FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Módulo 'O meu Mordomo' para Aplicações móveis e Domótica

João Miguel Mariz de Barros Zão

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Miguel Pimenta Monteiro

Co-orientador: José Morgado

29 de Junho de 2015

Resumo

O setor da domótica tem crescido de forma acentuada ao longo dos últimos anos, crescimento este influenciado pela grande evolução de áreas como as tecnologias de informação e automação. A integração dos dispositivos móveis permitiu adicionar portabilidade às soluções de automação residencial, registando-se cada vez mais aderentes às soluções existentes neste mercado. Contudo, na sua maioria, estas soluções apresentam problemas de interoperabilidade, pois estas são produzidas e distribuídas por vários fabricantes, cada um deles com diferentes objetivos. O mesmo acontece no desenvolvimento de aplicações móveis, dada a grande diversidade e heterogeneidade dos dispositivos móveis e seus sistemas operativos.

Esta dissertação vem precisamente dar ênfase à utilização dos dispositivos móveis no contexto da automação residencial. Combater a incompatibilidade existente entre os dispositivos móveis é um dos objetivos deste projeto. De modo a ultrapassar este problema de incompatibilidade recorreu-se à utilização de uma tecnologia que permitisse o desenvolvimento de uma aplicação multiplataforma para dispositivos móveis, neste caso o Xamarin. Esta aplicação móvel e a interligação de vários módulos permitiram a validação de uma primeira versão de uma arquitetura e prova de conceito para o mercado da automação residencial.

A validação do sistema passa por verificar a possibilidade de controlar e monitorizar dispositivos domésticos através de uma aplicação móvel. A arquitetura contemplou quatro grandes componentes nomeadamente uma aplicação multiplataforma para dispositivos móveis, um *web service*, uma base de dados e um Raspberry Pi. A aplicação multiplataforma para dispositivos móveis, compatível com Android, iOS e Windows Phone, permite que o utilizador monitorize e controle remotamente dispositivos associados ao sistema de forma fácil e intuitiva. O *web service* é responsável pelas comunicações entre os vários componentes. A base de dados é onde estão armazenados todas as informações relativas ao sistema. O Raspberry Pi foi utilizado de forma a permitir a atuação nos dispositivos associados ao sistema. O sistema desenvolvido, para além de permitir a monitorização e controlo de dispositivos domésticos associados ao sistema, é capaz de tomar decisões e realizar operações autonomamente, de acordo com o seu estado de funcionamento.

A implementação do sistema especificado e respetivos testes de validação da arquitetura e prova do conceito foram realizados com sucesso, concluindo-se que foram atingidos os objetivos propostos.

Abstract

The home automation sector has grown sharply over the past few years, this growth being influenced by the great development of areas such as information and automation technologies. The integration of mobile devices allowed to add portability to home automation solutions, registering an increasing number of solutions to this market. However, for the most part, these solutions have interoperability issues, since they are produced and distributed by various manufacturers, each with different objectives. The same is true in mobile application development, given the great diversity and variety of mobile devices and their operating systems. This thesis is precisely to emphasize the use of mobile devices in the context of home automation. To fight incompatibility between mobile devices is one of the objectives of this project. To overcome this incompatibility problem we have resorted to the use of a technology that would allow the development of a platform for mobile application, in this case, Xamarin. This mobile application and the interconnection of several modules allowed the validation of a first draft of an architecture and proof of concept for the home automation market. The system validation checks the possibility of controlling and monitoring the home devices via a mobile application. The architecture included four major components namely a multi-platform software for mobile devices, a web service, a database, and a Raspberry Pi. The multi-platform software for mobile devices, compatible with Android, iOS and Windows Phone allows the user to monitor and remotely control devices associated with it in an easy and intuitive way. The web service is responsible for communications between the various components. The database is where we store all information related to the system. The Raspberry Pi was used to enable the operation of the devices associated to the system. The developed system not only allows for the monitoring and control of home devices, but is also able to take decisions and carry out operations independently accordingly with its operation state. The implementation of the specified system and respective architecture validation and proof of concept tests were successfully carried out, concluding that the proposed objectives were achieved.

Agradecimentos

Em primeiro lugar gostaria de agradecer à CimSoft - Tecnologias de Informação, e ao meu supervisor Eng. José Morgado, pela forma como me acolheu nas suas instalações para a realização desta dissertação, facultado todas as condições para o desenvolvimento da mesma.

Quero expressar o meu sincero agradecimento ao meu orientador, Prof. Miguel Pimenta Monteiro, pelo seu interesse e acompanhamento, no desenvolvimento e escrita desta dissertação.

Agradeço também a todos os meus amigos que de uma forma ou de outra contribuíram para o meu sucesso como estudante.

Por último e o maior agradecimento vai para a minha família, em especial para os meus pais, irmã e ao meu grande amigo e cunhado João Miguel Machado, por me terem apoiado sempre em tudo e em todas as coisas.

João Miguel Mariz de Barros Zão

"Intelligence is the ability to adapt to change."

Stephen Hawking

Conteúdo

1 Introdução **1**
- 1.1 Enquadramento e domínio de intervenção . . . 1
- 1.2 Contexto e motivação . . . 2
- 1.3 Benefícios esperados com a solução do problema . . . 2
- 1.4 Objetivos . . . 3
- 1.5 Estrutura da Dissertação . . . 4

2 Domótica e dispositivos móveis **5**
- 2.1 Interfaces comerciais . . . 5
  - 2.1.1 Comparação entre soluções comerciais disponíveis . . . 5
- 2.2 Protocolos de comunicação usados na Domótica . . . 7
  - 2.2.1 X10 . . . 7
  - 2.2.2 INSTEON . . . 8
  - 2.2.3 Z-Wave . . . 9
  - 2.2.4 ZigBee . . . 9
  - 2.2.5 WiFi . . . 10
  - 2.2.6 Bluetooth . . . 11
  - 2.2.7 IEEE 802.15.4 . . . 12
- 2.3 Aplicações móveis e multiplataforma . . . 12
  - 2.3.1 Appcelerator Titanium . . . 14
  - 2.3.2 PhoneGap . . . 15
  - 2.3.3 Xamarin . . . 16
- 2.4 Internet das coisas e automação residencial . . . 19
  - 2.4.1 Raspberry Pi . . . 20
- 2.5 *Web Services* . . . 21
  - 2.5.1 REST . . . 21
  - 2.5.2 SOAP . . . 22
  - 2.5.3 WCF . . . 23
- 2.6 Conclusão . . . 23

3 Requisitos, funcionalidades e arquitetura **25**
- 3.1 Atores . . . 25
- 3.2 Requisitos Funcionais . . . 25
- 3.3 Requisitos não funcionais . . . 26
- 3.4 Casos de Uso . . . 26
  - 3.4.1 Autenticação . . . 27
  - 3.4.2 Utilização . . . 27
- 3.5 Arquitetura do sistema . . . 29

3.5.1 Aplicação Móvel - Interface com o utilizador . . . 30
3.5.2 Aplicação Raspberry Pi . . . 33
3.5.3 Base de dados . . . 34
3.5.4 *web service* . . . 36
3.5.5 Modo degradado . . . 38

4 Implementação **41**
4.1 Tecnologias implementadas . . . 41
4.2 Interface da aplicação móvel . . . 42
4.2.1 Autenticação . . . 42
4.2.2 Opções . . . 43
4.2.3 Menu principal . . . 44
4.2.4 Eventos Ocorridos . . . 45
4.2.5 Próximas atividades . . . 46
4.2.6 Serviços . . . 49
4.2.7 Extras . . . 55
4.2.8 Notificações . . . 55
4.3 *Webservice* . . . 57
4.4 Aplicação Raspberry Pi . . . 59

5 Resultados **61**
5.1 Operação em Android e iOS . . . 61
5.2 Operação e execução . . . 62
5.3 Testes e resultados de execução . . . 64
5.3.1 Testes de autenticação . . . 64
5.3.2 Teste de configuração à base de dados . . . 65
5.3.3 Testes de eventos ocorridos . . . 65
5.3.4 Testes de atuação em dispositivos . . . 66
5.3.5 Teste de regras . . . 66
5.4 Resultado da operação de um caso de utilização . . . 67

6 Conclusões e Trabalho Futuro **71**
6.1 Conclusão e satisfação dos objetivos . . . 71
6.2 Trabalho Futuro . . . 72

Referências **73**

Lista de Figuras

1.1 Diagrama do sistema . . . 3

2.1 HomeKit-enabled Insteon Hub. . . . 6
2.2 INSTEON peer-to-peer Network receptores . . . 8
2.3 Rede INSTEON com repetidores *peer-to-peer*. . . . 9
2.4 Tabela de frequências e potência . . . 11
2.5 Tabela de potência e alcance . . . 12
2.6 Native vs. Cross platform development . . . 13
2.7 Visão geral do Titanium SDK . . . 14
2.8 Diagrama PhoneGap . . . 15
2.9 Plataforma Xamarin . . . 16
2.10 Diagrama do projeto Xamarin, *Shared Porject* . . . 18
2.11 Diagrama do projeto Xamarin, *Portable Class Library*. . . . 19
2.12 Comparação Raspberry Pi Modelo A e B . . . 20
2.13 *Web Service REST* . . . 22

3.1 Modelo de caso de uso da autenticação . . . 27
3.2 Modelo de caso de uso da autenticação . . . 28
3.3 Arquitetura Mordomo . . . 29
3.4 Elementos gráficos Xamarin.Froms . . . 31
3.5 Diagrama da interface do Mordomo . . . 32
3.6 Diagrama da Aplicação do Raspberry Pi . . . 33
3.7 Modelo da base de dados . . . 35
3.8 Diagrama de composição do *web service* . . . 37
3.9 Possibilidades de funcionamento do sistema em modo degradado. . . . 39

4.1 Interface Opções . . . 43
4.2 Interface Opções . . . 44
4.3 Interface do menu principal . . . 45
4.4 Interface de eventos ocorridos . . . 46
4.5 Interface atividades esporádicas . . . 47
4.6 Interface atividades do dia a dia . . . 48
4.7 Interface Actividades do Fim-de-semana . . . 49
4.8 Interface de Serviços . . . 50
4.9 Interface do Alarme . . . 51
4.10 Interface do Aquecimento . . . 52
4.11 Interface da Iluminação . . . 53
4.12 Interface do Aquecimento . . . 54
4.13 Interface do Aquecimento . . . 55

4.14 Representação da notificação de sucesso e alerta 56
4.15 Representação da notificação de emergência e informação 57
4.16 Interface de um método presente na *Web Application* 58
4.17 Esquema utilizado para a ligação de componentes que simulam os dispositivos associados ao sistema 60

5.1 Comparação das interfaces entre dispositivos móveis Android e iOS 61
5.2 Diagrama de sequência de um caso de utilização 68
5.3 Imagem do dispositivo no estado ligado 69
5.4 Interface da aplicação alarme 69

Lista de Tabelas

2.1 Tabela de comparação entre protocolos na automação residencial [1] 7
2.2 Tabela de características 18
2.3 Tabela de características Raspberry Pi 2 21

3.1 Tabela resumo dos atores e requisitos. 26

5.1 Interação entre aplicação móvel e *Webservice* ao nível dos pedidos 62
5.2 Interação entre *Webservice* e a aplicação móvel ao nível das respostas 64

Abreviaturas e Símbolos

Android	Sistema operativo para dispositivos móveis desenvolvido pela Google Inc.
API	Application Programming Interface
ASP	Active Server Pages
CRUD	Create, Read, Update and Delete
DLL	Dynamic-link library
DSSS	Direct Sequence Spread Spectrum
HTTP	Hypertext Transfer Protocol
iOS	Sistema operativo para dispositivos móveis desenvolvido pela Apple Inc.
IEEE	Institute of Electrical and Electronics Engineers
LR-WPAN	Low-Rate Wireless Personal Area Network
IoT	Internet of Things
MAC	Media Access Control
OFDM	Orthogonal frequency-division multiplexing
REST	Representational State Transfer
RF	Radio frequency
RSS	Rich Site Summary
SOAP	Simple Object Access Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modeling Language
WiFi	Wireless Fidelity
WLAN	wireless local area network
WPAN	Wireless Personal Area Network
WSD	Web Services for Devices
WSDL	Web Services Description Language
WWW	World Wide Web

Capítulo 1

Introdução

1.1 Enquadramento e domínio de intervenção

A domótica, também conhecida como automação residencial, está no mercado há muitos anos e é responsável pelo projeto de edifícios inteligentes, que compreendem empreendimentos que combinam desenho arquitetónico e tecnologias avançadas, integradas e desenvolvidas conjuntamente. No entanto, apenas nos últimos anos teve um crescimento exponencial no contexto residencial, graças ao aumento da oferta em relação a dispositivos de baixo custo e conduzido pelo emergir de necessidades como poupança de energia, conforto, segurança, comunicação e serviços multimédia. [2]

O primeiro contacto com algo ligado a domótica surgiu em 1960. O projecto chamava-se casas com fio e na época foi desenvolvido em jeito de passatempo. Depois disso o primeiro nome oficial de automação residencial apareceu em 1984 através da *American Association of House Builders*. Este desenvolvimento pode ser considerado como a chave para as casas modernas inteligentes. As pessoas naquela época não tinham em consideração a forma de como a casa inteligente era construída, de como se usa o espaço ou se iria ser ambientalmente amigável. Apenas se interessavam pela interatividade das tecnologias que continha [2, 3].

Hoje em dia a domótica é um símbolo importante da sociedade humana. A primeira unidade de automação que entrou em casa das pessoas foi a máquina de lavar automática, automação de ar-condicionado e assim por diante. Quando se fala em automação residencial a maioria das pessoas imagina uma casa inteligente com um controlo remoto para todos os aparelhos domésticos. A ideia básica de automação residencial é monitorizar e controlar uma habitação utilizando sensores e sistemas de controle, em que através de vários mecanismos ajustáveis o utilizador pode desfrutar de um espaço personalizado em temperatura, ventilação, iluminação e outros serviços convenientes. No entanto o sentido de evolução indica que cada vez mais um sistema de automação residencial tenha autonomia na decisão de tarefas a realizar tendo em conta determinadas regras. É neste contexto que entra o conceito de *Internet of Things* em que todos os dispositivos têm capacidade de comunicar entre si. Com o cruzamento da informação e a parametrização de regras é possível tornar o sistema ainda mais inteligente, e mais importante ainda, com capacidades

de expansibilidade [4].

1.2 Contexto e motivação

As soluções de domótica são produzidas e distribuídas por vários fabricantes de componentes elétricos, cada um com diferentes objetivos funcionais e políticas de marketing, pondo em causa a sua interoperabilidade. Normalmente estas soluções de domótica são projetadas como uma evolução de componentes elétricos tradicionais como interrutores e relés, e neste caso apresentam uma grande desvantagem pois tornam-se incapazes de fornecer inteligência nativa além de cenários simples de automação. Isto era aceitável numa primeira fase de evolução onde as instalações eram poucas e isoladas, mas nos tempos que correm a domótica é vista como uma tecnologia que utiliza simultaneamente a electrónica e as tecnologias da informação em vários ambientes, permitindo realizar a gestão, local ou remota, e oferecer uma vasta gama de aplicações [4].

Os dispositivos móveis, com o aparecimento dos *Smartphones*, integram cada vez mais funcionalidades e aplicações que permitem ao utilizador o uso destes dispositivos para as mais variadas tarefas além do contacto telefónico. De todas as funcionalidades, o acesso à Internet é a que mais se destaca e que tende cada vez mais a estar presente em todos os dispositivos que nos rodeiam. Com esta funcionalidade é possível manter os dispositivos ligados a qualquer conteúdo e também a um sistema que controle um conjunto de equipamentos domóticos. Tendo em conta que grande parte das pessoas, nos dias que correm, estão em constantes deslocações, faz grande sentido que uma das principais necessidades seja a possibilidade de acesso em qualquer local onde o utilizador sinta a necessidade de controlar ou monitorizar o estado da sua habitação.

1.3 Benefícios esperados com a solução do problema

A abordagem nesta dissertação vem precisamente dar ênfase à utilização dos dispositivos móveis no contexto da automação residencial. Tendo em conta o avanço tecnológico nos *Smartphones*, o seu baixo custo e por consequência a ampla e quase unânime utilização destes dispositivos na sociedade dos dias de hoje, fazem deste dispositivo um dos aspectos chave na evolução da automação residencial. Hoje em dia o telemóvel pode ser considerado um pequeno mordomo, no sentido em que este já realiza imensas tarefas de forma autónoma facilitando a vida do seu utilizador. Seguindo esta linha de pensamento chega-se facilmente à conclusão de que o conceito de casa inteligente é deveras semelhante ao de um *Smartphone* e com a junção de ambos se obtém um sistema com enorme potencial.

Tal como foi descrito anteriormente, as soluções de domótica são produzidas e distribuídas por vários fabricantes o que resulta em problemas de interoperabilidade entre as várias marcas. O mesmo acontece no desenvolvimento de aplicações móveis, em que as incompatibilidades entre os sistemas operativos dos três gigantes dos dispositivos móveis, Android, iOs e Windows Phone são globalmente conhecidas. Recorrer a uma tecnologia que permita o desenvolvimento de software multiplataforma é uma das possíveis formas de abordar este problema.

1.4 Objetivos

Pretende-se como objetivo da dissertação, desenvolver software e interligar vários módulos que permitam a validação de uma primeira versão de uma arquitetura e prova de conceito para o mercado de automação residencial. A arquitetura deve incluir uma aplicação multiplataforma para dispositivos móveis, fácil e intuitiva de manipular, que permita aos utilizadores monitorizar e controlar outros dispositivos remotamente.

O requisito de que esta aplicação esteja acessível ao utilizador em qualquer local e a qualquer momento é sem duvida um dos principais objetivos. Para que tal seja possível é necessário que a aplicação seja compatível com os três principais sistemas operativos usados nos *smartphones* nos dias que correm, Android, iOs e Windows Phone. O que vai permitir ao utilizador usufruir de todas as suas utilidades independentemente do local onde se encontra e do dispositivo móvel utilizado.

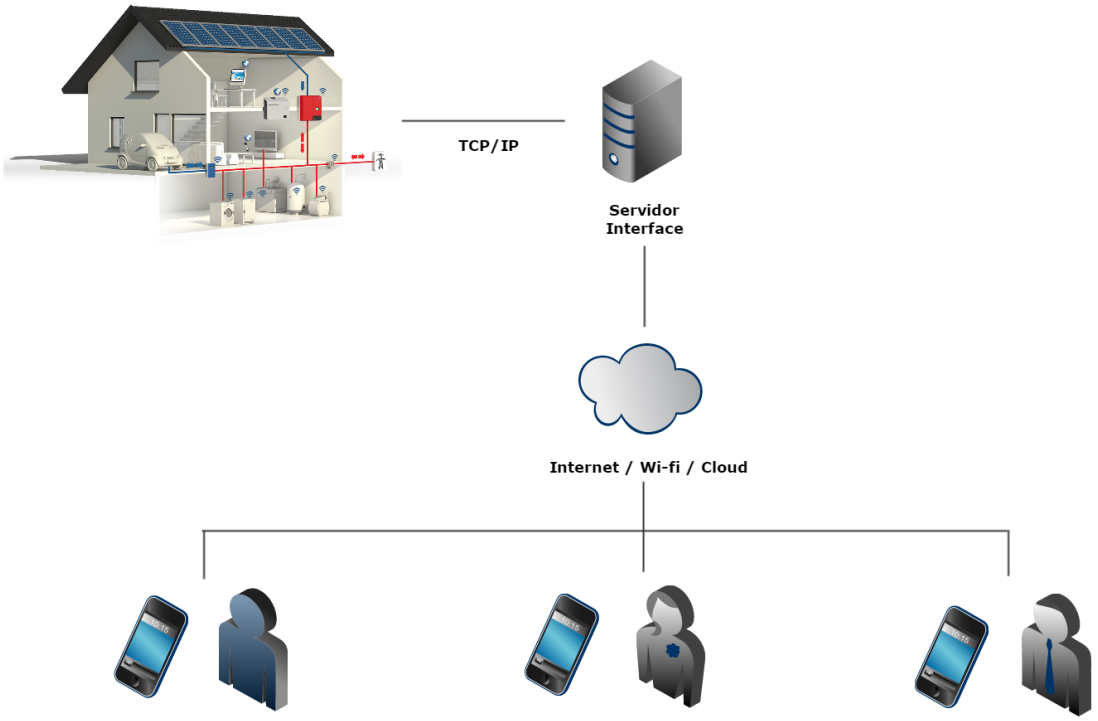


Figura 1.1: Diagrama do sistema

Outro objetivo é validar a utilização da aplicação móvel e a sua integração num sistema de automação residencial, demonstrando a capacidade e o potencial da utilização dos dispositivos móveis no controlo e monitorização de sistemas deste tipo. A Figura 1.1 demonstra a ligação existente entre os vários componentes deste projecto. Vários dispositivos de uma casa a comunicar com um servidor que por sua vez coloca e organiza a informação na *internet*. Por fim os utilizadores com recurso à aplicação que deve integrar o sistema terão acesso a todas as informações dos dispositivos do sistema, podendo controlá-los e fazer a monitorização da sua casa

inteligente onde quer que estejam.

1.5 Estrutura da Dissertação

Para além do presente capítulo, esta dissertação contém mais cinco capítulos. No capítulo 2 é feita a revisão bibliográfica e são apresentadas algumas soluções comerciais atuais. No 3 é feita uma análise dos requisitos, a descrição da arquitetura do projeto desenvolvido, a identificação de itens criados, bem como as suas funcionalidades, interações e ligações. No 4, é descrita toda a implementação do sistema, de modo a poder validar a arquitetura escolhida. No 5 são apresentados os resultados da implementação e apresentados os testes realizados. Finalmente, no 6 são retiradas as conclusões acerca da satisfação dos objetivos propostos e feitas considerações sobre possíveis desenvolvimentos futuros.

Capítulo 2

Domótica e dispositivos móveis

2.1 Interfaces comerciais

As interfaces têm uma função preponderante em qualquer aplicação disponível, pois é através destas que os utilizadores têm a possibilidade de interagir com o sistema.

Sendo os sistemas domóticos controlados pelos habitantes das casas onde estão instalados, estes por norma apresentam interfaces fáceis de manipular. Inicialmente estas interfaces eram apenas controlos físicos como interruptores e botões, evoluindo para o uso de aplicações em software que permitem facilmente controlar e visualizar algumas propriedades do sistema. Hoje em dia existem interfaces disponíveis tais como INSTEON, Revolv, WeMo ou SmartThings . Todas estas tecnologias, actualmente, permitem o controlo e acesso aos dispositivos remotamente através de *Smartphones*. Nestes sistemas as interfaces com acesso à Internet estão em expansão devido à evolução dos *Smarphones* e do aumento da velocidade e cobertura de acesso à Internet nos dispositivos móveis nos últimos tempos [5, 2].

A Internet das Coisas tem vindo lentamente a invadir o nosso quotidiano e as empresas estão a preparar-se para dar uma resposta capaz e à altura do que vai ser exigido. Um bom exemplo foi a aquisição da Revolv pela Nest que por sua vez foi recentemente comprada pela Google [6]. Esta aquisição da Google mostra a forte aposta que a empresa pretende fazer no contexto da Internet das coisas, até pelo seu novo sistema operativo, Brillo, já anunciado nas recentes conferências Google I/O, de Maio de 2015. A corrida para a criação do melhor e mais optimizado sistema operativo que irá equipar os novos dispositivos que se vão ligar à Internet está a ser encarada com seriedade não só pela Google mas também por empresas como a Microsoft, que tem uma versão do Windows 10 dedicada à Internet das coisas, e a Huawei que recentemente lançou o LiteOS, um sistema operativo de 10KB [7, 8].

2.1.1 Comparação entre soluções comerciais disponíveis

A INSTEON tem décadas de experiência na construção de dispositivos de controlo doméstico. Com o recente lançamento do *Hub* HomeKit, a INSTEON permite aos seus utilizadores

expandirem o conjunto de dispositivos pois o HomeKit permite a utilização de produtos de outras empresas.

Os dispositivos HomeKit comunicam através de Bluetooth e WiFi, no entanto os dispositivos que pertencem à INSTEON tais como luzes, tomadas interruptores,sensores ou câmaras para além de usarem WiFi também usam a rede elétrica para comunicar. Isto acontece porque a INSTEON é um sistema híbrido que combina a tecnologia X10 e redes sem fio, permitindo a quem já possuía dispositivos X10 que são conhecidos por usarem a rede elétrica de energia para comunicar, não necessite de descartar esses dispositivos caso queira migrar para INSTEON. Neste aspecto a INSTEON tem vantagem face a concorrentes como os já referidos acima WeMo ou SmartThings [9, 1].

Figura 2.1: HomeKit-enabled Insteon Hub.

A WeMo comparativamente à INSTEON tem, até ao momento, menos dispositivos. A comunicação é feita apenas através de redes em Rádio Frequência e a tecnologia usada é WiFi. Em termos de interface para dispositivos móveis a WeMo tem compatibilidade com iOS e Android.

A SmartThings tem disponíveis mais dispositivos que a WeMo mas ainda assim continua, actualmente, atrás da INSTEON. Em termos de comunicações utilizam redes de Rádio Frequência tal como a WeMo mas em termos de protocolos de domótica usam WiFi, ZigBee e Z-Wave. Tal como a INSTEON, em termos de dispositivos móveis, oferecem compatibilidade com iOS, Android e Windows Phone [1].

