Instituto Superior de
Engenharia de Coimbra

Departamento

de Engenharia Eletrotécnica

Conceção e Desenvolvimento de Aplicações Móveis

Estágios na Empresa Network Electronics 2000 Consultadoria e Redes Informáticas, lda

Relatório de Estágio apresentado para a obtenção do grau de
Mestre em Automação e Comunicações em Sistemas de Energia

Autor

Hugo Santos

Orientadores

Doutor Inácio Fonseca

Supervisor na Empresa

Engenheiro Eduardo Jorge Oliveira Emílio

Instituição

Instituto Superior de Engenharia de Coimbra

Coimbra, Setembro, 2014

Agradecimentos

Quero deixar, aqui, expresso o meu reconhecimento e agradecimento:

- Ao meu orientador de estágio, Professor Doutor Inácio Fonseca, pela disponibilidade, persistência, sabedoria e conselhos preciosos e por me encorajar ao longo do percurso do estágio.

- Ao supervisor da empresa Network Electronics 2000 – Consultadoria e Redes Informáticas, Lda, Eng. Eduardo Jorge Oliveira Emílio, por me ter possibilitado este estágio.

- Ao *staff* da organização do Congresso internacional *Maintenance Performance Measurement and Management Conference* 2014, Professor Doutor José Torres Farinha, Rúben Oliveira e Pedro Neves, por todo o apoio e suporte no desenvolvimento do *site* do congresso.

- Por fim, mas não menos importante quero agradecer à minha família e amigos pelo apoio prestado durante este longo percurso.

Resumo

Desde o início da revolução digital nos meados dos anos 70 com o advento dos primeiros computadores pessoais, passando pelo desenvolvimento do *World Wide Web* e dos primeiros dispositivos de comunicação móvel da década de 1990, até aos dispositivos *smartphones* e *tablets*, dos dias de hoje, que a dependência tecnológica da nossa sociedade tem vindo a aumentar a um ritmo cada vez mais acelerado, facilitando e automatizando muitas das tarefas rotineiras do dia-a-dia.

Com esta atual dependência tecnológica, omnipresente em todos os sectores na nossa sociedade é fundamental mais do que nunca que existam cada vez mais empresas de consultoria, desenvolvimento e venda de soluções informáticas, capazes de lidar com as necessidades deste mercado em constante mudança e que atualmente se foca cada vez mais no desenvolvimento de tecnologias móveis e *web*. Estes factos, por sua vez incentivam estas empresas a investir na formação dos seus quadros técnicos de forma a possibilitar o desenvolvimento de novas e melhores soluções digitais para os diversos sectores de mercado.

O presente documento pretende descrever o trabalho realizado durante o período de estágio de mestrado realizado na empresa Network Electronics 2000 – Consultadoria e Redes Informáticas, Lda.

Este período de estágio foi dedicado ao desenvolvimento de diversas soluções nos campos da tecnologia móvel e web. Entre estas soluções podemos destacar alguns sistemas de captura de imagem e assinaturas, tanto para um dispositivo *PDA* industrial com o sistema operativo *Windows Embedded Handheld 6.5*, como para um típico smartphone *Android*. Muito do trabalho desenvolvido para o *PDA* industrial consistiu ainda no desenvolvimento de bibliotecas em *C#* para interface de baixo nível às chamadas de sistema e/ou *hardware* específico do equipamento. Estas bibliotecas permitem o desenvolvimento em ambiente de simulação (sem o *PDA* – simulam o hardware) ou em ambiente real (com o *PDA*). Desta forma a tarefa de testes pode ser efetuada com/sem o equipamento, o que se revelou uma mais-valia. Durante este período também foi desenvolvido um sistema de catalogação de material para salas e laboratórios, ao qual foi dado o nome Listagem de Material – para ambos os equipamentos – *PDA* industrial e sistema *Android*. Este sistema permite a interligação a uma base de dados alojada num servidor remoto com diversos dispositivos móveis diferentes. Por fim, foram desenvolvidos e tidos em manutenção dois *sites web*: o primeiro corresponde ao evento internacional *Maintenance Performance Measurement and Management Conference 2014*; enquanto o segundo diz respeito ao evento Encontro Nacional de Engenharia e Gestão Industrial de 2014, ambos os eventos realizados no passado mês de Setembro.

Todos estes desafios foram realizados com sucesso, estando à data deste relatório em curso o estudo da possível adaptação de algumas destas aplicações na empresa. Acrescenta-se ainda que quanto à aplicação Listagem de Material foi possível realizar um artigo académico publicado no livro *Proceedings of Maintenance Performance Measurement and Management (MPMM) Conference 2014.*

Palavras-chave: Aplicações em dispositivos móveis; Sistemas e serviços *web*; Ferramentas de desenvolvimento de aplicações móveis.

Abstract

Since the beginning of the digital revolution in the middle 70s, with the advent of the first personal computers, through the development of the World Wide Web and the first mobile communication devices of the 1990s, up to Smartphones and Tablets devices of today, that the technological dependence of our society has been increasing fast, facilitating and automating many of the routine tasks of day-to-day.

With this current technological dependence, ubiquitous in all sectors in our society, is more critical than ever that there are increasingly consulting firms, development and sale of software solutions capable of dealing with the needs of this ever-changing market and currently focuses increasingly on the development of mobile and web technologies. These facts, in turn, encourage these companies to invest in training their technical staff in order to enable the development of new and better digital solutions for various market sectors.

This document is intended to describe the work done during the internship Masters held on the company Network Electronics 2000 - Computer and Network Consulting, Lda.

This internship was connected to the development of various solutions in the fields of mobile and web technology. Within these solutions one can highlight some systems of image and signatures capture, for both industrial PDA device with Windows Embedded Handheld 6.5 operating system, as well for a typical Android smartphone. Much of the work developed for the industrial PDA was the development of libraries in C # for low-level interface to system calls and/or specific hardware equipment. These libraries allow the developing within a simulated environment (without the PDA - simulate the hardware) or in real environment (with the PDA). Therefore allowing the testing task to be carried out with / without the equipment, which proved to be an asset. During this period was also developed a material cataloging system for rooms and laboratories which has been called Material Listing for both equipment - industrial PDA and Android system. This system allows linkage between a database hosted on a remote server with several different mobile devices. Finally, two websites were developed and taken into service: the first corresponds to the international event Maintenance Performance Measurement and Management Conference 2014; while the second relates to the event National Meeting of Industrial Engineering and Management, 2014, both events held last September.

All these challenges were successful, being studyed at the date of this report the possibility to adapt some of these applications to the company. In addiction, regarding to the application Material Listing was made an academic article published in the book Proceedings of Maintenance Performance Measurement and Management (MPMM) 2014 Conference.

Keywords: Applications for mobile devices; Systems and web services; Tools for developing mobile applications.

Índice

Agradecimentos I
Resumo III
Abstract V
Índice VII
Índice de Figuras XI
Simbologia e Abreviaturas XV
Capítulo 1 – Introdução ao Trabalho 1
1.1 Motivação e Contexto 1
1.2 Objetivo 2
1.3 Organização do Documento 2
Capítulo 2 – Estado da Arte 5
2.1 Introdução 5
2.2 Formar a Ideia 7
2.3 Escolha do sistema operativo e dispositivos alvos 8
2.4 A ferramenta de desenvolvimento usada 8
Capítulo 3 – Ferramentas de Desenvolvimento para Aplicações Móveis 9
3.1 Enquadramento 9
3.2 Porquê Desenvolver Para Sistemas Operativos Diferentes 9
3.3 Aplicações Nativas, *Web* e Híbridas 10
3.3.1 Aplicações Nativas 11
3.3.2 Aplicações *Web* 13
3.3.3 Aplicações Híbridas 14
3.3.4 Resumo 15
3.4 Principais Sistemas Operativos Móveis 16
3.4.1 *Windows Mobile* e *Windows Phone* 16
3.4.2 *Android* 21
3.4.3 *iOS* 22
3.5 Ferramentas de Desenvolvimento Disponíveis no Mercado 24
3.5.1 *Visual Studio* 24
3.5.2 *Xcode* 24
3.5.3 *Eclipse* 25
3.5.4 *Embarcadero Delphi Studio* 25
3.5.5 *Xamarin Studio* 26
3.5.6 *Rhomobile Suite* 27
3.5.7 *PhoneGap* (*Cordova*) 28
3.5.8 *Icenium* 29
3.5.9 Resumo 29

Capítulo 4 – Características dos Dispositivos Móveis Utilizados e Respetivas Bibliotecas 31
4.1 *Datalogic Lynx* 31
4.1.1 Empresa *Datalogic* 31
4.1.2 Dispositivo *Lynx* 32
4.2 *Sumsung Fresh* 33
4.2.1 Grupo *Samsung* 33
4.2.2 Dispositivo *Galaxy Fresh* 34
Capítulo 5 – Aplicações Desenvolvidas Para o Dispositivo *Datalogic* 37
5.1 Aplicação *Send Mail* 37
5.1.1 Enquadramento 37
5.1.2 Funcionamento 37
5.1.2.1 Classe *SmtpClient*. 39
5.1.2.2 Classe *Poutlook* 40
5.1.3 Resultados 41
5.2 Recolher Uma Foto 41
5.2.1 Enquadramento 41
5.2.2 Protótipo Aplicação *CameraCall* 42
5.2.3 Classe *ShortCamara* 46
5.3 Recolha de Assinatura 48
5.3.1 Enquadramento 48
5.3.2 Descrição de Funcionamento 48
5.3.3 Resultados 50
5.4 Acesso e Gestão de uma Base da Dados Interna. 50
5.4.1 Enquadramento 50
5.4.2 Base de Dados Local *SQLite* 51
Capítulo 6 – Aplicação de Listagem de Material – *ANDROID E WEH 6.5* 53
6.1 Enquadramento Desta Aplicação 53
6.2 Objectivos da aplicação Listagem de Materiais 54
6.3 Princípio de Funcionamento do Sistema. 54
6.4 Base de Dados 59
6.4.1 *MySQL* 59
6.4.2 Implementação da Base de Dados 59
6.5 Arquitetura *REST* 60
6.6 Desenvolvimento da Aplicação no Dispositivo *Lynx* 62
6.6.1 Implementação no dispositivo *Lynx* 62
6.6.2 Funcionamento do *GUI* 62
6.7 Desenvolvimento da Aplicação num Dispositivo *Google Android* 64
6.7.1 Implementação em *Google Android* 64

6.7.2 Funcionamento do *GUI* 64
6.8 *Paper* 70
Capítulo 7 – Desenvolvimento de *Sites Web* 71
7.1 Enquadramento 71
7.2 Eventos Associados aos *Sites* 71
7.2.1 MPMM 2014 71
7.2.2 ENEGI 2014 71
7.3 Ferramentas Utilizadas 72
7.3.1 *Dreamweaver* 72
7.3.2 *Xampp* 73
7.4 Composição dos *Sites* 74
7.4.1 Constituição de uma Página em Geral 74
7.4.2 Estrutura dos *Sites* 75
7.4.3 Implementação de uma Galeria de Imagens 77
7.4.4 Implementação do Formulário de Inscrição do *Site* MPMM 2014 78
7.4.5 Implementação de Envio de *Emails* em Ambos os *Sites* 83
Capítulo 8 – Conclusão 85
8.1 Conclusão 85
8.2 Trabalhos futuros 86
Referências Bibliográficas 87
Anexo A – Principais Classes Desenvolvidas para o Uso do *Datalogic Lynx* A-1
A.1 Classe *Hardware* A-1
A.1.1 Enquadramento A-1
A.1.2 Princípio de Funções do Sistema de Emulação de *Hardware* A-2
A.1.3 Função *RebootPhone* A-3
A.1.4 Função *setPhonePowerOnWPin* e *setPhonePINNumber* A-4
A.1.5 Função *GetMyIP*() A-5
A.1.6 *Has, Get, Set* A-5
A.1.7 Função *setKey2ScannerState* A-6
A.1.8 Classe *PowerNeedFromOs* A-7
A.1.9 Classe *SistemTimeLib* A-7
A.1.10 Classe *SendPing* A-8
A.1.11 Classe *SimLCK* A-8
A.1.12 Classe *GPRSConnection* A-9
A.1.13 Classes *NetworkAdapters* e *Infoadapters* A-9
A.1.14 Classe *PhoneClasse* A-9
A.2 Classe *ThreadStatus* A-10
A.3 Classe *GUIControl* A-11

A.4 Classe *SignatureControl* A-11
A.4.1 Descrição das Principais Funções A-12
A.4.2 Eventos da Classe A-12
A.5 Biblioteca *WinINet* A-13
A.5.1 Enquadramento A-13
A.5.2 Composição da Classe *WinINet*. A-13
Anexo B – Serviço *Web*: Gerador de Etiquetas B-1
B.1 Conversor de *Array* de Dados em Imagem B-1
B.1.1 Enquadramento da Aplicação B-1
B.1.2 Biblioteca *GD* B-1
B.1.3 Diagrama do Sistema B-2
B.1.4 Função *ImagConverter* B-2
B.2 Conversor de Imagem para *PDF* B-4
B.2.1 Enquadramento da Aplicação B-4
B.2.2 Biblioteca *FPDF* B-4
B.2.3 Diagrama do Sistema B-5
B.2.4 Função *ImagConvertToPDF* B-5
Anexo C – *Paper* Realizado: C-1
Anexo D – Bibliotecas utilizadas nos Dispositivos Móveis D-1
D.1 Bibliotecas utilizadas no Dispositivo *Lynx* D-1
D.1.1 *Datalogic C/C++, dotNET, JAVA SDKs* para *Windows Embedded Handheld 6.5* . D-1
D.1.2 *OpenNetCF* D-1
D.1.3 *ProcessCE* ou *Terranova Api* D-2
D.2 Bibliotecas utilizadas no Dispositivo *Android* D-3
D.2.1 Biblioteca *Zxing* D-3

Índice de Figuras

Figura 1 – Estudos de Mercado [1]. 10
Figura 2 – Diagrama de uma chamada de uma API Nativa [12]. 11
Figura 3 – Processo de compilação nativa [12]. 11
Figura 4 – Quadro de relações entre os principais Sistemas Operativos [12]. 12
Figura 5 – *Wikipedia home page* e *Dremel mobile home page* [12]. 13
Figura 6 – Diagrama de uma chamada de uma API *web* [12]. 14
Figura 7 – Diagrama de uma chamada de uma API híbrida [12]. 15
Figura 8 – Quadro de comparação [12] 15
Figura 9 – Ambiente de trabalho do *Windows CE*. 16
Figura 10 – *Dell Axim x51* da *HTC*. 17
Figura 11 – Logotipo do *Windows Mobile*. 17
Figura 12 – Ambiente de trabalho do *Windows Mobile* 5. 18
Figura 13 – Ambiente de trabalho do *Windows Mobile 6.* 18
Figura 14 – Ambiente de trabalho do *Windows Mobile 6.5.* 19
Figura 15 – *Interface* do *Windows Phone* 7. 20
Figura 16 – *Interface* do *Windows Phone* 8. 20
Figura 17 – Logotipo da *Android*. 21
Figura 18 – *HTC Dream (T-Mobile G1).* 21
Figura 19 – *Nexus 7.* 21
Figura 20 – Diagrama da arquitetura do *Android* [21]. 22
Figura 21 – Logotipo do *iOS* da *Apple*. 22
Figura 22 – Smartphone *iPhone*. 23
Figura 23 – *Interface* típico do *iOS*. 23
Figura 24 – Diagrama da arquitetura do *Rhomobile* [41]. 27
Figura 25 – Tabela de acesso a *hardware* do *PhoneGap*. 28
Figura 26 – Logotipo da *Datalogic*. 31
Figura 27 – Alguns modelos de *PDAs* da *Datalogic.* 31
Figura 28 – *Datalogic Lynx.* 32
Figura 29 – Representação do funcionamento do *A-GPS.* 32
Figura 30 – Códigos de barras 1D e 2D. 33
Figura 31 – Logotipo da *Samsung.* 33
Figura 32 – Alguns modelos *Galaxy* da *Samsung.* 34
Figura 33 – *Samsung Galaxy Fresh*. 34
Figura 34 – Esquema simplificado do funcionamento do *Send Mail.* 37
Figura 35 – GUI da aplicação *Send Mail.* 38
Figura 36 – Confirmação de envio de correio eletrónico. 38
Figura 37 – Insucesso no envio do correio eletrónico. 38
Figura 38 – Diagrama do envio de correio eletrónico através da classe *SmtpClient.* 39
Figura 39 – Diagrama do Envio de correio eletrónico Via *Outlook.* 40
Figura 40 – Caixa de correio do *Hotmail*, ilustrando o recebimento do *email* enviado pela aplicação. 41
Figura 41 – Menu inicial da aplicação. 43
Figura 42 – *Display* de recolha de foto. 43
Figura 43 – Menu do *display* de recolha de foto. 43
Figura 44 – Recolha da foto bem-sucedida 44
Figura 45 – Diagrama do acesso normal à câmara. 45
Figura 46 – *Display* recolha de foto da class *ShortCamera.* 46
Figura 47 – Menu da recolha de fotos. 46

Figura 48 – Recolha de imagem completa. 46
Figura 49 – Diagrama da class *ShortCamera*. 47
Figura 50 – Aspeto gráfico da classe *SignatureControl.* 48
Figura 51 – Diagrama do funcionamento desta aplicação. 49
Figura 52 – *GUI* da aplicação de captura de assinatura. 49
Figura 53 – Imagem original. 50
Figura 54 – Imagem assinada. 50
Figura 55 – Diagrama de Acesso a BD Interna. 51
Figura 56 – Aplicação de teste *SQLite*. 52
Figura 57 – Diagrama de interação entre o lado do servidor e o lado móvel. 54
Figura 58 – Diagrama do lado do servidor. 54
Figura 59 – Esquema básico de funcionamento da aplicação. 55
Figura 60 – Fluxograma da fase 1. 56
Figura 61 – Fluxograma da fase 2. 57
Figura 62 – Fluxograma da fase 3. 58
Figura 63 – Representação gráfica da base de dados do sistema. 60
Figura 64 – Interação do *REST Server* com os diversos componentes *Web.* 61
Figura 65 – Menu lista de equipamento em dispositivo *Lynx.* 62
Figura 66 – *GUI* de introdução de códigos de barras dispositivo *Lynx.* 63
Figura 67 – *GUI* da captura de imagens e captura de assinaturas. 63
Figura 68 – Menu inicial da aplicação para *Android*. 65
Figura 69 – Menu opções. 65
Figura 70 – Menu *IP Options*. 66
Figura 71 – Menu *User Options*. 66
Figura 72 – Listas de equipamento, escolha da área de trabalho. 67
Figura 73 – Listas de equipamento, escolhas completas. 67
Figura 74 – Menu *ListView* de equipamento em *Android.* 67
Figura 75 – *GUI* de aquisição de código de barras. 68
Figura 76 – *Barcode Scanner* de *Zxing*. 68
Figura 77 – *GUI* Recolha, (*Tab* “Fotos”). 68
Figura 78 – *GUI* Recolha, (Tab “Sign”). 68
Figura 79 – *GUI* Recolha, (*Tab* “Notas”). 69
Figura 80 – *GUI* de captura de assinaturas. 69
Figura 81 – Logotipo do evento MPMM 2014. 71
Figura 82 – Logotipo ENEGI Coimbra 2014. 72
Figura 83 – Logotipo da *Dreamweaver.* 73
Figura 84 – Logotipo do *Adobe System.* 73
Figura 85 – A esquerda o *site* MPMM 2014 e a direita o *site* ENEGI Coimbra 2014. 74
Figura 86 – Diagrama da composição de uma página. 75
Figura 87 – Diagrama de compilação de uma página. 75
Figura 88 – Diagrama da estrutura do *site* MPMM 2014. 76
Figura 89 – Diagrama da estrutura do *site* ENEGI Coimbra 2014. 76
Figura 90 – *Slideshow* na página inicial do *site* MPMM 2014. 78
Figura 91 – Diagrama de funcionamento do formulário. 79
Figura 92 – Exemplo de um resumo de inscrição. 80
Figura 93 – Mensagem de envio bem-sucedido. 81
Figura 94 – Mensagem em caso de problemas de envio. 81
Figura 95 – Impressão da inscrição. 81
Figura 96 – Formulário MPMM, campo *anti-spam*. 82
Figura 97 – Diagrama do funcionamento *geneImage().* 82
Figura 98 – Diagrama do funcionamento *imagesistem*(). 82
Figura 99 – Parte da tabela (imagem, código). 83
Figura 100 – Interface de envio *emails* no *site* ENEGI Coimbra 2014. 83

Figura 101 – Diagrama do processo de envio de *email* utilizando *phpMailer.* 84
Figura 102 – Diagrama da classe *Hardware*.....A-1
Figura 103 – Diagrama funcionamento da classe *Hardware* em modo emulador.....A-2
Figura 104 – Diagrama funcionamento da classe *Hardware* em modo normal.....A-2
Figura 105 – Representação do pedido de ativação de *hardware* A-3
Figura 106 – Botões de *scan*. A-6
Figura 107 – Exemplo do funcionamento desta classe A-10
Figura 108 – A esquerda com os botões Start e Done a direita sem eles. A-11
Figura 109 – Representação gráfica da class *SignatureControl* A-12
Figura 110 – Partes constituintes da biblioteca *WinINet API*.....A-13
Figura 111 – Diagrama Base da função *ImagConverter.* B-2
Figura 112 – Fluxograma da função *ImagConverter.* B-2
Figura 113 – Diagrama Base da função *imagConvertToPDF.* B-5
Figura 114 – Fluxograma da função *ImagConvertToPDF*. B-5
Figura 115 – Logotipo da Zxing..... D-3

Índice de Tabelas

Tabela 1 – Principais características dos IDEs. 30
Tabela 2 – Formatos suportados pelo Zxing D-3

Simbologia e Abreviaturas

AEGI-UC – "Associação de Engenharia e Gestão Industrial da Universidade de Coimbra"

A-GPS – "*Assisted Global Positioning System*"

API – "*Application Programming Interface*"

APP – "*Application*"

BD – "Base de dados"

CMMS – "*Computerized maintenance management system*"

DEM-FCTUC – "Departamento de Mecânica da Faculdade de Ciências e Tecnologia da Universidade de Coimbra"

DNS – "*Domain Name System*"

EDGE – "*Enhanced Data rates for GSM Evolution*"

ENEGI Coimbra 2014 – "Encontro Nacional de Engenharia e Gestão Industrial de 2014"

FTP – "*File Transfer Protocol*"

GIF – "*Graphics Interchange Format*"

GPS – "*Global* Positioning System"

GPRS – "*General Packet Radio Service*"

GPU – "*Graphics Processing Unit*"

GSM – "*Global System for Mobile Communications*"

GUI – "*Graphical user interface*"

HTTP – "*Hypertext Transfer Protocol*"

HTTPs – "*Hypertext Transfer Protocol Secure*"

IDC – "*International Data Corporation*"

IDE – "*Integrated Development Environment*"

IP – "*Internet Protocol"*

ISV – "*Independent software vendor*"

JPEG – "*Joint Photographic Experts Group*"

JSON – "*JavaScript Object Notation*"

LCD – "*liquid-crystal display*"

LTS – "*Long Term Support"*

MP – "*megapixel*"

MPMM 2014 – *"Maintenance Performance Measurement and Management Conference 2014"*

OS – *"Operative system"*

PDA – *"Personal digital assistant"*

PDF – *"Portable document format"*

PIN – *"Personal identification number"*

PNG – *"Portable Network Graphics"*

POP – *"Post Office Protocol"*

QVGA –*"Quarter Video Graphics Array"*

RAM – "*Random-access memory*"

REST – "*Representational state transfer*"

SDK – "*Software Development Kit*"

SIM – "*Subscriber identity module*"

SQL – "*Structured Query Language"*

SMTP – *"Simple Mail Transfer Protocol*"

VoIP – "*Voice over IP*"

TFT-LCD – "*Thin film transistor liquid crystal display*"

WEH 6.5 – "*Windows Embedded Handheld 6.5"*

WLAN – "*Wireless Local Area Network*"

WM – "*Windows Mobile*"

WP – "*Windows Phone*"

Capítulo 1 – Introdução ao Trabalho

1.1 Motivação e Contexto

No decorrer destes últimos dez anos, o desenvolvimento tecnológico no campo dos dispositivos móveis, têm causado mudanças profundas não só na indústria e nos processos industrias, como também na sociedade em geral. Estas mudanças originaram por sua vez:

- Novos mercados;
- Novas ideias;
- Novas tecnologias;
- Novas formas de pensar e de agir;
- Novas soluções para problemas existentes.

