e manter o sistema actualizado, assim como corrigir qualquer falha que venha eventualmente a ser encontrada.

8.3 Considerações Finais

O desenvolvimento de um projecto de raiz e cujo o carácter pluridisciplinar traça desafios que incluem *software* e *hardware* é sem dúvida uma excelente oportunidade de aprendizagem.

A formação em engenharia biomédica foi uma mais valia, uma vez que a abordagem de disciplinas tão diversas como a *Fisiologia*, *Electrónica* e *Integração de Sistemas* permitem a percepção do sistema como um todo de uma forma que seria impossível enveredando por uma formação mais especializada.

Por fim, o desenvolvimento de um projecto em ambiente empresarial e em contacto directo com clientes das áreas da saúde e prestação de cuidados, assim como profissionais responsáveis pelo desenvolvimento e manutenção de produtos nesta área revelaram-se importantíssimos no meu desenvolvimento como aluno, pessoa e futuro empregado.

Bibliografia

[1] Max Roser. Life expectancy at birth, 2015. URL http://ourworldindata.org/data/population-growth-vital-statistics/life-expectancy/.

[2] Rui Miguel Vieira De Sousa. Envelhecimento da populaÇÃo portuguesa – algumas decorrÊncias econÓmicas. *Review of Scientific Instruments*, 2009. URL http://www.repository.utl.pt/bitstream/10400.5/1728/1/TESE%20-%20RUI%20SOUSA%20%2002-2010.pdf.

[3] Center for Research and Prevention of Injuries. Prevention of falls among elderly, 2003.

[4] AgeUK. Falls in the over 65s cost nhs £4.6 million a day, 2010. URL http://www.ageuk.org.uk/latest-press/archive/falls-over-65s-cost-nhs/.

[5] Dementia Uk. Alzheimer's society, 2014. URL http://www.alzheimers.org.uk/site/scripts/documents_info.php?documentID=412.

[6] Kent Beck. Agile manifesto, 2001. URL http://www.agilemanifesto.org/.

[7] Trello. Trello, 2015. URL http://trello.com.

[8] Margaret Rouse. Sprint definition - software development, 2015. URL http://searchsoftwarequality.techtarget.com/definition/Scrum-sprint.

[9] Github. Github, 2015. URL https://github.com.

[10] Nachiappan Nagappan. Realizing quality improvement through test driven development: results and experiences of four industrial teams, 2008. URL http://research.microsoft.com/en-us/groups/ese/nagappan_tdd.pdf.

[11] Ansible. Ansible provisioning, 2015. URL http://ansible.com.

[12] Wercker. Wercker - continuous integration, 2015. URL http://wercker.com.

[13] Deloitte. Primary care: Working differently. 24 Março 2015. URL http://www2.deloitte.com/content/dam/Deloitte/uk/Documents/life-sciences-health-care/deloitte-uk-telehealth-telecare.pdf.

[14] Tunstall UK. Tunstall solutions, 23 Março 2015. URL http://www.tunstall.co.uk/solutions/products.

[15] Appello UK. Activity monitoring, 23 Março 2015. URL https://www.appello.co.uk/Consumer/Products/activity-monitoring.

[16] Just Checking. Helping people to stay at home, 23 Março 2015. URL http://www.justchecking.co.uk/#simple2.

[17] Consumers Advocate. Senior alert systems, 2015. URL http://www.consumersadvocate.org/medical-alerts/senior-alert-system-comparison.

[18] Consumer Reports. What to look for in a medical alert system, 2015. URL http://www.consumerreports.org/cro/2014/06/what-to-look-for-in-a-medical-alert-system/index.htm.

[19] Laing & Buisson. Care of older people, uk market report. 2014.

[20] Oracle. What are restful web services, 2013. URL http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html.

[21] Network Working Group. Http over tls. Maio 2000. URL http://tools.ietf.org/html/rfc2818.

[22] E.A Osman. An estimation of a passive infra-red sensor's probability of detection. 2009. URL http://www.physicsegypt.org/nuppac09/npc9060.pdf.

[23] DSC. Dsc lc-100-pi, 2015. URL http://www.dsc.com/index.php?n=products&o=view&id=93.

[24] Tempoiq, 26 Março 2015. URL https://www.tempoiq.com.

[25] Orchestrateio, 26 Março 2015. URL https://orchestrate.io/.