2.2 Protocolos de comunicação usados na Domótica

As formas de comunicação entre equipamentos são efetuadas através de protocolos, ou seja um equipamento só poderá comunicar com outro equipamento que obedeça ao mesmo protocolo. Existe uma grande variedade de protocolos normalizados que são direcionados para a Domótica.

Tabela 2.1: Tabela de comparação entre protocolos na automação residencial [1]

	X10	ZigBee	Z-Wave	INSTEON
Dual Mesh(Powerline e Wireless RF)	Não	Não	Não	Sim
Powerline Communication	Sim	Não	Não	Sim
Powerline Data Rate(bps)	20	N.A.	N.A.	2.400
RF Wireless Communication	Não	Sim	Sim	Sim
RF Wireless Data Rate(bps)	N.A.	20.000	9.600	38.400
Full Mesh Network Communication	Não	Não	Não	Sim
Automatic Network Enrollment	Não	Não	Não	Sim
Maximum Devices Per Network	256	65.536	232	16,777,216
All Devices are Peers	Não	Não	Não	Sim
Command Acknowledgement	Não	Sim	Sim	Sim
X10 Compatible	Sim	Não	Não	Sim

2.2.1 X10

O protocolo X10 é de momento a tecnologia mais acessível para a realização de uma instalação domótica não muito complexa pois usa a rede de distribuição de energia eléctrica de 230V existente como o principal meio de comunicação entre os vários dispositivos. Ao usar as linhas eléctricas da habitação, não se torna necessário ter novos cabos para ligar os dispositivos.

A grande vantagem desta tecnologia em relação a outros protocolos de domótica é que pode ser aplicada em qualquer momento, tanto na altura da construção como posteriormente. Um sistema X10 pode ser constituído por um conjunto de dispositivos que são comandados directamente pelo utilizador. Por exemplo, através de um telecomando RF, o utilizador poderá enviar um comando para o receptor X10/RF, que é transmitido através da rede eléctrica a um actuador X10 que, por sua vez, liga/desliga o aparelho. A comunicação do X10 recorre a um pequeno sinal de potência existente na rede eléctrica da habitação e modula esse sinal numa frequência maior (120KHz) e injecta-o de novo na rede eléctrica de 230V AC, através do módulo emissor, representando sinais binários (cada bit 1 numa transmissão X10 é um *burst* de 120KHz na origem do sinal AC, e cada bit 0 é a ausência deste *burst*). Este sinal é inserido logo a seguir à passagem pela origem da onda sinusoidal de 50Hz. O problema de comunicar pela rede eléctrica é ficar-se sujeito aos ruídos que essa rede possa ter. Os ruídos são sinais eléctricos indesejados que podem eventualmente existir na mesma rede eléctrica a par dos sinais desejados. Para reduzir a probabilidade de um sinal ser confundido com ruído, foi adoptada a seguinte política: Por cada bit é enviado o seu valor lógico e o seu complemento. Isto significa na prática que, sempre que se pretende enviar o bit 1, isso corresponde a enviar um 1 (sinal de 120kHz na origem) seguido de um 0 (ausência de sinal).

O envio do bit 0 corresponde a enviar um 0 (ausência de sinal) seguido de um 1 (frequência de 120kHz na origem). Tem como aspecto negativo reduzir o ritmo de transmissão que fica assim restrito a 50 bps (é enviado um bit por cada ciclo da rede eléctrica). Como consequência das baixas velocidades de transmissão, os transmissores apenas são capazes de realizar operações simples que envolvam poucos dados (ligar ou desligar aparelhos e luzes e regular a intensidade luminosa de lâmpadas) [9, 10]. O X10, como usa a rede eléctrica para comunicar não serve para lidar com sinais digitais de alta resolução como sinais de vídeo, televisão e hi-fi.[1].

2.2.2 INSTEON

A INSTEON permite que simples dispositivos e de baixo preço estejam ligados em rede usando a rede elétrica de energia, RF ou ambas. Todos os dispositivos INSTEON são *peers* o que quer dizer que qualquer dispositivo pode transmitir, receber ou repetir mensagens sem ter que ter permissão. Quantos mais dispositivos forem adicionados à rede INSTEON mais robusta esta fica pois os dispositivos INSTEON repetem as mensagens uns dos outros simultaneamente(*simulcasting*). Quanto maior for o número de dispositivos na rede mais forte o sinal fica. Quando a comunicação é feita através da rede elétrica a INSTEON usa os princípios da rede X10 descritos anteriormente [1].

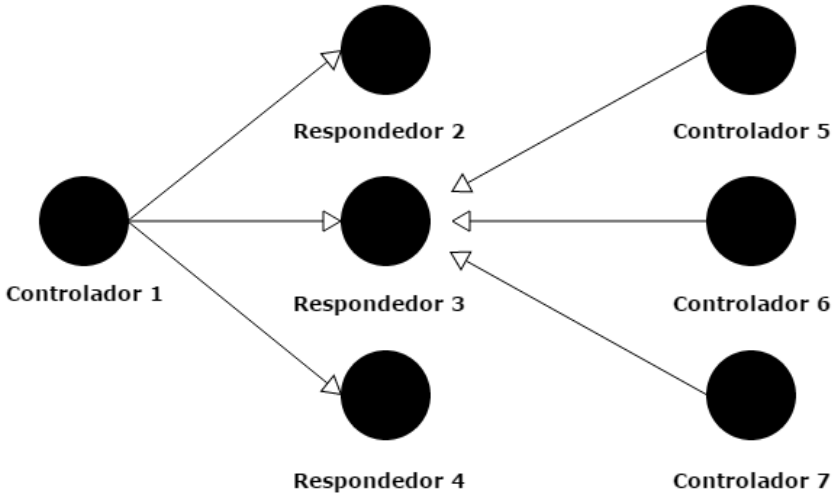


Figura 2.2: INSTEON peer-to-peer Network receptores

A Figura 2.2 ilustra o modo de funcionamento da rede *peer-to-peer* onde o dispositivo 1 está a actuar como um controlador, envia mensagens para os dispositivos 2, 3 e 4 que estão a actuar como recetores. Por outro lado também é possível ter vários controladores, como é o caso dos

[1] Hi-fi é a reprodução de áudio feita por um aparelho de reprodução de som com a maior fidelidade possível ao som real.

dispositivos 5, 6 e 7 que estão a actuar como controladores e enviam mensagens para um único receptor INSTEON, neste caso o dispositivo 3 que está a atuar como recetor.

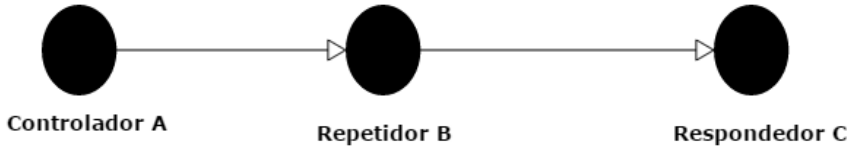


Figura 2.3: Rede INSTEON com repetidores *peer-to-peer*.

Já na Figura 2.3 podemos observar o modo de funcionamento com repetidor em que o dispositivo A está a actuar como controlador, envia uma mensagem que vai ser repetida pelo dispositivo B e chega então ao destino que neste caso é o dispositivo C.

Com isto é possível verificar o que foi afirmado anteriormente sobre o aumento da força do sinal da rede quando o número de dispositivos disponíveis é grande.

2.2.3 Z-Wave

O protocolo de comunicação Z-Wave foi desenvolvido pela Zensys, Inc. uma empresa sediada na Dinamarca em 2004. Baseado no conceito do protocolo ZigBee, o objectivo da Z-Wave foi a construção de uma tecnologia mais simples e não tão dispendiosa como a ZigBee. A Z-Wave opera a 908,42 MHZ nos EUA e a 868,42 MHZ na Europa. Teoricamente uma rede Z-Wave pode conter até 323 nós, no entanto há relatos de problemas com redes que contenham mais de 30-40 nós. Deve também ter-se cuidado na escolha dos dispositivos, pois certos fabricantes têm problemas de compatibilidade com dispositivos de outros fabricantes. A Z-Wave usa uma *routed network*, o que torna os seus produtos mais complexos na instalação e por sua vez mais caros. Os produtos Z-Wave comunicam via RF, resultando em potenciais problemas de confiabilidade e alcance. A Leviton é uma empresa que usa Z-Wave mas que obteve resultados um pouco dececionantes no mercado de automação residencial, onde a facilidade de uso é um dos principais desafios, muito devido à complexidade da sua instalação [9, 1].

2.2.4 ZigBee

A tecnologia utilizada no protocolo ZigBee é comparável às redes WiFi e Bluetooth e diferencia-se destas por desenvolver menor consumo, por um alcance reduzido, cerca de 100 metros, e a comunicação entre dois dispositivos poder ser repetida sucessivamente pelas unidades existentes na rede até atingir o destino final. Os dispositivos ZigBee são projetados para comunicarem através

de frequências de rádio, usando 2,4GHz como frequência padrão mundial. No entanto nos Estados Unidos da América usa 915MHz e na Europa 866MHz. Tanto a Z-Wave como a ZigBee usam interruptores como repetidores na sua *mesh network*. Assim a fiabilidade da rede é diretamente proporcional ao número de unidades instaladas. Quanto menor o número de unidades menos confiável é a *mesh network*. A adição de estação base[2] e de repetidores aumentam a fiabilidade, a distância e também o custo. Pontos únicos de falha podem resultar em efeitos dramáticos e em alguns casos pode até tornar partes da rede ou da casa inoperante [9, 1].

A ZigBee enfrenta os mesmo desafios que a Z-Wave, e mais um ainda, pois a especificação ZigBee é mais complexa do que a Z-Wave e é a tecnologia mais cara quando comparada com X10, INSTEON ou Z-Wave.

2.2.5 WiFi

WiFi é um conjunto de especificações para redes locais sem fio (WLAN - Wireless Local Area Network) baseada no padrão IEEE 802.11. O nome WiFi é tido como uma abreviatura do termo inglês *Wireless Fidelity*, embora a WiFi Alliance, entidade responsável principalmente pelo licenciamento de produtos baseados na tecnologia, nunca tenha afirmado tal conclusão. É comum encontrar o nome WiFi escrito como Wi-Fi ou até mesmo wifi. Todas estas denominações se referem à mesma tecnologia.

Com a tecnologia WiFi, é possível implementar redes que conetam computadores e outros dispositivos compatíveis que estejam próximos geograficamente. Estas redes não exigem o uso de cabos, já que efetuam a transmissão de dados por meio de RF. Este esquema oferece várias vantagens, entre as quais podemos destacar: utilização da rede em qualquer ponto dentro dos limites de alcance da transmissão; possibilidade de inserção rápida de outros computadores e dispositivos na rede; inexistência da necessidade de adaptação de paredes ou estruturas para a passagem de cablagem.

A flexibilidade do WiFi é tão grande que se tornou viável a implementação de redes que fazem uso desta tecnologia nos mais variados lugares, principalmente pelo facto de as vantagens citadas no parágrafo anterior muitas vezes resultarem em diminuição de custos.

A tecnologia WiFi pode ser facilmente utilizada para redes de sensores sem fio. Porém, se formos compará-la com outros protocolos, não é a mais recomendada, pelo alto preço, consumo de banda e energia. Atecnologia WiFi é mais indicada para aplicações que necessitem de transmissões de áudio e vídeo.

O padrão 802.11g foi disponibilizado em 2003 e é tido como o sucessor natural da versão 802.11b, uma vez que é totalmente compatível com este. Isso significa que um dispositivo que opera com 802.11g pode comunicar com outro que trabalha com 802.11b sem qualquer problema, exceto o facto de que a taxa de transmissão de dados é, obviamente, limitada ao máximo suportado por este último.

[2]Estação base - É por processar, manipular, armazenar numa base de dados e disponibilizar os dados em tempo real através de uma página de Internet

Padrão	Região/País	Frequência	Potência
802.11b & g	América do Norte	2,4 - 2,4835 GHz	1000 mW
802.11b & g	Europa	2,4 - 2,4835 GHz	100 mW
802.11b & g	Japão	2,4 - 2,497 GHz	10 mW
802.11b & g	Espanha	2,4 - 2,4875 GHz	100 mW
802.11b & g	França	2,4 - 2,4835 GHz	100 mW
802.11a	América do Norte	5,15 - 5,25 GHz	40 mW
802.11a	América do Norte	5,25 - 5,35 GHz	200 mW
802.11a	América do Norte	5,47 - 5,725 GHz	Não aprovado
802.11a	América do Norte	5,725 - 5,825 GHz	800 mW

Figura 2.4: Tabela de frequências e potência

O principal atrativo do padrão 802.11g é poder trabalhar com taxas de transmissão de até 54 Mb/s, assim como acontece com o padrão 802.11a. No entanto, ao contrário desta versão, o 802.11g opera com frequências na faixa de 2,4 GHz (canais de 20 MHz) e possui praticamente o mesmo poder de cobertura do seu antecessor, o padrão 802.11b. A técnica de transmissão utilizada nesta versão também é o *Orthogonal Frequency Division Multiplexing*, OFDM, todavia, quando é feita comunicação com um dispositivo 802.11b, a técnica de transmissão passa a ser o *Direct Sequence Spread Spectrum*, DSSS [1].

2.2.6 Bluetooth

Bluetooth é uma tecnologia criada pela Ericsson em 1994, sendo caracterizada como pessoal e de curta distância (WPAN). Opera na frequência 2,4 GHz ISM. Dentre outros aspetos foi destinada para substituir cabos de ligação, porém com altos níveis de segurança. As suas principais características são a robustez, baixo consumo e custo acessível, além de ser compatível com outras tecnologias. Vários dispositivos podem conectar-se formando redes limitadas a 8 nós, conhecidas como piconets, que podem juntar-se a estruturas maiores, chamadas scatternets. É atrativa para RSSF pelo facto de facilitar a formação de redes ad-hoc, pois possui um protocolo MAC específico [1].

O Bluetooth possibilita a comunicação entre dispositivos quando estes estão dentro do raio de alcance. Os dispositivos usam um sistema de comunicação via rádio, por isso não necessitam estar na linha de visão um do outro, e podem estar até em outros ambientes, contando que a transmissão seja suficientemente potente.

Classe	Potência máxima permitida	Alcance (Aproximadamente)
Classe 1	100 mW (20 dBm)	até 100 metros
Classe 2	2.5 mW (4 dBm)	até 10 metros
Classe 3	1 mW (0 dBm)	~ 1 metro

Figura 2.5: Tabela de potência e alcance

2.2.7 IEEE 802.15.4

O padrão 802.15.4 da IEEE é definido como um protocolo para interconexão entre dispositivos de comunicação de dados utilizando uma a taxa de dados mínima, uso pouco elevado de potência, baixa complexidade e transmissões de rádio frequência de pequeno alcance numa rede WPAN, que é uma rede sem fios utilizada para transportar dados em pequenas distâncias. Ao contrário das redes WLAN, as conexões WPAN têm pouca infraestrutura, permitindo soluções de baixo custo e com alta eficiência energética. O padrão IEEE 802.15.4 é padronizado para as camadas físicas e de acesso ao meio, e para redes WPAN com baixas taxas (LR-WPAN), que são redes de comunicação simples e de baixo custo que toleram conexões em aplicações com potência baixa e que precisam de desempenho sem muita rigidez. As redes LR-WPAN são fáceis de instalar, confiáveis em transferência de dados, têm suporte à operação de curto alcance, baixo custo e alta vida útil, com um protocolo flexível e simples [11].

2.3 Aplicações móveis e multiplataforma

Os dispositivos móveis ocupam cada vez mais tempo e espaço nas sociedades modernas. Isto deve-se ao facto de que praticamente tudo o que antes apenas era possível fazer através de um computador agora pode ser feito em qualquer lugar através de um dispositivo móvel. Com o emergir desta tecnologia, foram vários os sistemas operativos a surgir no mercado. De entre os quais se destacam: Android, iOS e Windows Phone.

O Android é um sistema operativo desenvolvido pela Google e é o sistema operativo móvel mais utilizado do mundo. É baseado no núcleo linux e tem uma interface de utilização baseada na manipulaçao direta, pois é projetado principalmente para dispositivos móveis com tela sensível ao toque como *smartphones* e *tablets*. O código deste sistema operativo é disponibilizado pela Google sob a licença de código aberto, sendo que isto permite que este possa ser adaptado e utilizado por outras empresas [12].

O iOS é o sistema operativo móvel da Apple Inc. inicialmente desenvolvido para o iPhone mas que posteriormente passou a ser utilizado em todos os dispositivos móveis da empresa tais como: iPod touch, iPad e Apple TV. Ao contrário a Google a Apple não permite que o seu sistema operativo móvel,iOS, seja executado em hardware de terceiros [13].

O Windows Phone o sistema operativo móvel da Microsoft, baseado no núcleo do Windows CE 6.0. Tem como principal característica poder ser sincronizado com o Windows 8. Tal como a Apple, a Microsoft não disponibiliza o código do seu sistema operativo móvel sob a licença de código aberto.

Ao longo dos últimos dois anos o mercado para o desenvolvimento de aplicações móveis multiplataforma para o meio empresarial aumentou significativamente. Grandes e médias empresas estão cada vez mais a adaptar-se ao mundo móvel e percebendo a necessidade de disponibilizar aplicativos da sua linha de negócios para *tablet* e *smartphone*. A necessidade da capacidade de rapidamente desenvolver e implantar aplicações em escala implica que estes estejam disponíveis de forma a que abranjam o maior número de possíveis utilizadores. As plataformas de desenvolvimento para múltiplos sistemas operativos móveis tem um papel importantíssimo neste contexto. O IDC, *International Data Corporation*, prevê que o mercado para estas plataformas de desenvolvimento terão uma taxa de crescimento anual composta de mais de 38%, atingindo os 4,8 biliões de dólares em 2017 e a Gartner espera que mais de 20 milhões de aplicativos empresariais venham a ser desenvolvidos em 2018. Este valor é evidente no número crescente de investimento no desenvolvimento móvel a nível empresarial [14].

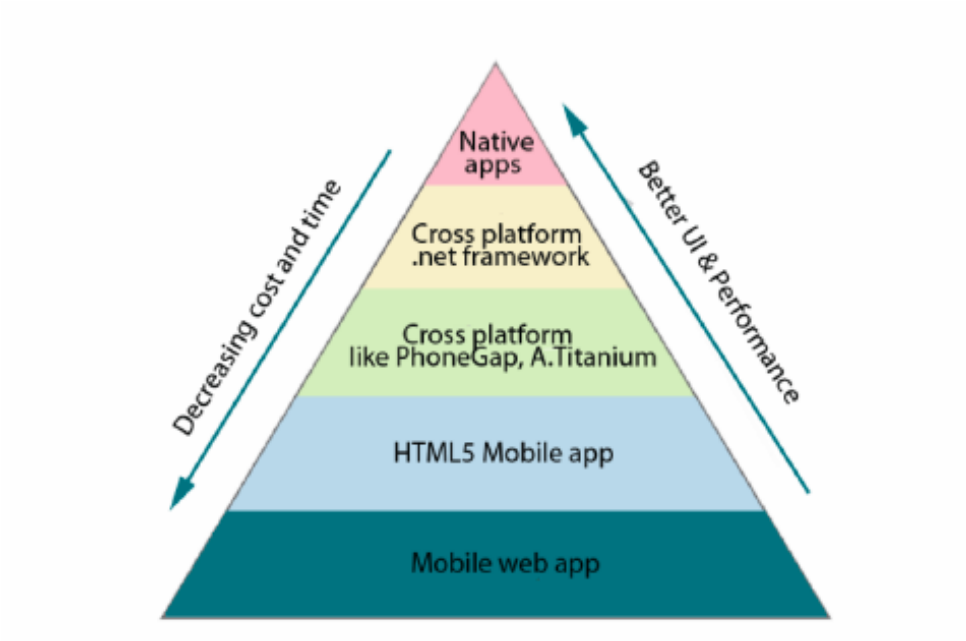


Figura 2.6: Native vs. Cross platform development

As multi-plataformas de desenvolvimento têm como objetivo a diminuição dos custos e tempo de desenvolvimento já que produzem um código base que está destinado a ser usado nos vários dispositivos e sistemas operativos. A 2.7 mostra a tendência de utilização das tecnologias de desenvolvimento móvel, onde se vê a importância relativa das plataformas nativas face ao desenvolvimento multiplataforma no que diz respeito aos factores de tempo e de custos.

2.3.1 Appcelerator Titanium

Appcelerator Titanium é uma *framework open source* utilizada para o desenvolvimento de aplicações móveis multiplataforma. A linguagem usada é o JavaScript, com recurso à API do Titanium fornecida numa biblioteca própria. O código, baseado em tecnologias Web tais como PHP, Python, Ruby, HTML, CSS e JavaScript é compilado e empacotado através das ferramentas nativas de cada plataforma, ou seja é gerado código nativo Objectiv-C através do XCode e iOS SDK, ou Java, através do Android SDK, para dispositivos Apple e Android respetivamente. O Appcelerator Titanium suporta bases de dados, ficheiros de média, localização, acesso a câmara, contactos e muitos outros aspetos nativos de um *smartphone*. A ferramenta Titanium permite aos *Web developers* criar aplicações móveis, para computador e tablet [14].

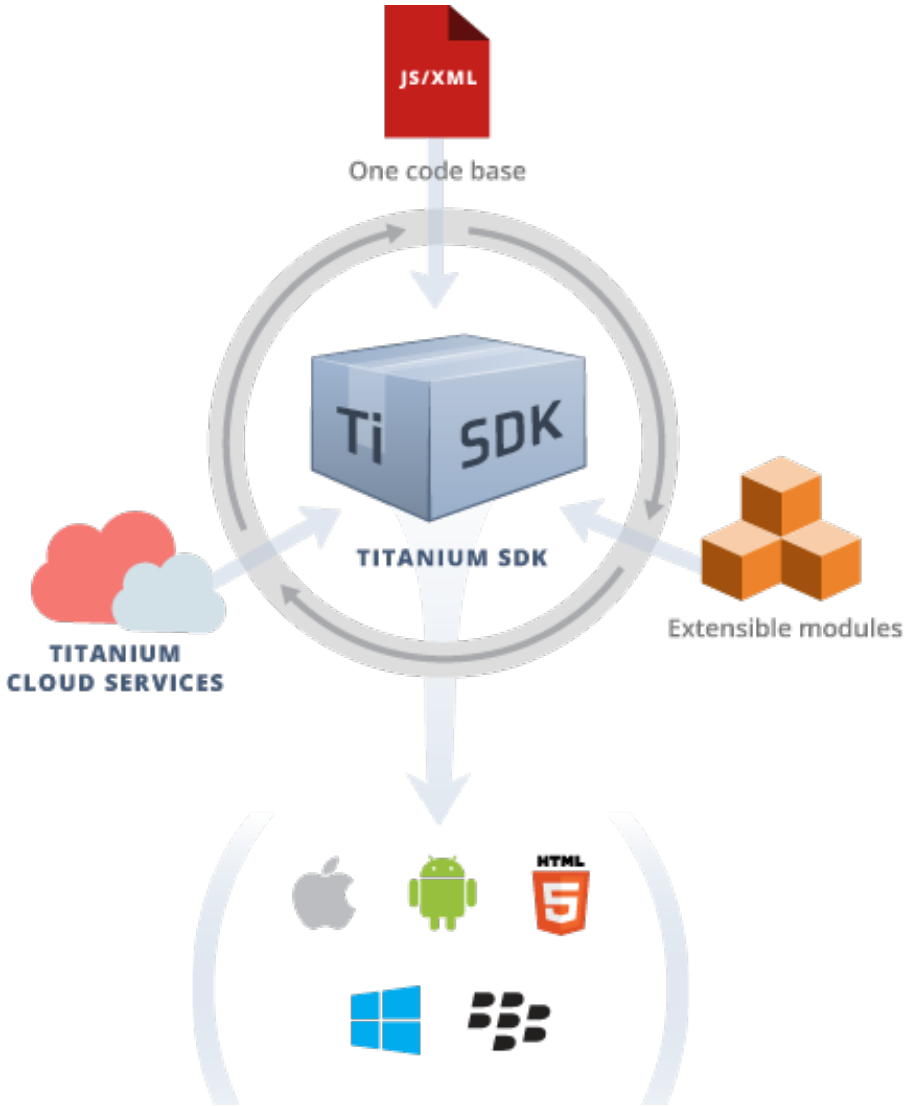


Figura 2.7: Visão geral do Titanium SDK

A Appcelerator Titanium Mobile permite a reutilização de código para plataformas diferentes, e não requer um *Browser* para execução e utilização de tecnologias web conhecidas. Estas características resultam na principal vantagem que é o baixo consumo de tempo na produção das aplicações. No entanto está limitada a algumas áreas suportadas diretamente pela API, o tempo de compilação é elevado e foram já relatados alguns problemas ou *bugs* da API.

2.3.2 PhoneGap

O PhoneGap tal como o Appcelerator Titanium, é uma *framework open source*. O PhoneGap é totalmente gratuito e permite a criação de aplicações móveis utilizando APIs padronizadas para Web. É indicado para desenvolvimento de aplicativos de pequeno e médio porte.