Todas estas alterações incentivaram o mercado de dispositivos móveis a crescer de uma forma nunca antes vista. Segundo alguns estudos este ritmo de crescimento deverá continuar a aumentar [1]. Mas nem sempre foi assim, desde o aparecimento do primeiro *PDA* comercial nos anos 80, (*Psion* da *Organizer* em 1984 [2], [3]), passando pelo primeiro *smartphone* Simon da *IBM* e da *Bell South* em 1993 [4], até ao aparecimento dos primeiros *smartphones* comerciais, estes dispositivos por norma eram caros, pouco práticos e desenvolvidos tendo em mente o mercado industrial ou empresarial. Somente com a introdução de dispositivos como *iPhone* da *Apple* e o *Nexus* do *Google* é que estes começaram a ser desenvolvidos em massa, o que por sua vez os tornou mais baratos, incentivando assim o mercado consumidor a aderir em força aos mesmos. Por outro lado, o desenvolvimento de novas técnicas de miniaturização levou à compactação de muitos dos componentes destes dispositivos, o que os tornou mais pequenos, rápidos e práticos.

Atualmente um dispositivo smartphone não é só um dispositivo para efetuar chamadas telefónicas. Estes aparelhos vêm acompanhados com inúmeras funcionalidades, que lhes conferem um estatuto similar a um "canivete suíço" moderno. Sistemas de *hardware* como *GPS*, giroscópio, sensores de luz ou proximidade, ou até câmaras de filmar, são componentes garantidos na maioria destes dispositivos. Aliado a isto, estes dispositivos possuem também uma grande capacidade de conexão a múltiplas redes de comunicação como por exemplo a rede telefónica, *web,* entre outras. Todas estas características permitem assim, um leque diversificado de soluções aplicáveis a estes dispositivos, que podem ser tão banais como formas de entretenimento, jogos ou aplicações audiovisuais, ou até sistemas integrados de manutenção e sistemas de gestão industrial, permitindo, a existência de um mercado diversificado e com ampla área de influência.

Com o advento de novas e melhores ferramentas de desenvolvimento, novas técnicas de desenvolvimento de aplicações para os vários sistemas operativos e formas de comercializar *software*, foi possível tornar o desenvolvimento de aplicações num negócio muito rentável e economicamente viável para muitas pequenas e médias empresas de desenvolvimento tecnológico. Isto incentivou por sua vez a expansão do mercado de desenvolvimento de aplicações e soluções nos últimos anos. Segundo os estudos realizados pelo analista *Gigaom's Mark Mulligan* cofundador da *Midia Consulting UK* e do analista *David Card*, este mercado em 2013 originou aproximadamente 17.5 bilhões de Euros em receitas e empregou acerca de 1.8 milhões de pessoas somente na Europa e segundo eles é esperado que estes números continuem a crescer ao longo dos próximos anos [5]. Resumidamente, o mercado de desenvolvimento e consultoria de soluções informáticas atualmente oferece:

- Um crescimento económico, que não deverá abrandar nos próximos anos;

- Um leque de nichos de mercados para explorar;
- Uma panóplia de ferramentas para facilitar o processo de desenvolvimento;
- Uma considerável capacidade de absorção de pessoal qualificado.

Por outras palavras, é atualmente um mercado extremamente interessante e potencial para desenvolver uma carreira profissional.

1.2 Objetivo

O principal objetivo deste estágio foi a obtenção do primeiro contato com o mundo do trabalho, no campo do desenvolvimento de soluções informáticas, especificamente no campo de dispositivos móveis e aplicações *web.*

Inicialmente foi necessário elaborar um estudo a respeito das várias ferramentas de desenvolvimento e sistemas operativos atualmente disponíveis no mercado.

Este estudo foi seguindo pelo desenvolvimento de diversas aplicações para o dispositivo *PDA Lynx* da *Datalogic* em linguagem *C#*, com o intuito de adquirir as diversas competências necessárias para desenvolver aplicações móveis para dispositivos com o sistema operativo *Windows Mobile*.

Em seguida foi necessário desenvolver um sistema servidor-cliente de Listagem de Material. Este sistema requereu a utilização de varios conhecimentos adquiridos ao longo do mestrado, mais especificamente no campo de desenvolvimento de Base de Dados e programação em linguagem *PHP* de forma a desenvolver o lado do servidor, enquanto no lado do cliente foi necessário desenvolver duas aplicações, uma em *C#* para um dispositivo *Windows Mobile* e a outra em *Java* para um dispositivo *Android, (*consultar capítulo 6 para mais informação a respeito deste sistema). Por outro lado este trabalho, também possibilitou o uso de vários conhecimentos no campo da manutenção adquiridos na licenciatura de engenharia electrotécnica, visto este sistema implementar várias facetas do que se designa por *software* de *computerized maintenance management system.*

Paralelamente a este trabalho, foi necessário desenvolver e manter em funcionamento dois *sites web*. Trabalho este que contribuiu para a aquisição de competências e conhecimento em algumas das linguagem *web* mais comuns, como por exemplo *HTML*, *JavaScript* e *CSS.*

Finalmente, o processo de estágio foi concluído com a elaboração deste documento.

1.3 Organização do Documento

Esta monografia está dividida em oito capítulos, tal como, seguidamente, se sintetiza:

- O primeiro capítulo contém a introdução à monografia, a contextualização, os objetivos e a organização do documento;

- O segundo capítulo, contém o Estado da Arte relativamente ao mercado de desenvolvimento de *software* e soluções móveis, referindo empresas e aplicações relevantes nesta área;

- No terceiro capítulo faz-se a apresentação de várias ferramentas de desenvolvimento de aplicações móveis e sistemas operativos móveis mais relevantes no mercado atual;

- O quarto capítulo apresenta a descrição técnica dos dispositivos móveis e principais bibliotecas de desenvolvimento utilizadas durante o período de estágio;

- O quinto capítulo descreve o funcionamento de múltiplas aplicações desenvolvidas para o dispositivo *Datalogic* em linguagem *C#*;

- O sexto capítulo contém a apresentação e descrição do funcionamento da aplicação de Listagem de Material;

- O sétimo capítulo descreve o desenvolvimento de dois *Sites Web*;

- O oitavo capítulo apresenta as conclusões;

- O final da monografia é constituído pelas referências bibliográficas e os vários anexos mencionados ao longo dos capítulos precedentes.

Capítulo 2 – Estado da Arte

2.1 Introdução

É impossível negar que nos últimos anos têm surgido cada vez mais empresas de desenvolvimento de aplicações móveis a nível mundial. Este facto pode ser atribuído a vários fatores, como a introdução de novas plataformas *online* de comercialização de *software* como a *Google Shop*, ou *Apple® Store*, entre outras, a facilidade de acesso ao conhecimento, documentação e informação através da rede global de *Internet*, ou até a quantidade monumental de potenciais clientes que compõem o mercado, ou então a introdução de novos e melhores programas de desenvolvimento como *Visual Studio* da *Microsoft*, ou *Android Studio* do *Google*. Independentemente das razões, este mercado tem tido um forte crescimento, faturando vários biliões de euros todos os anos. Por outro lado, este aumento progressivo do mercado obriga à constante inovação e busca de novas soluções por parte das empresas, de forma a manterem-se relevantes num mercado extremamente competitivo e em constante mudança.

Devido ao estado atual do mercado de aplicações, muitas das empresas têm-se dedicado ao desenvolvimento de ferramentas utilitárias, ideais tanto para o uso profissional como amador. Exemplos:

- *TeamViewer*[1] → Esta empresa alemã dedica-se exclusivamente ao desenvolvimento da *TeamViewer App.* Trata-se de uma aplicação de acesso remoto a dispositivos em funcionamento como *desktops*, *tablet* e *smartphones*, conferências *web* e transferência de ficheiros entre máquinas. Esta aplicação foi portada para todos os sistemas operativos relevantes no mercado atual e é ideal tanto para implementação de soluções e serviços de manutenção remota, como para controlo remoto de dispositivos;

- *TomTom*[2] → Esta empresa holandesa, fundada em 2001 por *Peter-Frans Pauwels* é a líder mundial em sistemas e soluções de navegação via *GPS*, tendo desenvolvido e comercializado vários dispositivos de navegação. Desde 2009 que esta, tem oferecido soluções de navegação tanto para dispositivos *Android* como *iPhone* e atualmente encontra-se numa parceria com a *Apple* com o objetivo de competir com o *Google Map*;

- *Whatsapp, Inc.*[3] → Esta empresa norte americana, foi fundada em 2009 por *Brian Acton* e *Jan Koum*, ambos antigos funcionários da empresas *Yahoo!,* com o intuito de desenvolver e comercializar a aplicação *Whatsapp*. Esta aplicação multiplataformas (*Android, Windows Mobile, iOS, Symbian* e *BlackBerry OS*) de comunicação instantânea para smartphones, similar a aplicação *Windows Live Messenger*, permite o envio de mensagens, imagens e outros formatos de ficheiros em geral. Posteriormente a *Whatsapp, Inc* foi adquirida pelo *Facebook, Inc.* em fevereiro 2014 [6].

[1] *TeamViewer home page*: https://www.teamviewer.com/pt/download/mobile.aspx (18/08/2014).
[2] *TomTom home page*: http://www.tomtom.com/pt_pt/ (18/08/2014).
[3] *Whatsapp home page*: http://www.whatsapp.com/ (18/08/2014).

Por sua vez, outras empresas preferiram dedicar-se ao desenvolvimento de soluções móveis exclusivamente para profissionais, dedicadas à gestão industrial e comercial, também conhecidas por *Computerized maintenance management system*. Empresas como:

- *Maintenance Assistant Inc.*[4]→ Esta empresa foi fundada em 2008 e tem-se concentrado no desenvolvimento de soluções CMMS ou também conhecidas por *computerized maintenance management system*, utilizando o melhor que a tecnologia tem para oferecer. Uma das suas aplicações mais conhecida, o *MA CMMS Mobile*[5], é compatível com dispositivos *Android* e *iOS* em combinação com um serviço *cloud*. Esta ferramenta foi desenhada tendo em mente o uso de dispositivos *smartphones*, aproveitando-se assim de todas as vantagens que estes conferem e desempenha todas as funções comuns de um CMMS de uma forma elegante, permitindo entre várias aspetos criar, consultar e completar ordens de trabalho, planear tarefas, adicionar notas e informação a respeito das tarefas, entre outras. oferecendo assim uma fácil e portátil ferramenta de manutenção;

- *Wasp Barcode Technology*[6] → Esta empresa americana foi fundada em 1994 e tem-se dedicado ao longo dos anos à produção de soluções para pequenas e médias empresas, sendo o seu lema, "*Productivity Solutions for Small Businesses*". Para isso, especializou-se no desenvolvimento de *software* e sistemas de código de barras. Muitas destas soluções, como por exemplo, o *Inventory Software & Inventory Management Systems*, ou o *Inbound Mail Package Tracking System,* ou até o *School Asset Tracking Software for Education*, entre outros, requerem o uso de dispositivos PDAs que suportem *Windows® Embedded Handheld*, *Windows Mobile* e *Windows CE,* com o intuito de recolher e enviar informação para o sistema central localizado num computador fixo a correr *software* proprietário desta empresa. Mais recentemente, sistemas como *Asset Tracking Systems*[7] ou o *School Asset Tracking Software for Education*[8] também permitem o uso de dispositivos *smartphones*, (*Android* e *iOS*), como forma de consultar informação do computador central;

- *Yardi Systems*[9] → Esta empresa californiana é uma das grandes líderes de mercado de *design*, desenvolvimento e suporte de *software* do sector industrial [7], tendo sido fundada em 1984 e atualmente emprega aproximadamente 3000 empregados espalhado por trinta escritórios em todo o mundo. Neste últimos anos, esta companhia tem alargado muitas das suas soluções para o ambiente móvel, oferecendo várias soluções no campo da gestão, do comércio, da manutenção, etc. Destas soluções, a mais relevante talvez seja a *Yardi Maintenance Mobile*[10]. Esta aplicação de CMMS existe disponível tanto em Android, como *iOS* e *BlackBerry OS* e permite a elaboração de muitas das tarefas esperadas deste tipo de sistemas como criação, consulta e fecho de ordens de trabalho, gestão de inventário, requisitos de compras, entre outros.

[4] *Maintenance Assistant Inc home page*: http://www.maintenanceassistant.com/ (19/08/2014).
[5] *MA CMMS Mobile* App page:http://www.maintenanceassistant.com/cmms/mobile-cmms/ (19/08/2014).
[6] *Wasp Barcode Technology home page*: http://www.waspbarcode.com/ (20/08/2014).
[7] *Asset Tracking Systems page*: http://www.waspbarcode.com/asset-tracking (20/08/2014).
[8] *School Asset Tracking page: http://www.waspbarcode.com/asset-tracking-schools* (20/08/2014).
[9] *Yardi Systems home page*: http://www.yardi.com/ (19/08/2014).
[10] *Yardi Maintenance Mobile page: http://www.yardi.com/products/yardi-maintenance* (19/08/2014).

Algumas outras optaram por desenvolver *software* de entretenimento numa tentativa de obter o sucesso financeiro de aplicações, como:

- *Angry Birds*[11] → Lançado Dezembro de 2009 para *iOS*, *Android* e outros, pela empresa Filandesa, *Rovio Entertainment*. Este *franchising* mundialmente conhecido é aclamado como sendo um dos jogos para dispositivos móveis mais bem-sucedido do mundo. Esta aplicação vendeu aproximadamente 12 milhões de cópias em menos de um ano através da loja online *Apple® Store* [8]*;*

- *Candy Crash*[12] → Lançado em Novembro de 2012 para a maioria das plataformas *smartphones*, pela empresa de *King Digital Entertainment plc*. Esta aplicação é considerada um dos melhores jogos de puzzles do mercado, tendo sido considerado pela *Eurogamer* o melhor do ano 2013 [9] e, segundo algumas fontes, esta aplicação gerou em julho de 2013, no mercado americano aproximadamente 474.000€/dia, através dos seus modelos de *microtransaction*[13] [10];

As descrições acima referidas, tratam-se de alguns exemplos do que se pode esperar neste vasto mercado.

Mas antes de obter uma aplicação funcional é necessário dar vários passos de desenvolvimento. Este processo de desenvolvimento pode ser resumido nas palavras do tenista americano e activista em diversas causas sociais, Arthur Ashe, "*Success is a journey, not a destination*." ou, por outras palavras, o que importa é a jornada e não o destino[14]. Podemos então separar o processo de desenvolvimento em três passos consecutivos os quais no fim resultam na aplicação desenvolvida. Estes passos são:

1) Formar a ideia;
2) Escolha do sistema operativo e dispositivos alvos;
3) A ferramenta de desenvolvimento usada;

2.2 Formar a Ideia

Sendo este o primeiro passo, é provavelmente um dos pontos mais importantes na fase de desenvolvimento, pois como é possível conceber uma aplicação sem saber o que pretendemos dele? Então, é necessário imaginar em primeiro lugar uma forma genérica do funcionamento da aplicação alvo. Idealmente devem-se representar os objectivos e o funcionamentodo da aplicação numa forma simples mas concreta, utilizando eventualmente para isso um diagrama de blocos como forma de representar a aplicação. Este passo pode ser observado no processo de desenvolvimento da diversas aplicações desenvolvidas ao longo deste estágio, capítulo 5 e 6.

[11] *Angry Birds home page*: https://www.angrybirds.com/ (18/08/2014).
[12] *Candy Crash home page*: http://www.candycrushsaga.com/ (18/08/2014).
[13] Conceito: Pequenos pagamentos com dinheiro real, que resultam em um bônus no jogo, como um item extra, caráter, etc.
[14] Sitação de Arthur Ashe: http://www.brainyquote.com/quotes/quotes/a/arthurashe371528.html

2.3 Escolha do sistema operativo e dispositivos alvos

A segunda fase consiste na escolha dos dispostivos alvo e seus correspondentes sistemas operativos. Desta forma, é possível escolher qual ou quais são os melhores dispositivos móveis para a aplicação em mente. Para isso é necessário equacionar varios factores, entre eles as vantagens e desvantagens de desenvolver para múltiplos sistemas operativos em simultâneo, como também é necessário ter um conhecimento geral dos diversos sistemas operativos atuais e suas subsequentes vantagens e desvantagens, tal como é útil reconhecer os potênciais do desenvolvimento de aplicações nativas, híbridas e *webs*.

Esta informação encontra-se muito mais descrita nos capítulos 3.3, 3.4 e 4 deste documento.

2.4 A ferramenta de desenvolvimento usada

Por fim, após escolher todos os parâmentros nos pontos acima referidos, só resta escolher a ferramenta de desenvolvimento ou Integrated Development Environment. Esta tem de ser compatível com as escolhas feitas e é necessário reconhecer que muitas destas ferramentas não são gratuitas e em alguns casos são extremamente caras. Por isso, é importante ter em mente qual é a melhor ferramenta para o trabalho a realizar e assim dar inicio ao desenvolvimento do código da aplicação.

É possível consultar mais informação a respeito deste tema no capítulo 3.5 deste relatório.

Capítulo 3 – Ferramentas de Desenvolvimento para Aplicações Móveis

3.1 Enquadramento

Existem neste momento no mercado inúmeras ferramentas de desenvolvimento de aplicações móveis e cada uma delas possui virtudes e defeitos. Por isso a escolha aleatória de uma ferramenta é um ato arriscado, visto que no pior dos casos pode resultar em incontáveis prejuízos.

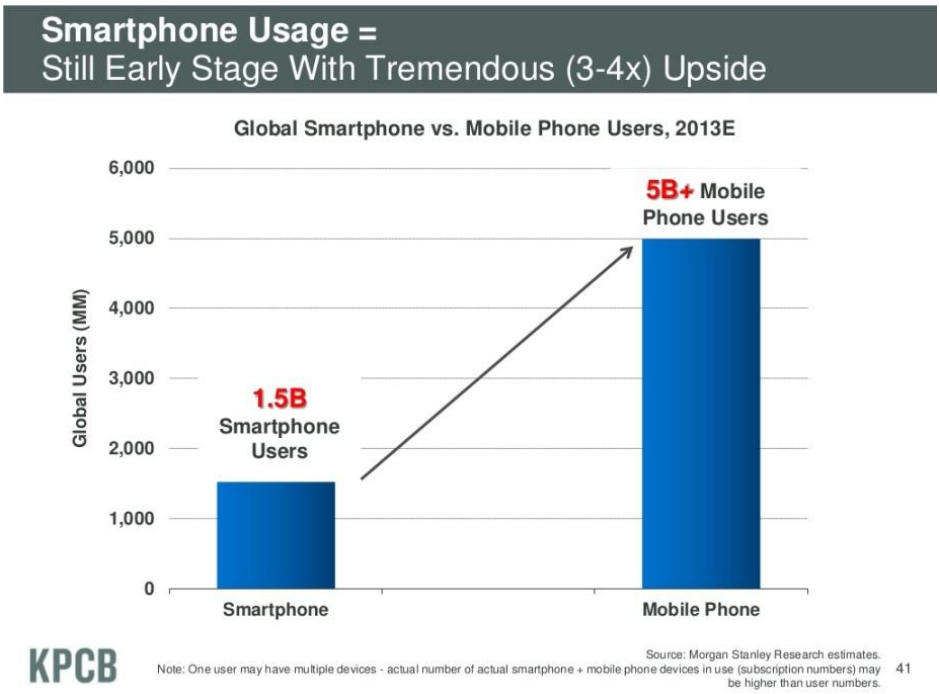
Antes de escolher um IDE (*Integrated Development Environment*) é sempre importante saber a resposta às seguintes perguntas:

- Permite somente desenvolver aplicações só para um sistema operativo, ou para múltiplos?
- Já tem incorporado uma boa ferramenta de *debugging*, ou é necessário adquirir *software* de terceiros para o fazer?
- De que forma compila as aplicações? São compiladas de forma nativa ou utiliza outros tipos de compilação?
- Que linguagem de programação usa?
- Quanto custa?

Este capítulo pretende responder a algumas destas perguntas de forma a ser possível fazer uma escolha adequada e informada. Pretende-se também efetuar uma análise de algumas das ferramentas IDE existentes atualmente no mercado.