[26] Dirk Bartels Jonathan Robie. A comparison between relational and object oriented databases for object oriented application development. 1994. URL

http://www.fing.edu.uy/inco/grupos/csi/esp/Cursos/cursos_act/2000/DAP_DisAvDB/documentacion/OO/POET_rel_vs_obj.pdf.

[27] E. F. Codd. A relational model of data for large shared data banks. 1970.

[28] Microsoft Developer Network. Acid properties, 2015. URL https://msdn.microsoft.com/en-us/library/aa480356.aspx.

[29] DB Engines. Database usage ranking, Março 2015. URL http://db-engines.com/en/ranking.

[30] Rick Cattell. Scalable sql and nosql data stores. 2010. URL http://www.cattell.net/datastores/Datastores.pdf.

[31] Oracle. Nosql and sql introspective. 2012.

[32] Martin Fowler. Schemaless data structures, 2013. URL http://martinfowler.com/articles/schemaless/.

[33] David Heinemeier Hansson. Ruby on rails, 2003. URL http://rubyonrails.org/.

[34] Django Software Foundation. Django, 2005. URL https://www.djangoproject.com/.

[35] Expressjs. URL http://expressjs.com/.

[36] Adam Bard. Which web framework is best?, 2015. URL http://www.quora.com/Which-is-the-best-web-application-framework-to-start-with-if-I-know-PHP-Ruby-Jav

[37] Deepanshu Mehndiratta. Django vs rails vs node, 2015. URL http://www.quora.com/Django-vs-Rails-vs-Node-js-vs-Code-Igniter-YII-Which-is-better.

[38] Diogo Nunes. Rails vs django vs play: Battle of frameworks, 2014. URL http://www.diogonunes.com/blog/rails-vs-django-vs-play-frameworks/.

[39] Global System for Mobile Communications. Digital cellular telecommunications system. URL http://www.etsi.org/deliver/etsi_ts/145000_145099/145001/12.01.00_60/ts_145001v120100p.pdf.

[40] Google. Spdy: An experimental protocol for a faster web. URL http://dev.chromium.org/spdy/spdy-whitepaper.

[41] Eclipse. Mqtt and coap, iot protocols. URL http://eclipse.org/community/eclipse_newsletter/2014/february/article2.php.

[42] Pragmateek. Json vs. xml: Some hard numbers about verbosity, 2013. URL http://www.codeproject.com/Articles/604720/JSON-vs-XML-Some-hard-numbers-about-verbosity.

[43] Google. Protocol buffers - developer guide, 2015. URL https://developers.google.com/protocol-buffers/docs/overview.

[44] Liam R. E. Quin. Xml essentials, 2015. URL http://www.w3.org/standards/xml/core.

[45] Digital ocean. URL http://digitalocean.com.

[46] Daniel Rice. Rails hosts: Amazon aws vs. digital ocean vs. heroku vs. engine yard, 2014. URL https://www.airpair.com/ruby-on-rails/posts/rails-host-comparison-aws-digitalocean-heroku-engineyard.

[47] Heroku. URL http://heroku.com.

[48] Rackspace. Rackspace, 2015. URL http://www.rackspace.co.uk/cloud/servers/pricing.

[49] Alasdair Allan. Which board is right for me?, 2014. URL http://makezine.com/magazine/make-36-boards/which-board-is-right-for-me/.

[50] Atmel Corporation. Interfacing gsm modems, 2006. URL http://www.atmel.com/Images/doc8016.pdf.

[51] Raspberry Pi foundation. Raspberry pi, 2015. URL https://www.raspberrypi.org/.

[52] Raspberry Pi Foundation. Raspberry pi - community, 2015. URL http://www.raspberrypi.org/community/.

[53] Inc. Electric Imp. Electric imp, 2015. URL https://electricimp.com/product/.

[54] Matt Asay. Ranking programming languages: Swift's growth is "unprecedented", 15 Janeiro 2015. URL http://readwrite.com/2015/01/15/popular-programming-languages-swift-go-r-redmonk.

[55] Python Software Foundation. Comparing python to other languages, 2015. URL `https://www.python.org/doc/essays/comparisons/`.

[56] Microsoft. Microsoft application architecture guide, 2009. URL `https://msdn.microsoft.com/en-us/library/ee658099.aspx#PresentationLayerConsiderations`.