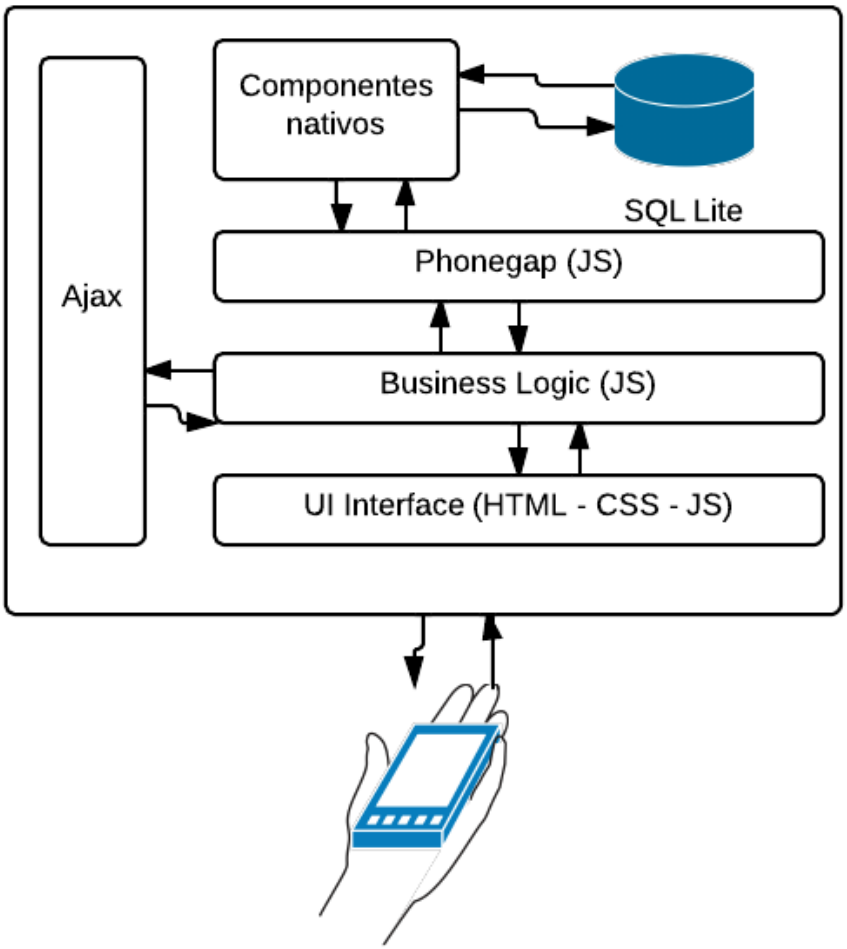


Figura 2.8: Diagrama PhoneGap

Esta ferramenta foi desenvolvida pela Nitobi em 2008, suportando inicialmente apenas iOS, Android e Blackberry 4 e posteriormente Symbian, WebOS e Windows Phone. Em 2011, a Adobe compra a Nitobe e o PhoneGap foi doado à Apache Software Foundation [14].

O PhoneGap é um conjunto de APIs que permitem o acesso a funções nativas dos dispositivos, e tal como acontece no Appcelerator Titanium, este acesso é feito através de linguagens Web tais como JavaScript, HTML5 e CSS, em vez de linguagens específicas de dispositivos móveis, como o Objectiv-C. O desenvolvimento de aplicações usando PhoneGap é similar ao desenvolvimento de aplicações web, permitindo a quem já tenha experiência em desenvolvimento web uma melhor e mais fácil experiência no desenvolvimento de aplicações móveis. Estes aspectos resultam em custos menores na elaboração de projectos em ambiente empresarial [15, 16].

2.3.3 Xamarin

O Xamarin é uma ferramenta comercial para o desenvolvimento de aplicações móveis. Numa altura em que os 3 grandes rivais do mundo *mobile* competem pelo mercado, o Xamarin promete o verdadeiro desenvolvimento multiplataforma em C#, que partilha em média 75% do código entre plataformas. Ao contrário das ferramentas de desenvolvimento descritas anteriormente, Appcelerator Titanium e PhoneGap, o Xamarin permite aceder aos elementos nativos UI de desenvolvimento de cada plataforma, Android, iOS e Windows Phone [17].

Figura 2.9: Plataforma Xamarin

Xamarin é baseado numa implementação *open-source* do .NET chamada mono. Esta implementação inclui o seu próprio compilador C# , ambiente de desenvolvimento e as principais bibliotecas .NET [18].

Existe uma diferença fundamental entre iOS e Android quando se trata do desempenho das aplicações, que é a compilação preliminar dos mesmos. Nos dispositivos Android é usada uma máquina virtual Java Dalvik para executar as aplicações. As aplicações feitas em Java já são pré-compiladas em *bytecode*, que o Dalvik interpreta em comandos para o processador quando a aplicação está em execução. A isto chama-se compilação *Just-in-time* [17].

Já a compilação em dispositivos iOS é chamada de compilação *Ahead-of-time*. O Xamarin cuida dessa diferença e fornece compiladores individuais para cada plataforma, de modo a obter como resultado aplicações nativas.

O Xamarin permite a utilização de mecanismos de desenvolvimento de UI nativas para cada plataforma, o que de certa forma vai contra o objetivo principal de uma ferramenta de desenvolvimento multiplataforma em que o pretendido é que se escreva apenas uma aplicação para várias plataformas. Mas caso o utilizador queira usar estes mecanismos de criação nativa de UI, tem que se sujeitar a isso. No entanto a única camada a escrever especificamente para cada plataforma é a camada UI. Por forma a combater esse aspeto a Xamarin tratou de encapsular as APIs nativas de cada sistema operativo para criar a API Xamarin.Forms que veio permitir compartilhar ainda mais código no desenvolvimento de aplicações [17, 19].

O Xamarin tem a sua própria loja com componentes *third-party*, que está integrado no IDE e permite adicionar componentes diferentes, escritos tanto por especialistas Xamarin bem como por programadores independentes. Apesar de alguns destes componentes serem pagos, a maior parte deles são gratuitos mas nem todos são multiplataforma pois alguns estão disponíveis apenas para uma plataforma específica.

Como ambiente de desenvolvimento existem duas opções, uma delas é o Xamarin Studio e a outra é o bem conhecido Visual Studio. Para utilização do Visual Studio é necessário adquirir a versão *Business* [18].

A Tecnologia Xamarin é actualmente um instrumento sério para resolver tarefas que vão desde o domínio fácil ao mais complexo no desenvolvimento de aplicações móveis. O Xamarin está em constante actualização e o suporte prestado tanto pela sua equipa de *suporters* como pela comunidade em geral são bastante positivos. O número de programadores que escolhem desenvolver com Xamarin tem aumentado a cada dia o que a recomenda como sendo uma tecnologia muito promissora.

Tendo em conta que esta foi a tecnologia escolhida para o desenvolvimento da aplicação móvel serão de seguida descritas as duas formas existentes ne abordar o desenvolvimento de aplicações multiplataforma em Xamarin: os *Shared Projects* e as *Portable Class Libraries*.

2.3.3.1 *Shared Projects*

Como o próprio nome sugere, este tipo de abordagem permite que se escreva código partilhado, ou seja na mesma aplicação podem definir-se regiões de código específicos de cada plataforma. O código é compilado como uma parte de cada projecto e pode incluir diretivas que ajudem a incorporar funcionalidades específicas de cada plataforma.

Ao contrário do que acontece na maioria dos projetos, um Shared Project não produz qualquer biblioteca executável independente (DLL), pois o código é compilado em cada um dos projetos nos quais está referenciado. Um *Shared Project* é um agrupamento de arquivos que contém código que vai ser incluído noutros projetos ao serem compilados. Os *Shared projects* não podem ser referenciados noutro tipo de projetos que não os do seu tipo.

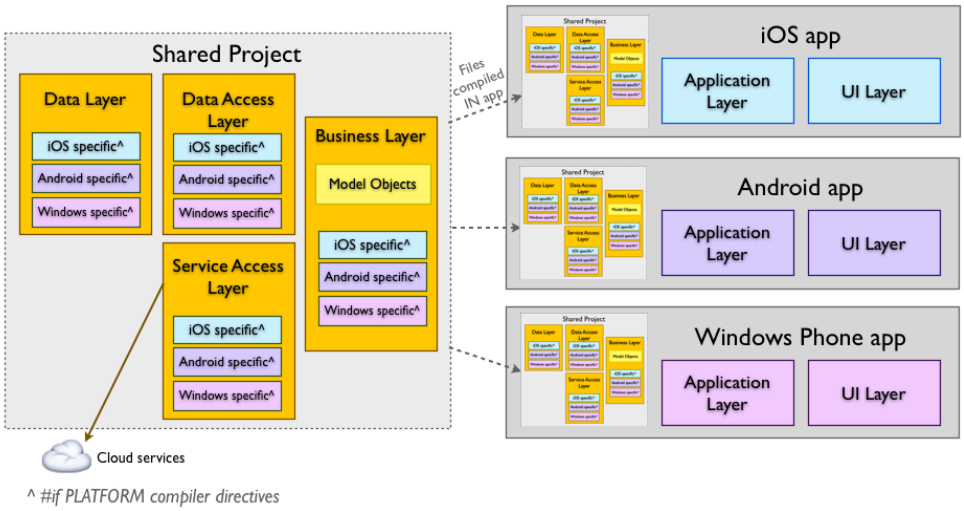


Figura 2.10: Diagrama do projeto Xamarin, *Shared Porject*

Os *Shared Projects* são versáteis e permitem facilmente adicionar características específicas de cada plataforma. Permitem também utilizar bibliotecas externas muito mais facilmente. No entanto, se a aplicação a desenvolver envolver várias regiões de código partilhadas para plataformas especificas, é necessário ter cuidado, pois o projeto pode-se tornar muito difícil de ler ou manter [17].

2.3.3.2 *Portable Class Libraries*

Num *Application Project* ou num *Library Project*, a DLL que resulta é restrito a uma plataforma específica para o qual foi criada. Isto evita que se crie um projeto para Windows Phone e que se volte a usar esse projeto para outra plataforma, Xamarin.IOS ou Xamarin.Android.

Um projecto do tipo PCL, *Portable Class Libraries*, apesar de também resultar numa DLL para uma plataforma específica, permite que se escolha uma combinação de plataformas para as quais a nossa aplicação se destina. As escolhas de compatibilidade que se faz ao criar um PCL ficam num perfil que descreve quais as plataformas e que bibliotecas suporta.

A tabela abaixo mostra algumas das características que variam de acordo com a plataforma .NET.

Tabela 2.2: Tabela de características

Característica	.NET Framework	Windows Store Apps	Silverlight	Windows Phone	Xamarin
Core	Sim	Sim	Sim	Sim	Sim
LINQ	Sim	Sim	Sim	Sim	Sim
IQueryable	Sim	Sim	Sim	7.5+	Sim
Serialization	Sim	Sim	Sim	Sim	Sim
Data Annotations	4.03+	Sim	Sim	N.A.	Sim

A coluna Xamarin reflete o facto de que Xamarin.IOS e Xamarin.Android suportam todos os perfis fornecidos com o Visual Studio 2013, e a disponibilidade de recursos em todas as bibliotecas que se criarem apenas serão limitados pelas outras plataformas que se quiser incluir, tais como Windows Phone ou Windows Store.

Apesar de todos os benefícios provenientes da criação de projetos do tipo PCL, tais como a centralização do código partilhado, tudo está concentrado no mesmo projeto. Essa característica facilita a referenciação do PCL noutros projetos. Tem também as suas desvantagens, pois comparando com os *Shared Projects*, quando é necessário incluir bibliotecas externas poderão ser encontradas algumas barreiras, apesar de grande parte destas barreiras poderem ser contornadas com o recurso a *dependency injection* [17].

O diagrama abaixo mostra a arquitetura de uma aplicação multiplataforma usando PCL.

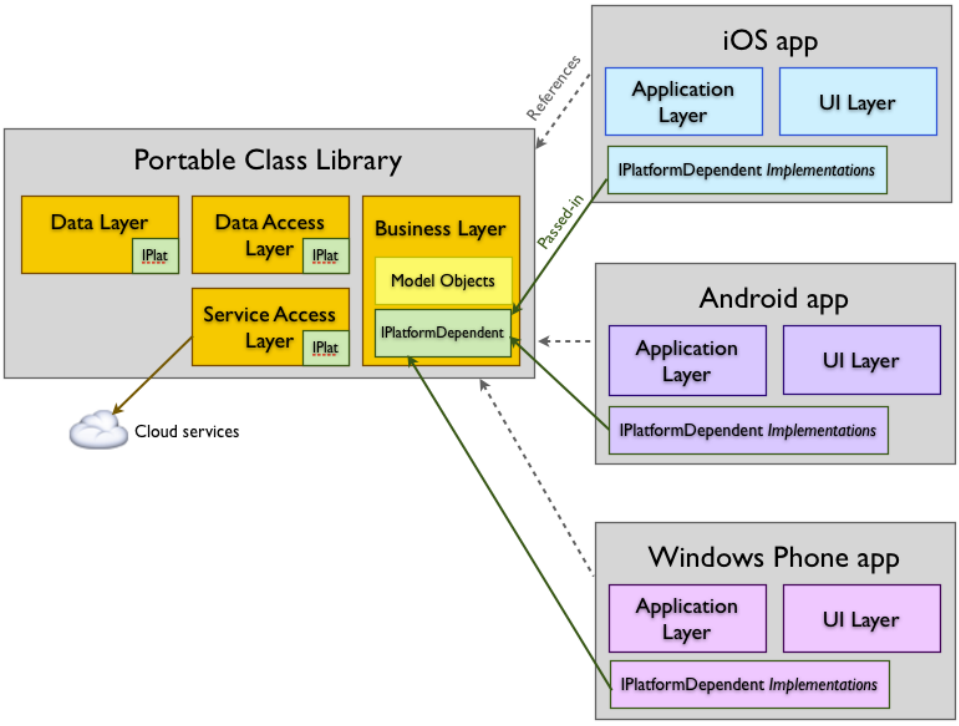


Figura 2.11: Diagrama do projeto Xamarin, *Portable Class Library*.

2.4 Internet das coisas e automação residencial

A Internet das coisas, IoT, já começa a fazer parte do presente mas prevê-se que no futuro biliões de objetos irão estar ligados através da Internet. Estes objetos, *things*, irão ter perceção do que os rodeia, comunicar entre si, bem como irão ter inteligência para atuar consoante as circunstâncias. A recolha de dados a partir destes objetos é crucial, pois permite que os sistemas de software entendam melhor o ambiente em que os objetos estão. Poderão estar muitos dispositivos

diferentes envolvidos na recolha dessas informações que serão depois processadas na Internet, possivelmente através de sistemas na *Cloud* [20].

Embora muitas vezes os conceitos de Internet das coisas e automação residencial sejam considerados idênticos, na verdade eles formam partes distintas do conceito de casas inteligentes. A automação residencial é onde os dispositivos elétricos de uma casa estão conetados a um sistema central que torna esses dispositivos autómatos baseados na acção do utilizador. Por exemplo, fechar persianas elétricas através de um botão ou ligar a iluminação da sala através de um comando de voz. A Internet das coisas é o que torna uma casa automatizada numa casa inteligente. O que torna uma casa inteligente é a combinação entre sensores, sistemas inteligentes e os objetos do dia-a-dia ligados numa rede permitindo a estes completar tarefas e comunicar entre si, sem que haja a acção do utilizador.

2.4.1 Raspberry Pi

O Raspberry Pi é um computador de baixo custo do tamanho de um cartão de crédito. O objetivo inicial era o de promover e incentivar, no setor da educação, o início à programação e conhecimento do hardware básico de um computador. No entanto hoje em dia os propósitos da utilização deste dispositivo vão muito além disso.

	Model A	Model B
SoC	Broadcom BCM2835 (CPU, GPU, DSP e SDRAM)	
CPU	700 MHz núcleo ARM1176JZF-S (família ARM11)	
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 h.264/MPEG-4 AVC high-profile decoder	
Memória (SDRAM)	256 MB (partilhado com GPU)	256 MB (partilhado com GPU)
Portas USB 2.0	1	2 (via USB hub integrado)
Saídas de vídeo	Composite RCA, HDMI	
Saídas de som	3.5 mm jack, HDMI	
Armazenamento	Slot para cartões SD, MMC, SDIO	
Rede LAN	Não disponível	Ethernet 10/100 (RJ45)
Periféricos baixo-nível	8 × GPIO, UART, I²C bus, SPI bus com selecção de 2 chips, +3.3 V, +5 V, Ground	
Corrente (potência)	500 mA (2.5 W)	700 mA (3.5 W)
Alimentação	5 volt via MicroUSB ou GPIO header	
Dimensões	85.60 × 53.98 mm (3.370 × 2.125 in)	
Sistemas operativos	Debian GNU/Linux, Fedora, Arch Linux	

Figura 2.12: Comparação Raspberry Pi Modelo A e B

A primeira versão lançada do Raspberry Pi surgiu logo com dois modelos, modelo A e B. Comparando a tabela de características dos dois modelos podemos verificar que a principal diferença é a existência de duas portas USB e Ethernet no modelo B.

O recente lançamento da nova versão, o Raspberry Pi 2, com o seu CPU ARMv7 que é uma das muitas evoluções presentes neste novo dispositivo, permite que seja possível correr as mais variadas distribuições Linux e até uma versão do novo Windows 10 que ao que tudo indica será lançado em Julho de 2015. Esta versão do Windows 10 direcionada para a Internet das coisas vem

Tabela 2.3: Tabela de características Raspberry Pi 2

Especificações Pi 2	
CPU	Cortex ARMv7
Cores	4
Clock	900MHz
GPU	Videocore IV
Mem	1GB
USB	4
Storage	microSD
Network	10/100 Ethernet
GPIO	40-pin
Wifi	No
Bluetooth	No
Flash	No

mostrar a aposta que grandes empresas como a Microsoft estão a fazer neste ramo, prevendo que o futuro da Internet depende da evolução nesta área [21].

2.5 *Web Services*

Quando se pretende interligar sistemas que contêm mais do que um subsistema independente, uma possível abordagem no que toca a comunicações entre subsistemas é a utilização de *Web Services*, pois estes foram criados e desenvolvidos de forma a suportar a interoperabilidade entre subsistemas, numa determinada rede. Num *Web Services* há sempre que ter em quanta dois agentes principais que estão envolvidos na comunicação que são o fornecedor do serviço e o consumidor desse mesmo serviço. O fornecedor do serviço é uma entidade, pessoa ou empresa, que disponibiliza um serviço através de um agente fornecedor. O consumidor de serviço é tambem um entidade que necessita de uma ou mais funcionalidades fornecidas pelo serviço e comunica com o agente fornecedor através de pedidos. Para que a ligação entre as duas partes seja possível é estabelecido um "contrato"entre o fornecedor e o consumidor, sendo este "contrato"constituído por duas partes que são a descrição do serviço e a semântica. A descrição do serviço é a definição do formato das mensagens trocadas, os tipos de dados, os protocolos de transporte e os formatos de serialização do transporte, tudo isto fica documentado numa WSD, *Web service description*. A WSD é especificada na Web WSDL, service description language [22].

2.5.1 REST

REST, acrónimo de *Representational State Transfer*, é um estilo arquitetural aplicado a componentes, conectores e elementos de dados dentro de um sistema de hipermédia distribuído. É destinado ao desenvolvimento de sistemas distribuídos dependentes de recursos, num padrão cliente-servidor. Quando se fala em REST automáticamente surge o termo RESTful, como referido a um serviço e seu conjunto de operações implementado no estilo REST e como que definindo uma API

a ser usada pelo cliente [23]. Uma API RESTful utiliza explicitamente os verbos HTTP de forma consistente com a definição do protocolo para a definição de operações CRUD sobre um recurso acedido remotamente. A relação direta entre as operações CRUD e os verbos HTTP são feitas da seguinte forma:

- *Create* com *POST*;
- *Read* com *GET*;
- *Update* com *PUT*;
- *Delete* com *DELETE*;

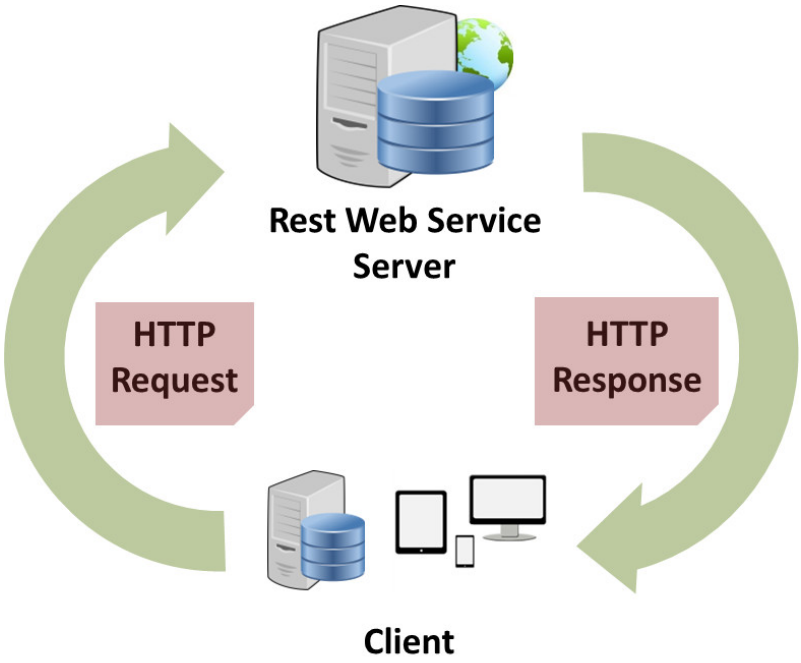


Figura 2.13: *Web Service REST*

2.5.2 SOAP

SOAP, acrónimo de *Simple Object Access Protocol*, é o protocolo utilizado por outro estilo de *web services* baseado na troca de mensagens na Linguagem de Marcação Extensível, XML. É um protocolo para troca de informações estruturadas indicado para plataformas descentralizadas e distribuídas. O protocolo de rede mais utilizado no envio de mensagens é o HTTP mas também podem ser utilizados outros protocolos tais como SMTP, FTP ou protocolos definidos por entidades [24].

2.5.3 WCF

WCF, do acrónimo *Windwos Communication Foundation* é uma *framework* desenvolvida pela Microsoft, como parte integrante da .NET framework, que permite a criação e implementação de *Web Services*. Esta solução tem como principais características e funcionalidades a interoperabilidade com outros tipos de *web services*, permitindo facilmente a extensibilidade da aplicação para a inclusão de novos tipos de clientes e contendo aspetos de segurança pois permite a encriptação das mensagens trocadas. Este tipo de serviço é composto por três elementos distintos, que são a classe do serviço, implementada em C# ou *Visual Basic* com um ou mais métodos, o servidor que aloja o processo e um ou mais *endpoints*, que são pontos de acesso ao serviço, através dos quais o serviço é acedido pelos clientes [25].

2.6 Conclusão

Atualmente, na área de automação residencial, estamos perante um clima de bastantes mudanças e de expetativas. Os pequenos fabricantes vão estando atentos ao que as grandes empresas vão anunciando por forma a estarem preparados para emergir no mercado. Estas pequenas empresas deverão optar por fornecer dispositivos compatíveis com sistemas abertos como o que a Google pretende lançar ao ter anunciado o seu novo sistema operativo e nova plataforma de comunicação, Brillo e Weaver respectivamente, dedicada à Internet das coisas. Por outro lado, prevê-se também que empresas como a Samsung ou Apple, que detêm grande parte do mercado de dispositivos móveis, lancem o seu sistema proprietário para automação residencial.

Capítulo 3

Requisitos, funcionalidades e arquitetura

Neste capitulo irão ser apresentados os atores pertencentes ao sistema, os requisitos funcionais e os não funcionais e serão também apresentados as funcionalidades do sistema através de casos de uso. Por fim irá ser feita uma análise da arquitetura adotada para o sistema.

3.1 Atores

Nesta primeira versão desenvolvida do sistema optou-se por se ter a camada de administração do sistema de forma externa. Assim o único ator do sistema é o utilizador, ou seja quem utilizar a aplicação móvel para aceder aos dispositivos associados ao sistema Mordomo.

3.2 Requisitos Funcionais

Os requisitos funcionais representam as ações que o sistema deve permitir que os utilizadores realizem. O sistema Mordomo tem então os seguintes requisitos funcionais:

- Autenticação do utilizador. Esta autenticação é verificada através das credenciais do utilizador.
- Monitorizar dispositivos. O utilizador deve ser capaz de ter informações acerca do estado dos dispositivos.
- Controlar dispositivos. O utilizador deve ser capaz de controlar o estado dos dispositivos.
- Programar atividades. O utilizador deve ser capaz de especificar tarefas, de forma sistemática ou esporádica, que constam de atividades a ocorrer no sistema.
- Acesso a histórico. O utilizador deve ser capaz de verificar todos e quaisquer eventos que ocorram no dispositivo.

3.3 Requisitos não funcionais

Os requisitos não funcionais são os requisitos indispensáveis ao funcionamento do sistema em geral. Podem então ser distinguidos dois grupos de requisitos, sendo eles os requisitos de qualidade e os requisitos de interface.