3.2 Porquê Desenvolver Para Sistemas Operativos Diferentes

É difícil prever qual será o sistema operativo dominante no futuro próximo. Segundo alguns estudos realizados pela IDC (*International Data Corporation*), é esperado em 2016 que os sistemas *Android* do *Google* dominem 53% do mercado, tendo como competidores diretos o *iOS* da *Apple* e o *Windows Phone* da *Microsoft*, ambos com 19% de mercado respetivamente [11], (ver Figura 1). Por outro lado, existem estudos que indicam que em 2013 existiam 5 biliões de utilizadores de telemóveis dos quais apenas 1.5 biliões eram dono de um *smartphone* [1]. Estes dados indicam a existência de um grande potencial de crescimento neste mercado.



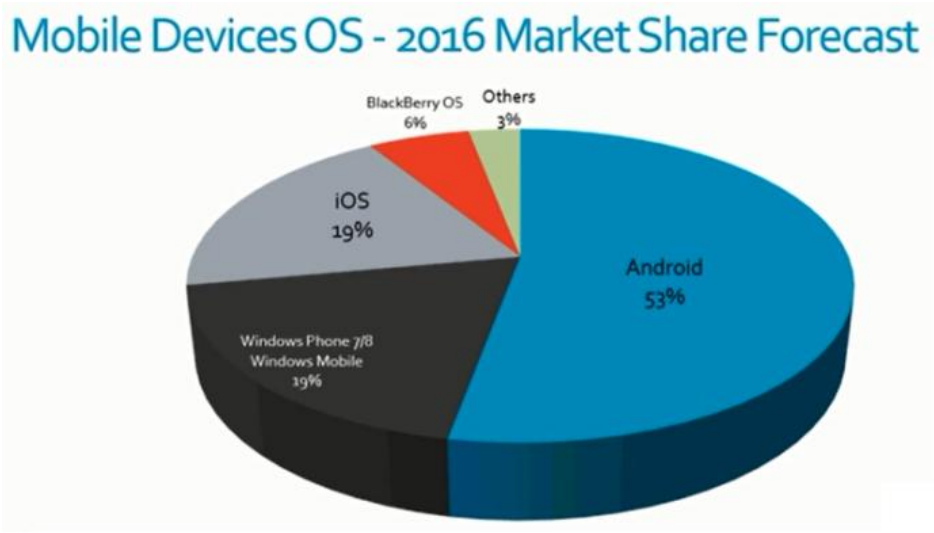


Figura 1 – Estudos de Mercado [1].

Sendo este mercado tão incerto, a capacidade de difundir uma aplicação para sistemas operativos diferentes, pode ser um fator decisivo entre o sucesso e o insucesso de uma empresa.

3.3 Aplicações Nativas, *Web* e Híbridas

Durante vários anos só foi possível desenvolver aplicações de tipo nativo para os vários dispositivos móveis, mas com o aparecimento de novos protocolos *web*, como o *HTML5* e o *JavaScript,* começaram a surgir aplicações do tipo *web* e mais recentemente aplicações do tipo híbridas. Cada um destes tipos acarreta vantagens e desvantagens no processo de

desenvolvimento, os quais qualquer empresa de desenvolvimento de aplicações necessita de conhecer adequadamente, antes de iniciar o processo de desenvolvimento.

3.3.1 Aplicações Nativas

Uma aplicação nativa é geralmente caracterizada por três características principais. A primeira é a existência de um executável, ou também conhecida por *binary* ou binário, este tem de ser descarregado pelo utilizador ou em alguns casos pelo sistema operativo e instalado no dipositivo móvel. A segunda característica é o modo de arranque da aplicação, esta deve ser chamada pelo sistema operativo sem necessitar do auxílio de aplicação intermédias. Finalmente, a aplicação deve ser capaz de aceder ao API's do sistema operativo, tendo assim acesso a todos os recursos disponíveis do telemóvel, (ver Figura 2).

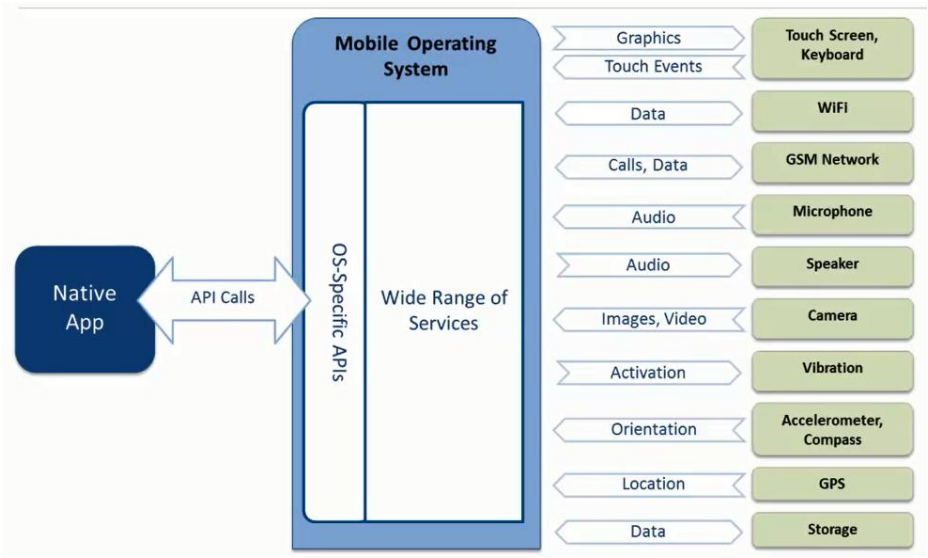


Figura 2 – Diagrama de uma chamada de uma API Nativa [12].

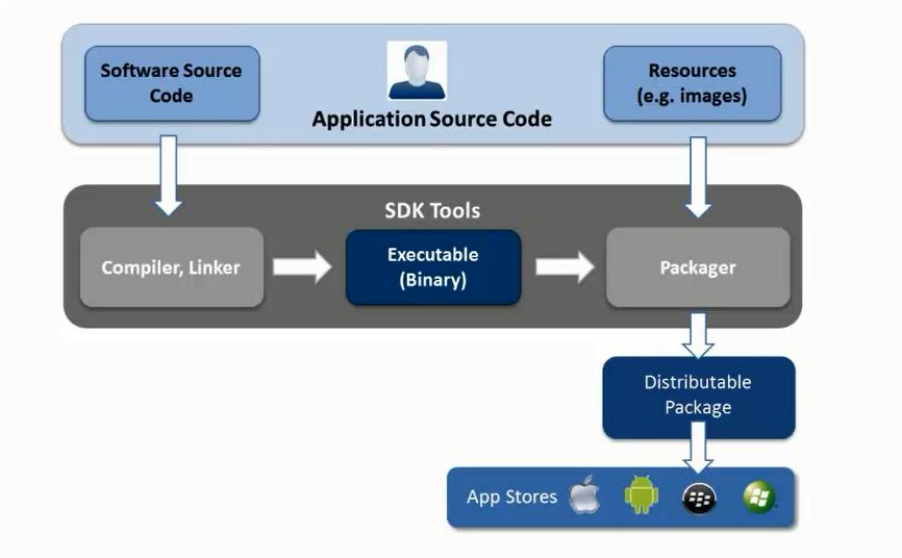


Figura 3 – Processo de compilação nativa [12].

O processo de desenvolvimento de uma aplicação nativa requer geralmente vários passos antes de se obter o aplicativo final, (Figura 3). Primeiramente, é necessário desenvolver o código fonte. Este deve ser escrito com a linguagem apropriada para o sistema operativo móvel em questão, por exemplo *Java* para *Android*, *C#* para *Windows Mobile*, etc.., (Figura 4) de seguida, com o auxílio das ferramentas fornecidas pelo fornecedor do sistema operativo, também conhecidas por SDK ou *Software Development Kit*, é necessário compilar o código fonte, de forma a gerar um executável, o qual por fim é empacotado juntamente com outros recursos da aplicação, para obter um pacote de distribuição.

Languages	Obj-C, C, C++	Java (Some C, C++)	Java	C#, VB.NET, etc
Tools	Xcode	Android SDK	BB Java Eclipse Plug-In	Visual Studio, Windows Phone Dev Tools
Executable Files	.app	.apk	.cod	.xap
Application Stores	Apple iTunes	Android Market	BlackBerry App World	Windows Phone Market

Figura 4 – Quadro de relações entre os principais Sistemas Operativos [12].

As aplicações nativas trazem algumas desvantagens. Como foi referido anteriormente, não só é necessário programar o código fonte na linguagem específica, como é necessário utilizar o SDK correspondente ao sistema operativo pretendido. Isto pode resultar em severos custos, caso uma empresa pretenda desenvolver aplicações para plataformas distintas.

O primeiro grande problema reside na impossibilidade de reutilização de código, logo é necessário desenvolver códigos fonte diferentes para cada OS. Em segundo lugar, obriga ao conhecimento de linguagem de programação dedicados a cada sistema operativo, isto resulta em custos acrescidos com o pessoal e por sua vez aumenta o custo de desenvolvimento. Finalmente, existem ainda custos associados às licenças de desenvolvimento, pois quase todos os fornecedores de sistemas operativos cobram algum tipo de comissão.

Por outro lado, com este tipo de aplicações é garantido o máximo desempenho do dispositivo. Isto é ideal para aplicações como jogos ou processamento de vídeo que requerem altos níveis de processamento gráfico e de dados. Outra grande vantagem deste tipo de aplicações é o acesso livre a todas as chamadas da API do OS, isto permite o acesso a todo o *hardware* do dispositivo, tal como o acesso aos serviços de nível alto como *Browsers*, *GUI Tool Kit*, realizar chamadas, entre outros [12] [13, pp. 9-10], permitindo assim o aumento do potencial de desenvolvimento da aplicação.

3.3.2 Aplicações *Web*

A evolução das ferramentas e protocolos *web*, nomeadamente o do *HTML 5, CSS3* e do *JavaScript*, deu origem a opções poderosas no campo de desenvolvimento *web*. Onde antes se encontravam páginas *web* tradicionais, passaram agora a estar páginas *web* capazes de se adaptar as diversas caracteristicas físicas dos dispositivos móveis, possibilitando assim o uso dos diversos componentes de *hardware* como forma de obter uma melhor experiência de utilização por parte do utilizador. Temos, como exemplo disso, as páginas do *Wikipedia* [14] e da *Dremel* [15], uma subsidiária da *Bosch* (Figura 5), as quais estão tão bem concebidas que aparentam ser à primeira vista aplicações nativas ao invés de páginas web no ponto de vista do utilizador.

Figura 5 – *Wikipedia home page* e *Dremel mobile home page* [12].

Existem atualmente duas formas opostas de desenvolver aplicações *web* para dispositivos móveis. Numa das formas, temos o desenvolvimento de páginas *web* para dispositivos móveis. Estas são normalmente caracterizadas por serem visualizadas diretamente através do *browser*, terem um interface de navegação standard e serem executadas no lado do servidor, o que obriga a uma ligação à *internet*. A outra forma consiste no desenvolvimento de aplicações *web*, estas são caracterizadas por serem idênticas às aplicações nativas em termos de navegação e aspeto, conseguirem aceder a alguns dos recursos de *software* e *hardware* do dispositivo móvel e serem executadas no lado do dispositivo, não requerendo assim ligação à rede *web* permanente para desempenhar algumas das suas tarefas.

Independentemente da forma como a aplicação *web* é desenvolvida, o *browser* continuará a ser o elemento comum e é este que realiza a ligação entre a componente web da aplicação e o dispositivo. O *browser* é uma aplicação garantida em todos os dispositivos móveis modernos, desde que estes suportem as linguagens de programação *web*, isto permite desenvolver aplicações universais para todas as plataformas, o que resulta diretamente numa diminuição do tempo de desenvolvimento. Por outro lado, o *browser* tem acesso limitado aos restantes recursos do dispositivo, o que torna difícil ou até impossível o acesso a certos componentes de *hardware*. Outra desvantagem de utilizar o *browser* como meio intermediário é o baixo nível de otimização, eficiência e rapidez da aplicação, pois esta não está a utilizar diretamente os recursos do dispositivo mas sim o *browser (*Figura 6*)*.

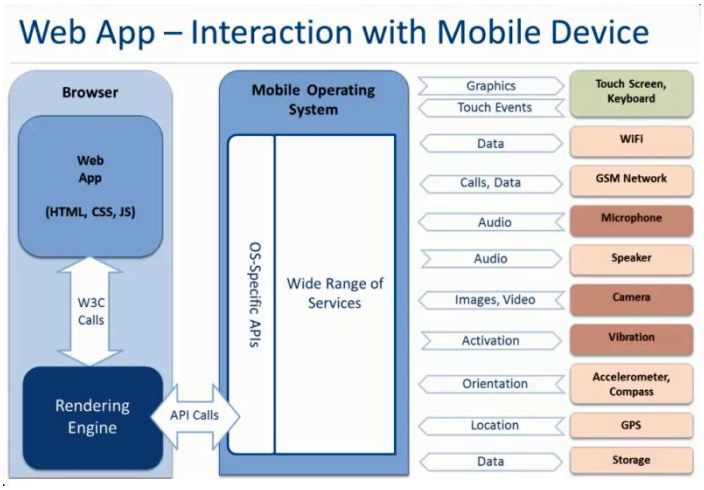


Figura 6 – Diagrama de uma chamada de uma API *web* [12].

É esperado que muitas destas desvantagens sejam ultrapassadas ou minimizadas com o contínuo desenvolvimento do *HTML5*, [13, pp. 10-11], mas de momento são mais recomendadas para aplicações que requeiram médio ou alto nível de conectividade à rede.

3.3.3 Aplicações Híbridas

Uma aplicação híbrida, como o nome sugere, é o resultado da fusão entre uma aplicação nativa e uma aplicação *web*. Este tipo de aplicação tem uma camada nativa, a qual comunica diretamente com o sistema operativo, obtendo assim a mesma velocidade, eficiência e controlo sobre os pedidos API, como qualquer outra aplicação nativa. Esta contém também uma camada *web* universal para todos os sistemas operativos. Isto permite o reaproveitamento de parte do código, tal como uma maior conectividade com outras aplicações *online*, como base de dados, serviços *cloud*, etc. (Figura 7).

Estas aplicações basicamente usufruem de todas as vantagens de uma aplicação nativa e *web*, sem quase nenhumas das suas desvantagens, o que resulta num tipo de aplicação com bastante potencial de desenvolvimento e fácil implementação em múltiplos sistemas operativos [12] [13, pp. 11-12].

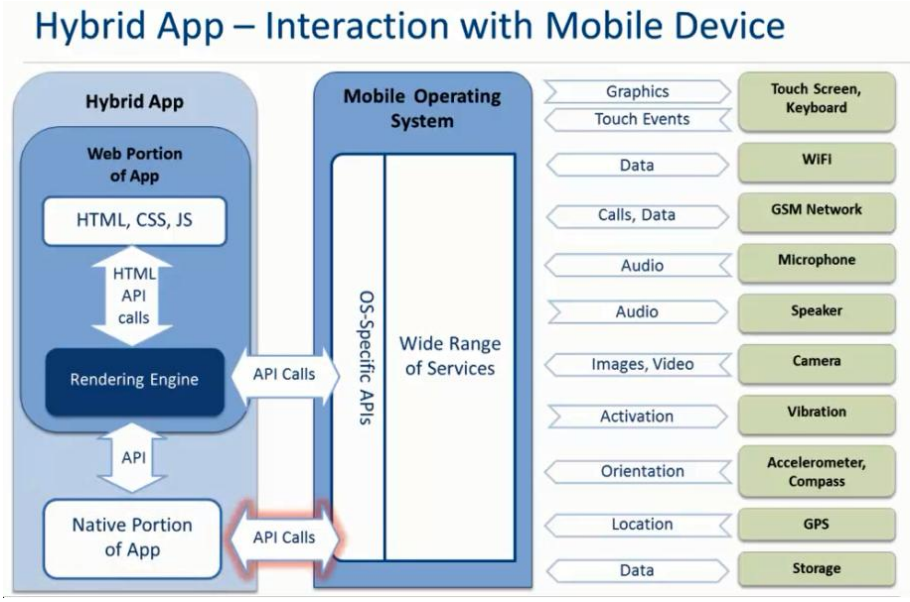


Figura 7 – Diagrama de uma chamada de uma API híbrida [12].

3.3.4 Resumo

Como é possível observar na Figura 8, nenhum dos três tipos de aplicações é claramente melhor ou pior que os outros, simplesmente são diferentes e têm os seus próprios campos de utilização ideais. De uma forma muito simples, podemos concluir que as aplicações nativas são óptimas para aplicações que requeiram um desempenho máximo do dispositivo móvel, enquanto as aplicações *web,* são excelentes para o desenvolvimento de multiplataformas ou que requeiram uma ligação quase contínua a internet. Por fim, as aplicações híbridas tentam obter o melhor das soluções nativas e web.

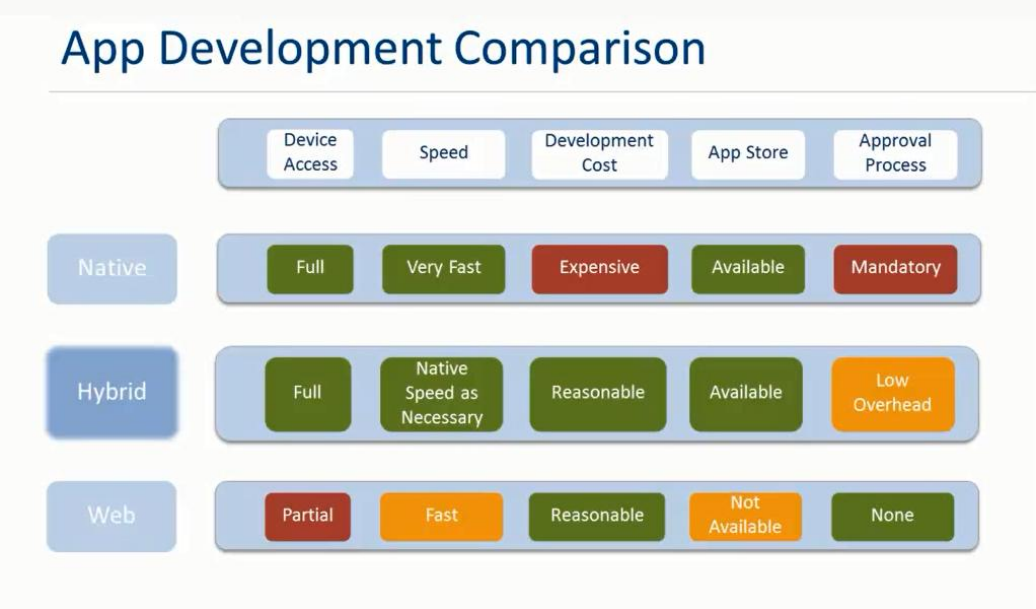


Figura 8 – Quadro de comparação [12]

Portanto a escolha do tipo de aplicação a usar, recai sobre o responsável pelo desenvolvimento, o qual deve decidir qual dos tipos resultará num desenvolvimento mais eficiente e eficaz.

3.4 Principais Sistemas Operativos Móveis

Desde dos primeiros sistemas operativos OS móveis nos anos 90 como o *Windows CE* e o *Palm OS* até aos mais recentes como, *Android* e *iOS 8*, a tecnologia dos sistemas operativos móveis tem vindo a sofrer uma contínua e acelerada evolução, tornando-os cada vez mais rápidos, potentes e com melhores *interfaces* gráficos. Tendo isso em conta, vejamos alguns dos sistemas operativos mais relevantes neste mercado.

3.4.1 *Windows Mobile* e *Windows Phone*

Desde o lançamento do *Windows CE* em 1996, que a *Microsoft* tem participado ativamente no mercado de dispositivos móveis, tendo lançado vários sistemas operativos ao longo destes últimos 18 anos. Destes, os mais relevantes foram o *Windows CE, Windows Mobile 5, Windows Mobile 6, Windows Phone 7* e *Windows Phone 8*.

O *Windows CE,* também conhecido por *Windows Embedded CE* foi desenvolvido para atender ao mercado de sistemas embebidos. Teve como base de desenvolvimento o *Windows 95* e suportava processadores *Intel x86* e *ARM.* A importância fundamental deste sistema operativo reside no facto de ter servido de base para quase todos os sistemas operativos móveis lançados posteriormente pela *Microsoft* (Figura 9).

Figura 9 – Ambiente de trabalho do *Windows CE*.

Em 2005 a *Microsoft* lança o *Windows Mobile 5* (ou WM5) em conjunto com o *smartphone* da *HTC Dell Axim x51* (Figura 10). Este sistema operativo ao contrário do seu antecessor *Windows CE* foi concebido especificamente para dispositivos móveis, como smartphone ou PDAs, no intuito de captar o mercado empresarial.

Figura 10 – *Dell Axim x51* da *HTC*.

O WM5 (Figura 11 e Figura 12), introduziu o suporte a várias funcionalidades anteriormente inexistentes ou pouco usadas em dispositivos móveis, entre elas, suporte de *GPS*, *Bluetooth* e *DirectDraw*. Este sistema operativo foi também um dos primeiros a utilizar *Persistent storage capability* ou capacidade de memória persistente, aumentando assim em quase 50% o tempo médio de duração de carga na bateria, isto porque anteriormente os dados eram armazenados na memória RAM a qual requeria energia para os manter.

Figura 11 – Logotipo do *Windows Mobile*.

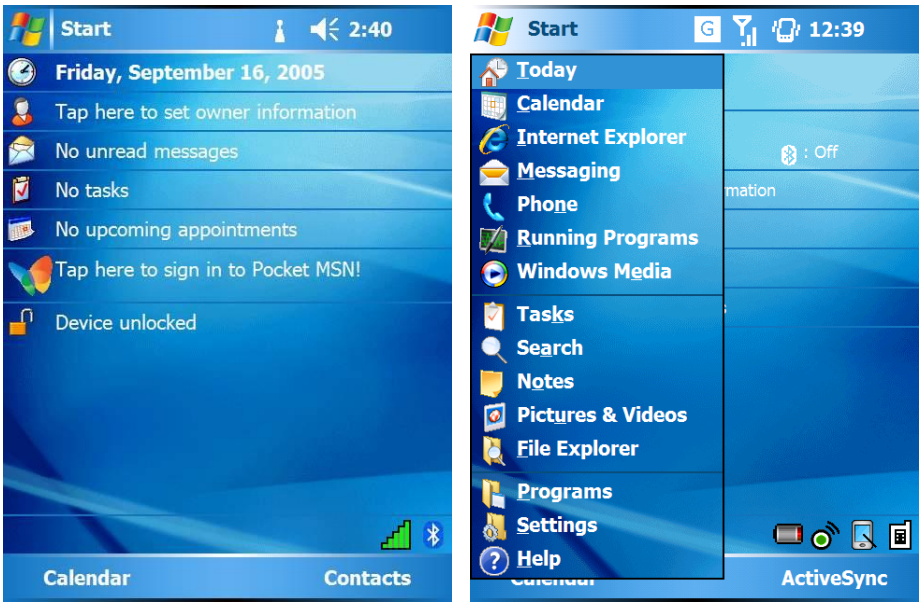


Figura 12 – Ambiente de trabalho do *Windows Mobile* 5.