- Comunicações. O sistema deve assegurar a integridade nas mensagens trocadas entre subsistemas.
- Usabilidade. O sistema deve ser *user friendly*, ou seja simples e de fácil utilização de forma a atenuar o tempo de aprendizagem dos utilizadores.
- Eficiência. O sistema deve ser rápido e fluído tanto no que diz respeito à navegação na interface dos dispositivos móveis, bem como no processamento dos pedidos a *web services* e na resposta de atuação no que toca ao controlo dos dispositivos associados ao Mordomo.
- Segurança. O sistema deve assegurar que apenas utilizadores autenticados possam ter acesso ao sistema. Para garantir uma maior segurança o processo de autenticação deve estar blindado com um sistema de encriptação. As configurações de portas na *firewall* para o acesso ao *web service* devem estar configuradas de forma a que apenas os consumidores confiáveis do sistema possam realizar pedidos.
- Modo degradado. O sistema deve assegurar o seu funcionamento, mesmo que limitado em algumas funcionalidades. O funcionamento do sistema no modo degradado será explicado no capítulo referente à arquitetura do sistema.

Tabela 3.1: Tabela resumo dos atores e requisitos.

Atores	**Requisitos Funcionais**	**Requisitos Não Funcionais**
Utilizador	**Autenticação do utilizador**	**Comunicações**
	Monitorizar dispositivos	**Usabilidade**
	Controlar dispositivos	**Eficiência**
	Programar atividades	**Segurança**
	Acesso a histórico	**Modo degradado**

3.4 Casos de Uso

As funcionalidades disponíveis ao utilizador para interagir com o sistema apresentam-se através dos casos de uso da solução. Com recurso a narrativas de texto e com o apelo a ilustrações fornece-se uma noção mais elucidativa das funcionalidades e interação do utilizador com o sistema. Apesar do sistema não se limitar à aplicação móvel, esta é a única parte de interesse do ponto de vista do utilizador. Portanto apenas serão descritos os casos de uso do ponto de vista do utilizador.

3.4.1 Autenticação

A figura 3.1 ilustra o modelo de caso de uso no que diz respeito à autenticação.

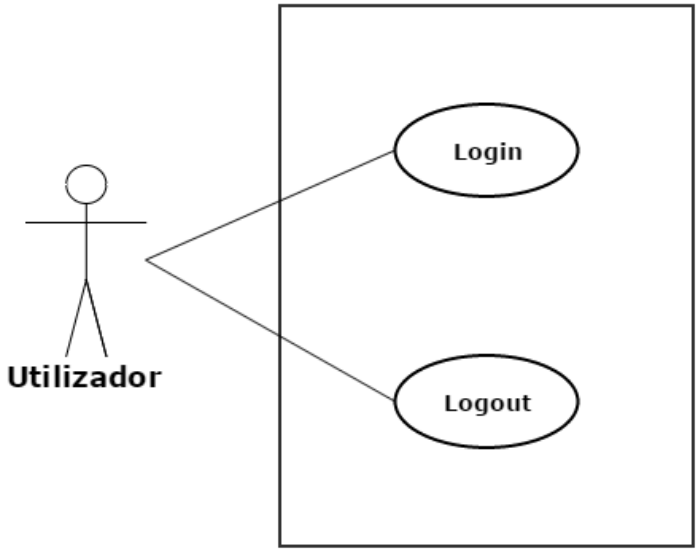


Figura 3.1: Modelo de caso de uso da autenticação

- CU01 - Login. O utilizador deve ser capaz de se autenticar no sistema recorrendo para isso às suas credenciais, nome de utilizador e senha.
- CU02 - Logout. Após o utilizador realizar a autenticação com sucesso este deve ser capaz , em qualquer momento, de dar por terminada a sua sessão, ou seja fazer *logout*.

3.4.2 Utilização

A figura 3.2 ilustra o modelo de caso de uso no que diz respeito à utilização do sistema pelo utilizador, mostrando as operações realizáveis por este.

- CU03 - Ver eventos ocorridos. O utilizador deve ser capaz de verificar todos e quaisquer eventos que tenham acontecido no sistema, tenham estes eventos acontecido por ação do utilizador através da aplicação móvel, por ação manual nos dispositivos ou por ter sido realizada uma ação pelo próprio sistema.
- CU04 - Ver Próximas atividades. O utilizador deve ser capaz de verificar quais as próximas atividades a realizar pelo sistema e a data/hora da sua realização.
- CU05 - Programar próximas atividades. O utilizador deve ser capaz de inserir e remover atividades no sistema. O utilizador deve ser capaz de programar atividades para que estas se

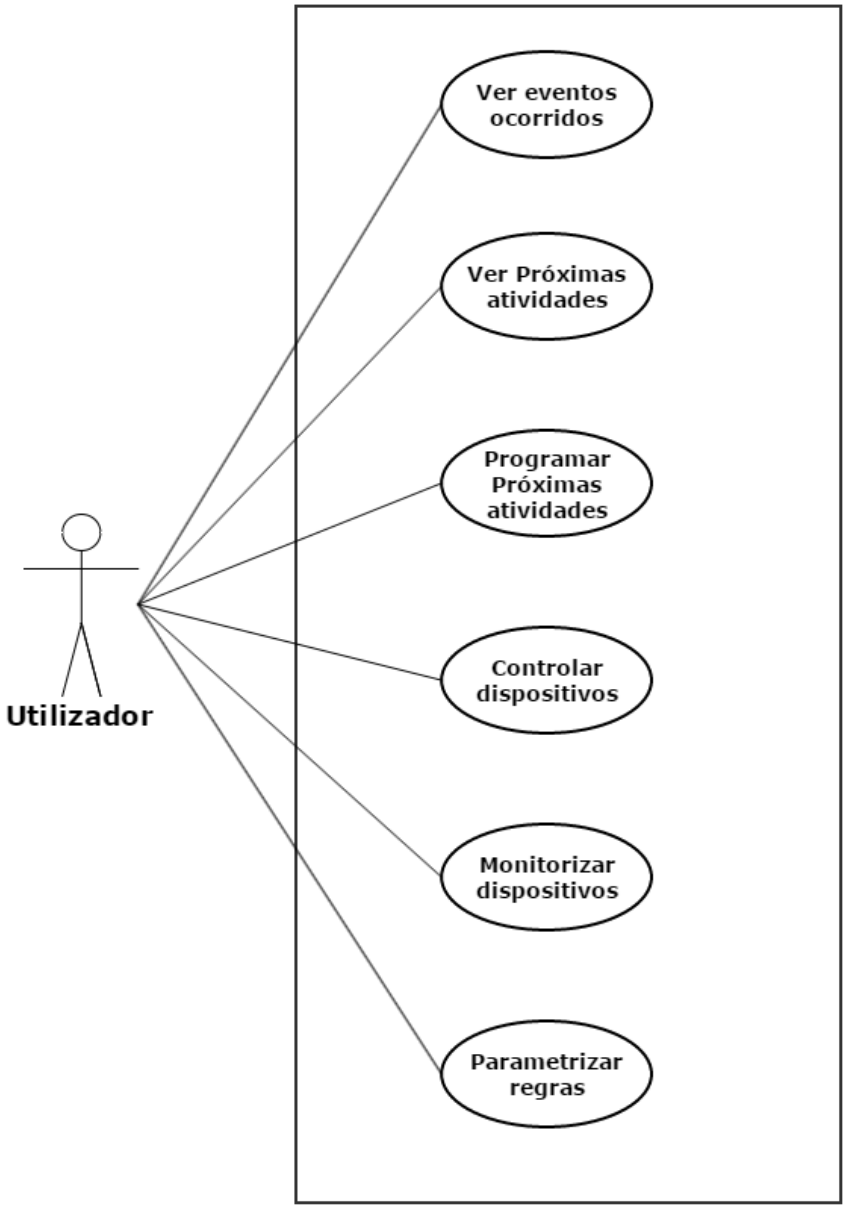


Figura 3.2: Modelo de caso de uso da autenticação

realizem a uma certa hora sistemáticamente, apenas aos fins de semana, durante a semana ou escolher uma hora para a atividade ocorrer no próprio dia.

- CU06 - Controlar dispositivos. O utilizador deve ser capaz de escolher um dispositivo e controlá-lo. Dependendo do dispositivo o utilizador poderá simplesmente ligar ou desligar e em alguns casos atribuir um ou mais valores ao dispositivo.

- CU07 - Monitorizar dispositivos. O utilizador deve ser capaz de verificar qual o estado atual do dispositivo.

- CU08 - Parametrizar regras. O utilizador deve ser capaz de definir valores limite máximo e mínimo e limiares de atuação, para certos dispositivos.

3.5 Arquitetura do sistema

Como já se referiu o projeto Mordomo tem como objetivo o desenvolvimento de uma aplicação multiplataforma para dispositivos móveis que permita aos utilizadores monitorizar e controlar outros dispositivos remotamente. Para tal foi pensado um sistema cuja arquitetura é ilustrada na figura 3.3. Este sistema está dividido em três módulos: a aplicação multiplataforma móvel, a base de dados e camada de acesso a hardware.

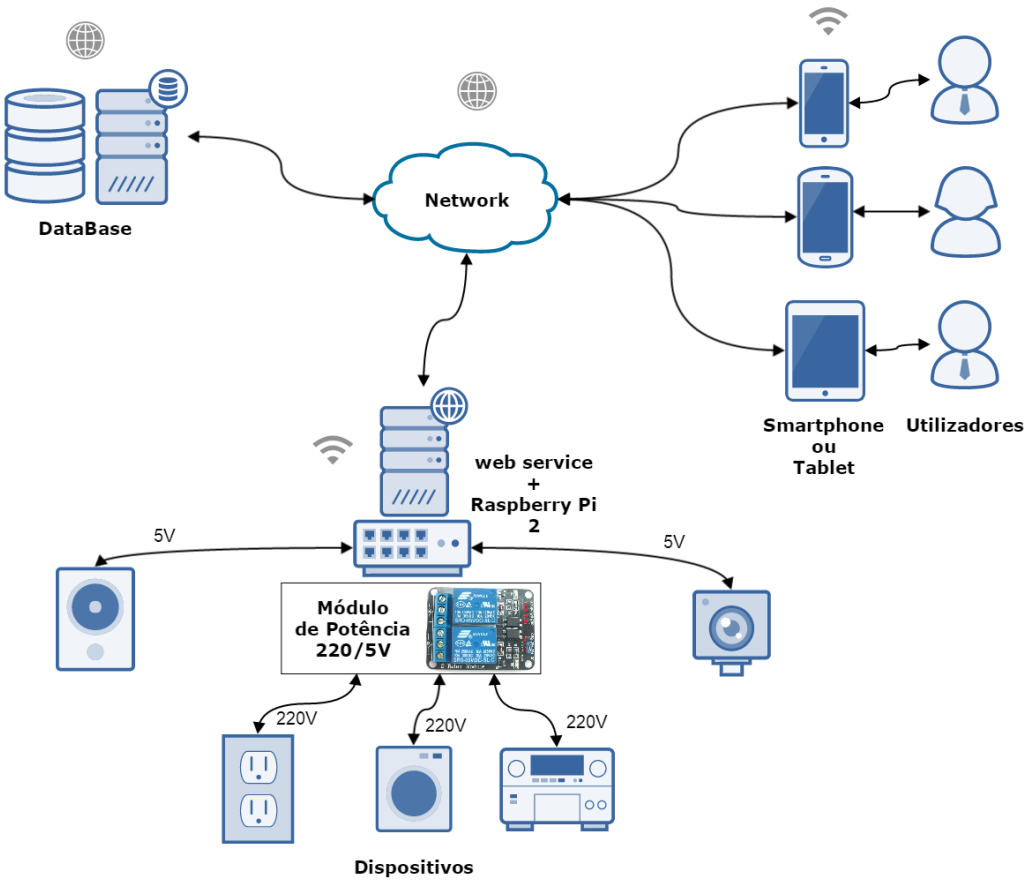


Figura 3.3: Arquitetura Mordomo

A aplicação multiplataforma, interface do sistema, é responsável por gerar todos os ecrãs visíveis ao utilizador num dispositivo móvel, permitindo a parametrização de alguns aspetos ou regras, o agendamento de atividades, a visualização de todo o histórico de atividades, alertas ou alarmes e ainda que o utilizador controle todos os dispositivos presentes no sistema.

No que diz respeito ao acesso de hardware, por forma a tornar este acesso possível recorreu-se a um mini-computador, neste caso um Raspberry Pi 2. Neste nível do sistema foram desenvolvidas duas aplicações, ambas a correr no Raspberry Pi. Uma delas é responsável pela aplicação de regras para determinar alertas, atuação nas saídas e aquisição do sinal nas entradas dos dispositivos ligados ao Raspberry Pi. Por forma a tirar partido das potencialidades do Raspberry Pi e consequentemente poupando na utilização de outros recursos, o que resulta numa diminuição de custos do produto a desenvolver, o Raspberry Pi foi configurado de forma a poder ser utilizado como um *web server* onde está a correr a outra aplicação que neste caso é uma aplicação Web contendo um *web service*. Neste *web service* é onde são realizadas todas as operações relacionadas com o acesso e escrita na base de dados e onde também é possível ter acesso direto às portas GPIO do Raspberry Pi onde estão ligados os dispositivos do sistema. Esta funcionalidade é essencial para o funcionamento do sistema em modo degradado que será descrito mais à frente.

A base de dados armazena todas as informações referentes ao sistema. Foi desenhada uma base de dados relacional que contém desde os dispositivos e informações referentes a estes até ao registo de todo o histórico de actividades do sistema. Foi também implementado no sistema o acesso a uma base de dados de documentos, LDAP, onde é possível armazenar imagens, que neste contexto tem como ponto de interesse o armazenamento de imagens provenientes de uma câmara de vigilância, por exemplo. A escolha de uma base de dados LDAP foi influenciada por esta ser utilizada noutros projetos da empresa onde foi realizado este projeto.

3.5.1 Aplicação Móvel - Interface com o utilizador

3.5.1.1 Interface

A interface é responsável por permitir ao utilizador interagir com o sistema e usar as suas funcionalidades. A interface foi desenvolvida para dispositivos móveis, sendo compatível com Android, iOS e Windwos Phone. Para que a mesma aplicação móvel fosse compatível com os sistemas operativos móveis referidos, recorreu-se à plataforma de desenvolvimento Xamarin. As interfaces foram criadas com recurso à API Xamarin.Forms que permite criar interfaces multiplataforma. Esta API está dividida em três categorias, *Pages*, *Layouts* e *Controls*. As *Pages* são as páginas ou ecrãs propriamente ditos tais como *TabbedPage* ou *ContentPage*. Os *Layouts*, tais como *StackLayout*, representam as possibilidades de representação dos mais variados componentes na página. Os *Controls*, tais como *Entry* ou *Button*, são os componentes com os quais o utilizador vai interagir.

Na figura 3.4 é possível ver os elementos que estão presentes na API. No entanto o Xamarin.Forms não se restringe apenas aos elementos ilustrados, pois embora o Xamarin.Forms encapsule as APIs nativas de cada plataforma, é ainda assim possível ter acesso às APIs nativas caso seja necessária alguma implementação específica para cada plataforma, como por exemplo, as APIs de voz e o acesso a ficheiros. Essas implementações específicas são feitas de duas formas: usando *Custom Renderers* e usando *DependencyService* [19].

Figura 3.4: Elementos gráficos Xamarin.Froms

A primeira vez que a aplicação for iniciada por qualquer utilizador, verifica-se que a configuração para aceder à sua base de dados ainda não foi efetuada. Essa configuração deve então ser feita na interface de opções, gravando-se de seguida num ficheiro no próprio dispositivo móvel de modo a que que em próximos acessos já se possa utilizá-la sem mais qualquer operação. A aplicação está munida de um sistema de autenticação com codificação MD5, que é um algoritmo de *hash* de 128 bits unidireccional [26]. Após a autenticação é desenhado o menu principal de onde estão acessíveis todas as interfaces.

A figura 3.5 ilustra a forma de como as interfaces principais da camada de apresentação do Mordomo estão dispostas.

3.5.1.2 Notificações

Visto tratar-se de um sistema de controlo e monitorização de dispositivos é indispensável a existência de um sistema de notificações na aplicação móvel que interage diretamente com o utili-

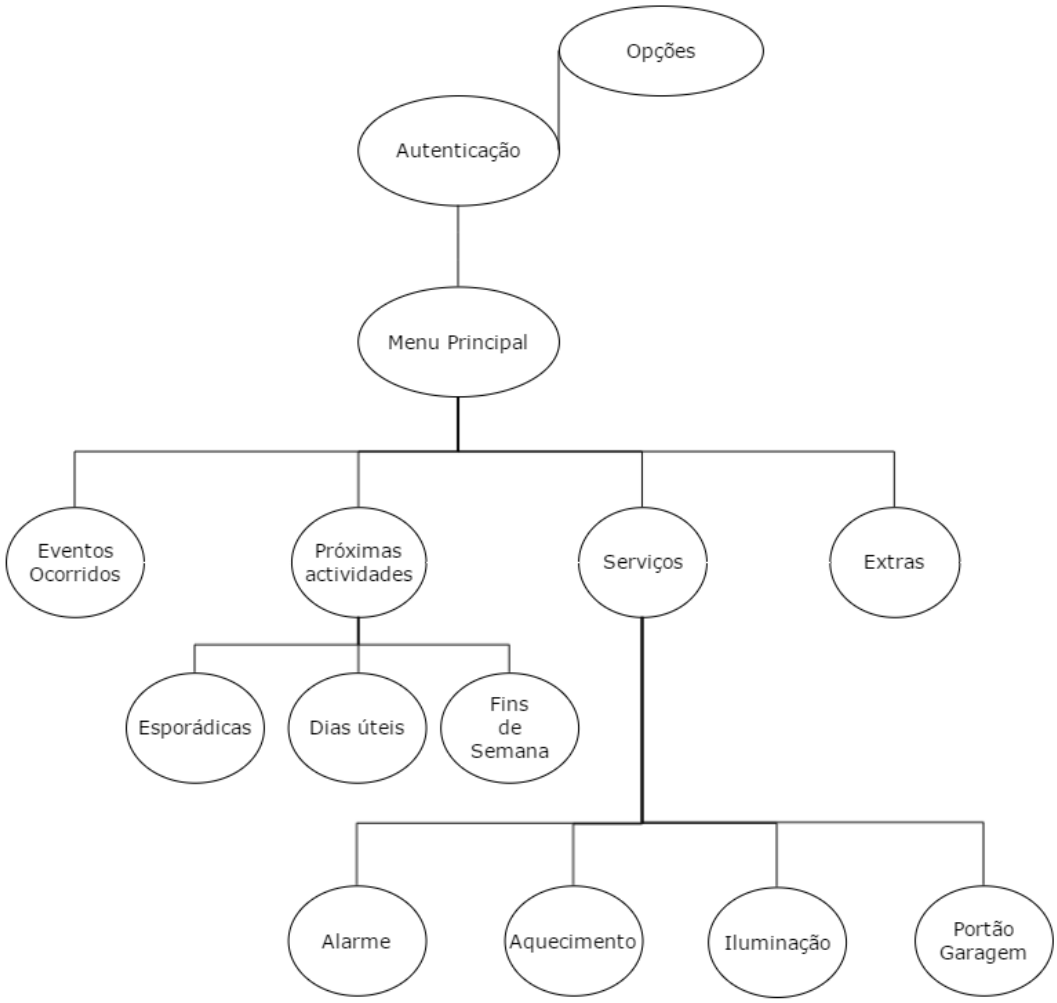


Figura 3.5: Diagrama da interface do Mordomo

zador. Sendo esta aplicação desenvolvida para correr em várias plataformas, optou-se por utilizar um plugin de notificações para Xamarin.Forms. Por forma a diferenciar os vários tipos de notificações optou-se por se fazer a divisão das notificações por grupos em que a cada grupo está associada uma cor sugestiva ao grau de segurança do sistema, sendo estes os seguintes:

- ***Info***, simples informação ao utilizador representada pela cor azul.
- ***Success***, notificação que ocorre em caso de sucesso de uma ação, representada pela cor verde.
- ***Warning***, notificação que ocorre em caso de alerta, representada pela cor amarela.
- ***Emergency***, notificação que ocorre em caso de emergência, representada pela cor vermelha.

3.5.2 Aplicação Raspberry Pi

No sistema desenvolvido, por forma a minimizar a utilização de recursos e a potenciar as mais valias do Raspberry Pi, estão duas aplicações independentes a correr em simultâneo. Uma delas é o *web service*, que será descrito posteriormente, e a outra é a aplicação responsável pela atuação nos dispositivos a controlar. Embora o grande objectivo seja o desenvolvimento de uma aplicação para dispositivos móveis capaz de controlar e monitorizar dispositivos de uma casa inteligente, sem esta aplicação seria impossível validar o conceito proposto para esta dissertação. Foi escolhido então o Raspberry Pi devido às características deste produto ,já mencionadas anteriormente, que se adequam a este propósito e que permitiram realizar com sucesso a prova deste conceito.

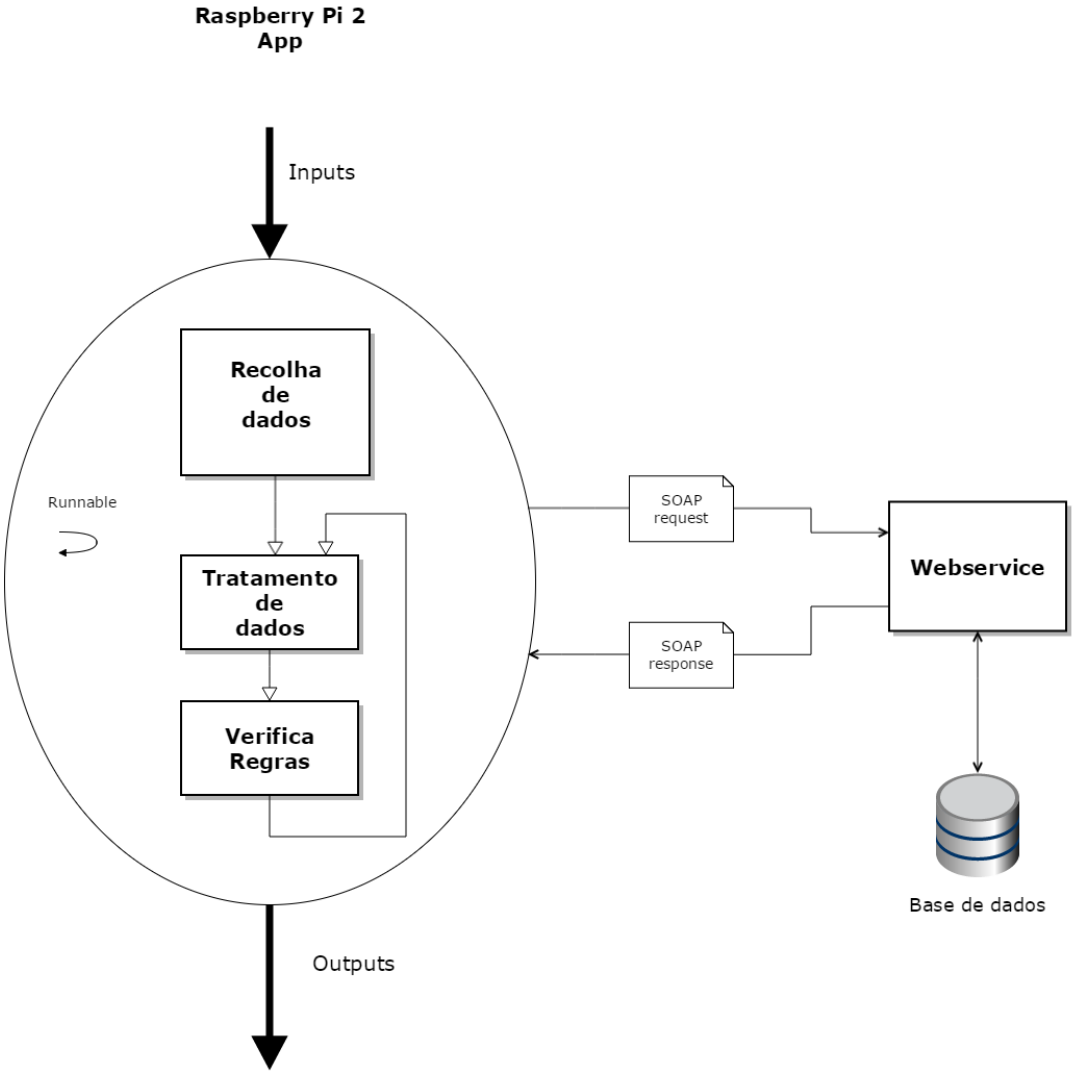


Figura 3.6: Diagrama da Aplicação do Raspberry Pi

Para que o Raspberry Pi fosse capaz de responder a um pedido vindo de um dispositivo móvel foi pensada a arquitetura anteriormente descrita no capítulo. Esta arquitetura que influenciou

a abordagem da aplicação a correr no Raspberry Pi, sendo que esta desempenha três funções principais ilustradas na figura 3.6: recolha de dados, tratamento de dados e verificação de regras.

A recolha de dados está constantemente à espera de dados vindos tanto das entradas do Raspberry Pi como também está à espera de dados vindos do dispositivo móvel. Esta recolha de dados é feita com recurso a *threads* e a constantes pedidos de leitura da base de dados através do consumo do *Web service*.