Dois anos após a introdução do WM5 no mercado, foi lançado o *Windows Mobile 6*, oferecendo três versões diferentes na sua compra, (Figura 13):

- WM6 *Standard* – dedicado a *smartphones* sem ecrã táctil;
- WM6 *Classic* – dedicado a *Pocket PC* sem capacidade rádio celularer;
- WM6 *Profissional* – dedicado a *Pocket PC* com capacidade rádio celularer.

Este sistema operativo veio com várias melhorias. O aumento da resolução de ecrã foi uma das mais notáveis, passando a suportar resoluções até 800x480. O aumento da segurança foi outra grande melhoria, com a introdução *Storage Card Encryption* o qual apagava todas as chaves de encriptação aquando de um *cold-booted*. O WM6 suportava também o envio de *emails* via *Outlook* e o suporte de todos os documentos do *Office 2007* (pptx, docx, xlsx).

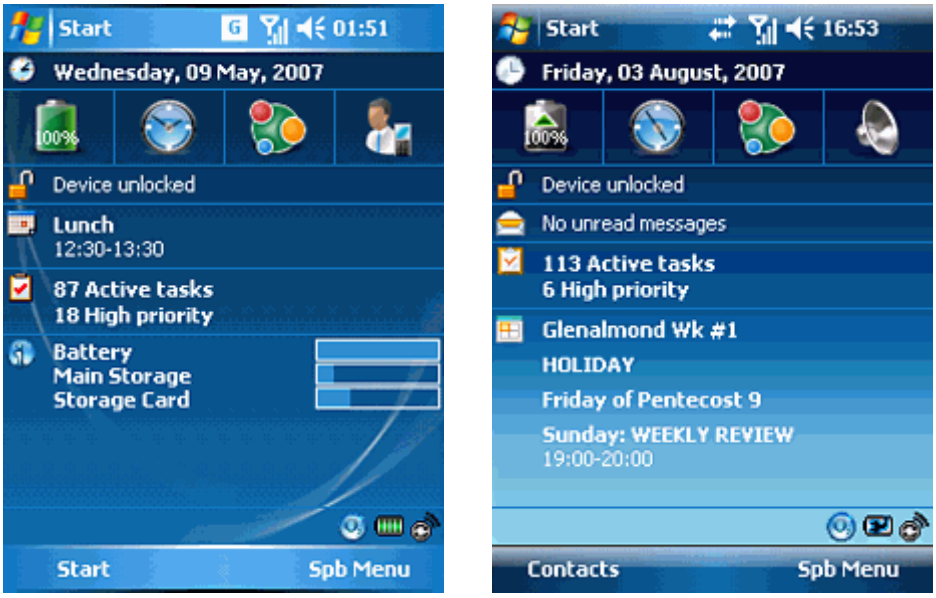


Figura 13 – Ambiente de trabalho do *Windows Mobile 6.*

Em 2010 foi lançado uma nova atualização para o WM6, passando este a ser o *Windows Mobile 6.5* (Figura 14). Esta atualização trouxe uma melhoria significativa no *interface* gráfico,

com o intuito de melhorar a experiência táctil do utilizador, oferecendo melhores e maiores ícones gráficos em vez de botões texto. O desempenho do sistema operativo também foi melhorado, tornando-o mais rápido e fiável. Este OS introduziu o suporte ao *A-GPS*, tal como vários serviços *Cloud* como *Windows Marketplace* e era compatível com o *Internet Explorer 6* e o *Office Mobile 2010*.

Figura 14 – Ambiente de trabalho do *Windows Mobile 6.5.*

Nesse mesmo ano a *Microsoft* lançou o *Windows Phone 7* (WP 7). Mesmo sendo este um descendente direto do *Windows Mobile*, por questões de incompatibilidade com dispositivos antigos, tal como por *marketing*, a *Microsoft* optou por lhe chamar *Windows Phone 7* em vez de *Windows Mobile 7*. Segundo uma entrevista feita ao gerente sênior da *Microsoft Mobile Developer Experience, Larry Lieberman*, este afirmou que não houve tempo, nem recursos suficientes para rever a questão de compatibilidade com dispositivos anteriores, *"If we'd had more time and resources, we may have been able to do something in terms of backward compatibility."* [16].

Ao contrário dos seus antecessores, o alvo principal do WP7 era o mercado de venda ao público. Para isso a *Microsoft* optou por um sistema operativo, interativo, com suporte completo de ecrã táctil e fácil acesso a aplicações de terceiros, (Figura 15).

O *Kernel* do WP7 era baseado na versão *Windows Embedded Compact 7*, requerendo no mínimo um processador 800MHz e 256MB de RAM para operar devidamente e era capaz de desligar algumas das suas funcionalidades mais intensivas em termos de recursos caso o dispositivo não fosse capaz de as suportar.

Com o lançamento posterior de algumas atualizações o WP7, passou a suportar também alguns protocolos standards da *web* mais recentes, permitindo assim compatibilidade com *Internet Explorer* 9, tal como o suporte *multi-tasking* para aplicações de terceiros, acesso ao *Windows Live SkyDrive* e suporte a algumas aplicações desenvolvidas para sistemas *Windows NT*.

Figura 15 – *Interface* do *Windows Phone* 7.

Em 2012 a *Microsoft* lançou o seu mais recente sistema operativo móvel o *Windows Phone 8* (WP8).

Figura 16 – *Interface* do *Windows Phone* 8.

Este OS é esteticamente muito similar ao seu antecessor WP7, continuando a ênfase no aspeto gráfico e alto nível de interação táctil como forma de atrair o público em geral. Um bom exemplo disso é o *design* dos botões, geralmente são grandes, coloridos e dinâmicos, o que os torna mais apelativos, ao toque e à vista, como é possível ver na Figura 16.

Mas as semelhanças terminam aqui. O WP8 ao contrário de todos os seus antecessores utiliza um *kernel* baseado no *Windows NT* em vez de uma arquitetura baseada no *Windows CE*, o que resulta numa inerente incompatibilidade com dispositivos que suportem o WP7. Esta mudança foi motivada pelo desejo de tornar o WP8 mais compatível com o sistema operativo *Windows 8*, facilitando assim a conversão de software deste sistema operativos para o WP8.

O WP8 suporta também uma resolução maior de ecrã que os seus antecessores, indo até 1920X1080, tal como possui já funções internas de suporte a *VoIP* (voz por IP), aumentando assim ainda mais a sua versatilidade.

3.4.2 *Android*

Figura 17 – Logotipo da *Android*.

Android ou como é também conhecido *Google Android* (Figura 17), é o sistema operativo móvel da *Google*, lançado comercialmente em 2008 em conjunto com o *smartphone HTC Dream* (*T-Mobile G1*) da *HTC* (Figura 18), posteriormente ficou associado às séries de *smartphones Nexus* desenhada pela *Google* e fabricada em parceria com outras empresas, como a *LG Electronics* ou a *Samsung Electronics* (Figura 19). Inicialmente desenvolvido na empresa *Android Inc., fundada em 2003 por Andy Rubin (*co-fundador *da Danger)* [17]*, Rich Miner* (co-fundador da *Wildfire Communications, Inc.*) [18], *Nick Sears* [19] e *Chris White* [20]. A partir de 2005 esta empresa foi adquirida pelo *Google*.

Figura 18 – *HTC Dream (T-Mobile G1).*

Figura 19 – *Nexus 7.*

Este sistema operativo em termos de *hardware* é atualmente compatível com arquiteturas de processadores *ARM* V7 32 bits, MIPS e x86. Necessita no mínimo de 340MB de memória RAM para operar devidamente, mas o recomendado é 512MB. Por fim, é necessário um processador gráfico GPU (*graphics processing unit*) compatível com *OpenGL ES 2.0*.

A *Google* tem vindo continuamente a atualizar o seu sistema operativo, geralmente de 18 em 18 meses, encontrando-se à data, da escrita deste texto, na versão 4.4.3 com o nome de código "*KitKat*".

Através do diagrama da Figura 20, é possível dividir em várias partes a arquitetura de funcionamento do *Android*. O núcleo deste OS é *Linux kernel* (baseado na bifurcação *Linux kernel long-term support (LTS),* atualmente na versão 3.4) e que tanto este como as bibliotecas e o API's do sistema são programados em *C*. Por outro lado aplicações de *software* e de *framework* são programadas em *Java*, que gerem todos os GUI's e interfaces do OS. Por fim, no meio a arquiteturas, existe o *Dalvik virtual machine*. Este compilador converte *Java* em linguagem *bytecode* em tempo real, o que permite a comunicação entre todas as partes.

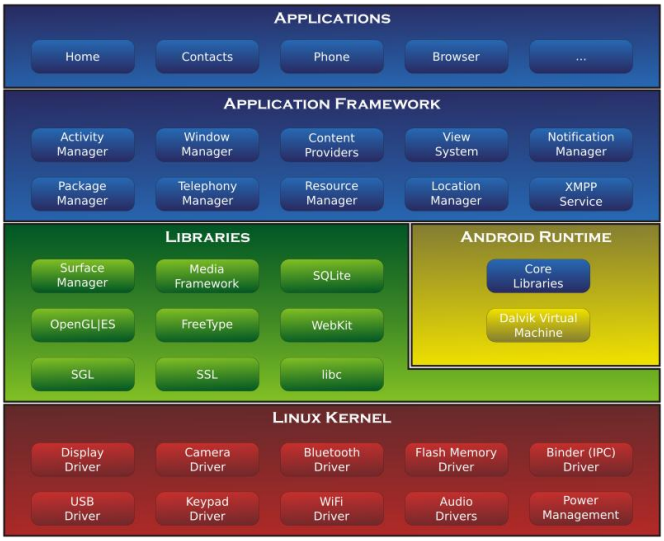


Figura 20 – Diagrama da arquitetura do *Android* [21].

O *Android* está licenciado como sendo código *open-source* (código aberto), o qual é normalmente disponibilizado ao público sempre que a *Google* fornece uma nova atualização. A licença em questão é a *non-copyleft Apache License version 2.0* a qual permite a modificação e redistribuição deste código.

Sendo o líder atual de sistemas operativos móveis, segundo alguns estudos e funcionando em mais de 50% de todos os dispositivos móveis em todo o mundo [22] é esperado que este sistema operativo se torne cada vez mais proeminente ao longo dos anos.

3.4.3 *iOS*

Figura 21 – Logotipo do *iOS* da *Apple*.

Em 2007 a *Apple* iniciou-se no mercado de dispositivos móveis lançando o seu primeiro *smartphone* "*iPhone*" (Figura 22). No coração deste dispositivo encontrava-se o sistema operativo *iPhone OS*, posteriormente renomeado como *iOS* (Figura 21).

Com a ajuda do visionário *Steve Jobs*, a *Apple* foi uma das primeiras empresas a focar-se na venda ao público, tendo vendido um milhão de *iPhones* em menos de um ano. Este facto provavelmente impulsionou muito o mercado de *smartphones*, o qual por sua vez incentivou o aparecimento dos sistemas operativos *Android* do *Google* e *Windows Phone* da *Microsoft* no mercado.

Figura 22 – Smartphone *iPhone*.

O *interface* desde sistema foi desenhado tendo em conta a filosofia de manipulação direta, logo todos os menus e ícones deste OS foram concebidos de forma a terem uma fácil manipulação via toque, Figura 23.

Figura 23 – *Interface* típico do *iOS*.

Este OS utiliza o *kermel XNU*, é um derivado do *kernel Darwin* e as aplicações para este OS são desenvolvidas de forma nativa na linguagem *Objective-C 2.0*, baseada na norma da ISO 9899:1999 ou mais conhecida por C99. Por outro lado, todos os dispositivos *Apple* são de fabrico próprio garantindo assim a máxima compatibilidade com o seu OS.

Desde do seu lançamento até a data, a *Apple* tem lançado periodicamente versões atualizadas do seu sistema operativo, estando atualmente na versão *iOS 8*. Ao contrário do *Google*, o seu código fonte é privado e fechado. Este facto resultou em inúmeras críticas por parte de várias ligas defensoras dos direitos digitais. Apesar disso, a *Apple* dominou 60% do mercado de dispositivos *smartphones* e *tablet* em 2011 e mesmo tendo perdido o lugar de líder para a *Google* continua a ser um dos grandes líderes de sistemas operativos móveis [23].

3.5 Ferramentas de Desenvolvimento Disponíveis no Mercado

3.5.1 *Visual Studio*

O *Visual Studio* foi desenvolvido pela *Microsoft* e lançado comercialmente em 1997. Desde então sofreu inúmeras atualizações e de momento encontra-se na versão*, Visual Studio 2013*. Este permite desenvolver vários tipos de aplicações, desde *interfaces* gráficos, até aplicações *web*, sendo todas elas compatíveis com os diversos sistemas operativos da *Microsoft*, como o *Windows CE*, *Windows Phone*, *Microsoft Windows*, entre outros.

Este IDE por defeito permite usar quatro línguas de programação *C*, *C++, C#* e *F#,* sendo *C#* e *C++* as mais utilizados no desenvolvimento de aplicações. Porém, podem ser adicionadas outras linguagens como *Python*, *JavaScript*, *Ruby*, *CSS* e *HTML*, com a adição de aplicações extra. Outra vantagem deste produto é a extensa documentação e quantidade de tutoriais, fornecidos pela comunidade *online*, tal como pela própria *Microsoft*.

Infelizmente a versão profissional do *Visual Studio*, ao contrário da sua versão para estudante, não é gratuita e tem um custo inicial de 1000 €, e um custo anual de 600 € tornando-a assim numa ferramenta relativamente dispendiosa [24].

3.5.2 *Xcode*

Esta ferramenta da *Apple* foi lançada no ano 2003, com o intuito de desenvolver aplicações para os sistemas operativos da *Apple OS X* e *iOS*. Baseada num IDE anterior, chamado *Project Builder*, o qual fora desenvolvido pela *Next Software*, *Inc*., companhia adquirida pela *Apple* em 1997 [25], mas com um novo *interface*, melhor compilador e uma ferramenta de *debug* incorporada.

O *Xcode* suporta o formato *Mach-O*, o que lhe permite criar *Universal binary,* sendo assim possível desenvolver aplicações nativas para processadores baseados em arquiteturas *Intel* (x86), tal como para processadores *ARM* através do *iOS* SDK. Este IDE suporta também várias linguagens de programação, entre eles *C, C++, Objective-C, Objective-C++, Java, Python*, entre outros.

Devido à popularidade deste IDE, para esta plataforma é comum encontrar muita documentação e tutoriais disponíveis acerca deste. Existe também no mercado um leque variado de livros à disposição, alguns mais dedicados a iniciantes como o "*Programming Objective-C 2.0*" de *Stephen G. Kochan*, ou o "*Head First iPhone Development*" de *Dan Pilone* e outros mais focados para profissionais, como o **"***Professional iPhone and iPad Database Application Programming***"** de *Patrick Alessi*, ou o "*iOS 6 Programming Cookbook*" de *Vandad Nahavandipoor*.

Esta ferramenta pode ser descarregada gratuitamente da página oficial da *Apple*. Por outro lado, é necessário pagar uma licença de desenvolvimento e este custo ronda os 80 €/ano [26].

3.5.3 *Eclipse*

O *Eclipse* foi idealizado na *IBM*. Esta empresa cansada de ter problemas de incompatibilidade entre ferramentas de desenvolvimento de departamentos diferentes, decidiu desenvolver a sua própria ferramenta. Esta deveria ser de desenvolvimento comum, mas capaz de ser adaptada pelas diversas equipas de desenvolvimento, facilitando assim a interação entre projetos [27] [28]. Com este conceito em mente, a *IBM* desenvolveu o *Visual Age* e o *Visual Age for Java*, os quais foram bem-sucedidos no mercado da altura. Infelizmente estes antecessores do *Eclipse* eram produtos proprietários e nunca foram concebidos de forma a suportar sistemas modulares, o que impossibilitou a sua continuação.

Não se deixando desanimar com este problema uma pequena equipa formada por alguns dos engenheiros responsáveis pelo desenvolvimento do *Visual Age*, desenvolveu a primeira versão do *Eclipse* e em novembro de 2001, foi formado um consórcio para impulsionar o desenvolvimento deste como sendo uma ferramenta *open-source*. Este consórcio era composto inicialmente pela *IBM*, *Borland, Merant, QNX Software Systems, Rational Software*, *Red Hat, SuSE, TogetherSoft* e *WebGain*, mas já em 2003 contemplava mais de 80 empresas [29]. Finalmente em janeiro de 2004, foi criada a fundação *Eclipse* e em junho desse mesmo ano foi lançado o *Eclipse 3.0*.

Esta ferramenta é normalmente usada para desenvolver aplicações em *java*, contudo existe vários *Plug-in* que permitem expandir o seu reportório linguístico para *C, C++, COBOL, Fortran, JavaScript, PHP, Python*, entre outros. É possível também aplicar outras ferramentas de desenvolvimento como emuladores ou ferramentas *debug*. Um bom exemplo disso, é a instalação do *Android* SDK que configura o *Eclipse* de forma a poder desenvolver aplicações *Android*, acrescentando a este todas as bibliotecas necessárias, tal como ferramentas de *debug* e de emulação necessárias para o desenvolvimento destas aplicações. O código desenvolvido para *Android* é compilado num binário nativo, obtendo assim eficiência máxima no processador *ARM*, o qual é o coração da maioria dos *smartphones Android* à data da escrita deste texto [30].

Devido à sua natureza *open-source*, este IDE incentivou o aparecimento de uma das maiores comunidades *online* do momento, a qual contribui diariamente com centenas de artigos, conselhos e tutoriais.

3.5.4 *Embarcadero Delphi Studio*

O *Embarcadero Delphi Studio*, originalmente chamado somente por *projeto Delphi* [31], foi desenvolvido pela companhia *Borland*, como o intuito de ser uma ferramenta de desenvolvimento de aplicações para o sistema operativo *Windows*. Esta, resultou num sucesso comercial aquando do seu lançamento em 1995, dando assim origem a vários outros produtos com *interface* similares como o *Borland Pascal* e o *C++ Builder*. Por escolha da *Borland*, todos os seus produtos foram transferidos em 2006 para a companhia *CodeGear* [32], a qual foi posteriormente vendida em 2008 à corporação *Embarcado Technologies*, a atual proprietária do *Delphi Studio*.

A última versão deste produto foi lançada em setembro de 2013, com o nome *Embarcadero Delphi EX5*. Esta ferramenta atualmente é capaz de desenvolver aplicações tanto para *Android* como para *iOS*, tornando-a assim numa plataforma de desenvolvimento multifacetada.

Este IDE utiliza *objective Pascal* como linguagem de desenvolvimento de aplicações o que permite a este gerar código nativo para as arquiteturas de 32 e 64 bits. Esta plataforma oferece também uma extensa lista de ferramentas capazes de personalizar o ambiente gráfico das aplicações, tal como um grande suporte a base de dados locais e globais, dando assim mais valor a este IDE.

Infelizmente este produto acarreta alguns custos, nomeadamente no tempo, o *delphi* requer profissionais que saibam programar em *objective Pascal*, o que torna necessário tempo de aprendizagem, pois não é uma linguagem de programação comum, isto impossibilita também a reutilização de código vindo de outras plataformas. Outra desvantagem deste produto é a incompatibilidade com versões anteriores, devido à introdução de um novo compilador nessa última versão.

Resumidamente esta plataforma de desenvolvimento é uma boa solução para o desenvolvimento de aplicações multiplataformas e ideal para aplicações que requeiram bases de dados, principalmente se forem bases de dados locais. Por outro lado, a impossibilidade de reutilizar código e o custo desta aplicação podem ser fatores muito negativos. A versão Profissional desta aplicação custa cerca de 670 €/ano e a versão *Ultimate* ronda os 2300 €/ano, isto pode provocar um desinteresse nas empresas de pequeno e médio porte [33].

3.5.5 *Xamarin Studio*

Xamarin Studio ou também conhecido por *Xamarin 2.0* foi lançado em fevereiro de 2013 [34] e é a ultima de uma longa linha de ferramentas de desenvolvimento produzida pela empresa *Xamarin*, pertencente ao *Sr. Miguel de Icaza*, um dos criadores do *desktop environment GNOME*, e *Nat Friedman* [35] [36].

Em 2004 a ferramenta de desenvolvimento *Mono* foi lançada pela *Ximian* e posteriormente adquirida pela *Novell* [37]. Esta tinha como objetivo principal permitir o uso do *Framework .NET* da *Microsoft* em sistemas multiplataformas. Tendo isso em mente, foi criado em conjunto com este IDE, um compilador capaz de compilar linguagem *C#,* pois esta é basicamente a línguagem franca do *dotNet*, respeitando as normas Ecma-334 e Ecma-335, tornando-o num compilador completamente gratuito.

Por fim em 2011, a *Novell* formou uma parceria com a *Xamarin*, tendo ficado esta última com uma licença perpétua de todos os produtos *Mono*. Ainda nesse ano foram lançados duas novas versões do *Mono*: o *MonoTouch* para desenvolvimento *iOS* e *Mono for Android* para o OS do *Google*. Em 2013, o *MonoTouch* passou a ser chamado *Xamarin.iOS* e o *Mono* para *Android* por *Xamarin.Android*. Finalmente o *Xamarin Studio* foi lançado com o intuito de unificar o *Xamarin.iOS* e *o Xamarin.Android* num único IDE.

Tal como os seus antecessores, utiliza o *C#* convertendo-o posteriormente para a linguagem nativa do OS, o que permite a reutilização de código para as diversas plataformas e a rápida implementação de API's e códigos *dotNET*. É incluído também nesta ferramenta um sistema de *debugg* e outro de emulação, capazes de simular e testar as aplicações em vários sistemas operativos diferentes. É possível também interligar o *Xamarin Studio* em conjunto com o *Visual Studio* dando a este último as mesmas capacidades de desenvolvimento, permitindo assim o desenvolvimento de aplicações para *android* e *iOS* no IDE da *Microsoft* [38].

Infelizmente este produto não é gratuito, a versão profissional custa aproximadamente 740 €/ano, o que pode revelar-se bastante dispendioso para muitos utilizadores.

3.5.6 *Rhomobile Suite*

Em meados de 2009, a *Motorola* decidiu desenvolver uma plataforma de desenvolvimento multiplataformas que utilizasse *HTML* como base, tendo como filosofia, "*your one application run on All your devices*", por outras palavras, uma mesma aplicação em todos dispositivos. Para desenvolver esta visão, a *Motorola* adquiriu em *2011 Rhomobile Inc*. e no ano seguinte lançou o IDE *Rhomobile Suite* [39].

O *Rhomobile Suite* é uma ferramenta de desenvolvimento multiplataforma gratuita, capaz de desenvolver aplicações para *Android*, *BlackBerry*, *iOS* e *Windows*. É possível dividir este IDE em três componentes básicos *RhoElements, RhoConnect* e *RhoStudio*. O *RhoElements* é a *framework* que gere a interação entre o programa desenvolvido e o dispositivo. O *RhoConnect* por sua vez é o responsável pela ligação da aplicação móvel a uma rede ou serviço *online*. E por fim o *RhoStudio*, permite o desenvolvimento das aplicações [40].

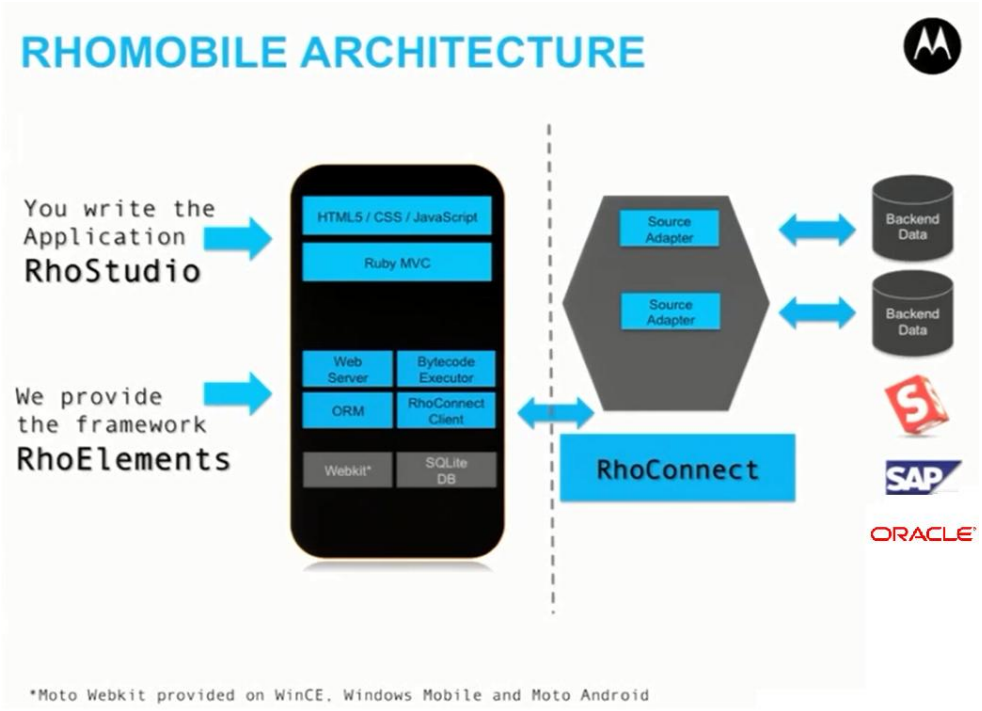


Figura 24 – Diagrama da arquitetura do *Rhomobile* [41].

A *Motorola* concebeu então esta ferramenta de forma a permitir o desenvolvimento de aplicações através de *HTML5*, *Ruby*, *JavaScript* e *CSS*, permitindo assim combinar a versatilidade do desenvolvimento *web* com a rapidez típica de uma aplicação nativa, gerando assim uma aplicação híbrida [42], Figura 24. Esta também permite a fácil interligação a outras aplicações *online* como *Oracle, SAP e WebService*. Esta IDE oferece também uma das melhores ferramentas de *debug* no mercado, o *Rhosimulator*. Este emulador permite simular um variado número de dispositivos e sistemas operativos em tempo real, facilitando e diminuindo o tempo de desenvolvimento das aplicações. A segurança é outro fator importante, considerando isso, o *Rhomobile Suite* oferece a capacidade de encriptação de aplicação através do protocolo *Independent Software Vendor* (ISV), impedindo assim o roubo e uso malicioso destas.

Resumidamente o *Rhomobile Suite* é uma boa ferramenta de desenvolvimento, não só é gratuita, como permite um grande reaproveitamento de código e um aumento na eficiência no tempo de desenvolvimento de aplicações, tal como oferece um dos melhores sistemas de *debug* disponíveis no mercado, tornando-o assim numa das melhores ferramentas da atualidade.

3.5.7 *PhoneGap* (*Cordova*)

Originalmente desenvolvido e lançado pela empresa *Nitobi*, foi posteriormente adquirida pela *Adobe* em 2011, passando a ser chamado *PhoneGap* (*Cordova*) [43].

O *PhoneGap* é uma *framework* gratuita que permite o desenvolvimento de aplicações utilizando protocolos de desenvolvimento *web*, como o *HTML 5, CSS3* e o *JavaScript* [44] e através do uso do *Cordova Application Container* é capaz de compilar aplicações híbridas [45]. Devido à forma como esta *framework* foi desenvolvida é capaz de contornar uma das maiores desvantagens deste tipo de aplicações, sendo capaz de aceder a uma boa parte dos componentes de *hardware* de uma longa lista de dispositivos, como é possível verificar na Figura 25.

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 6.0+	Blackberry 10	Windows Phone 8	Ubuntu	Firefox OS
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓
Compass	x	✓	✓	x	✓	✓	✓	✓
Contacts	✓	✓	✓	✓	✓	✓	✓	✓
File	✓	✓	✓	✓	✓	✓	✓	x
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	x	✓	✓	✓	x
Network	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓

Figura 25 – Tabela de acesso a *hardware* do *PhoneGap*[15].

A *Adobe* em conjunto com a *PhoneGap* fornece também um serviço chamado *PhoneGap Build Service*. Este consiste num sistema *cloud* que recebe o *upload* de uma aplicação *web* desenvolvida em *PhoneGap* e converte-a numa aplicação híbrida, melhorando assim o desempenho desta [46] e facilitando o processo de publicação *on-line* da mesma. Este serviço no entanto não é gratuito custando a volta de 55 €/mês [47].

Resumidamente esta *framework* tem bastante potencial, desde que não seja requerido um alto nível de desempenho, pois de momento as aplicações híbridas mesmo sendo mais rápidas que

[15] Lista de suporte *PhoneGap*: http://phonegap.com/about/feature/ (20/08/2014).

as aplicações *web* não têm o mesmo desempenho que aplicações nativas, o que as torna impróprias para certos tipos de aplicações.

3.5.8 *Icenium*

A *Icenium* é um serviço cloud de desenvolvimento de aplicações multiplataformas pertencente a empresa *Telerik inc*., e lançado no mercado em Outubro de 2012 [48].

Este serviço baseia-se no desenvolvimento de aplicações hibridas através de *HTML5*, *CSS3* e *JavaScript*. Tem um processo de desenvolvimento de aplicações *cloud* muito similar ao *PhoneGap Build Service*, pois tal como ele utiliza o *Apache Cordova* para gerar as suas aplicações híbridas [49]. Com este serviço é possível utilizar duas ferramentas de desenvolvimento relativamente distintas*, Icenium Mist* e *Icenium Graphite*. A primeira é uma IDE *online*, acessível via *browser*, que fornece algumas das funções necessárias para o desenvolvimento de aplicações, mas requer ligação à *web* e não é gratuita. A segunda é uma IDE local, gratuita, onde é possível desenvolver aplicação para qualquer dispositivo móvel, tal como inúmeras funções de *debugging* e teste [50].

Este serviço oferece vários escalões de pagamento mas é esperado um pagamento mensal entre os 141 € e os 700 €.

As diversas capacidades oferecidas por este produto são bastante interessantes para quem pretenda desenvolver para múltiplos OS, mas os custos associados e o facto de não ser capaz de desenvolver aplicações nativas podem ser razões suficientes para evitar este serviço.

3.5.9 Resumo

É possível ver de seguida na Tabela 1, um resumo das principais características de cada ferramenta de desenvolvimento anteriormente descritas. De mencionar que esta informação rapidamente se tornará obsoleta devido à constante evolução tecnológica.

	Linguagem	**Multiplataformas**	**Debug tool**	**Tipo de App**	**Custo**
Visual Studio	*C, C++, C#*	*Windows Phone*	Sim	Nativa	600 €/Ano
Xcode	*Objective C*	*iOS, Mac*	Sim	Nativa	Free
Eclipse	*Java,* outros	*Android, outros*	Sim	Nativa, Híbrida e Web	Free
Embarcador Delphi Studio	*objective Pascal*	*iOS, Android, Windows Phone*	Sim	Nativa	(670 – 2300) €/Anos
Xamarin Studio	*C#*	*iOS, Android*	Sim	Nativa	735 €/Ano
Rhomobile Studio	*HTML5, Ruby, JavaScript e CSS*	*iOS, Android*, outros	Sim	Web, Híbrida	Free
PhoneGap	*Html5, Css, JavaSript*	*iOS, Android,* outros	Sim	Híbrida	Free 55 €/Mês

Iceniem	*Html5, Css, JavaSript*	*iOS, Android,* outros	Sim	Híbrida	Free (141 - 700) €/Mês

Tabela 1 – Principais características dos IDEs.

À data da escrita deste texto, este são alguns dos melhores e mais usados IDEs disponíveis no mercado mas não os únicos. Cada um deles oferece vantagens e desvantagens, muitos deles são de desenvolvimento polivalentes e é impossível afirmar que qualquer um deles seja melhor que os outros. Todos estes factos podem ser resumidos numa simples frase, “liberdade de escolha”.

Capítulo 4 – Características dos Dispositivos Móveis Utilizados e Respetivas Bibliotecas

4.1 *Datalogic Lynx*

4.1.1 Empresa *Datalogic*

Figura 26 – Logotipo da *Datalogic*.

O grupo *Datalogic* (Figura 26), é um dos atuais líderes no mercado de produção de equipamento industrial e é proprietário de 9 centros de pesquisa e 7 locais de produção espalhados por Itália, USA e Vietnam, dando emprego a mais de 2000 funcionários.

Segundo o jornal *La Stampa* [51], foi uma das empresas com maior penetração de mercado no sector de saúde em 2012, tendo fornecido aproximadamente 60.000 unidades ao *Hospital Corporation of America.*

Fundado em Bologna, Itália, pelo engenheiro *Romano Volta* [52] em 1972, a *Datalogic* foca-se em dois mercados distintos:

- Captura de Informação Automática ou *Automatic Data Capture*;
- Automatização Industrial.

Esta empresa concentra-se fundamentalmente no desenvolvimento de novos produtos e soluções, tendo para isso investido em média quase 26 milhões de euros em pesquisas quase todos os anos e é atualmente dona de mais de 1000 patentes.

Há alguns anos que a *Datalogic* tem vindo a lançar dispositivos móveis inteligentes ou *PDAs*, sendo este um dos seus produtos mais populares. Entre estes *PDAs* industriais temos o *Falcon X3, Elf, Lynx*, entre outros, Figura 27. Estes equipamentos são ideais para aplicações hospitalares, industriais, serviços, transporte e logística, sendo assim bastante versáteis e de fácil utilização.

Figura 27 – Alguns modelos de *PDAs* da *Datalogic.*

4.1.2 Dispositivo *Lynx*

Figura 28 – *Datalogic Lynx.*

O *Datalogic Lynx PDA* (Figura 28), foi lançado pela *Datalogic* em dezembro de 2012, com o intuito de ser um computador portátil pequeno, robusto e ideal para diversas áreas. O *Lynx* tenta combinar o melhor do mundo empresarial e comercial, para oferecer um produto prático e eficiente [53].

Este dispositivo pesa aproximadamente 270g, com uma altura de 14.4 cm, um comprimento de 6.8cm e uma largura de 2.7cm, o que o torna num dos mais pequenos dispositivos industriais móveis.

Tem um microprocessador *XScale PXA 310*, o que lhe dá uma velocidade de processamento de aproximadamente 806MHz, 256MB de RAM e 512MB de memória *Flash*. Esta característica permite-lhe correr o sistema operativo *Windows Embedded Handheld 6.5.*

Em termos de comunicação sem fios é capaz de fazer transferências de voz e dados, de forma idêntica a qualquer telemóvel, através *GSM, GPRS, EDGE, 3G* e *4G*. É também capaz de enviar dados via *Bluetooth* e *Wireless*. O *GPS* tem um sistema de auxílio integrado, ou também conhecido por *A-GPS*, (Figura 29), o que lhe permite utilizar não só o posicionamento da rede de satélites do sistema *GPS*, como também utilizar a própria rede de comunicação para auxiliar a sua localização [54].

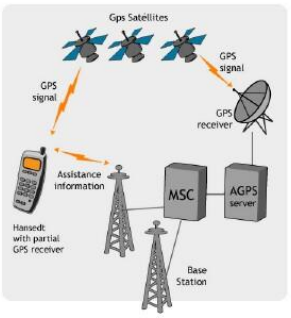


Figura 29 – Representação do funcionamento do *A-GPS*[16].

Este dispositivo tem incorporado um sistema de *scan* a *laser*, capaz de ler códigos de barras 1D e 2D (Figura 30), tal como uma câmara com autofoco capaz de captar imagens até 3 Megapixels e gravar vídeos com uma resolução de 752 H x 480 V *pixels* a 60 frames por segundo.

[16] *GPS vs. aGPS: A Quick Tutorial*: http://www.wpcentral.com/gps-vs-agps-quick-tutorial (20/08/2014).

Figura 30 – Códigos de barras 1D e 2D.

Finalmente este dispositivo tem um ecrã táctil, *TFT-LCD* a cores com uma resolução de *QVGA* (320 X240 pixels).

Todas estas características tornam o *Datalogic Lynx* numa ferramenta prática, robusta e eficiente, tornando-o ideal para o uso industrial e empresarial.

4.2 *Sumsung Fresh*

4.2.1 Grupo *Samsung*

Figura 31 – Logotipo da *Samsung.*

O grupo *Samsung (Figura 31),* é atualmente um dos líderes do mercado das novas tecnologias, sendo responsável pelo desenvolvimento e produção de centenas de componentes eletrónicos como ecrãs de cristal líquido (*liquid-crystal display*) e chips, até equipamentos de uso geral como *smartphones* e televisões.

Esta empresa multinacional é dona de várias infraestruturas espalhadas pelo mundo fora, desde fábricas de componentes e equipamentos, até centros de desenvolvimento, dando emprego a milhares de pessoas. O seu quartel-general encontra-se localizado em Seoul, Coreia do Sul.

Fundada em 1938 pelo coreano *Lee Byung-chul*, como uma empresa de comércio, sendo que ao longo dos anos seguintes ramificou-se em áreas tão diversas como seguros, alimentação, produção têxtil, etc., só tendo entrado no mercado eletrónico nos anos 60, com a produção do seu primeiro modelo de televisão a preto e branco.

Posteriormente nos anos 80, a *Samsung* começou a desenvolver sistemas de comunicação como telefones e faxes. Nesse mesmo ano decidiu investir fortemente no processo de desenvolvimento com o objetivo de se tornar na líder global da indústria eletrónica. Este processo de inovação permitiu-lhe em 1992 ser a segunda maior produtora de chips eletrónicos atrás da *Intel* e em 2005 ser a maior produtora de painéis de *liquid-crystal* (*LCD*).

Atualmente é uma das principais produtoras de dispositivos móveis do mundo, competindo diretamente com a *Nokia* e a *Apple*, com os seus modelos *Galaxy* (Figura 32).

Figura 32 – Alguns modelos *Galaxy* da *Samsung*.

4.2.2 Dispositivo *Galaxy Fresh*

Figura 33 – *Samsung Galaxy Fresh*.

Lançado em 2013, o *Samsung Galaxy Fresh*, (Figura 33) é um entre vários modelos *Galaxy* da *Samsung*.

Este dispositivo tem um comprimento de 121.5 mm, uma largura de 63.1 mm, um comprimento de 10.9 mm e pesa um total de 126 g. Na parte da frente deste modelo encontra-se um ecrã tátil capacitivo com um tamanho diagonal de 4'' e na parte de trás uma câmara de 3.15 MP, capaz de captar imagens e vídeo.

Internamente utiliza um processador *single core* com uma velocidade de processamento de 1GHz, uma memória RAM de 512MB, uma memória física de 4GB, um sistema de localização *GPS* e um sensor de aceleração.

Este *smartphone* suporta os sistemas *standards* de comunicação rádio, *2G* e *3G*, tal como os de comunicação de dados como *GPRS*, *WLAN* e *Bluetooth* e já vem de fábrica com o sistema operativo *Android* versão 4.1.2.

Mesmo não sendo nenhum topo de gama, o seu baixo custo em combinação com as suas características de *hardware*, torna-o numa plataforma ideal de desenvolvimento de aplicações para *Android*.

Capítulo 5 – Aplicações Desenvolvidas Para o Dispositivo *Datalogic*

Durante as primeiras etapas do estágio, foi requerido periodicamente o desenvolvimento de aplicações, com o intuito de incentivar a aprendizagem e familiarização do processo de desenvolvimento de aplicações.

Neste capítulo é pretendida a apresentação e demonstração do funcionamento de algumas destas aplicações desenvolvidas. De mencionar que foi necessário desenvolver e implementar em conjuntos com estas aplicações inúmeras classes (programação *C#*, orientada a objetos) que devido à sua importância, algumas encontram-se descritas no Anexo A.

5.1 Aplicação *Send Mail*

5.1.1 Enquadramento

A interconectividade entre aplicações e os serviços *web* é cada vez mais importante no âmbito dos sistemas móveis, sendo hoje em dia o envio de *e-mails* um dos meios mais comuns. É possível através deste meio, requerer informação, confirmar ou autenticar entregas, confirmar novas atividades, etc., tornando-o assim num excelente meio de comunicação entre o dispositivo e os serviços administrativos.

Tendo esta problemática em mente, foi pedido o desenvolvimento de uma classe capaz de adquirir todos os recursos necessários ao envio de correio eletrónico. Como forma de facilitar a implementação e teste desta classe foi desenvolvida a aplicação *Send Mail*.

5.1.2 Funcionamento

O intuito desta aplicação é desenvolver, testar e demonstrar uma forma simples de enviar correio eletrónico em *C#* através do dispositivo *Lynx*. Tendo este conceito em mente, foi seguido o esquema descrito na Figura 34, como base para a aplicação.

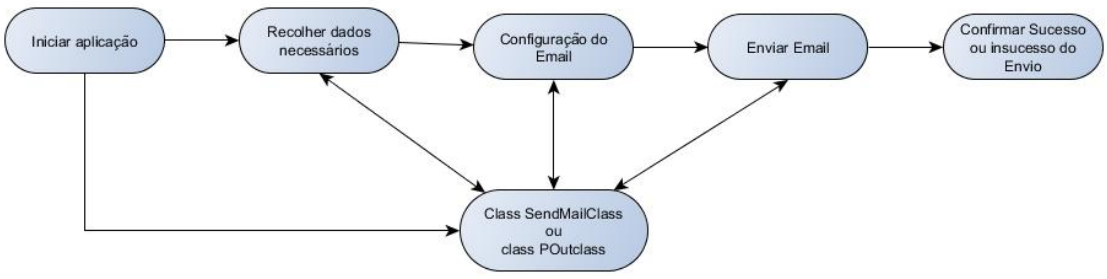


Figura 34 – Esquema simplificado do funcionamento do *Send Mail.*

Para o processo de envio de correio eletrónico em si, foram desenvolvidos dois métodos de envio, a pedido da empresa. Em primeiro lugar foi desenvolvida a classe *SmtpClient*. Esta utiliza a biblioteca *openNETCF.Net.Mail* como núcleo das suas funções, no anexo D.1.2 encontra-se mais informação a respeito desta biblioteca. Em segundo lugar, foi desenvolvida a classe *Poutlook*. Esta por sua vez utiliza no seu âmago a biblioteca do sistema *Microsoft.WindowsMobile.PocketOutlook.*

Figura 35 – GUI da aplicação *Send Mail.*

Figura 36 – Confirmação de envio de correio eletrónico.

Figura 37 – Insucesso no envio do correio eletrónico.

Em ambos os casos, o ambiente gráfico da aplicação é idêntico (Figura 35). Este foi desenvolvido tendo em conta todos os campos necessários ao envio normal de um *email*, como exemplo, endereço, corpo da mensagem, etc. tal como também foi implementada uma forma de confirmar o sucesso ou insucesso do envio do *email*, (ver Figura 36 e Figura 37).

5.1.2.1 Classe *SmtpClient*.

Como foi referido anteriormente, a classe utiliza a biblioteca *openNETCF.Net.Mail.* Esta biblioteca permite a rápida implementação desta aplicação, pois já contém todas as funções e classes necessárias, nomeadamente a classe *SmtpClient*, a qual é a responsável pelo envio de correio eletrónico, necessitando apenas de código de ligação entre o *GUI* e as funções preexistentes nesta biblioteca, como é possível ver pelo diagrama descrito na Figura 38.