O tratamento de dados é a camada responsável por identificar se os dados recolhidos são entradas, ou seja sinais dos dispositivos ligados ao Raspberry Pi, ou pedidos de ações do utilizador. Dados estes que após serem analisados pela camada de verificação de regras serão registados na base de dados, através do consumo do *Web service*.

A verificação de regras, tal como o próprio nome sugere, verifica se os novos valores estão dentro dos parâmetros estipulados, pelo utilizador ou pelo sistema. Caso algum valor não cumpra estes limites podem ocorrer duas situações: gerar apenas um alerta e registar o mesmo no histórico de atividades ou para além disso também automaticamente, sem interferência do utilizador, fazer atuar algo no sistema que contrarie a situação de alerta por forma a normalizar o sistema.

3.5.3 Base de dados

A base de dados tem como propósito armazenar todas as informações do sistema, desde informação sobre os utilizadores, dispositivos, divisões da casa inteligente, atividades e registo de todos os eventos. O modelo de dados construído baseia-se em três elementos representados na figura 3.7.

3.5.3.1 Utilizadores

Na base de dados os utilizadores são constituídos por um identificador único, ID, por um nome de utilizador também ele único, por uma senha e um nível de acesso. O nome e a senha são os elementos que identificam o utilizador perante a aplicação. O nível de acesso de cada utilizador vai determinar as suas permissões no uso das funções disponíveis na aplicação através da comparação do nível de acesso do utilizador com o nível de acesso mínimo dos dispositivos ou atividades com as quais pretende operar.

A aplicação por razões de segurança utiliza codificação MD5 tanto no nome de utilizador como na senha, portanto os valores destes dois elementos estão apresentados na base de dados com o seu valor na forma de um *hash*.

3.5.3.2 Dispositivos

No que diz respeito a dispositivos, estes são caracterizados por um identificador único, nome, tipo, divisão, valor, valor máximo, valor mínimo e por fim nível de acesso. A cada dispositivo é atribuído um ID único pois podem existir dispositivos com o mesmo nome, sendo necessário conseguir distinguir os vários dispositivos do sistema. Quanto ao tipo, este atributo refere-se ao facto de poderem existir dois tipos de dispositivos: digitais ou analógicos. Cada dispositivo

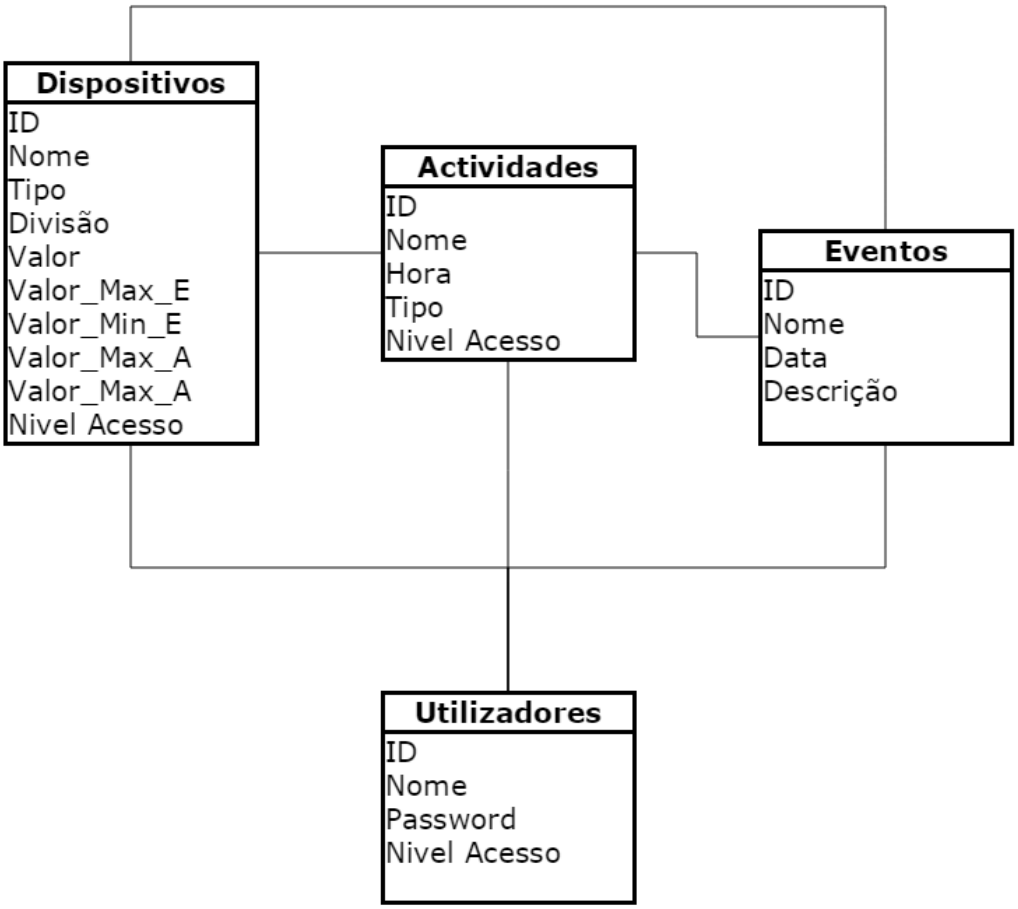


Figura 3.7: Modelo da base de dados

tem associada uma divisão de uma habitação que é onde o dispositivo se encontra instalado. O valor refere-se ao valor atual que o dispositivo possui, o 'Valor Max E' e o 'Valor Min E' são os parâmetros definidos pelo utilizador ou seja os valores máximos e mínimos estipulados. Os 'Valor Max A' e 'Valor Min A' são os valores máximos e mínimos já predefinidos pelo sistema que determinam os limites máximo e mínimo admissíveis que o valor atual do dispositivo pode atingir. O nível de acesso de cada dispositivo é o parâmetro que vai ser comparado com o nível de acesso do utilizador por forma a determinar se esse utilizador tem permissões para aceder a esse dispositivo.

3.5.3.3 Atividades

Uma atividade tem associada a ela um dispositivo que tem como atributo uma divisão. Uma atividade representa uma ação que o utilizador queira agendar para uma determinada hora. As

atividades estão divididas em três tipos: esporádicas, dias da semana e fins-de-semana. Nem todos os utilizadores podem programar atividades sendo então necessário incluir como atributo das atividades um nível de acesso que servirá de comparação com o dos utilizadores por forma a verificar a permissão dos utilizadores em programar atividades.

3.5.3.4 Eventos

Os eventos representam todo o histórico do sistema. Cada entrada no *log* de eventos tem um ID único, o nome do evento ocorrido, a data à qual ocorreu o evento e uma descrição do acontecimento. Toda e qualquer alteração que ocorra no sistema irá ser registada, desde uma simples acção de ligar uma luz até ao alarme que dispara por ter existido a quebra de uma regra, como a de a temperatura estar superior ao limite máximo.

3.5.4 *web service*

Esta componente é também indispensável ao funcionamento ideal deste sistema pois todos os dados relevantes para o funcionamento do sistema estão alojados numa base de dados. Este acesso à base de dados é feito através do *web service* pois do ponto de vista do utilizador é importante que este acesso seja completamente transparente e eficiente, como se os dados estivessem disponíveis no próprio dispositivo. Do ponto de vista do desenvolvimento é importante que o acesso aos serviços seja feito de forma simples e direta, com o mínimo de código possível, ou seja, sem a necessidade de tratamento de baixo nível na chamada e receção da resposta do servidor. Como a base de dados utilizada é uma base de dados PostgreSQL e o Web *Service*, SOAP, foi desenvolvido em C#, o recurso à biblioteca NPGSQL foi indispensável para realizar as ligações e as operações necessárias. Esta biblioteca é um provedor de dados .NET para PostgreSQL que permite que qualquer aplicação desenvolvido sobre a plataforma .NET consiga ter acesso a essa base de dados.

Foi adotada uma metodologia de desenvolvimento do *web service* a três camadas, que são a camada de interface do serviço, camada lógica e a camada de acesso a dados tal como está ilustrado na figura 3.8.

A interface do serviço, *service interface*, implementa o "contrato"entre o consumidor do serviço e o fornecedor. Este "contrato"permite que eles troquem informação mesmo que estejam em sistemas diferentes. A interface do serviço é responsável por toda a implementação dos detalhes necessários ao funcionamento da comunicação [22]. Alguns dos aspetos por que é responsável são:

- ***Network protocol***. Encapsulamento de todos os aspetos relacionados com o protocolo de rede usado nas comunicações entre o consumidor e o fornecedor do serviço. Por exemplo, supondo que um serviço é exposto aos consumidores através de HTTP por uma rede TCP/IP, pode-se implementar a interface do serviço como um componente ASP.NET publicado num URL conhecido. O componente ASP.NET recebe o pedido HTTP, extrai a informação necessária pelo serviço para processar o pedido, invoca a implementação do serviço, empacota a resposta do serviço e envia a resposta de volta para o consumidor como uma

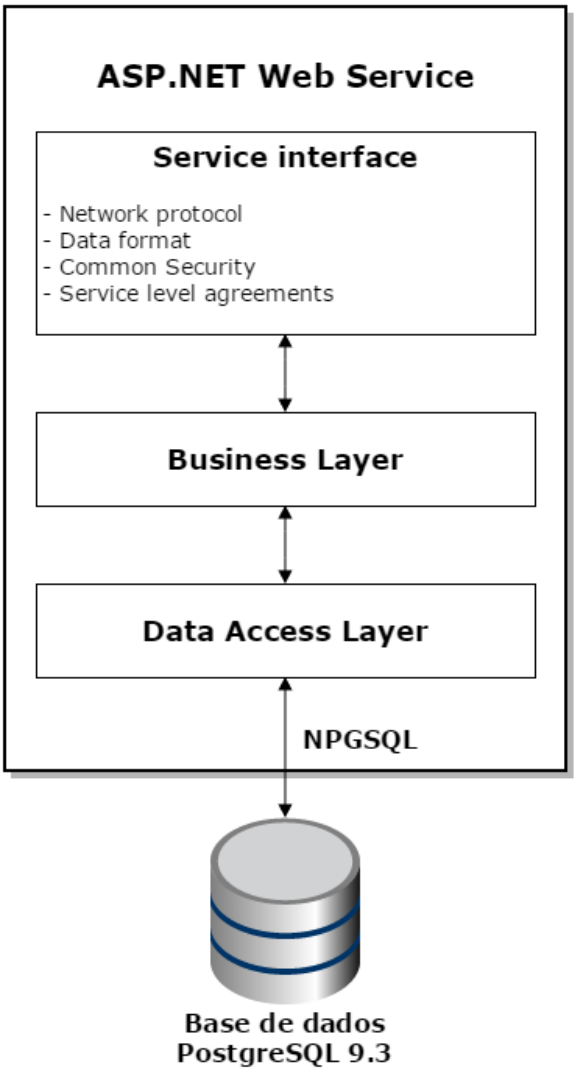


Figura 3.8: Diagrama de composição do *web service*

resposta HTTP. Do ponto de vista do serviço, o único componente do serviço em contacto com HTTP é a interface o serviço. A implementação do serviço tem a sua própria forma de comunicar com a interface do serviço e não deve ter nenhuma dependência em relação às tecnologias usadas pelos consumidores para comunicar com a interface do serviço [22].

- ***Data formats***. A interface de serviço faz a ligação entre os formatos de dados de consumo e os formatos de dados que o serviço recebe. Por exemplo, consumidores externos podem enviar um pedido e esperar uma resposta com dados no formato XML que está de acordo com um esquema XML padrão. A interface de serviço é responsável por transformar e mapear ambos os formatos de dados num formato que o serviço possa usar. A implementação do

serviço é independente no que diz respeito aos formatos específicos usados entre a interface de serviço e o consumidor [22].

- ***Security***. Cada consumidor do serviço pode ter diferentes requisitos de segurança, portanto é a camada de interface do serviço que fica responsável por implementar os requisitos específicos do consumidor. No caso do sistema desenvolvido, como o *Web Service* faz parte do próprio sistema os consumidores dos subsistemas integrantes deste sistema são implicitamente confiáveis [22].

- ***Service level agreements***. A interface de serviço tem um papel preponderante no sentido de garantir que o serviço atenda aos compromissos para um conjunto específico de consumidores. Por exemplo, que estas interfaces de serviço podem aumentar o tempo de resposta e reduzir o consumo de largura de banda recorrendo a memória cache [22].

3.5.5 Modo degradado

Como referido anteriormente, o sistema é composto por três componentes principais, aplicação móvel, *web service* e aplicação no Raspberry Pi. É um requisito o sistema ser capaz de manter o seu funcionamento mesmo em caso de falha de algum desses componentes. Existem então quatro formas de funcionamento do sistema em modo degradado, que permitem ainda o seu funcionamento mesmo que de forma limitada. Na figura 3.9 estão representadas as possibilidades de funcionamento do sistema em modo degradado.

1. **Raspberry Pi para aplicação móvel**. Caso o sistema esteja debilitado no acesso à base de dados este deve ser capaz de realizar operações como o envio de alertas para a aplicação móvel, por forma a que o utilizador seja notificado de acontecimentos importantes que ocorram no sistema.

2. **Aplicação móvel para base de dados.** Caso haja uma falha no Raspberry Pi o sistema fica sem ter acesso tanto à aplicação do Raspberry Pi como ao *web service*, pois ambas as aplicações correm no Raspberry Pi. Neste caso o acesso é feito diretamente através da aplicação móvel de forma a garantir um serviço mínimo de consulta em termos de eventos ocorridos e outras informações presentes na base de dados. Esta opção de utilizar o Raspberry Pi como *host* do *web service* foi apenas para fins de aproveitamento de recursos. Idealmente o *web service* deveria estar alojado num elemento externo ao sistema.

3. **Raspberry Pi para base de dados**. Caso exista uma falha na aplicação móvel o sistema continua o seu funcionamento normal, e quando a aplicação móvel voltar a estar disponível o utilizador irá ter acesso aos eventos ocorridos durante o tempo em que o sistema esteve a correr em modo degradado.

4. **Aplicação móvel para Raspberry Pi**. Tal como no caso dois, em que há uma falha na base de dados, o sistema, agora visto do lado da aplicação móvel, permite que o utilizador

continue a ter acesso aos dispositivos a ele associados. Para que não sejam perdidos dados quando o sistema voltar ao funcionamento ideal, o Raspberry Pi deveria ter uma mini base de dados que permitisse o registo dos eventos ocorridos temporariamente por forma a ser possível atualizar a base de dados com os eventos ocorridos durante o tempo de funcionamento em modo degradado. Esta última funcionalidade, da mini base de dados, não foi implementada.

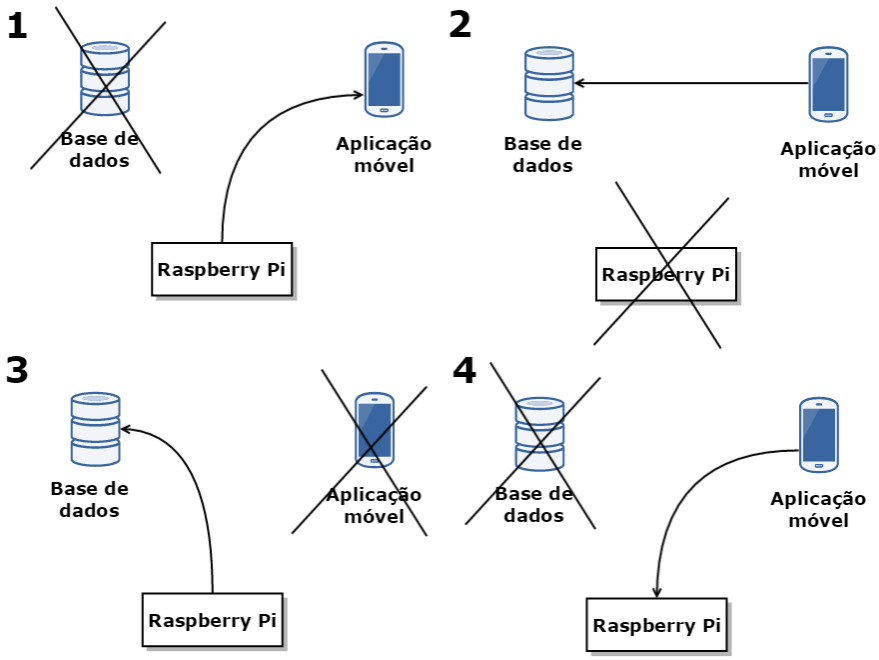


Figura 3.9: Possibilidades de funcionamento do sistema em modo degradado.

O funcionamento em modo degradado é um aspeto bastante importante quando se trata de uma abordagem de sistemas constituídos por vários componentes que funcionam de forma independente. Qualquer sistema está sujeito a falhas e quando esta abordagem é feita de forma eficaz podem minimizar-se os problemas que delas resultam, tornando a experiência do utilizador perante o sistema muito mais agradável, o que resulta numa maior satisfação.

Capítulo 4

Implementação

Neste capítulo é feita uma análise das tecnologias utilizadas e posteriormente é feita uma apresentação detalhada da implementação dos três módulos principais que constituem o sistema, cuja arquitetura foi apresentada anteriormente. No que toca à implementação, primeiramente é feita uma abordagem da interface desenvolvida para dispositivos móveis incluindo uma descrição dos ecrãs desenvolvidos. Depois são descritas as implementações do *web service* e da aplicação da camada de acesso aos dispositivos ligados ao Raspberry Pi.

4.1 Tecnologias implementadas

O grande objetivo deste projeto era a implementação de uma aplicação móvel multiplataforma, como tal foi necessário escolher tecnologias e ferramentas que permitissem realizar o desenvolvimento deste tipo de aplicações. Anteriormente foram abordadas várias tecnologias existentes dedicadas para este tipo de desenvolvimento, de entre as quais a escolhida foi o Xamarin. Esta escolha foi feita tendo em conta todas as qualidades da tecnologia já referidas tendo como ponto determinante ser uma tecnologia que tem o .NET como base do seu desenvolvimento o que permite a reutilização de código já desenvolvido pela empresa onde foi realizada a dissertação, CimSoft - Tecnologias de Informação. A utilização de C# foi de facto o ponto que mais influenciou na escolha das tecnologias de todo o sistema, visto que não só a aplicação móvel mas também o *web service* e a camada de acesso a dispositivos no Raspberry foram desenvolvidos com recurso a C#. A versatilidade desta linguagem de programação permitiu que todo o sistema fosse desenvolvido recorrendo apenas a uma linguagem o que permitiu a reutilização de código mesmo entre os módulos que formam o sistema desenvolvido. No que diz respeito ao *web service* o estilo escolhido foi o SOAP. Tendo em conta que qualquer uma das tecnologias relacionadas com *web services*, abordadas no capítulo da revisão bibliográfica, poderiam ser utilizadas e seriam igualmente bem sucedidas neste contexto, escolheu-se no entanto o SOAP por permitir facilmente a utilização de outros serviços associados, como as comunicações seguras e protocolos fiáveis. No entanto, como já foi referido, a escolha da linguagem C# no desenvolvimento do *web service* deveu-se à possibilidade de haver reutilização de código, tal como aconteceu por exemplo no acesso à base de

dados em que foi reutilizado algum código já desenvolvido noutros projetos pela empresa. Por forma a tirar partido das potencialidades do Raspberry Pi e também tendo em conta a poupança de recursos, sabendo ao mesmo tempo que o sistema operativo instalado no Raspberry Pi foi o Rasbpian, uma distribuição Linux, realizou-se uma configuração para que este fosse capaz de alojar o *web service*. Para que tal fosse possível recorreu-se à tecnologia XSP que é um servidor simples, independente, escrito em C# e que suporta a tecnologia ASP.NET em distribuições linux.

Na aplicação desenvolvida para o Raspberry Pi, apesar da linguagem mais utilizada no desenvolvimento de aplicações para este dispositivo ser Python, com recurso a bibliotecas, tais como *Raspberry.IO.GeneralPurpose*, foi possível obter os resultados pretendidos sem que fosse necessário recorrer a outra linguagem de programação que não C#. A base de dados utilizada foi PostgreSQL versão 9.3. E esta escolha foi determinada pela já utilização da mesma noutros projetos da empresa.

4.2 Interface da aplicação móvel

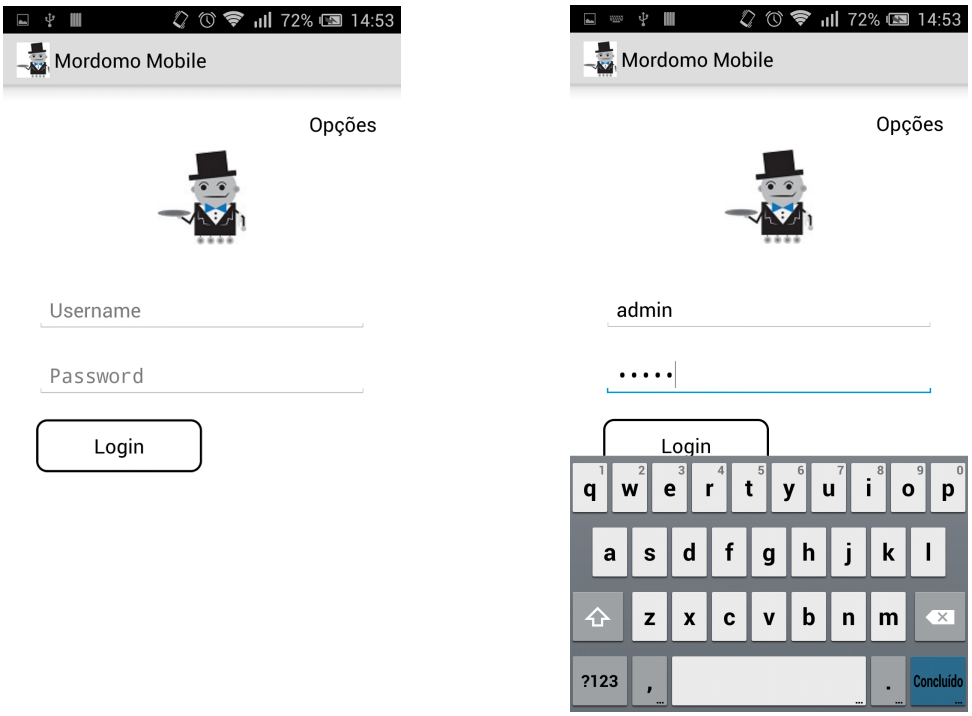
O objetivo principal do desenvolvimento da aplicação móvel é a criação de uma interface de acesso remoto para controlar e monitorizar um sistema domótico. O utilizador, com recurso a um dispositivo móvel Android, iOS ou Windows Phone, pode aceder a qualquer momento a partir do seu dispositivo móvel com acesso à internet.

Por forma a facilitar e a tornar mais agradável a experiência do utilizador na utilização da aplicação, foi pensado um desenho das interfaces que fosse caracterizado por uma apresentação leve, limpa e funcional. Na demonstração das funcionalidades da aplicação móvel, serão apresentados os ecrãs, a forma de navegação e de que forma o utilizador pode interagir com o sistema a partir do seu *smartphone*.

Todos os componentes utilizados no desenvolvimento das interfaces da aplicação móvel foram implementados com recurso apenas a Xamarin.Forms, executando as notificações que foram desenvolvidas com recurso a um *plugin* desenvolvido para ser usado com Xamarin.Forms.

4.2.1 Autenticação

Nesta interface o objetivo é o utilizador poder realizar a autenticação no sistema. Como foi dito anteriormente, caso a aplicação esteja em funcionamento no modo degradado, em que o acesso ao *web service* não se encontre disponível, a aplicação móvel deve ser capaz de aceder à base de dados diretamente do dispositivo móvel. Para que tal seja possível basta o utilizador aceder ao ecrã de opções presente no ecrã de autenticação. A figura 4.1 representa o ecrã de autenticação. Neste ecrã estão presentes apenas dois botões, dois campos de texto e o logótipo do mordomo. O campo destinado à senha este contém uma propriedade própria para campos desse tipo em que cada caráter escrito é substituído por um símbolo, não o mostrando diretamente . O botão de opções está estrategicamente localizado na interface de forma a ser visto como um botão secundário em relação ao botão de *Login*. Foi pensado desta forma pois o botão de Opções, tendo em conta uma utilização normal da aplicação, será de menor importância do ponto de vista do utilizador.