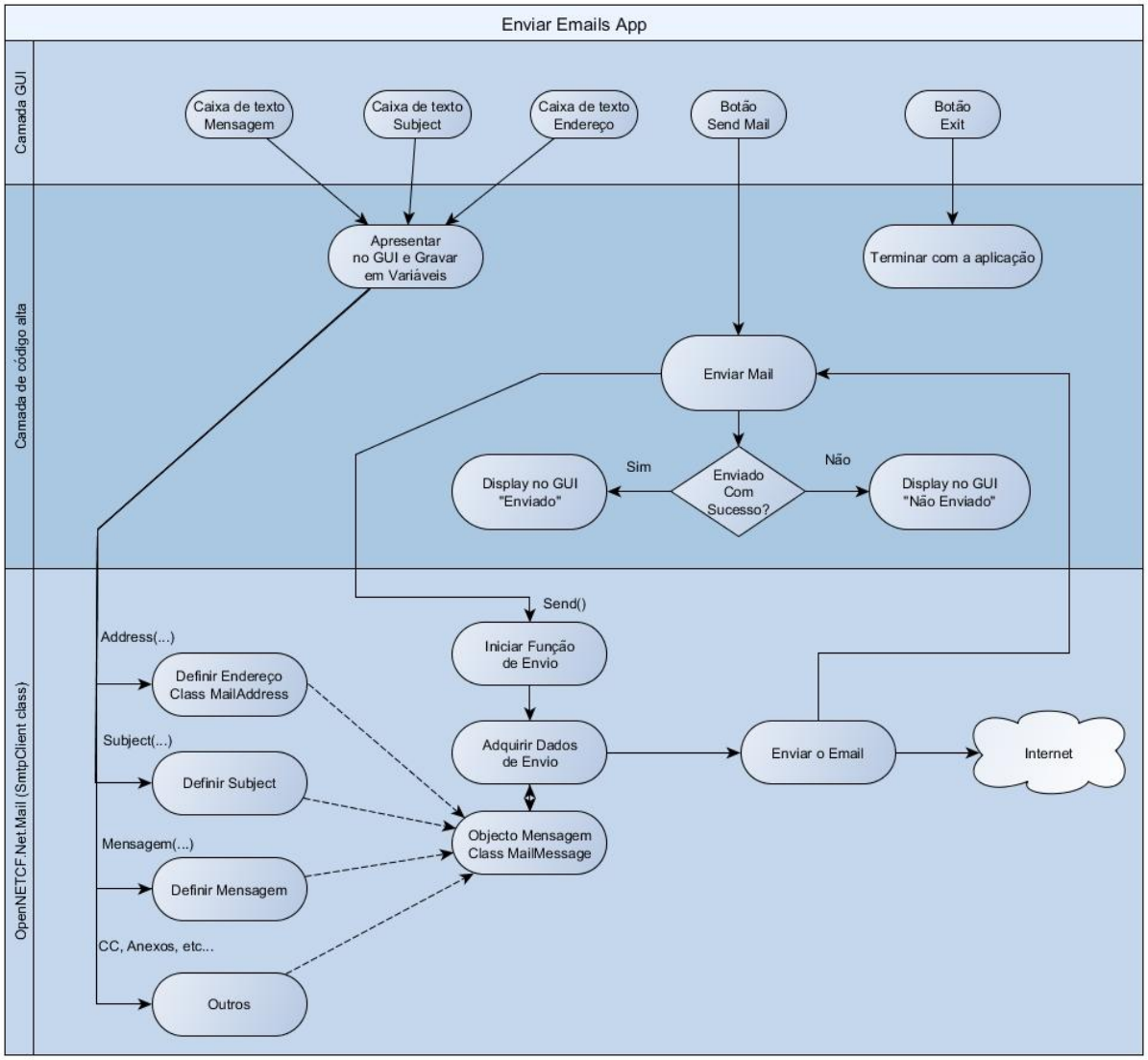


Figura 38 – Diagrama do envio de correio eletrónico através da classe *SmtpClient.*

Ao ser iniciada a aplicação, deve ser criada a classe *SmtpClient*, a qual deve ser de seguida configurada com a conta do utilizador, com o nome do utilizador, palavra-chave, domínio do correio eletrónico, endereço *IP* e porto.

A classe *SmtpClient* é a responsável não só pelo armazenamento de todos os dados a enviar no *email*, como endereço, assunto, anexo, etc. mas também pelas instruções de envio do correio em si.

Infelizmente estes métodos contêm uma grande desvantagem, a biblioteca *openNETCF.Net.Mail* é propriedade da companhia *openNETCF Consulting*, mesmo sendo uma biblioteca *open-source*, o que dificulta a aplicação deste método em aplicações comerciais as quais são geralmente aplicações fechadas.

5.1.2.2 Classe *Poutlook*

Este método, ao contrário do método descrito anteriormente, utiliza as funções disponibilizadas pelo próprio sistema, mais especificamente utiliza as funções e classes disponibilizadas pelo *namespace Microsoft.WindowsMobile.PocketOutlook*, as quais permitem controlar a aplicação *Windows Mobile Pocket Outlook* (ver Figura 39).

A aplicação deve iniciar a ligação numa secção *Pocket Outlook.* Para isso basta construir a classe *POutclass*, a qual é a responsável pela gestão e acesso de todos os serviços de correio eletrónico disponibilizado pela aplicação *Pocket Outlook.* O *POutclass* ao ser construído requer a introdução do nome de uma conta *Outlook* anteriormente configurada no *Pocket Outlook*.

Figura 39 – Diagrama do Envio de correio eletrónico Via *Outlook.*

Este método, infelizmente apresenta os seus próprios problemas. Em primeiro lugar a classe *Poutlook* não tem acesso direto à aplicação *Pocket Outlook*, o que por vezes resulta num pedido de envio por parte desta classe, a qual o *Pocket Outlook* ignora ou demora a enviar. Em segundo lugar este método obriga ao uso de uma conta de correio eletrônico *Outlook* da Microsoft, não permitindo a utilização de contas de correio eletrônico da *Gmail* ou do *Hotmail*. Por fim, este método requer uma configuração externa à aplicação, nomeadamente todas as configurações necessárias a configurar o *Pocket Outlook.*

5.1.3 Resultados

Como é possível observar pela figura seguinte, (Figura 40), a aplicação enviou com sucesso o correio eletrónico para uma conta *hotmail*, provando assim o seu correto funcionamento.

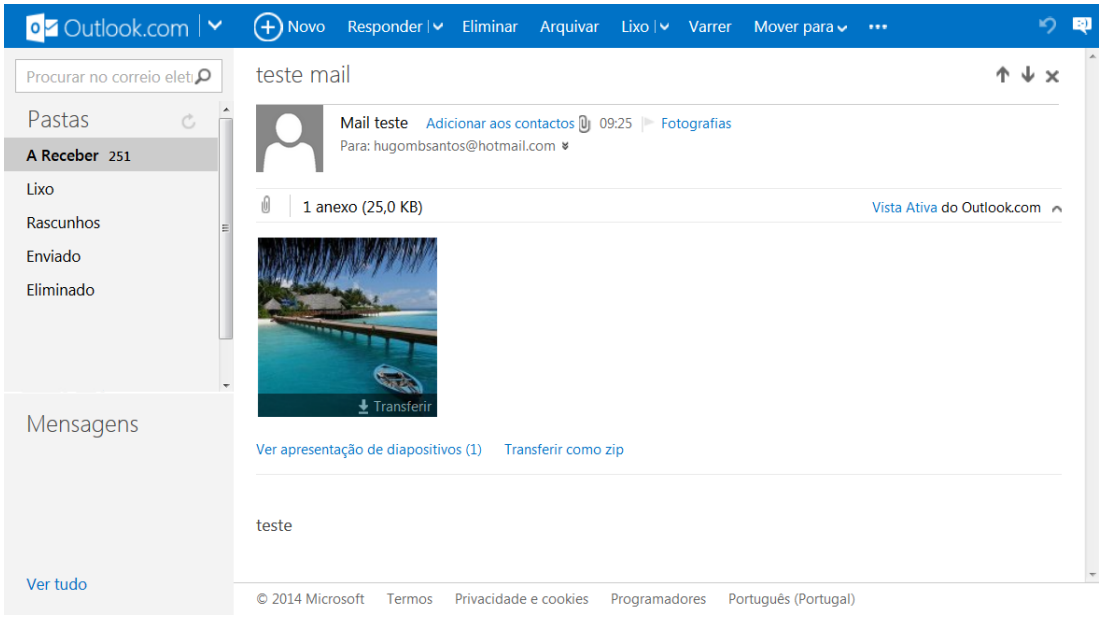


Figura 40 – Caixa de correio do *Hotmail*, ilustrando o recebimento do *email* enviado pela aplicação.

5.2 Recolher Uma Foto

5.2.1 Enquadramento

Há vários anos que os dispositivos móveis partilham uma relação simbiótica com os sistemas de captura de imagem, tendo esta relação resultado numa quase extinção de dispositivos dedicados à captura de imagem.

A cada nova série de *smartphones* e *PDAs*, a qualidade da imagem capturada aumentou e em conjunto com a conectividade que estes dispositivos oferecem, originou a emergência de diversas aplicações que usufruem destas capacidades, como por exemplo, *Barcode Scanner*

de *ZXing*[17] que utiliza as capacidades de captura de imagem para identificar diversos tipos de códigos de barras, ou a rede social *Instagram*[18] que partilha centenas se não milhares de imagens em cada segundo que passa, ou ainda a aplicação *Word Lens Translator*[19], a qual permite traduzir para várias línguas texto captado pela câmara.

Como o antigo provérbio dizia "uma imagem vale mais que mil palavras", a captura de imagens, oferece uma panóplia de soluções para diversas necessidades em diversas áreas, destacando-se as duas seguintes:

- Sempre que é necessário descrever os danos num dado equipamento. Podemos recolher uma foto dos estragos;

- Sempre que é necessária a rápida identificação de um dado componente entre centenas de outros. Basta recolher previamente a imagem deste e armazená-la numa base de dados.

Resumidamente esta é uma função essencial em muitas aplicações profissionais e de lazer e é sempre necessário ter alguns conhecimentos neste campo.

Tendo esta temática em mente, foi requerido o desenvolvimento e implementação de um sistema de captura de imagem no dispositivo *Lynx*. Este sistema deveria disponibilizar apenas o necessário para o utilizador capturar a imagem, sendo todos os parâmetros da captura, como resolução, qualidade, *path* da imagem e formato definidos internamente e não pelo utilizador, isto com o intuito de o tornar num sistema modular e fácil de implementar em outras aplicações.

5.2.2 Protótipo Aplicação *CameraCall*

O processo de desenvolvimento deste trabalho resultou em primeiro lugar numa aplicação protótipo, chamada *CameraCall*. Esta aplicação teve como base toda a documentação a respeito de *CameraCaptureDialog* oferecida pela Microsoft [55] e da aplicação de demonstração acompanhado com o *Developer Tool Kit* do *Windows Mobile 6.5.3,* de referir que as aplicações para WM 6.5.3 são compatíveis com o sistema operativo *Windows Embedded Handheld 6.5* [56] .

[17] *Google Shop*: https://play.google.com/store/apps/details?id=com.google.zxing.client.android (08/08/2014);
[18] *Instagram Home Page*: http://instagram.com/ (08/08/2014);
[19] *Google Shop*: https://play.google.com/store/apps/details?id=com.questvisual.wordlens.demo (08/08/2014).

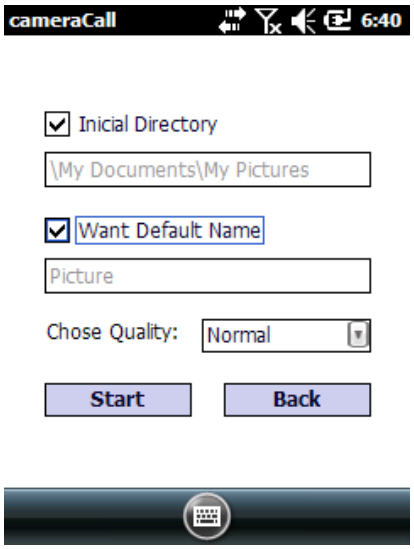


Figura 41 – Menu inicial da aplicação.

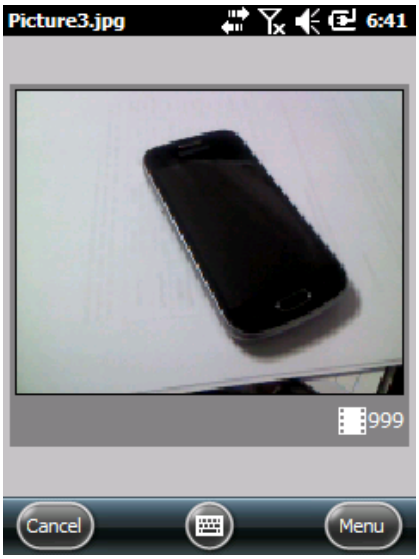


Figura 42 – *Display* de recolha de foto.

A aplicação ao ser iniciada deve apresentar o menu da Figura 41, no qual o utilizador pode escolher alguns dos parâmetros do processo de recolha das fotos, como nome do ficheiro, diretório da foto e qualidade.

Ao carregar no botão "*Start",* como é possível ver pelo diagrama descrito na Figura 45, todos os dados são recolhidos e a janela de diálogo da câmara é iniciada, como se pode constatar nas Figura 42 e Figura 43. De referir que neste protótipo, não chegou a ser implementado o controlo interno das definições da câmara, sendo esta apresentada com as definições *standard* do sistema.

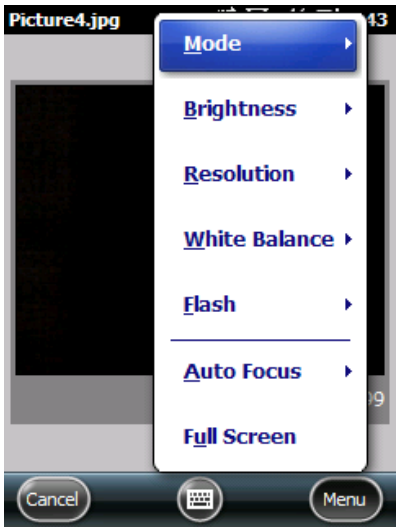


Figura 43 – Menu do *display* de recolha de foto.

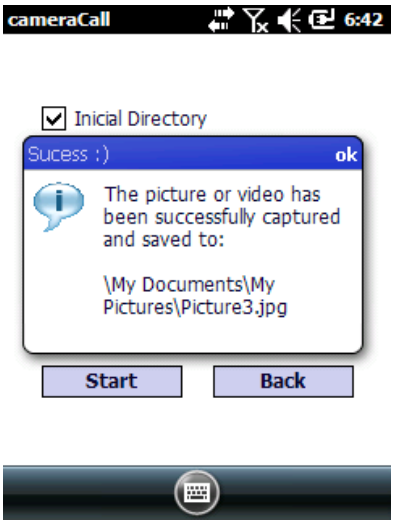


Figura 44 – Recolha da foto bem-sucedida

Após o utilizador confirmar a recolha de foto, através no botão "*Enter"* ou duplo clique do ecrã do dispositivo, a seguinte mensagem de sucesso deve ser apresentada, *"The picture or vídeo has been sucessfully captured and save to:"*, como se pode ver na Figura 44, finalizando assim o processo de captura da imagem.

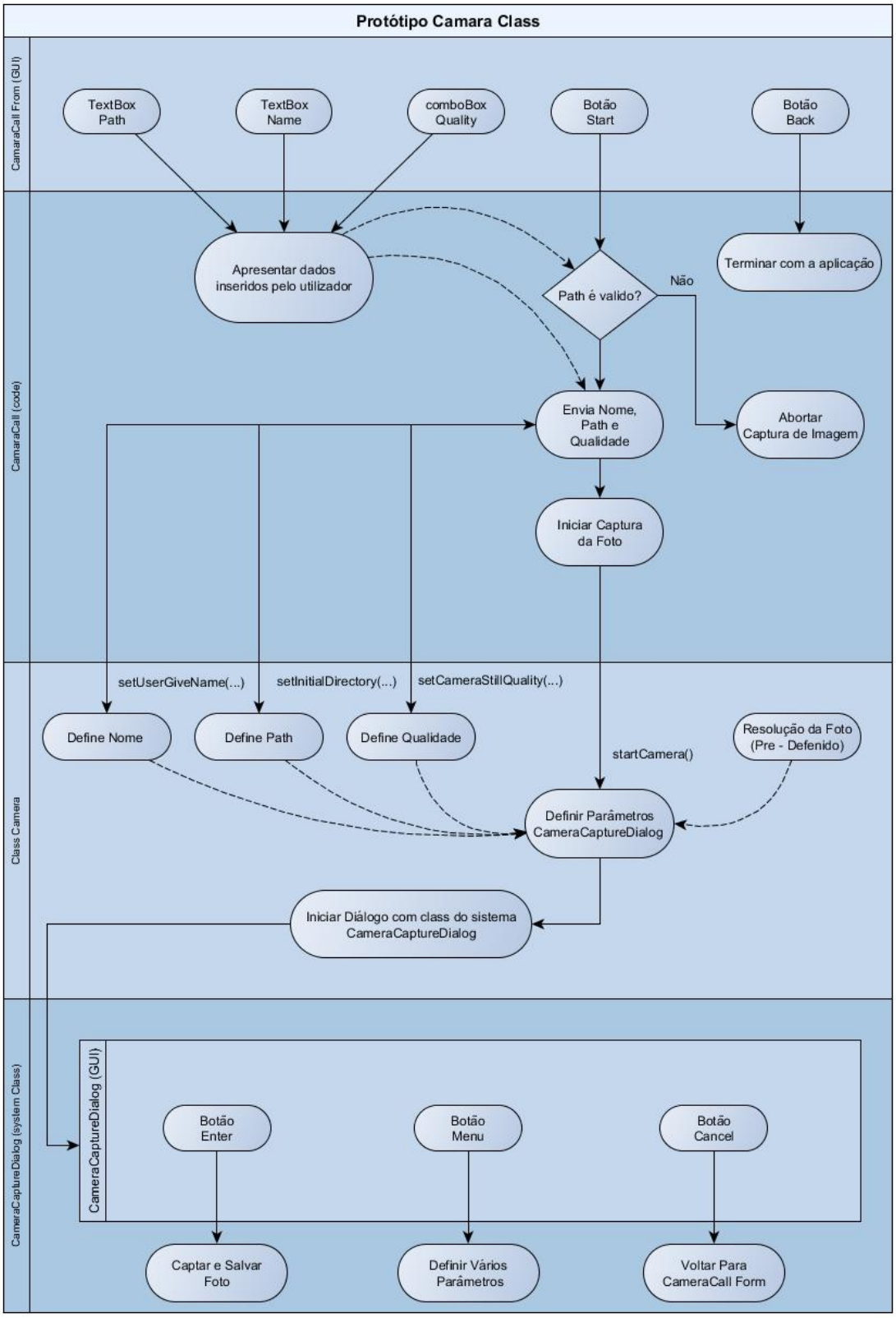


Figura 45 – Diagrama do acesso normal à câmara.

5.2.3 Classe *ShortCamara*

Como foi referido anteriormente, o objetivo final deste trabalho era implementar a recolha de fotos num sistema modular de forma a simplificar a integração deste sistema em outras aplicações. Tendo em conta todos os conhecimentos adquiridos durante o desenvolvimento da aplicação protótipo *CameraCall*, foi desenvolvido a class *ShortCamera,* (ver diagrama da Figura 49).

A classe ao ser iniciada deve receber todos os parâmetros necessários à sua configuração, caso contrário, esta assume os valores *standard* do sistema.

Após a fase de recolha de dados e configuração, a classe deve entrar nos registos locais do dispositivo de forma a configurar os parâmetros da câmara, tal como os aspetos no *interface* gráfico do *GUI* da câmara.

Figura 46 – *Display* recolha de foto da class *ShortCamera.*

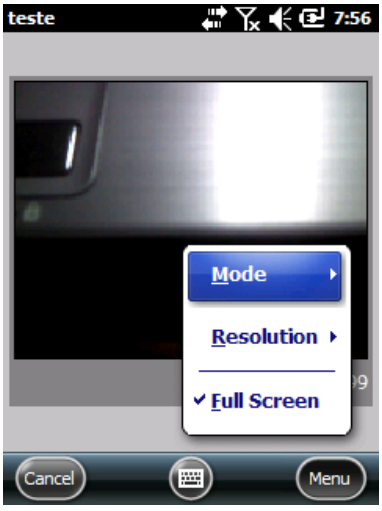


Figura 47 – Menu da recolha de fotos.

Por fim, deve iniciar o diálogo com a câmara em modo personalizado. Como é possível ver na Figura 46, este *form* ao contrário da aplicação *CameraCall,* (ver Figura 42), inicia em modo ecrã completo, e com um *click* no ecrã, apresenta as opções relevantes a o utilizador (Figura 47).

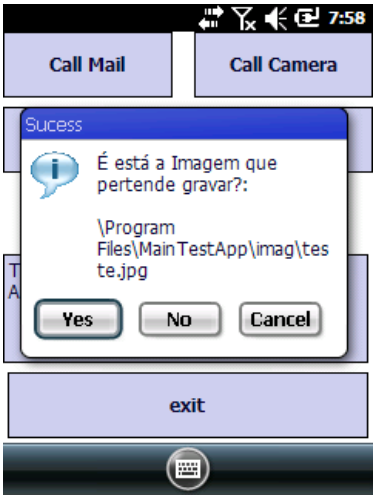


Figura 48 – Recolha de imagem completa.

Tal como a versão protótipo, após a recolha da imagem a classe deve confirmar o sucesso da captura e perguntar se o utilizador pretende gravar a foto no disco rígido do dispositivo (Figura 48).

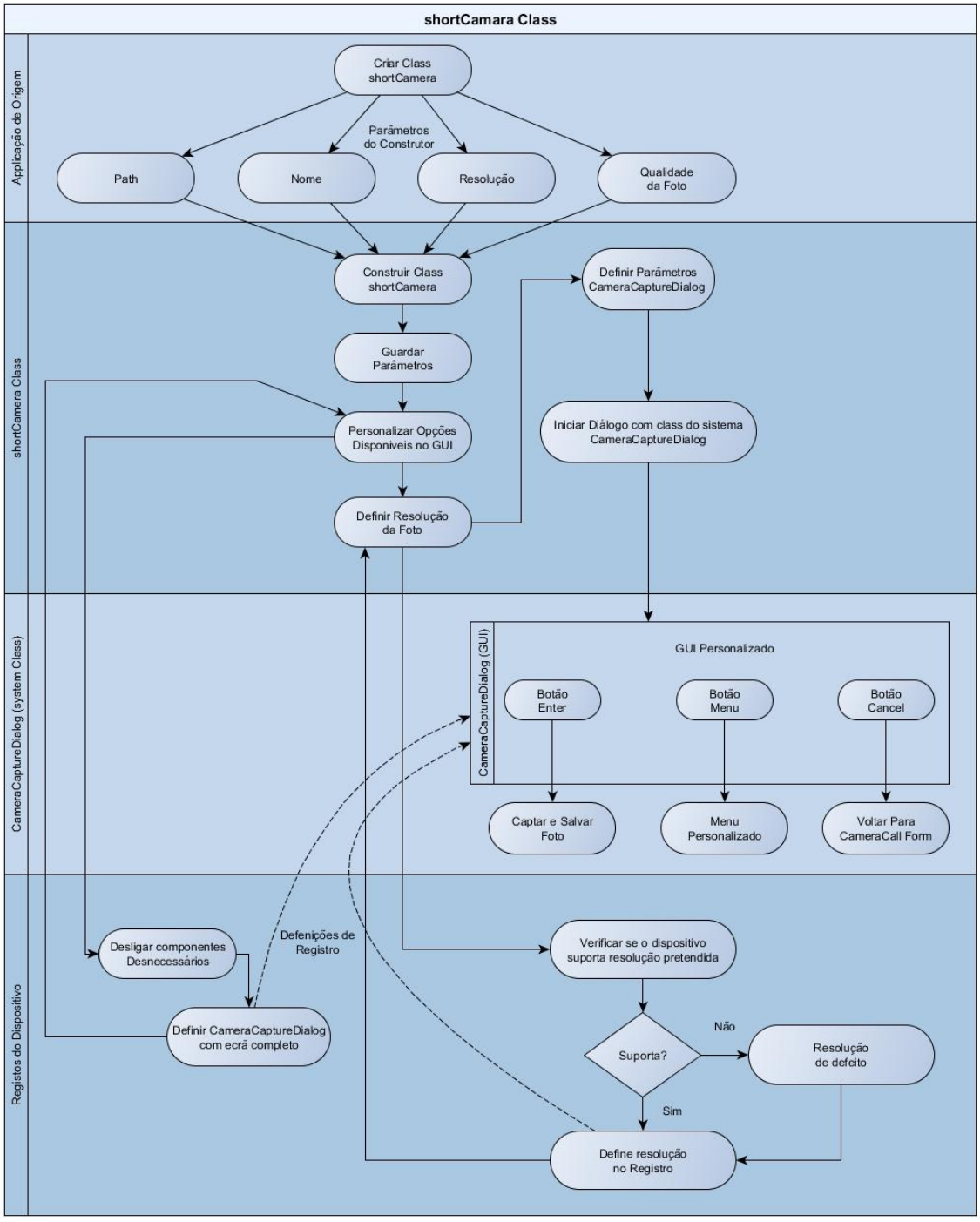


Figura 49 – Diagrama da class *ShortCamera*.

5.3 Recolha de Assinatura

5.3.1 Enquadramento

Atualmente assinar um documento é uma ação tão comum e usual como entrar num autocarro e viajar para o trabalho, mas sem ela o nosso mundo seria muito diferente. Há vários séculos que usamos este simples acto como forma de autenticar documentos, confirmar a validade de contratos e determinar as responsabilidades de um indivíduo.

Sendo o processo de assinatura de documentos e comprovativos uma ação normal e essencial, com o evoluir das tecnologias existem atualmente várias formas de certificar e autenticar programas e documentos através de assinaturas digitas, normalmente em forma de código binário. Mas com o advento de novos equipamentos, principalmente móveis com capacidade de deteção de toque, como por exemplo a maioria dos dispositivos *smartphones* e PDAs, por uma razão de comodidade, surgiram meios de captar a assinatura de uma pessoa em forma inteiramente digital e vários sistemas e programas rapidamente preencheram esta necessidade.

Tendo este conceito em mente foi pedido o desenvolvimento de uma aplicação capaz de capturar uma assinatura e introduzi-la numa imagem como forma de a certificar.

5.3.2 Descrição de Funcionamento

Tendo como base o artigo "*Signature Box that Makes the Signature Look Right"* publicado pelo *CEO* da companhia *Gravel Innovation Inc*. o Professor *Jean-Philippe Gravel* [57], foi desenvolvida a classe *SignatureControl*.

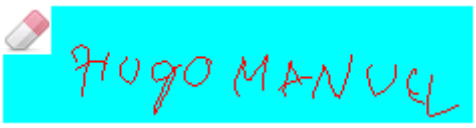


Figura 50 – Aspeto gráfico da classe *SignatureControl*.

Esta classe foi criada com o intuito de gerir e controlar uma zona específica do *GUI* de forma a ser possível detetar o toque no ecrã e através disso inscrever a assinatura. É possível também a remoção da assinatura do campo através de um clique no ícone em forma de borracha no canto superior esquerdo.

Devido à forma como foi desenvolvida, esta classe oferece múltiplas formas de personalização, como definição de tamanho de área, de fundo da classe e até escolha da cor da assinatura. Neste caso foi escolhido um fundo similar à Figura 50.

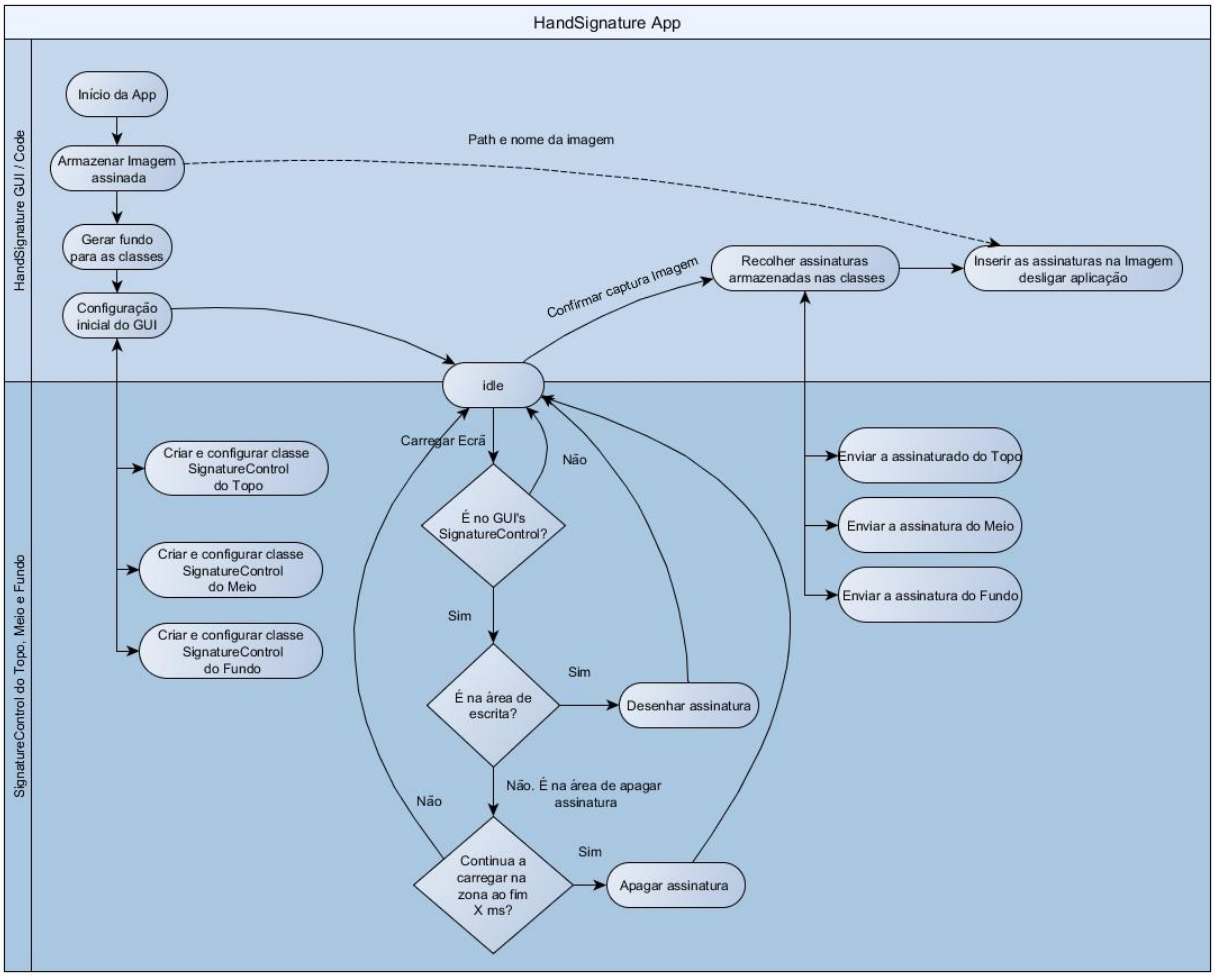


Figura 51 – Diagrama do funcionamento desta aplicação.

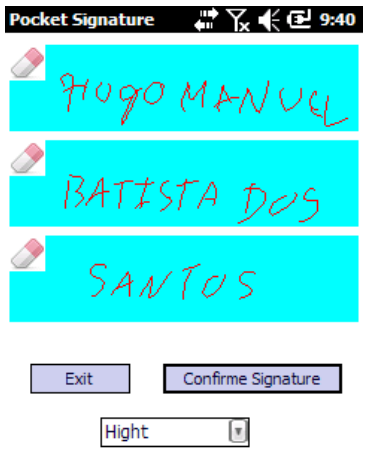


Figura 52 – *GUI* da aplicação de captura de assinatura.

Como é possível ver pelo diagrama descrito na Figura 51 e do *GUI* da Figura 52, a aplicação ao iniciar, gera em primeiro lugar o fundo a ser usado pela classe *SignatureControl*.

Em segundo lugar, esta aplicação cria e configura três classes *SignatureControl*, a do topo, a do meio e a do fundo. Cada uma destas classes é responsável por um dos campos e devem responder de forma individual e conveniente a um clicar do utilizador.

Por fim, após o utilizador pressionar o botão "*Confirme Signature*", a aplicação deve verificar e recolher todas as inscrições em cada um dos campos e transferi-las para a imagem alvo, finalizando assim o processo.

5.3.3 Resultados

Por fim é possível observar os resultados desta aplicação em funcionamento, através da comparação entre uma foto original (Figura 53) e a mesma após a introdução da rubrica (Figura 54).

Figura 53 – Imagem original.

Figura 54 – Imagem assinada.

Infelizmente esta aplicação não é muito viável para uso no dispositivo *Lynx*, devido à sua natureza requerer demasiada memória virtual para executar todo o processamento de imagem e mesmo após várias horas dedicadas à sua otimização, continua a consumir uma elevada percentagem de memória disponibilizada pelo dispositivo, durante o processo de introdução da assinatura da imagem.

5.4 Acesso e Gestão de uma Base da Dados Interna.

5.4.1 Enquadramento

Há várias décadas que os sistemas de gestão de base de dados (*Database Management System)* se tornaram na forma padrão de armazenar e gerir informação no mundo informático. Sem estes sistemas, *sites* como *Google* ou o *Youtube* seriam impossíveis, tal como uma boa parte de todas as soluções de gestão industrial, financeira, etc.. Em resumo estes sistemas de gestão de informação são a espinha dorsal da nossa atual era informática.

Nos últimos anos, o surgimento de dispositivos móveis cada vez mais potentes, incentivou o desenvolvimento de novos métodos que permitem utilizar este novo recurso no desenvolvimento de novas arquiteturas de base de dados. Destas pesquisas surgiram soluções

como *PortaBase*[20], *IBM Mobile Database* da *IBM*, *SQL Server Compact* da *Microsoft*, *SQLite*, entre outros.

Com o intuito de explorar esta temática foi desenvolvida uma simples aplicação com a finalidade de aceder a informação guardada numa base de dados local (a funcionar no *PDA*).

5.4.2 Base de Dados Local *SQLite*

Para poder implementar esta arquitetura de base de dados local foi necessário utilizar a biblioteca *SQLite*[21]. Esta biblioteca permite a conversão da linguagem do sistema operativo, neste caso *C#,* para *SQL* o que por si possibilita a execução de todas as instruções básicas de *SQL*, como criar, consultar e modificar a base de dados.

Tendo em conta o diagrama da Figura 55, foi desenvolvido um pequeno programa com o intuito de demonstrar este processo em funcionamento (Figura 56). Esta aplicação ao iniciar cria uma classe *DataBaseClass*, a qual representa a base de dados. A partir deste ponto é possível utilizar a função *QueryCommand*, para executar o código *SQL* pretendido.

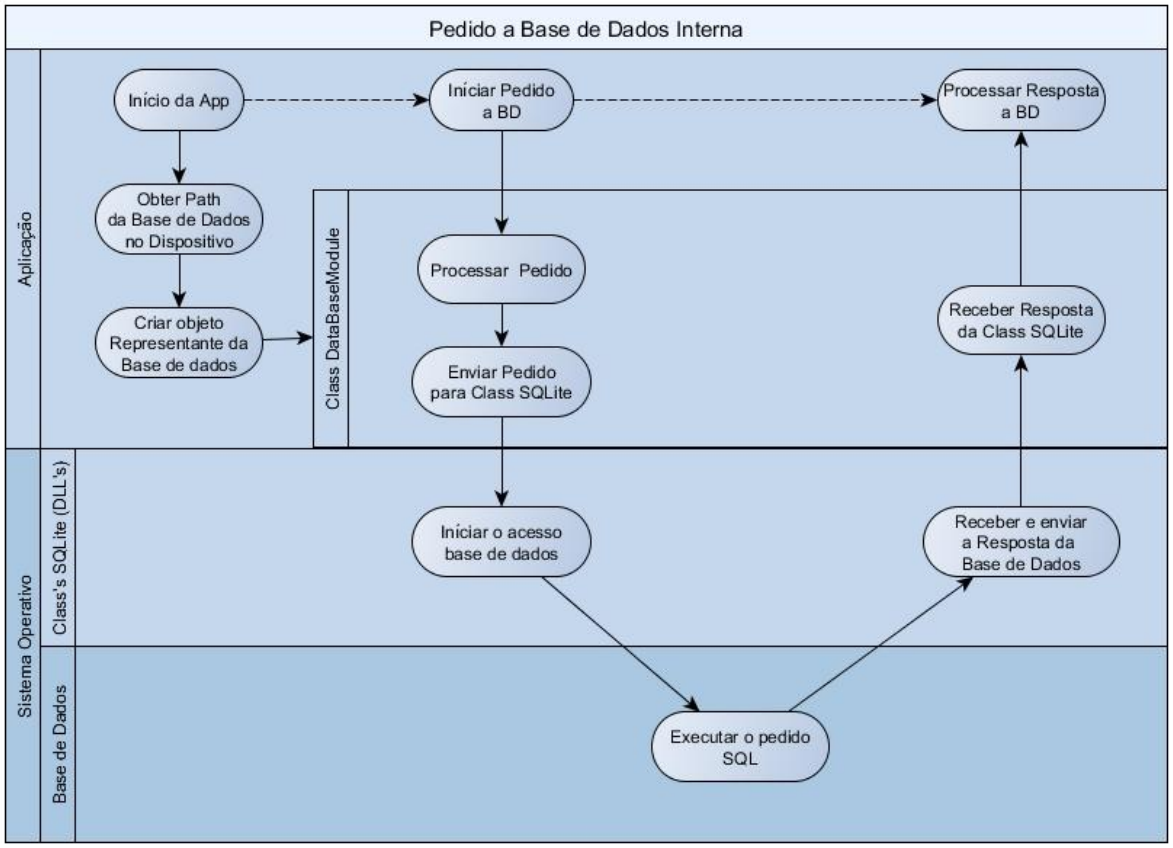


Figura 55 – Diagrama de Acesso a BD Interna.

[20] *Homa page*: http://portabase.sourceforge.net/ (06/08/2014).
[21] *Home Page*: http://www.sqlite.org/ (05/08/2014).

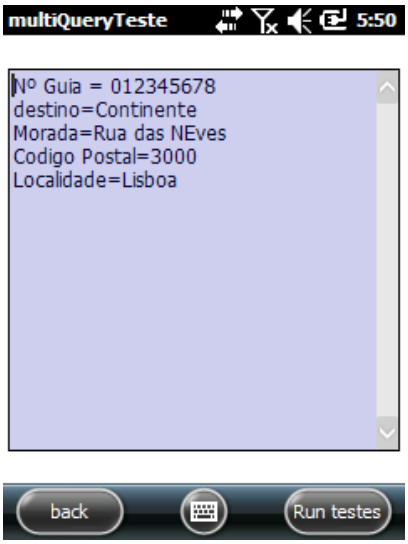


Figura 56 – Aplicação de teste *SQLite*.

A aplicação está a executar um comando *select*, com o intuito de adquirir a informação guardada numa tabela numa base de dados previamente criada e enviada aquando da instalação da aplicação. De mencionar que é possível por exemplo criar uma tabela completamente nova em vez de a enviar em conjunto com a aplicação.

Em resumo este tipo de arquitetura é excelente para a gestão de grandes quantidades de informação a um nível local, pois é de rápida implementação em código, tem uma comunicação direta entre a aplicação e a base de dados e execução de comados *SQL* quase imediata. Por outro lado, existe a questão de limitação de recursos disponíveis para a base de dados e o aumento da complexidade da aplicação, caso seja necessário a partilha de informação com outros dispositivos.

De uma forma geral é ideal para a gestão de informação relativamente pequena e local, mas não recomendada para implementações em média e grande escala.

Capítulo 6 – Aplicação de Listagem de Material – *ANDROID E WEH 6.5*

6.1 Enquadramento Desta Aplicação

Atualmente, é prática comum utilizar na indústria soluções de gestão de manutenção, também conhecidas por CMMS, (*Computerized Maintenance Management Systems)*. Estes sistemas CMMS permitem a monitorização de todo o processo industrial, a partir do ponto de vista da manutenção, permitindo entre vários aspetos a maximização no tempo de vida útil dos mecanismos de produção, o estudo e certificação da qualidade dos produtos produzidos, a eficiência do pessoal de produção e manutenção, a catalogação de inventários, etc.. Mas de uma forma geral, é possível separar estes sistemas em quatro funções fundamentais:

- Assistência no processo de manutenção;
- Gestão das ordens de trabalho;
- Gestão do planeamento de manutenção;
- Gestão de inventário.

Cada uma destas funções contribui com a sua quota-parte para o sistema de manutenção.

Estes sistemas, apenas, começaram a surgir em força a partir dos anos 90, impulsionados pela melhoria da tecnologia informática, (melhores computadores e sistemas de comunicação). Várias empresas como a *ManWinWin Software*[22]*,* o *Bigfoot Maintenance Software*[23] e a *TMW Systems*[24]*,* entre outras, têm vindo a desenvolver e a melhorar novas soluções CMMS de forma a manterem-se relevantes neste mercado competitivo.

Este desenvolvimento interativo também não ignorou os recentes desenvolvimentos no campo dos dispositivos móveis, visto estes serem ideais para estas soluções, permitindo a introdução ou a consulta de dados do processo de manutenção, (em tempo real e no local). Nestes últimos anos, empresas como a *Wasp Barcode Technologies*[25]*,* ou a *RealPage*[26], têm-se dedicado a oferecer soluções de gestão de manutenção, que são compatíveis com vários dispositivos comerciais (*smartphones* e *PDAs*), consolidando o uso destes dispositivos como sendo futuros componentes de qualquer CMMS comercial.

Tendo esta temática em mente, foi desenvolvido um sistema móvel de catalogação de material perecível de manutenção. Este sistema, tem como principal objetivo a demonstração das capacidades dos dispositivos móveis como meio de recolha de informação a respeito de equipamento e materiais alojados num dado espaço geográfico.

[22] Manwinwin *Home Page: http://www.manwinwin.com/PT/index.htm (12/08/2014).*
[23] *Bigfoot Maintenance Software Home page: http://www.bigfootcmms.com/ (12/08/2014).*
[24] *TMW Systems Home page: http://www.tmwsystems.com/ (12/08/2014).*
[25] *Wasp Barcode Technologies Home page: http://www.waspbarcode.com/ (12/08/2014).*
[26] *RealPage Home page: http://www.realpage.com/ (12/08/2014).*

6.2 Objectivos da aplicação Listagem de Materiais

Como acima foi referido, esta aplicação foi concebida tendo em mente a necessidade de catalogar material ou equipamento numa dada sala, escritório ou laboratório. Para isso, foi decidido que este sistema deveria:

- Operar num dispositivo PDA Lynx da Datalogic e num SmartPhone Samsung Galaxy Fresh;
- Recolher todas as informações de equipamentos a catalogar para dentro de uma base de dados ou servidor remoto;
- Identificar todos os objetos a catalogar através do seu código de barras idenficativo.
- Ser capaz de capturar imagens do objeto, assinatura do técnico e se possível a recolha de notas a respeito do objeto, como por exemplo localização, estado de manutenção, etc.

6.3 Princípio de Funcionamento do Sistema.

Este sistema pode ser separado em duas partes distintas. O lado do servidor, onde todas as informações são armazenadas e processadas e o lado do dispositivo móvel, onde todas as informações são consultadas ou inseridas (Figura 57).

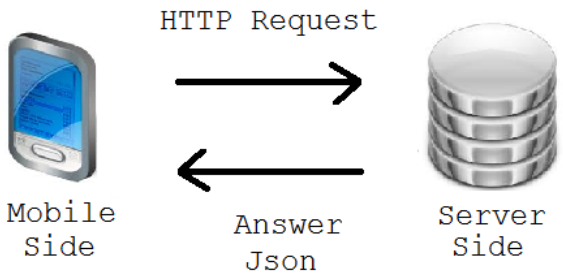


Figura 57 – Diagrama de interação entre o lado do servidor e o lado móvel.

O lado do servidor deve ser executado num servidor com arquitetura do tipo *REST* (Figura 58), o qual recebe e gere todas as solicitações à base de dados. A aplicação desenvolvida nesta arquitetura é também a responsável por responder de volta ao cliente, sendo esta resposta em formato *Json* ou também conhecido por *JavaScript Object Notation*[27].

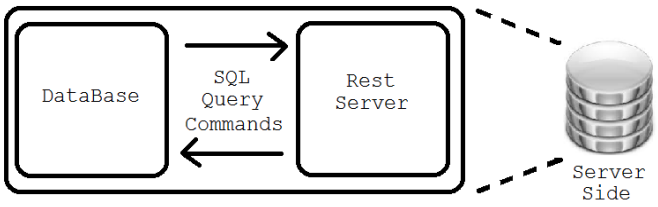


Figura 58 – Diagrama do lado do servidor.

Por sua vez, no lado do dispositivo móvel, este deve ser capaz de comunicar com o servidor, a fim de obter todas as informações necessárias. Para isso, o dispositivo deve conectar-se à

[27] Introdução ao *Json*: http://json.org/ (15/08/2014).

rede *web* via *Wireless* ou via *GPRS*. É necessário também que o dispositivo leia os códigos de barras. Isto, pode ser conseguido por meio de captura de imagem ou *scan*, o que requer uma unidade móvel com câmara ou um *scanner* a *laser*. O dispositivo também deve ter na sua lista de *hardware,* um ecrã táctil sensível o suficiente para capturar assinaturas, uma câmara para tirar fotos e um sistema de *GPS* para reunir a localização do dispositivo. Durante vários anos, todos esses recursos só podiam ser encontrados em dispositivos industriais, geralmente caros, como o *PDA Datalogic Lynx*. Hoje em dia, qualquer *smartphone* é suficiente para preencher esses requisitos, sendo esta uma das razões que incentivou a implementação deste sistema simultaneamente para um *PDA*, (*Datalogic Lynx*), como para um *smartphone*, (*Samsung GT-S7390*).