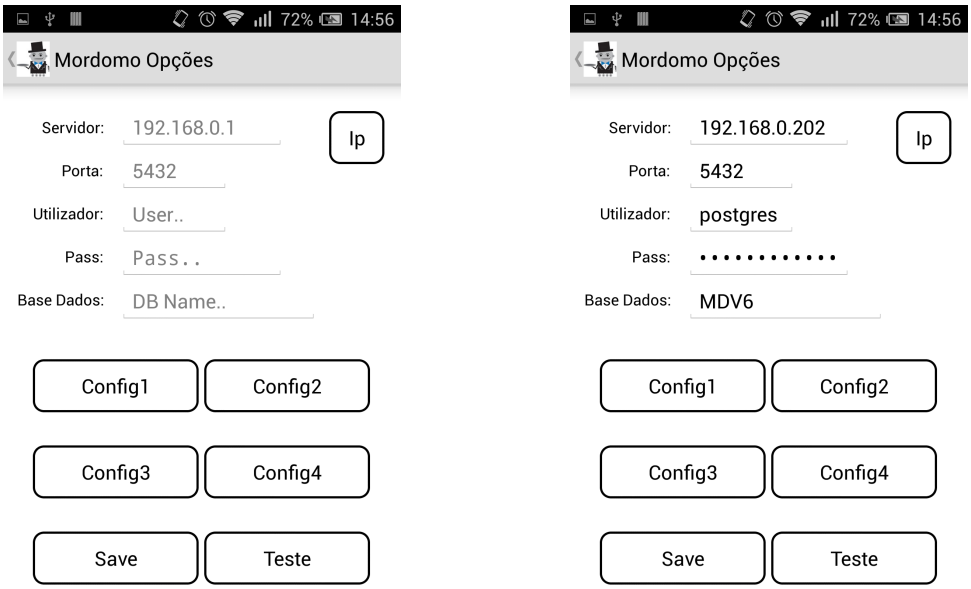
(a) Interface de autenticação (b) Interface de autenticação preenchida

Figura 4.1: Interface Opções

Esta interface é composta por dois *StackLayouts* dentro de uma *StackLayout* principal, normalmente chamado de *mainLayout* que representa todo o ecrã. O primeiro dos dois *StackLayouts* contém como filhos o botão Opções e o logótipo, enquanto que o segundo contém os dois *Entry*, onde o utilizador insere as suas credenciais, seguido do botão de *Login*. Como é possível ver na figura 4.1 todos os *StackLayouts* estão com uma orientação vertical, ou seja todos os seus filhos irão ser colocados numa pilha na vertical.

4.2.2 Opções

A interface Opções interface permite o utilizador configurar a ligação a qualquer base de dados Postgres. No âmbito deste projeto, esta interface tem como objetivo permitir que a aplicação móvel possa comunicar diretamente com a base de dados do sistema. Esta funcionalidade apenas existe por forma a responder a falhas que possam existir no sistema, nomeadamente falhas momentâneas no acesso ao *web service*. Permitindo assim que mesmo em caso e falhas o utilizador continue a ter acesso a algumas funcionalidades do sistema. Apesar de cada vez ser mais fácil escrever nos *smartphones*, esta ainda é um dos pontos mais criticados pelos seus utilizadores. Para facilitar estas configurações foram criados quatro botões que funcionam como uma espécie de configurações favoritas do utilizador. O utilizador pode então gravar até quatro configurações que serão armazenadas num ficheiro localmente. O botão de teste, tal como o próprio nome sugere, serve para testar a ligação à base de dados que foi parametrizada.



(a) Interface Opções no estado inicial (b) Interface Opções depois de preenchida

Figura 4.2: Interface Opções

Em cada configuração o utilizador, recorrendo ao botão *Ip*, pode definir dois endereços no servidor. Esta funcionalidade foi implementada por forma ao utilizador poder facilmente alternar em cada configuração entre *ip* interno e *ip* externo, que se referem respetivamente a um acesso quando estão ligados à mesma rede que o sistema e quando estão a comunicar através da Internet. Esta funcionalidade foi pensada para os casos em que o utilizador se encontra em casa e por uma das imensas razões que possam existir o utilizador pretende utilizar o sistema através da sua rede em vez de um acesso via Internet.

Esta interface é composta, para além do *mainLayout*, por dois *StackLayouts* que contém vários *StackLayouts*. Este trabalho com os *StackLayouts* verificou-se de bastante importância pois pretende-se que a aplicação mantenha o seu aspecto independentemente da dimensão dos ecrãs dos dispositivos móveis usados pelo utilizador, sejam eles *tablets* ou *smartphones*. Em ecrãs com bastantes componentes como este, é necessário ter bastantes cuidados.

4.2.3 Menu principal

Após o utilizador realizar a autenticação o próximo ecrã, a aparecer ao utilizador, é obrigatoriamente o Menu. Neste ecrã o utilizador tem acesso a todas as funcionalidades do sistema Mordomo, ou seja é a unidade central de navegação. Como é possível verificar na figura 4.3, optou-se por um menu simples e intuitivo, com os seguintes botões que permitem ao utilizador navegar na aplicação.

Figura 4.3: Interface do menu principal

- Eventos Ocorridos
- Próximas Atividades
- Serviços
- Extras

Isto permite que o utilizador facilmente e em poucas iterações consiga atingir os seus objetivos. No momento em que o utilizador realiza a autenticação, dá-se o início de uma *Thread* responsável pelas notificações, sendo o utilizador notificado de qualquer alerta que aconteça estando nesta interface ou nas seguintes.

4.2.4 Eventos Ocorridos

Esta interface aparece em primeiro lugar na lista do menu, precisamente por ser a primeira coisa que o utilizador deve verificar. Aqui encontra-se o histórico de todos e quaisquer eventos que tenham ocorrido no sistema Mordomo. Mesmo os alertas ou outros eventos que ocorram na ausência de utilização da aplicação móvel serão listados por data de ocorrência. Os eventos são apresentados ao utilizador mostrando uma breve descrição, qual o utilizador responsável pelo

evento, bem como a data e hora em que ocorreu o evento e o *id* respetivo, permitindo ao utilizador saber o número total de eventos ocorridos.

Esta interface está ainda munida de uma ferramenta de pesquisa inteligente em que o utilizador não precisa de se preocupar nem com a utilização de maiúsculas ou minúsculas bem como com possíveis carateres especiais tais como acentos. Também não é necessário escrever a palavra ou texto completo. Esta ferramenta de pesquisa foi implementada com recurso ao componente *searchbar*. Já os itens com os eventos ocorridos foram implementados com recurso a uma *listview*. Ambas as componentes estão acessíveis através de Xamarin.Forms.

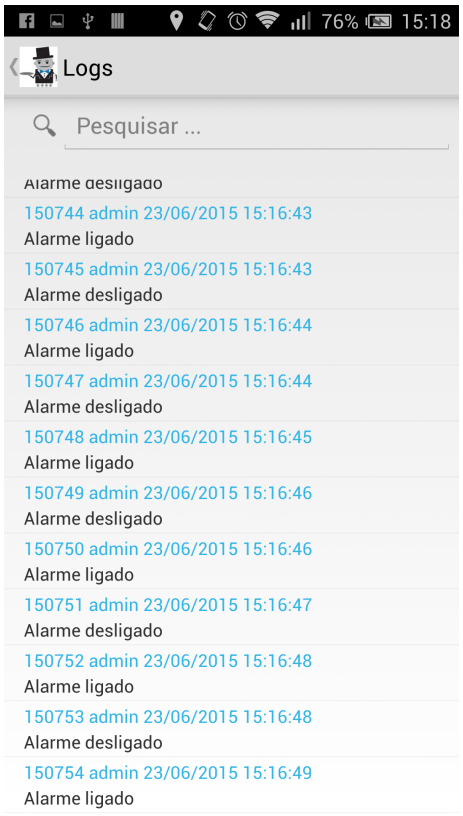


Figura 4.4: Interface de eventos ocorridos

4.2.5 Próximas atividades

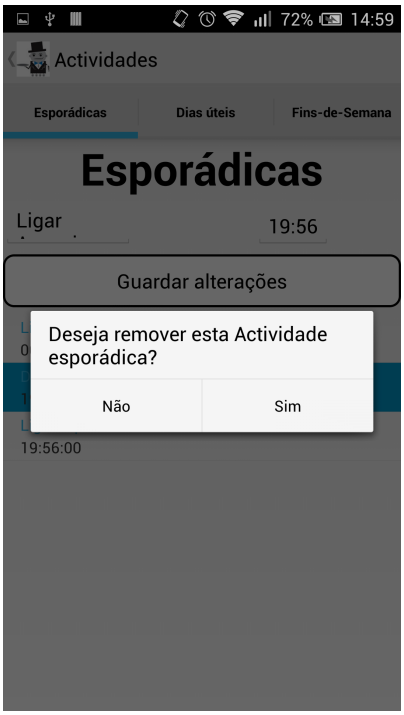
Para o agendamento de próximas atividades do sistema foi pensada uma interface que facilitasse a interação do utilizador com a mesma. A ideia desta interface é o utilizador ser capaz de programar tarefas de forma sistemática ou a uma determinada hora no próprio dia. A divisão do ecrã em três separadores correspondente a situações diferentes da mesma operação.

4.2.5.1 Esporádicas

Neste separador o utilizador pode adicionar, de entre uma lista predefinida, atividades a realizar no próprio dia podendo escolher a hora que deseja para que a atividade ocorra. Este tipo de agendamento é singular ou seja uma atividade programada neste separador apenas acontecerá uma vez.

Como é possível ver na figura 4.5 o utilizador tem acesso a dois *Pickers* sendo um dedicado para a escolha da hora enquanto que o outro contém a lista das atividades que o utilizador pode agendar. Após feito o agendamento o utilizador pode ou guardar as alterações ou pode facilmente remover alguma atividade. Para tal basta tocar no item desejado sendo então solicitada, através de uma caixa de dialogo, a confirmação da remoção, como se mostra na figura 4.7b.

(a) Agendamento de atividades esporádicas



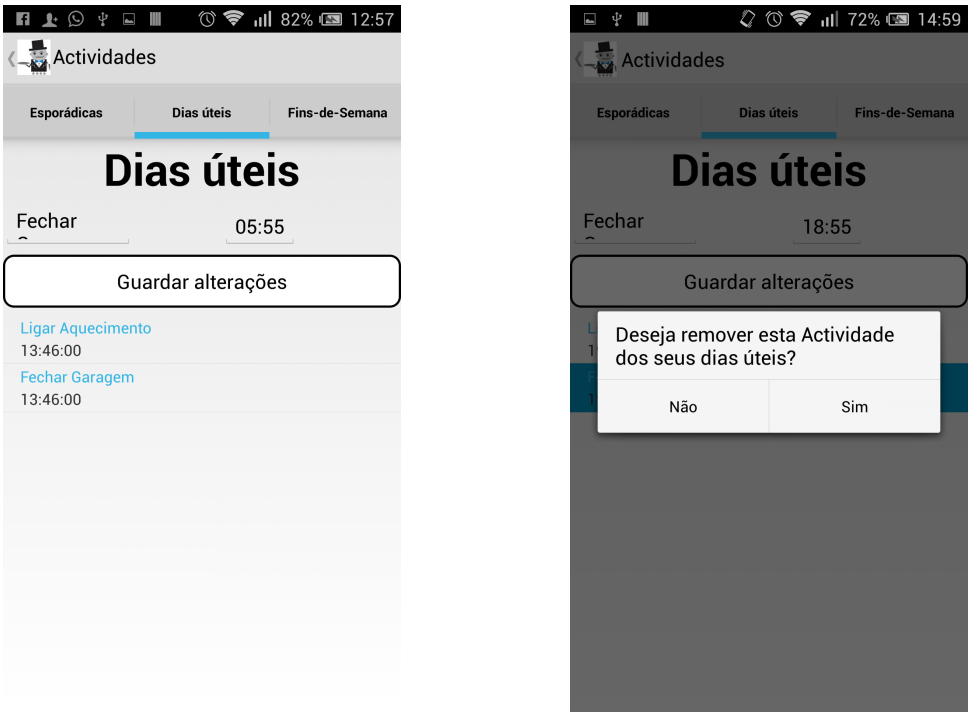
(b) Dialogo para eliminar actividade

Figura 4.5: Interface atividades esporádicas

4.2.5.2 Dias úteis

Este separador, como é possível ver na figura 4.6, permite ao utilizador programar atividades para que estas aconteçam em todos os dias úteis à mesma hora. Para a maior parte das pessoas existe um ritmo de vida completamente diferente entre os dias úteis e os fins de semana, tal como acontece por exemplo com uma tarefa muito comum como a de ter o despertador sempre para a mesma hora durante a semana e ao fim de semana nem sequer ter despertador. Tendo isto em conta foi então pensada esta divisão de agendamento de tarefas associada a tarefas comuns e sistemáticas, o que vem diminuir as preocupações do utilizador e por consequência aumentar o conforto. A

forma de agendamento é idêntica à já explicada anteriormente nas atividades esporádicas, havendo também a possibilidade de eliminar atividades e acrescentar novas sempre que o utilizador desejar.



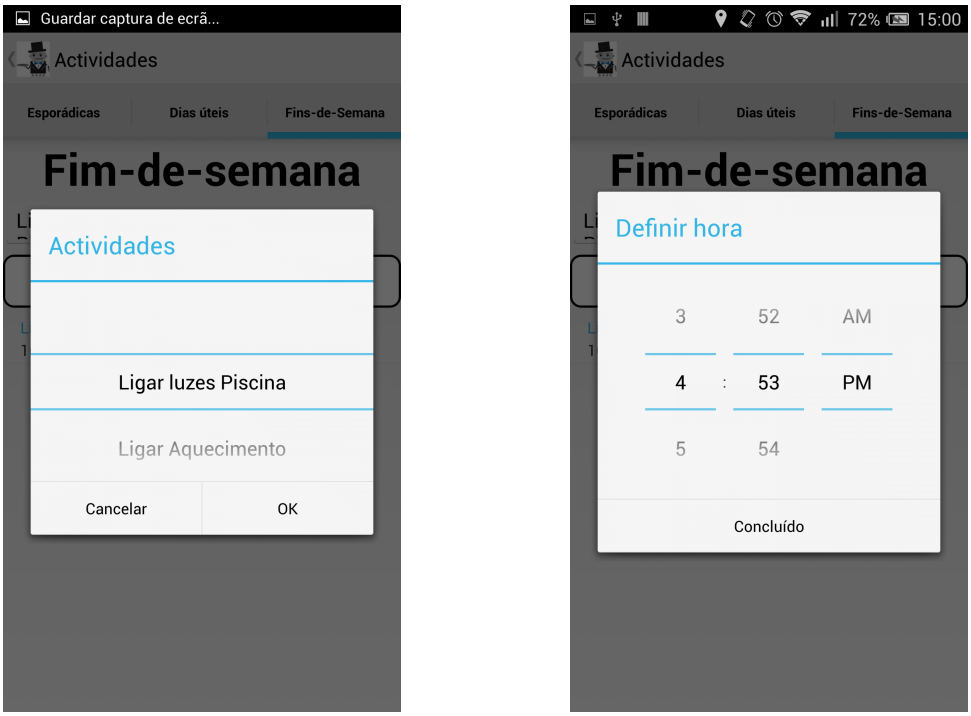
(a) Agendamento de atividades para os dias úteis (b) Dialogo para eliminar actividade

Figura 4.6: Interface atividades do dia a dia

4.2.5.3 Fins de semana

Neste separador todas e quaisquer atividades adicionadas pelo utilizador apenas ocorrem aos fins de semana, Sábados e Domingos. Esta funcionalidade de agendamento de atividades por espaços temporais serve, de certa forma para dar liberdade ao utilizador para definir os modos de funcionamento do seu sistema consoante a situação. Esta linha de pensamento oferece ao sistema uma capacidade de expansibilidade enorme, visto que poderiam ser adicionadas por exemplo, modos de funcionamento tais como: modo férias, modo inverno, etc.

Na figura 4.7a é possível ver a escolha da atividade através do componente *Picker*. O mesmo acontece para a escolha da hora ilustrado na figura 4.7b.



(a) Interface que permite definir atividades (b) Interface que permite definir hora

Figura 4.7: Interface Actividades do Fim-de-semana

4.2.6 Serviços

Nesta interface é onde o utilizador tem acesso aos dispositivos disponíveis no sistema. O acesso aos dispositivos do sistema está limitado pelo nível de acesso de cada utilizador. Como tal, o menu de serviços é feito por uma lista dos dispositivos aos quais o utilizador tem acesso. Dado que este se trata de um sistema expansível, este ecrã está munido de um meio de pesquisa pois, caso haja um aumento significativo da lista de dispositivos, esta funcionalidade permitirá que facilmente o utilizador consiga aceder ao dispositivo desejado de forma rápida e eficaz.

Cada dispositivo tem uma porta associada e devidamente configurada no Raspberry Pi, sendo a aplicação que corre no mesmo responsável por inserir e remover da base de dados todos os dispositivos de forma a que na interface dos serviços apenas surjam dispositivos que estejam em funcionamento com o sistema e com os quais o utilizador possa interagir.

Nesta primeira versão apenas foram feitos testes simples que comprovam a possibilidade de controlo e monitorização de dispositivos através de uma aplicação móvel. Os nomes e imagens presentes na figura 4.8 devem ser vistos como possíveis dispositivos associados ao sistema mordomo. No entanto as interfaces foram pensadas e implementadas de forma a permitirem ao utilizador interagir com cada um desses tipos de dispositivos.

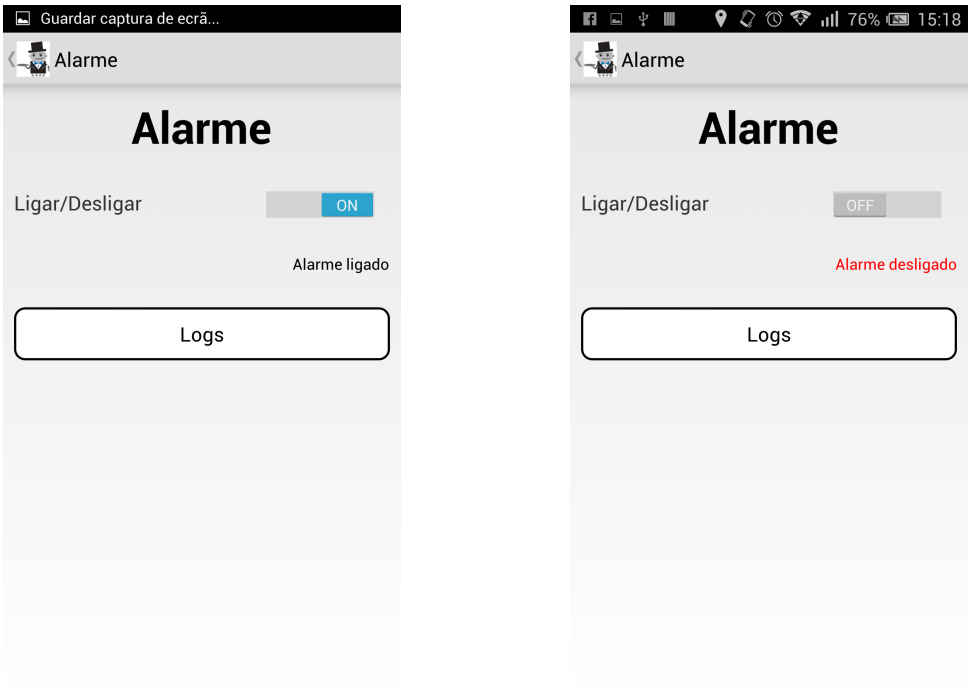
De seguida são descritas as interfaces pensadas e implementadas para alguns tipos de dispositivos comuns e com potencial no contexto da automação residencial.

Figura 4.8: Interface de Serviços

4.2.6.1 Alarme

Esta interface permite ao utilizador controlar, monitorizar e ainda ter acesso a uma interface que lhe permita verificar todo o histórico referente aos alarmes. O controlo é feito através de um interrutor. Permitindo que o utilizador, onde quer que esteja desde que tenha acesso à Internet e com recurso a um simples toque no seu *smartphone*, consiga controlar o estado do alarme da sua propriedade. Esta implementação do controlo do dispositivo é feita através de um pedido ao *Web service* que trata de informar o Raspberry Pi da ação a tomar.

Na figura 4.9a e 4.9b é possível verificar a forma de como a interface foi pensada de modo a evidenciar as diferenças de estado do dispositivo a controlar e monitorizar. Esta coerência no desenho é mantida em todas as interfaces como será possível verificar posteriormente.



(a) Interface do Alarme quando este está ligado (b) Interface do Alarme quando este está desligado

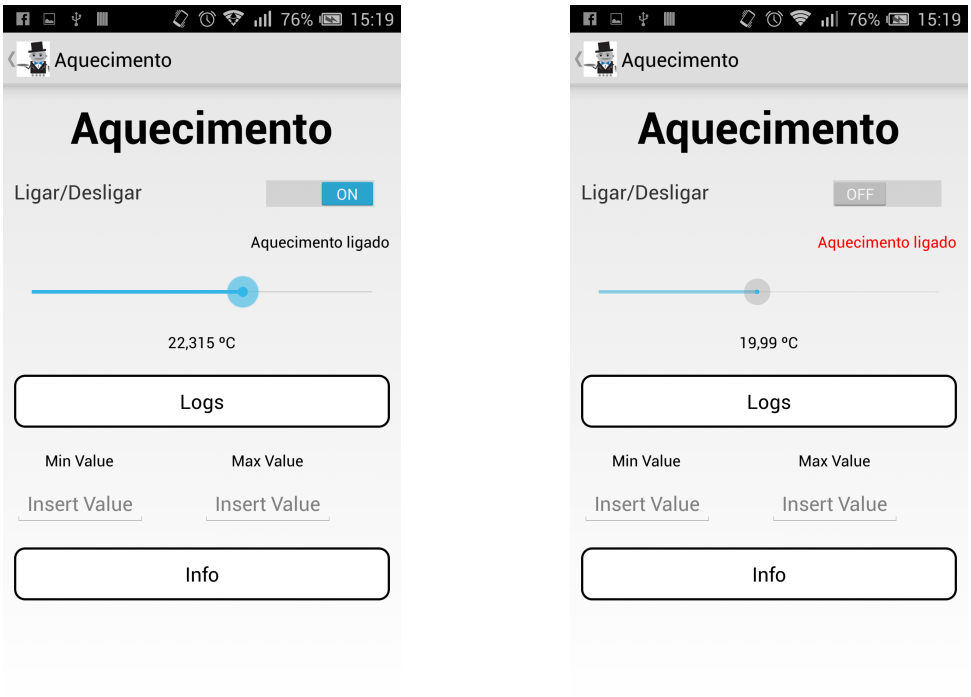
Figura 4.9: Interface do Alarme

4.2.6.2 Aquecimento

Nesta interface o utilizador pode atribuir valores numéricos de acordo com a temperatura que pretende, pode também parametrizar regras para o aquecimento, e verificar informações relacionadas com este dispositivo. Isto tudo para além de poder controlar, monitorizar e ainda ter acesso a uma interface que permite verificar todo o histórico referente ao aquecimento.

Como é possível ver em qualquer uma das duas partes da figura 4.10 o utilizador pode definir o valor da temperatura que pretende através de um *slider*. Este componente foi escolhido por forma a facilitar a interação do utilizador com o sistema. Como é possível reparar na figura 4.10a o utilizador não pode definir a temperatura que deseja, caso o dispositivo se encontre desligado.

Como é possível ver na figura 4.10 o utilizador nesta interface tem a possibilidade de estabelecer limites de temperatura para um máximo e para um mínimo. Com a introdução destes valores o sistema ao detetar uma temperatura superior ou inferior às que foram parametrizadas reage automaticamente por forma a manter a gama pretendida. Este tipo de abordagem tem bastante margem de crescimento num sistema neste contexto, quanto maior for o número de regras implementadas maior será a inteligência e autonomia do sistema.



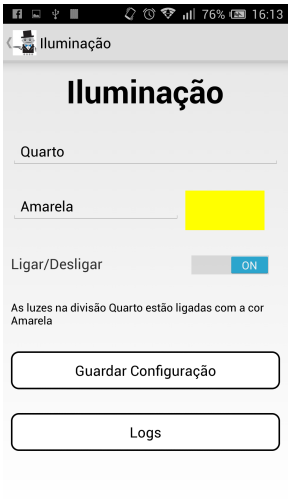
(a) Interface do Aquecimento ligado (b) Interface do Aquecimento desligado

Figura 4.10: Interface do Aquecimento

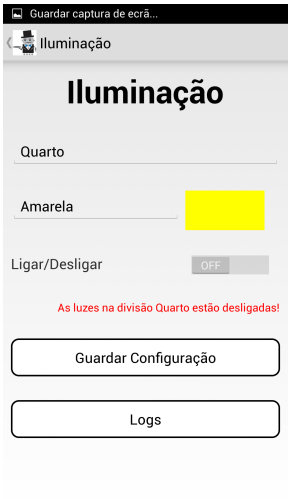
4.2.6.3 Iluminação

No conjunto de figuras 4.11 é possível ver as interfaces que permitem ao utilizador fazer o controlo e monitorização da iluminação. Para tal foi pensada uma interface com dois componentes em que o utilizador pode selecionar a divisão e a cor da iluminação que deseja, supondo que ao sistema está associado um subsistema de iluminação de leds que permitem escolher a cor. Nas situações em que o objetivo é permitir ao utilizador escolher uma de várias coisas, foi implementado o componente denominado por *Picker* que está representado na figura 4.11c.

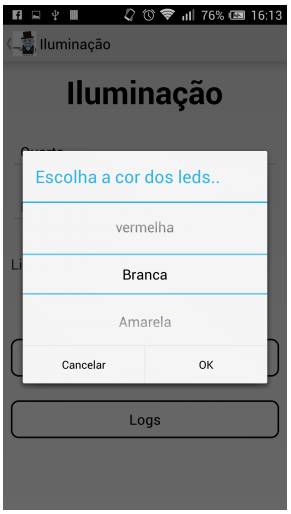
Após ter sido feita a escolha da divisão e da cor da iluminação o utilizador pode então ligar ou desligar a iluminação na respetiva divisão, as mudanças de estado são apresentadas nas interfaces tal como é possível ver nas figuras 4.11a e 4.11b. Tal como acontece para outros dispositivos o utilizador é capaz de aceder ao histórico dos eventos relacionados com a iluminação e pode também gravar a última configuração em que a cada divisão fica associada uma cor. Isto permite que o utilizador tenha as cores que pretende associadas a cada divisão sem ter que estar constantemente a selecionar.



(a) Interface do Iluminação ligado



(b) Interface do Iluminação desligado

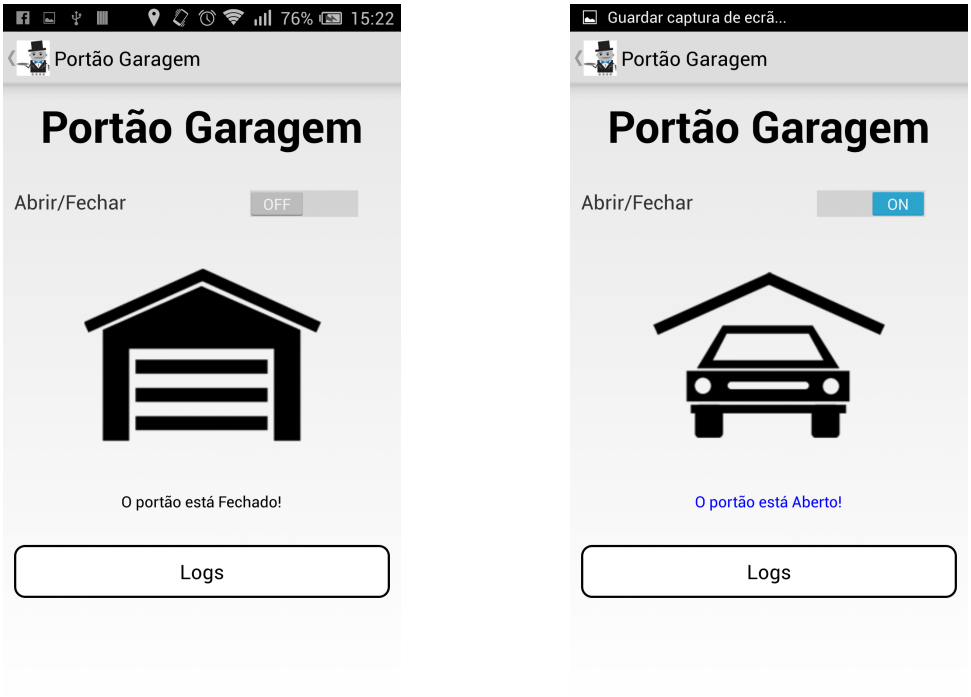


(c) *Picker* utilizado para selecionar cor

Figura 4.11: Interface da Iluminação

4.2.6.4 Portão Garagem

Nesta interface o utilizador tem a possibilidade de abrir e fechar e monitorizar o portão da garagem onde quer que esteja. Como é possível ver no conjunto das figuras 4.12, nesta interface o estado do portão da garagem é indicado não só através de texto mas também através de demonstração gráfica. Mais uma vez é possível verificar o histórico filtrado aos eventos ocorridos relacionados com o portão da garagem.



(a) Interface do Aquecimento ligado (b) Interface do Aquecimento desligado

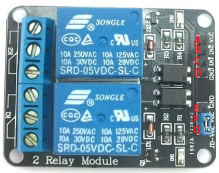
Figura 4.12: Interface do Aquecimento

Apesar de não ter sido realizado qualquer teste no controlo de um portão real, a sua implementação foi pensada. Para que tal implementação fosse possível, para além do Raspberry Pi, seria então necessário recorrer a:

- **Interruptor magnético**. Este componente seria necessário para determinar quando é que a garagem estaria aberta ou fechada. Metade do interruptor estaria estrategicamente fixado na parede enquanto que a outra metade estaria presa ao portão na sua parte superior, tal como ilustrado na figura 4.13a.
- ***Sainsmart Relay Module***. Uma pequena placa, como ilustrado na figura 4.13b muito utilizada em projetos com Raspberry Pi. Esta placa placa representa o módulo de potência que foi referido na arquitetura do sistema e que permite a ligação de dispositivos comuns de uma casa que estão ligados à rede elétrica de 220V. Esta placa contém relés isolados já embutidos. Separa o fornecimento de energia do sinal através de uma acoplador óptico protegendo assim os outros componentes quando existem colapsos do campo magnético da bobina.

- **Resistências**.

(a) Montagem do interruptor magnético num portão



(b) Módulo de relés

Figura 4.13: Interface do Aquecimento

4.2.7 Extras

Esta secção é reservada a funcionalidades extra. Essas funcionalidades são:

- Acesso às imagens da base de dados de documentos do sistema.
- Acesso à localização do utilizador por GPS.
- Acesso à meteorologia das principais cidades de vários países.

No que toca ao primeiro item, este foi implementado por forma a pensar na integração de um sistema de vigilância em que uma câmara associada ao sistema inseria imagens na base de dados de documentos, e o utilizador com recurso a esta funcionalidade teria possibilidade de monitorizar o seu sistema de segurança através do *smartphone*. O acesso à localização do utilizador foi implementado a pensar na possibilidade de o utilizador desejar estabelecer regras que tenham como parâmetros a sua localização em relação ao sistema, tal como por exemplo o sistema ser capaz de verificar que o utilizador estava fora de casa e automaticamente fosse ligado o alarme caso este estivesse desligado. Estas são implementações a pensar na expansibilidade do sistema desenvolvido.

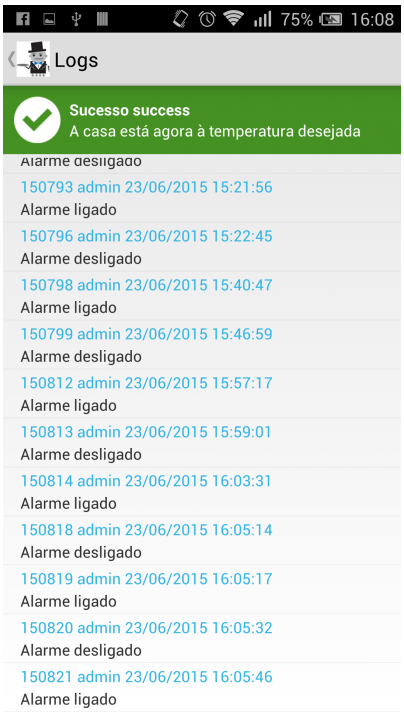
Já a funcionalidade que permite ao utilizador verificar a meteorologia foi implementada a pensar num outro lado da mordomia que a aplicação pode oferecer nas próximas versões, tornando-se assim mais completa. Esta funcionalidade, através do consumo de um *web service open-source*, permite que o utilizador saiba a meteorologia sem ter que para isso sair da aplicação para abrir uma outra que faça algo semelhante neste contexto.

4.2.8 Notificações

A aplicação móvel contém um sistema de notificações tal como foi referido anteriormente. Este sistema foi implementado com recurso a um *plugin* feito para Xamarin.Forms.

As implementações de notificação foram feitas de forma a serem bem visíveis e demonstrativas do tipo de alerta. Para tal foi utilizado um sistema de cores, que já foi explicado anteriormente e que foi implementado tal como as figuras 4.14 e 4.15 mostram.

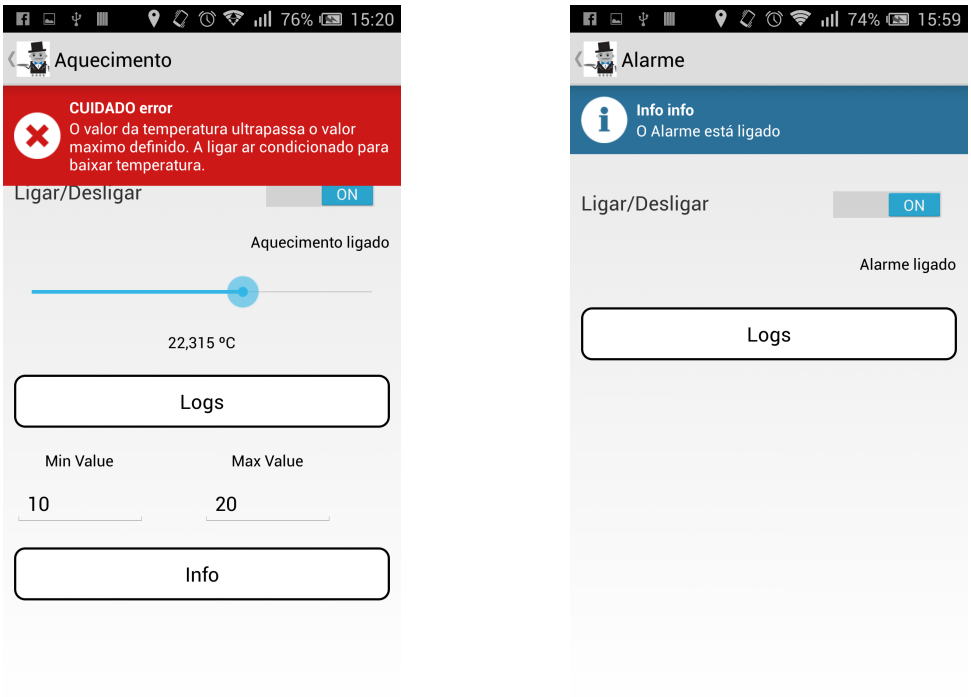
Esta implementação foi feita recorrendo a uma *thread* que executa verificações de notificações de trinta em trinta segundos. Como se pode ver na figura 4.14a as notificações aparecem em qualquer ecrã, desde que o utilizador faz a autenticação, e não só na altura em que a ação é realizada como dão a entender as outras imagens de notificações.



(a) Representação da notificação de sucesso

(b) Representação da notificação de alerta

Figura 4.14: Representação da notificação de sucesso e alerta



(a) Representação da notificação de emergência (b) Representação da notificação de informação

Figura 4.15: Representação da notificação de emergência e informação

4.3 *Webservice*

O *web service* é a camada responsável pela comunicação entre a aplicação móvel e o Raspberry Pi. No modo de funcionamento ideal do sistema, o *web service* apenas é responsável pelas funcionalidades relacionadas com a base de dados, tais como inserir, atualizar ou eliminar dados. No entanto, e tendo em conta que o Webservice está a correr no Raspberry Pi e visto que este foi implementado em C# com recurso à biblioteca *Raspberry.IO.GeneralPurpose*, o *web service* é capaz de atuar diretamente nos dispositivos associados ao sistema. Isto permite que o utilizador tenha acesso aos dispositivos através da aplicação móvel mesmo que a base de dados não esteja em funcionamento, tal como foi explicado no capítulo da arquitetura do sistema no tópico do funcionamento do sistema em modo degradado.

Este trabalho centra-se acima de tudo no desenvolvimento da aplicação móvel, onde nesta primeira versão desenvolvida, foram apenas implementadas funções básicas no *web service* que permitem ao utilizador usar o sistema Mordomo que foi anteriormente especificado.

Foi então desenvolvida uma *Web Application*, ASP.NET em C# com métodos acessíveis pela aplicação móvel e pela aplicação do Raspberry Pi, designados por *web Methods*. Estes métodos de funcionalidade comum incluem:

- **inputRasp_DB**. Este método é chamado pela aplicação do Raspberry Pi quando existe a atuação de alguma entrada e tem como propósito inserir e atualizar a base de dados nos respetivos campos e tabelas. Um exemplo a mudança de estado de um interruptor. Como se

pode ver na figura 4.16 este método tem como parâmetros de entrada duas variáveis do tipo *String*, ou seja texto, cujos nomes são dispositivo e estado. E tem como saída uma variável do tipo Boolean, ou seja uma variável lógica à qual apenas estão associados dois valores, verdadeiro ou falso para os casos de sucesso ou insucesso respetivamente.

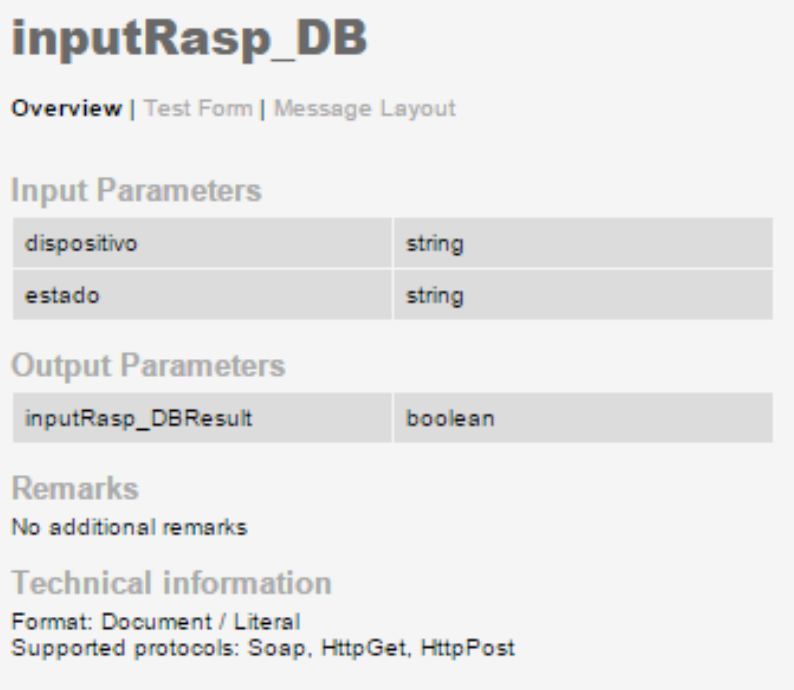


Figura 4.16: Interface de um método presente na *Web Application*

- **toggleOnOffDevice**. Este método é chamado pela aplicação móvel quando o utilizador quer ligar ou desligar um dispositivo. Este método para além de inserir e atualizar a base de dados faz também a atuação direta no dispositivo. Este método tem como parâmetros de entrada o nome do utilizador, o dispositivo, o estado e descrição. Tem como saída uma variável lógica que retorna um de dois valores que representam sucesso ou insucesso.

- **getData**. Este método é consumido pela aplicação móvel quando o utilizador pretende aceder ao histórico de eventos e quando é necessário listar os dispositivos associados ao sistema. Tem como parâmetros de entrada o nome do utilizador e do dispositivo. Retorna um vetor de dados.

- **setRules**. Este método é chamado quando o utilizador pretende definir uma regra. Tem como parâmetros de entrada o nome do utilizador e do dispositivo e os valores máximo e mínimos a atribuir ao dispositivo. O método tem como retorno uma variável lógica que indica ao sistema o sucesso ou insucesso da operação.

- **testLogin**. Este método é chamado sempre que o utilizador faz a autenticação no sistema. Tem como parâmetros de entrada o nome do utilizador e respetiva palavra-chave. Retorna

também uma variável lógica que informa a aplicação móvel do sucesso ou insucesso da autenticação.

Para além da camada lógica, foi também implementada a camada de acesso à base de dados. Para tal foi necessário recorrer à biblioteca Npgsql que permite comunicar e fazer o acesso a dados de bases de dados PostgreSQL.

4.4 Aplicação Raspberry Pi

A aplicação Raspberry Pi foi, tal como todas as outras, implementada em C#. Tendo em conta a finalidade desta aplicação, esta foi implementada como uma aplicação de consola. Esta aplicação foi a última a ser desenvolvida visto que o sistema sem esta aplicação é capaz de satisfazer todas as funcionalidades pretendidas à exceção da funcionalidade de ler valores dos dispositivos quando estes estão configurados como entradas no sistema. Permitir que o sistema reconheça a atuação de um interruptor é um exemplo de uma funcionalidade deste tipo.

Tal como na *Web application*, foi necessário recorrer à biblioteca *Raspberry.IO.GeneralPurpose*. Esta biblioteca permite ter acesso, de forma conveniente, aos pinos GPIO do Raspberry Pi. GPIO, *General purpose Input/Output*, são as portas programáveis de entrada e saída de dados, neste caso, do Raspberry Pi. Com esta biblioteca foi possível configurar as entradas tanto como interrutores bem como botões de pressão. Mesmo não tendo sido testado, foi estudada a potencialidade oferecida por esta biblioteca e foi verificado que a expansibilidade do sistema não está comprometida pois esta biblioteca suporta os mais variados tipos de interface de periféricos, tal como por exemplo periféricos com interface série.

Foi então implementada uma *Thread* que está constantemente a verificar o estado das entradas. Para o caso dos dispositivos configurados como entradas a aplicação ao detetar alterações do estado faz a recolha e identificação do mesmo, e depois de feita a análise das implicações que essa mudança possa ter desencadeado é feito o pedido ao *web service* de forma a executar a atualização da base de dados com a devida informação. Foram predefinidas regras dedicado às entradas por forma a verificar a potencialidade do sistema neste aspeto. Um exemplo implementado foi para o caso de o interrutor que simula um sensor de presença, tenha sido atuado mais de cinco vezes num minuto. É então feito o registo de um alerta seguido de uma ação, como por exemplo ligar um Led simulando o trancar de uma porta, que corrija o problema existente no sistema. Esta mesma *thread* é também responsável por estar constantemente a consumir o *web service* por forma a verificar possíveis ações que o utilizador faça. Esta parte da aplicação é responsável pela verificação constante dos valores dos dispositivos por forma a detetar incumprimento de regras relacionadas com as ações do utilizador da aplicação móvel, e por consequência ter de realizar as operações necessárias para que o sistema volte à normalidade.

Na figura 4.17 é mostrado o esquema do circuito utilizado na ligação dos componentes que permitem fazer a simulação dos dispositivos associados ao sistema, com os quais o utilizador pode interagir.

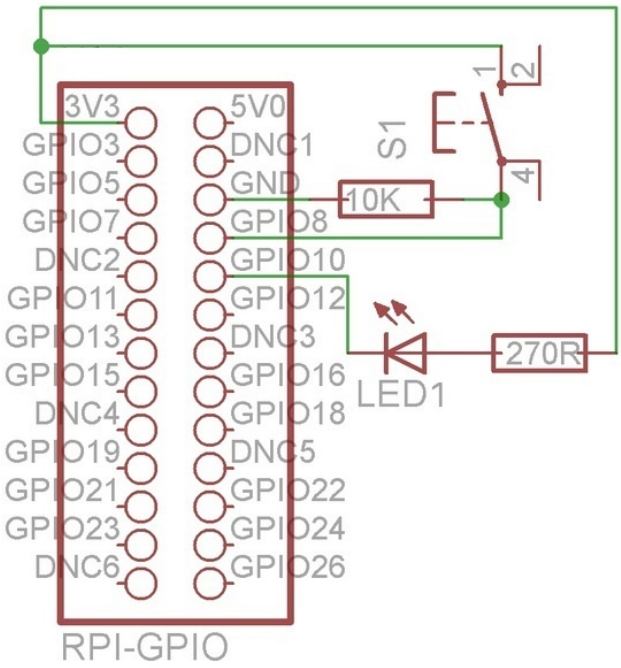


Figura 4.17: Esquema utilizado para a ligação de componentes que simulam os dispositivos associados ao sistema

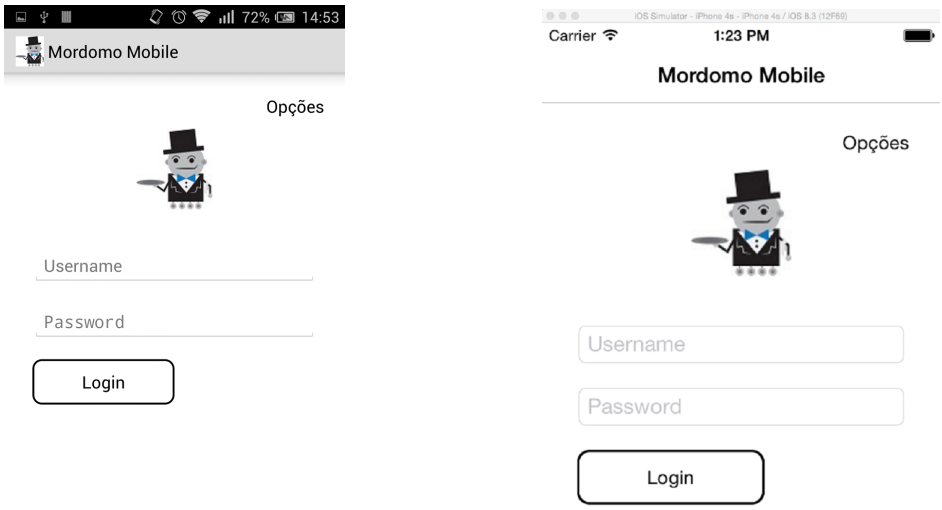
Capítulo 5

Resultados

Neste capitulo são apresentados os resultados e operações do sistema. Primeiramente será dada uma explicação das operações realizadas no sistema e a forma como os utilizadores interagem com o mesmo. Depois serão apresentados os testes realizados ao sistema e o seu resultado, utilizando exemplos de utilização e as vantagens que este apresenta.

5.1 Operação em Android e iOS

Um dos resultados obtidos com um nível elevado de importância é sem dúvida o resultado apresentado no conjunto das figuras 5.1, que compara o aspeto gráfico da aplicação multiplataforma a executar em Android e em iOS.



(a) Interface da aplicação num dispositivo móvel Android

(b) Interface da aplicação num dispositivo móvel iOS

Figura 5.1: Comparação das interfaces entre dispositivos móveis Android e iOS

Na figura 5.1a temos então representado o aspeto da interface num dispositivo móvel Android e na figura 5.1b a mesma aplicação mas a correr num dispositivo iOS. Como é possível ver, apesar da aplicação ser a mesma, os componentes presentes em cada uma das interfaces são os componentes nativos da plataforma específica em que estão a correr proporcionando ao utilizador uma experiência familiar, com o *look and feel* da sua plataforma. Este resultado apenas foi possível de concretizar com a utilização do Xamarin.Forms que traduz o mesmo código em C# para os objetos nativos das plataformas de destino.

5.2 Operação e execução

A aplicação móvel é o que permite ao utilizador interagir com o sistema. Como tal sempre que o utilizador realiza alguma operação na interface móvel, outras operações de caráter interno ao sistema são realizadas de forma a que o utilizador atinja os seus objetivos enquanto utilizador do sistema Mordomo.

Tabela 5.1: Interação entre aplicação móvel e *Webservice* ao nível dos pedidos

Aplicação móvel - Webservice			
#	**Tipo**	**Dados**	**Respostas possíveis**
1	login	nome de utilizador, senha	(12,13)respostaLogin
2	logout	-	(14,15)respostaLogout
3	testaBD	servidor, porta, utilizador, senha, base de dados	(16,17)respostaTestaBD
4	listaHistórico	utilizador, dispositivo	(18)respostaLista
5	toggleOn	utilizador, dispositivo, estado, descrição	(19,20)respostaToggleOn
6	toggleOff	utilizador, dispostivio, estado, descrição	(21,22)respostaToggleOff
7	addAtividade	nome, hora, tipo	(23,24)respostaAddAtividade
8	remAtividade	id	(25,26)respostaRemAtividade
9	listaDispositivos	utilizador	(27)respostaListaDisp
10	atribuiValor	utilizador, dispositivo, valor	(28,29)respostaAtribuiVal
11	defineRegra	utilizador, dispositivo, valor máximo, valor mínimo	(30,31)respostaDefRega

As operações envolvendo a aplicação móvel e o *web service* estão resumidas nas tabelas 5.1 e 5.2. Na tabela 5.1 estão representados os pedidos que a aplicação móvel faz ao *Webservice*, estes pedidos são caraterizados por um tipo, que basicamente é o nome do pedido, e os dados que são enviados nesse pedido e que tornam possível a realização da operação. Nessa mesma tabela são indicadas as possíveis respostas a obter desses mesmos pedidos, respostas estas que estão representadas na tabela 5.2. Estas respostas, vindas do *Webservice* para a aplicação móvel, representam o sucesso ou insucesso dos pedidos.

Para aceder ao sistema mordomo o utilizador tem obrigatoriamente que iniciar a aplicação móvel desenvolvida. Tem também que realizar a autenticação na mesma. No que diz respeito a autenticação, sempre que o utilizador tenta realizar uma autenticação este tem obrigatoriamente que preencher os campos de utilizador e senha. É então enviado um pedido ao *web service*, que verifica a existência e a validação das credenciais do utilizador com a base de dados e retorna uma resposta à aplicação móvel. O insucesso dessa verificação pode estar relacionado com dois aspetos. Um dos motivos poderá ser a introdução errada das credenciais, sendo então necessário

apenas reintroduzir as credenciais e voltar a tentar a operação de autenticação. O outro motivo da falha pode dever-se ao facto do *web service* não estar acessível no momento da autenticação, devendo o utilizador proceder à configuração da ligação da base de dados, no próprio *smartphone*, de forma a que o seu dispositivo móvel consiga ter acesso direto à base de dados e assim realizar a operação de autenticação. Depois de realizada a autenticação o utilizador passa a ter acesso ao sistema e suas funcionalidades. Uma dessas funcionalidades é a de verificar o histórico de todo o sistema ou apenas o histórico relacionado com um único dispositivo. Para a realização desta operação o utilizador tem no menu principal a opção de ir para a interface de eventos ocorridos. Ao aceder a esta interface a aplicação móvel faz um pedido ao *web service* sem especificar nenhum dispositivo, obtendo como resposta, um vetor de dados relativo ao histórico de todos os dispositivos. Quando o utilizador pretende aceder ao histórico de dispositivos específicos, esse acesso é feito nas interfaces referentes a cada dispositivo. Nesses casos a operação entre aplicação móvel e *web service* ocorre de igual modo, à exceção de que o pedido no campo dispositivo terá especificado o nome do dispositivo. Para o utilizador poder atuar sobre um dispositivo associado ao sistema, é necessário que este aceda, através do menu principal, à interface dos serviços. Ao aceder a esta interface a aplicação móvel faz um pedido ao *web service* enviando apenas como parâmetro o nome do utilizador, de forma a que na resposta apenas surja um vetor com os dispositivos aos quais o utilizador pode aceder. Após realizada esta operação o utilizador pode então aceder à interface de cada um dos dispositivos que lhe estão acessíveis para então atuar sobre eles. Dependendo do tipo de dispositivo escolhido pelo utilizador este pode ser um dispositivo em que o utilizador apenas pode ligar ou desligar ou ser um dispositivo em que é possível também atribuir um valor ao mesmo. As operações de ligar ou desligar podem falhar se o seu estado não estiver atualizado devido a uma operação manual quase simultânea. Quanto o utilizador pretende atribuir um valor, como por exemplo definir a temperatura que deseja, este é enviado no pedido ao *Webservice* bem como o dispositivo a que se refere e o nome do utilizador que está a executar a operação. Uma outra funcionalidade implementada foi o agendamento de atividades, em que o utilizador pode programar atividades a ocorrer no sistema. Para poder realizar tal operação o utilizador necessita de aceder à interface de Próximas Atividades, interface que está dividida em três separadores como já foi explicado anteriormente na implementação. Em termos operacionais o seu funcionamento é muito semelhante, sendo a única diferença a forma como o sistema vai abordar em termos lógicos os três diferentes tipos de agendamento. Essa distinção é feita através do parâmetro que diz respeito ao tipo de atividade e que é enviado no pedido sempre que o utilizador pretende adicionar uma nova atividade. Para remover uma atividade o utilizador apenas necessita de selecionar na interface o item correspondente à atividade que pretende eliminar e confirmar o desejo da eliminação daquele item numa caixa de diálogo que surge após a seleção do item. Para a realização desta operação a aplicação móvel apenas necessita de fornecer, no pedido ao *web service*, o identificador da atividade que se pretende eliminar. Por último vem a operação referente às regras, que operação não está disponível para todos os dispositivos. Um bom exemplo é o da definição dos valores máximos e mínimos admissíveis que a temperatura pode atingir. Para realizar esta operação o utilizador necessita de aceder à interface relativa ao aquecimento. Nesta

operação a aplicação móvel faz um pedido ao *web service* com o nome do utilizador, dispositivo e claro com o valor máximo e mínimo definidos pelo utilizador.

Tabela 5.2: Interação entre *Webservice* e a aplicação móvel ao nível das respostas

Webservice - Aplicação móvel		
#	**Pedido**	**Resposta**
12	respostaLogin	True
13	respostaLogin	False
14	respostaLogout	True
15	respostaLogout	False
16	respostaTestaBD	True
17	respostaTestaBD	False
18	respostaLista	Array[dados]
19	respostaToggleOn	True
20	respostaToggleOn	False
21	respostaToggleOff	True
22	respostaToggleOff	False
23	respostaAddAtividade	True
24	respostaAddAtividade	False
25	respostaRemAtividade	True
26	respostaRemAtividade	False
27	respostaListaDisp	Array[dispositivos]
28	respostaAtribuiVal	True
29	respostaAtribuiVal	False
30	respostaDefRega	True
31	respostaDefRega	False

5.3 Testes e resultados de execução

No diagrama de sequência representado na figura 5.2 é possível verificar os resultados das execuções entre cada um dos módulos do sistema.

No que diz respeito aos testes, por forma a cobrir todas as funcionalidades optou-se por dividir as operações em vários tipos de testes. O acesso à Internet é uma pré-condição para todos os seguintes testes.

5.3.1 Testes de autenticação

- ***Login***. Este teste tem como entradas as credenciais do utilizador, nome de utilizador e senha. Tem como pré-condições o acesso à base de dados e a necessidade de o utilizador ter uma conta criada no sistema. A pós-condição é a autenticação do utilizador no sistema. O procedimento deste caso de utilização segue a seguinte ordem: a aplicação ser iniciada, inserir o nome de utilizador, inserir a senha, pressionar o botão *Login* e por fim é apresentada a interface do menu principal. Este teste foi sempre executado com sucesso à exceção de

quando o *web service* não está disponível. Esta situação +pde ser remediada procedendo-se à ligação direta da base de dados através da aplicação móvel.

- ***Logout***. Neste teste verifica-se a possibilidade do utilizador terminar a sua sessão na aplicação. Quando o utilizador realiza esta operação com sucesso é então apresentada a interface de autenticação. Este caso de utilização tem como pré-condição o utilizador estar autenticado na aplicação e tem como pós-condições o utilizador deixar de estar autenticado na aplicação e a apresentação da interface de autenticação. Este teste foi sempre executado com sucesso.

5.3.2 Teste de configuração à base de dados

O acesso à base de dados pode falhar quando o *web service* não se encontra disponível no caso de se tratar do primeiro acesso no dispositivo móvel, visto que após a primeira configuração, esta é guardada num ficheiro no próprio *smartphone*. Este caso de utilização tem como entradas os dados de acesso à base de dados. O utilizador tem acesso a uma funcionalidade de teste das suas credenciais por forma conseguir verificar a correção das mesmas. A pré-condição neste caso é o facto de ser obrigatório a existência de espaço suficiente para a gravação do ficheiro de configuração no dispositivo móvel. As ações para a realização deste teste são: pressionar o botão opções presente na interface de autenticação, preencher o dados relativos à conexão da base de dados, pressionar o botão teste e pressionar o botão gravar que automaticamente grava o ficheiro com as configurações e redireciona o utilizador à interface de autenticação. Este teste foi sempre realizado com sucesso.

5.3.3 Testes de eventos ocorridos

- **Listar Histórico**. Neste teste pretende-se mostrar a possibilidade do utilizador verificar todas as ocorrências no sistema. O utilizador apenas pode verificar as ocorrências dos dispositivos aos quais ele tem permissão de acesso e como tal este teste apenas tem como entrada o nome de utilizador. Tem como pré-condição a necessidade do utilizador estar autenticado e como pós-condição a apresentação de todo o histórico de eventos ocorridos. Para a realização deste teste é apenas necessário pressionar o botão eventos ocorridos no menu principal e são apresentados os resultados. Este teste foi sempre executado com sucesso.

- **Listar Histórico filtrado**. Este teste pretende realizar o mesmo que o teste anterior mas com a diferença de se verificar todas as ocorrências no sistema relativamente a um dispositivo. Para além do nome de utilizador, este teste tem como entrada o dispositivo. Tem como pré-condição a necessidade do utilizador estar autenticado e como pós-condição a apresentação de todo o histórico de eventos ocorridos relativos a um dispositivo especifico. Para a realização deste teste são necessárias as seguintes ações: no menu principal pressionar o botão serviços, pressionar um dos dispositivos disponíveis, pressionar o botão de *logs* e são então apresentados os resultados. Este teste foi sempre realizado com sucesso.

5.3.4 Testes de atuação em dispositivos

- **Ligar**. Neste teste pretende-se comprovar que o utilizador é capaz de ligar um dispositivo remotamente através de um dispositivo móvel. Este teste tem como entradas o nome do utilizador, o dispositivo, o estado e a descrição. A presença da descrição nas entradas deste teste não é de todo normal mas é justificada pelo facto de a funcionalidade de ligar ser também responsável por inserir na base de dados a informação sobre esta ocorrência que é utilizado na listagem dos eventos ocorridos nos dispositivos. Tem como pré-condições o utilizador estar autenticado no sistema e o dispositivo a ser controlado estar desligado. Tem como pós-condições o dispositivo estar ligado. Para a realização deste teste são necessárias as seguintes ações: na interface de serviços o utilizador pressionar o item associado ao dispositivo que pretende controlar, mudar o estado do botão interrutor na interface do dispositivo para ligado e por fim é atualizado o estado atual do dispositivo tanto através de texto como por imagens dependendo do dispositivo. Este teste foi sempre realizado com sucesso.

- **Desligar** Neste teste pretende-se comprovar que o utilizador é capaz de desligar um dispositivo remotamente através de um dispositivo móvel. Este teste tal como o anterior tem como entradas o nome do utilizador, o dispositivo, o estado e a descrição. Tem como pré-condições o utilizador estar autenticado no sistema e o dispositivo a ser controlado estar ligado. Tem como pós-condições o dispositivo estar desligado. A única diferença relativamente ao teste anterior é a mudança de estado do botão interrutor de ligado para desligado. Este teste foi sempre realizado com sucesso.

- **Atribuir valor** Neste teste pretende-se comprovar que o utilizador é capaz de controlar um dispositivo analógico atribuindo um valor ao mesmo. Este teste tem como entradas o nome do utilizador, o dispositivo e o valor a atribuir. Tem como pré-condições o utilizador estar autenticado no sistema e o dispositivo estar ligado. Tem como pós-condições o dispositivo apresentar o valor atribuído. Para a realização deste teste são necessárias as seguintes ações: na interface de serviços pressionar o botão associado ao dispositivo analógico a controlar, ligar o dispositivo caso este esteja desligado e por fim pressionar o indicador do *slider* e arrastar no sentido desejado até se atingir o valor pretendido. Este teste não foi realizado com sucesso visto não ter sido implementado no sistema qualquer tipo de dispositivo analógico. No entanto foi implementada uma interface na aplicação móvel preparada para fazer este tipo de operações. Apesar de não existir qualquer ação num dispositivo físico, o valor atribuído através da interface móvel é inserido e atualizado na base de dados, pois este é usado no teste relacionado com as regras que será descrito na próxima subsecção.

5.3.5 Teste de regras

Neste teste pretende-se comprovar que o sistema é capaz de autonomamente realizar ações sobre o sistema sem que haja influencia direta do utilizador. Foram então testados dois tipos diferentes de regras.

- **Limites máximo e mínimo**. Este teste serve para comprovar a capacidade de verificar que o valor atual de um dispositivo ultrapassou os limites máximo ou mínimo definidos pelo utilizador e por consequência realize uma ação com o intuito de normalizar o estado do sistema. A pré-condição deste teste é o utilizador estar autenticado e o valor atual ser superior ao valor máximo ou interior ao valor mínimo. A pós-condição é o valor atual estar dentro dos limites máximo e mínimo definidos. Para a realização deste teste é necessário realizar os seguintes passos: na interface de serviços pressionar o botão associado ao dispositivo analógico pretendido, ligar o dispositivo caso este esteja desligado, introduzir o valor mínimo pretendido, introduzir o valor máximo pretendido e por fim pressionar o indicador do *slider*e arrastar no sentido desejado até se atingir um valor superior ou inferior ao limite máximo e mínimo respetivamente. O sistema ajusta automaticamente o valor atual para o valor máximo ou mínimo dependendo se este se encontra mais próximo do limite máximo ou mínimo. Este teste foi realizado com sucesso.

- **Número máximo de ocorrências**. Este teste pretende comprovar que caso as entradas do Raspberry Pi sejam atuadas manualmente e ultrapassem um número máximo de ocorrências o sistema reage de forma autónoma e por consequência realize uma ação com o intuito de normalizar o estado do sistema. As pré-condições deste teste são estar predefinido o número máximo de ocorrências, visto a aplicação que corre no Raspberry Pi não ter interface que permita interação com o utilizador e o número de ocorrências ser superior ao número de ocorrências máximo predefinido. A pós-condição é ligar um dispositivo. Para a realização deste teste é necessário realizar os seguintes passos: iniciar a aplicação do Raspberry Pi, atuar manualmente no dispositivo de forma a ultrapassar o valor máximo definido e como resultado ser ligado automaticamente um dispositivo. Este teste foi realizado com sucesso. Neste teste foi verificado, com recurso à biblioteca usada para aceder às portas do Raspberry Pi já mencionada anteriormente, que a taxa de aquisição do sinal da porta a que está associado o dispositivo configurado como entrada é de 4hz. Esta verificação permite concluir que o sistema apenas é capaz de detetar mudanças de estado, do dispositivo à entrada do Raspberry Pi, que ocorram em intervalos iguais ou superiores a 0,25 segundos.

5.4 Resultado da operação de um caso de utilização

Para melhor descrever um caso de utilização do sistema, foi então elaborado o diagrama de sequência da figura 5.2. O caso de utilização escolhido foi o de o utilizador pretender ligar um dispositivo, neste caso o alarme. O utilizador então realiza o pedido ao sistema de ligar um dispositivo, através da aplicação móvel, executando esta um pedido ao *web service* que por sua vez faz um pedido ao Raspberry Pi para ligar o dispositivo. O Raspberry Pi, que está constantemente à espera de pedidos, realiza ou não a operação dependendo do estado atual em que se encontra o dispositivo. Após essa verificação, devolve o resultado do pedido ao *Webservice*, fazendo este, por sua vez, um pedido à base de dados no sentido de atualizar o estado do dispositivo bem como de

inserir o evento ocorrido, consoante o tipo de resposta dado pelo Raspberry Pi. Depois de feitas as operações na base de dados o *web service* devolve a resposta do pedido à aplicação móvel e esta de imediato faz outro pedido por forma a verificar as regras de modo a detetar anomalias no sistema. O *web service* realiza um pedido à base de dados por forma a obter os dados atualizados das regras e do dispositivo, pois o estado deste pode ter sido entretanto alterado por parte de uma atuação manual à entrada do Raspberry Pi. Com os dados atualizados, a camada lógica do *web service* verifica as regras e devolve uma resposta à aplicação que por sua vez devolve uma resposta ao utilizador que em caso de sucesso é a notificação de que o dispositivo se encontra ligado.

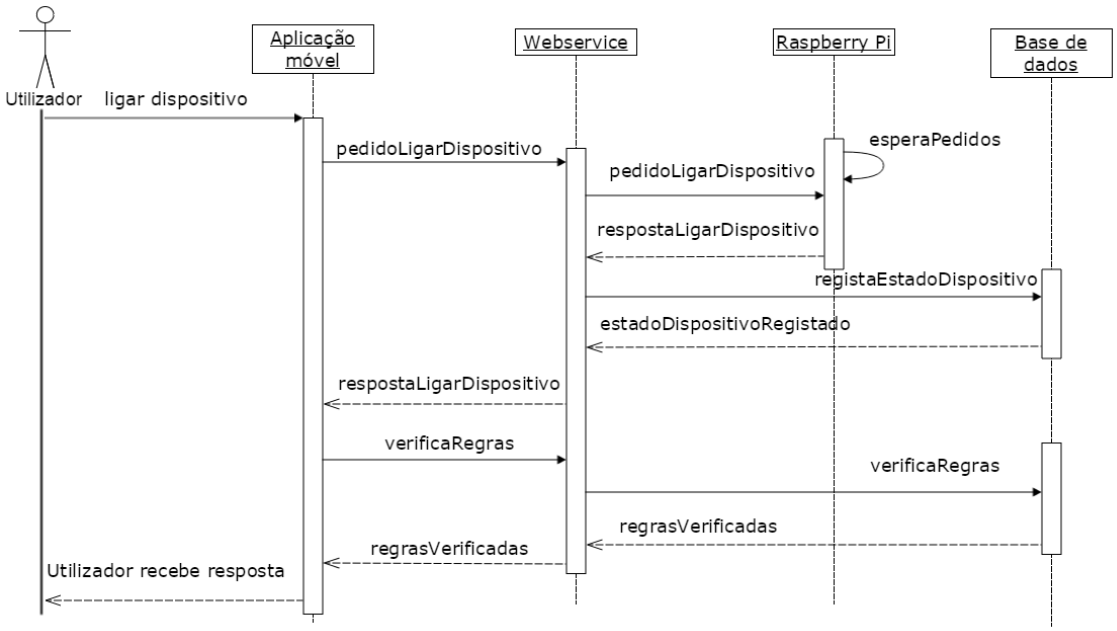


Figura 5.2: Diagrama de sequência de um caso de utilização

Em caso de sucesso o sistema deverá encontrar-se no estado representado pelo conjunto das figuras 5.3 e 5.4. O diodo emissor de luz representa um dispositivo ligado, neste caso o alarme.

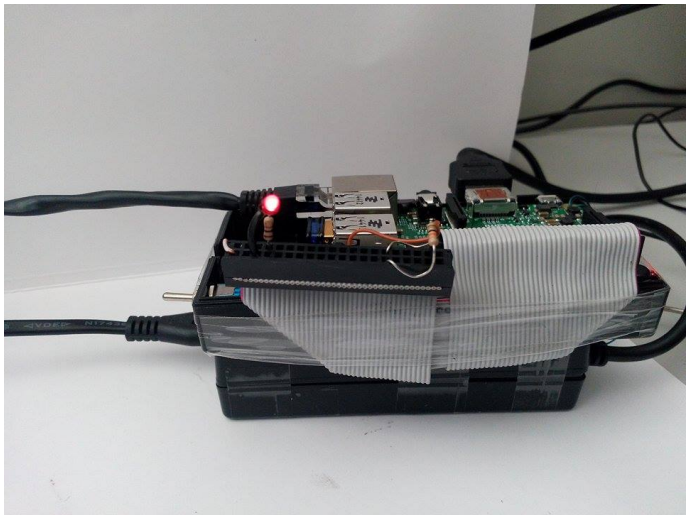


Figura 5.3: Imagem do dispositivo no estado ligado

Figura 5.4: Interface da aplicação alarme

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusão e satisfação dos objetivos

As tecnologias de informação e automação são áreas que estão em constante em evolução, praticamente todos os dias surge uma novidade tecnológica. Uma das áreas que tem beneficiado com esta evolução é a domótica, visto que cada vez mais os dispositivos domésticos se tornam mais sofisticados e permitem que das mais variadas formas seja possível interligar os dispositivos uns com os outros. A revisão bibliográfica, realizada na elaboração deste trabalho, permitiu concluir que embora atualmente existam várias alternativas no mercado que permitem realizar o controlo e monitorização de dispositivos domésticos através de dispositivos móveis, esta é uma área que está em constantes mudanças não existindo ainda uma normalização neste mercado. Ao longo do desenvolvimento deste projeto foram surgindo mudanças nas perspetivas futuras desta área, pois grandes empresas como a Google e a Microsoft recentemente afirmaram estar interessadas neste segmento do mercado e prometem trazer muitas novidades ao mesmo, tais como o novo protocolo, Weave, dedicado à internet das coisas recentemente apresentado pela Google.

A abordagem e arquitetura definida para este projeto foi pensada tendo em conta esta constante mutação no mercado deste segmento. Assim o objetivo deste projeto foi o desenvolvimento de uma aplicação para dispositivos móveis que permitisse aos utilizadores monitorizar e controlar dispositivos domésticos remotamente. Dada a existência de uma grande diversidade e heterogeneidade dos dispositivos móveis e seus sistemas operativos, é facilmente detetada uma barreira no que toca ao desenvolvimento da mesma aplicação móvel para mais do que um sistema operativo de dispositivos móveis. Por isso, outro dos objetivos iniciais foi a disponibilidade desta aplicação para a grande maioria dos utilizadores de *tablets* e *smartphones*. Este objetivo foi cumprido com recurso a uma *framework* de desenvolvimento multiplataforma, Xamarin, que permitiu grande economia de tempo no desenvolvimento, visto apenas ter sido necessário desenvolver uma aplicação para que esta ficasse disponível nas três plataformas mais utilizadas atualmente, Android , iOS e Windows Phone.

A implementação e os resultados obtidos permitiram concluir que a abordagem feita e a arquitetura escolhida foram um sucesso, pois foram atingidos os objetivos de controlar e monitorizar

dispositivos domésticos de muitos tipos remotamente através de um *smartphone*. Indo de encontro à denominação deste sistema, O Meu Mordomo, foram ainda atingidos os objetivos de comprovar a possibilidade de tornar o sistema inteligente através do uso de regras. Com a possibilidade dessa definição. o sistema foi capaz de realizar operações autonomamente, de acordo com o seu estado de funcionamento.

6.2 Trabalho Futuro

Embora todos os objetivos definidos inicialmente tenham sido atingidos com sucesso, trata-se de um sistema de grande expansibilidade existindo assim muitos aspetos que podem ser explorados e aprofundados. O primeiro ponto a explorar seria a utilização de dispositivos analógicos, utilizando conversores para que o sinal fosse reconhecido pelas portas do Raspberry Pi. Um desses dispositivos poderia ser o portão da garagem em que é necessária a utilização de um módulo de relés para o condicionamento do sinal, tal como foi explicado no capítulo da implementação. Outras expansões poderiam incluir a implementação de mais regras e de mais dependências entre as operações de forma a tornar o sistema mais robusto. A implementação de gráficos que permitissem ao utilizador visualizar o funcionamento do sistema no passado e ao longo do tempo tornaria, sem dúvida, a aplicação mais útil e apelativa. Através das coordenadas GPS, o sistema também poderia ser capaz de saber onde se encontra o utilizador e autonomamente realizar uma série de operações.

Este tipo de soluções não está limitada à automação residencial pois sistemas deste tipo podem ser aplicados a outras áreas. Como por exemplo na medicina, em que os médicos ou outros sistemas poderiam monitorizar remotamente as condições dos seus pacientes a qualquer momento. Na verdade com a chegada da internet das coisas, cenários que apenas existiam em ficção científica começam a tornar-se realidade.

Referências

[1] INSTEON. Insteon whitepaper:compared, 2013.

[2] Zeya Zheng. Home Automation, 2013.

[3] S. Soucek, S. Soucek, G. Russ, G. Russ, C. Tamarit, e C. Tamarit. The Smart Kitchen Project–An Application of Fieldbus Technology to Domotics. 2000.

[4] Diana Sobreiro da Costa Palma. *Domótica KNX / EIB de Baixo Custo*. Tese de doutoramento, Faculdade de Engenharia da Universidade do Porto, 2008.

[5] Somak R Das, Silvia Chita, Nina Peterson, Behrooz Shirazi, Medha Bhadkamkar, et al. Home automation and security for mobile devices. Em *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, 2011.

[6] Aaron Tilley. Google's nest acquires smart home hub startup revolv to control every device in your home, 2014. Disponível em http://www.forbes.com/sites/aarontilley/2014/10/24/googles-nest-acquires-smart-home-hub-maker-revolv/.

[7] Google announces Brillo, an operating system for the Internet of Things. Disponível em http://www.theverge.com/2015/5/28/8677119/google-project-brillo-iot-google-io-2015.

[8] Microsoft to support raspberry pi 2 with a free version of windows 10. Disponível em http://www.theverge.com/2015/2/2/7962179/raspberry-pi-windows-10.

[9] Dean Walsh. An introduction to z-wave home automation, 2014. Disponível em http://electronician.hubpages.com/hub/An-Introduction-to-Z-Wave-Home-Automation.

[10] Tecnologia x10. Disponível em http://www.x10.com/x10-basics.html.

[11] Giuseppe Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*, 18(3):535–547, 2000.

[12] Reto Meier. *Professional Android 4 application development*. John Wiley & Sons, 2012.

[13] ios: A visual history. Disponível em http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad.

[14] Ten of the best cross-platform mobile development tools for enterprises. Disponível em http://appindex.com/blog/ten-best-cross-platform-development-mobile-enterprises/.

[15] Mobithinking. Mobile applications: native v web apps – what are the pros and cons?, 2015. Disponível em http://mobiforge.com/news-comment/mobile-applications-native-v-web-apps-what-are-pros-and-cons?

[16] Sarah Allen. Pro smartphone cross-platform:iphone, blackberry, windows mobile and android development and distribution. 2010.

[17] Build cross-platform apps with xamarin in visual studio. Disponível em https://msdn.microsoft.com/en-us/library/dn879698(v=vs.140).aspx.

[18] Xamarin reference. Disponível em http://developer.xamarin.com/.

[19] Cross-platform user interfaces with xamarin.forms. Disponível em http://developer.xamarin.com/guides/cross-platform/xamarin-forms/.

[20] Rolf H Weber e Romana Weber. *Internet of Things*. Springer, 2010.

[21] Eben Upton e Gareth Halfacree. *Raspberry Pi user guide*. John Wiley & Sons, 2014.

[22] Gustavo Alonso, Fabio Casati, Harumi Kuno, e Vijay Machiraju. *Web services*. Springer, 2004.

[23] Alex Rodriguez. Restful web services: The basics. Relatório técnico, IBM, Novembro 2008.

[24] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, e Donald F Ferguson. *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR, 2005.

[25] Craig McMurtry, Marc Mercuri, Nigel Watling, e Matt Winkler. *Windows Communication Foundation Unleashed (Wcf)(Unleashed)*. Sams, 2007.

[26] Janaka Deepakumara, Howard M Heys, e R Venkatesan. Fpga implementation of md5 hash algorithm. Em *Electrical and Computer Engineering, 2001. Canadian Conference on*, volume 2, 2001.