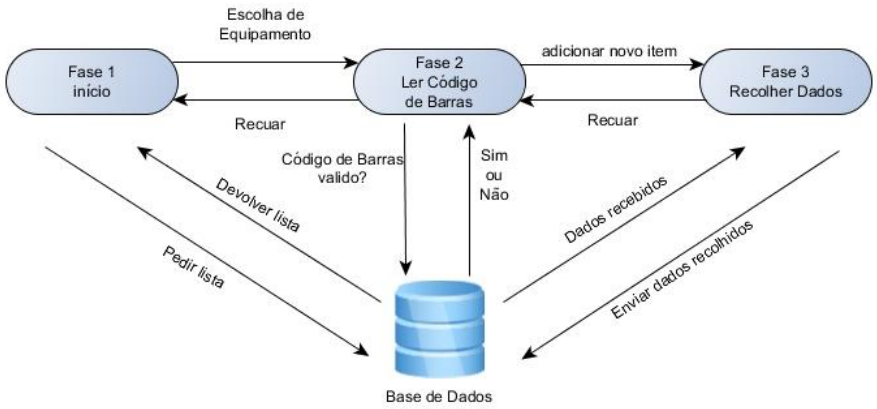


Figura 59 – Esquema básico de funcionamento da aplicação.

A aplicação no dispositivo móvel ao ser iniciada, (Figura 60), deve verificar se o dispositivo se encontra ligado à rede e se é possível estabelecer ligação ao servidor, antes de iniciar o seu funcionamento. Após esta verificação, a aplicação deve pedir ao servidor para enviar a lista de todos os tipos de equipamento disponíveis e caso o pedido seja efetuado devidamente apresentá-los ao utilizador. Após isso, o utilizador deverá escolher o tipo de item que pretende catalogar. Foi decidido que a estrutura da lista de todos os tipos de equipamento possíveis de catalogar na sala seria organizado na seguinte forma, “área de negócios”, “familias de equipamentos”, “marcas de equipamentos” e “modelos de equipamentos”.

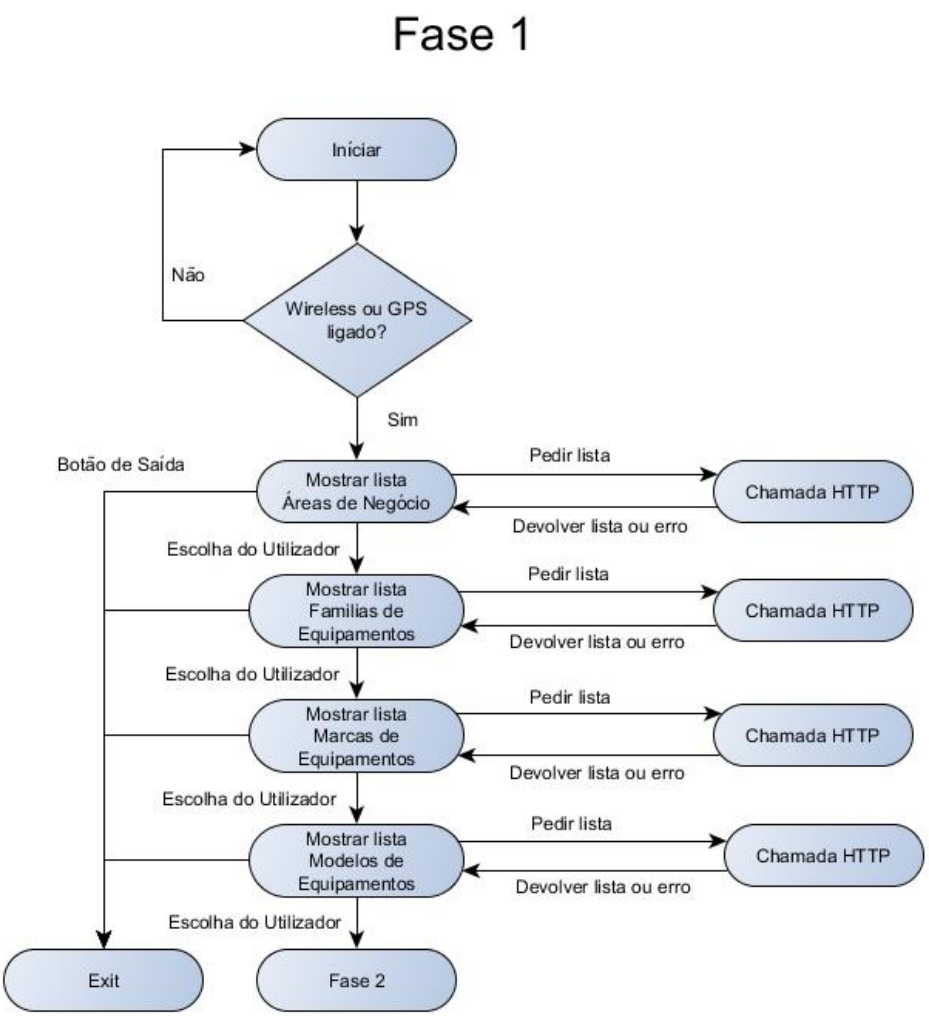


Figura 60 – Fluxograma da fase 1.

Na fase seguinte, (ver Figura 61), a aplicação permite utilizar o leitor de códigos de barras, ou a introdução manual como forma de obter o código identificativo do objeto. Após fazê-lo, a aplicação deve perguntar ao servidor se o código de barras já foi ou não anteriormente inserido, por outras palavras, se for uma entrada nova, podemos considerar o código de barras como sendo valido. Se a resposta for afirmativa, a aplicação deve perguntar ao utilizador se este pretende substituir a entrada anterior. Caso o utilizador confirme a substituição ou esta for uma nova entrada, a aplicação deve prosseguir para a terceira fase (Figura 62).

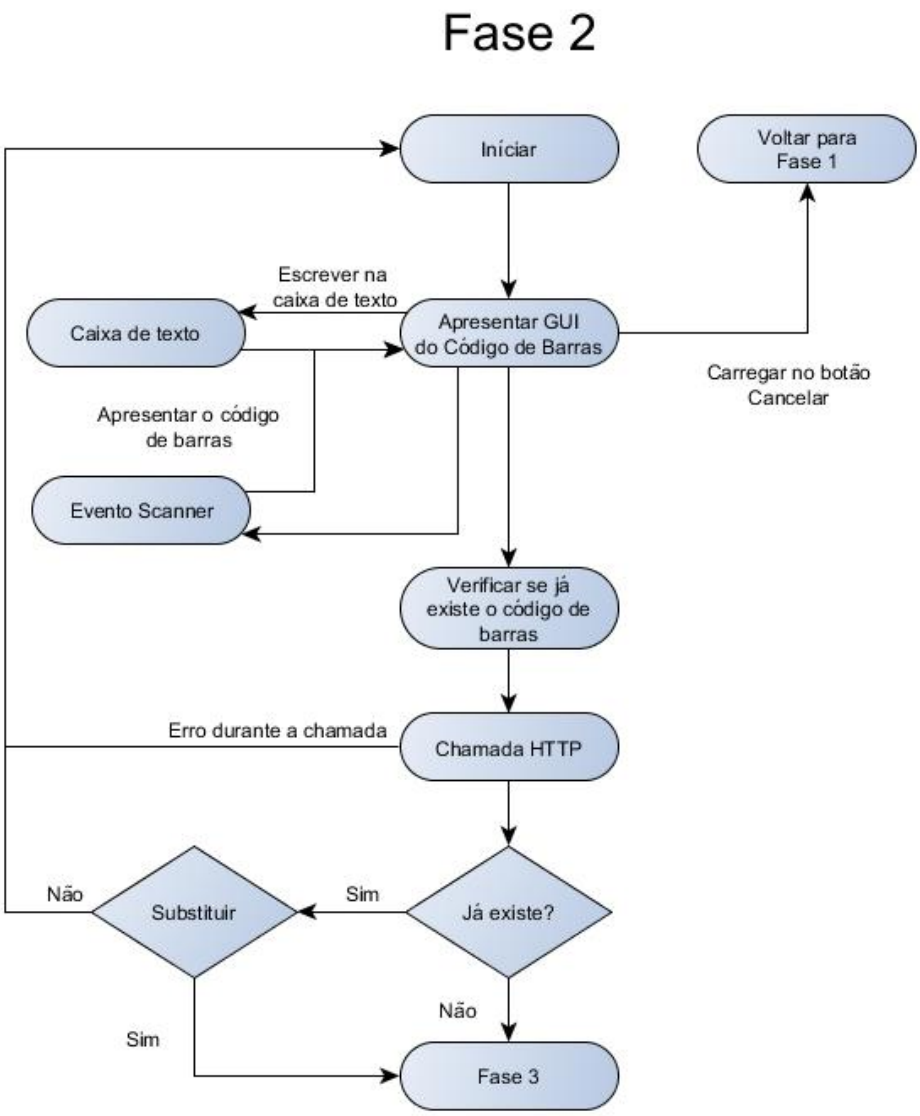


Figura 61 – Fluxograma da fase 2.

Na terceira fase, o utilizador deve recolher fotos, informações sobre a localização de equipamentos, comentar sobre o estado do equipamento e obter a assinatura do utilizador. Note-se que no caso de uma atividade decorrer dentro de portas, o *GPS* não é o método ideal para obter a localização do dispositivo. Assim sendo, outros meios devem ser encontrados, por exemplo, através da utilização de um sistema de código identificativo da sala, que possa ser introduzido manualmente ou através de um marcador de uma etiqueta de código de barras.

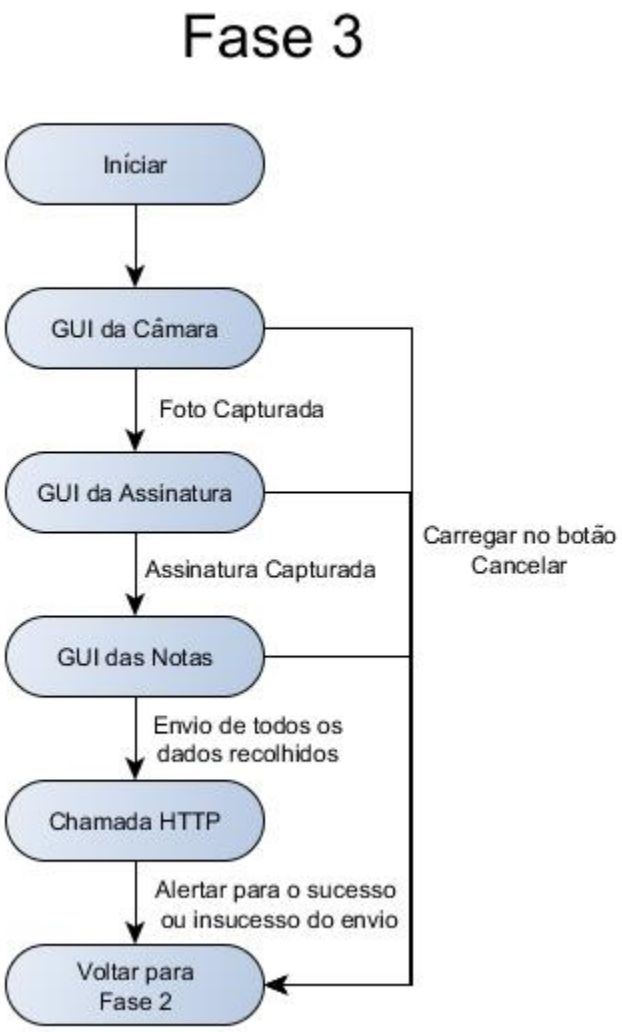


Figura 62 – Fluxograma da fase 3.

Por fim, todas as informações recolhidas anteriormente devem ser enviadas e processadas pelo servidor, enquanto a aplicação deve reverter para a segunda fase e esperar por um novo processo de leitura de código de barras.

Tendo em conta que existe a possibilidade de ocorrerem falhas ou erros de comunicação entre a aplicação e o servidor, foi necessario conceber a aplicação de forma a esta reagir correctamente a estes eventos nocivos, permitindo a continuação de funcionamento da aplicação.

De referir que é possível observar a interação entre as três fases através do diagrama da Figura 59.

6.4 Base de Dados

6.4.1 *MySQL*

O *MySQL* é um sistema genérico de desenvolvimento de base de dados e utiliza a linguagem padrão *SQL* (*Structured Query Language*). Foi desenvolvido inicialmente pela empresa *MySQL AB,* fundada por *Michael Widenius, David Axmark* e *Allan Larsson*, em 1995 [58].

Este *software* teve uma história atribulada ao longo dos tempos, cinco anos após da sua criação passou a ser *open-source*, mas após várias ações judiciais movidas pela empresa Americana *NuSphere,* em 2001, passou a ser propriedade desta. Ao longo dos anos seguinte os direitos do *MySQL* foram sucessivamente comprados e vendidos, enquanto este se tornava cada vez mais popular no mercado. Atualmente é a companhia *Oracle Corporation* a detentora dos direitos do *MySQL* [59].

6.4.2 Implementação da Base de Dados

Como foi referido anteriormente, é pretendido catalogar equipamentos num dado local, então para isso é necessário em primeiro desenvolver uma base de dados que armazene uma listagem de todos os equipamentos perecíveis de sofrer manutenção. Esta tem de ser capaz de armazenar toda a informação relevante a cada equipamento catalogado. Para isso, utilizou-se o *MySQL workbench*. Esta ferramenta de desenvolvimento de base de dados, desenvolvida pela *Oracle Corporation*, permitiu o rápido desenvolvimento e implementação da base de dados representada na Figura 63.

Como é possível observar pela Figura 63, a base de dados suporta a existência de múltiplas áreas de negócio. Cada uma destas áreas contém as suas famílias de equipamento, as quais por sua vez, contêm as diversas marcas e modelos de todos os equipamentos perecíveis de manutenção, permitindo assim consultar, escolher e diferenciar todos os equipamentos catalogados. Paralelamente, existe a tabela de ”equipamento”. Esta é a responsável pelo armazenamento de toda a informações relativas aos equipamentos catalogados, como por exemplo, o seu código de barras, fotos, marca, modelo, etc. Posteriormente, foi adicionado um conjunto de tabelas com o intuito de armazenar notas acerca dos equipamentos catalogados.

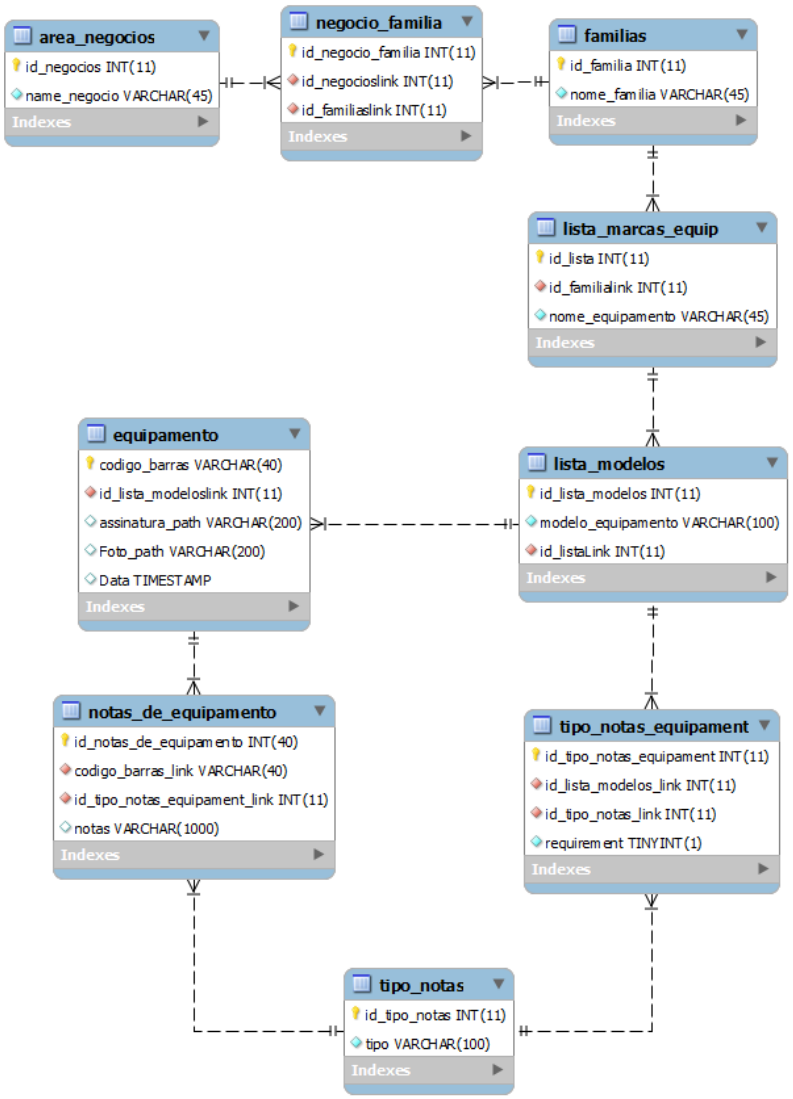


Figura 63 – Representação gráfica da base de dados do sistema.

6.5 Arquitetura *REST*

Como já foi anteriormente referido, no lado do servidor foi implementado uma arquitetura *REST* ou também conhecida por *Representational State Transfer*. Esta arquitetura foi inicialmente proposta em 2000 pelo engenheiro de computação *Roy Thomas Fielding* cofundador da *Apache HTTP Server* [60]. Tendo em conta os conceitos básicos desta arquitetura, foi desenvolvido um servidor *REST* em *PHP,* o qual lida com todos os pedidos de acesso à base de dados, servindo assim, como uma primeira fronteira de proteção ao acesso à base de dados e ao mesmo tempo oferece um acesso universal a todos os dispositivos que respeitem os protocolos *HTTP/HTTPs*, tal como se pode observar na Figura 64.

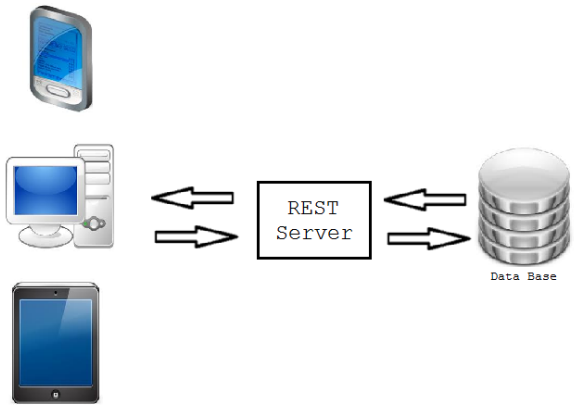


Figura 64 – Interação do *REST Server* com os diversos componentes *Web*.

A comunicação entre o cliente e o servidor *REST* é estabelecida através dos protocolos *HTTPs*. O cliente ao enviar um pedido ao servidor *REST*, utiliza o caracter '?' para separar os requisitos do endereço do servidor. O servidor por sua vez deve responder ao cliente em formato *Json*. Este princípio de funcionamento é demonstrável através do seguinte exemplo:

> O seguinte pedido é emitido pelo cliente quando este pretende requisitar ao servidor o nome de todas as áreas de negócio na base de dados.
>
> https://localhost/RestServerV2/?method=GET&areanegocios=getAll&format=json
>
> A resposta devolvida pelo servidor ao cliente:


```
{"code":1,"status":200,"data":{"Line[0]":{"Row[0]":"Electrónica","
Row[1]":"4"},"Line[1]":{"Row[0]":"Energia","Row[1]":"2"},"Line[2]"
:{"Row[0]":"Informática","Row[1]":"3"},"Line[3]":{"Row[0]":"Saude"
,"Row[1]":"0"},"Line[4]":{"Row[0]":"teste1","Row[1]":"0"},"Line[5]
":{"Row[0]":"teste2","Row[1]":"0"}},"NLine":5,"Querysucesso":"true
","Probleme":"Null"}
```


Toda a informação contida até ao '?' diz respeito ao endereço do servidor. Neste caso encontra-se alojado no "localhost/RestServerV2" e por sua vez toda a informação posterior a este caracter, representa o pedido em si.

O pedido, neste caso, pode ser separado em três partes, cada uma destas, separada pelo caracter '&'. Por sua vez, cada uma destas partes pode ainda ser separada por palavra-chave e por "valor".

A primeira parte "method=GET", informa o servidor que tipo de método está a ser requisitado, neste caso, o método é de consulta, alternativamente o *method* poderia ser *POST* para indicar introdução de novos valores na base de dados, *PUT* para a alteração de valores já existentes e o *DELETE* para a remoção de valores.

A segunda parte do pedido em si, "areanegocios=getAll", informa o servidor do tipo de tarefa que tem de realizar, neste caso enviar a lista de todas as áreas de negócio armazenadas na base de dados.

Por fim "format=json", refere o formato de resposta pretendido por parte do servidor *REST*, neste caso, como é possível ver pela resposta no exemplo acima, é em *Json*, mas poderia ser *XML*, entre outros, dependendo da configuração do servidor.

6.6 Desenvolvimento da Aplicação no Dispositivo *Lynx*

6.6.1 Implementação no dispositivo *Lynx*

Como em outras aplicações para este sistema operativo (ver capítulo 5), foi utilizado o IDE *Microsoft Visual Studio* para desenvolver esta aplicação, tal como também foi usado o dispositivo *Lynx* da *Datalogic* para executar testes práticos.

Durante o processo de desenvolvimento desta aplicação, foi necessário implementar os protocolos de comunicação *HTTPs (HyperText Transfer Protocol Secure) e FTP* (*File Transfer Protocol)*, como meios principais de comunicação com o servidor. Infelizmente estes protocolos só são suportados nativamente pelo sistema operativo através da *dll WinINet*, não tendo qualquer equivalente na *framework .NET Compact.* Isto obrigou ao desenvolvimento de uma biblioteca capaz de fazer a ligação entre o código nativo em *C* do sistema operativo e a linguagem de programação de alto nível *C#*, (consultar anexo A.5 para mais informação).

6.6.2 Funcionamento do *GUI*

A aplicação ao iniciar deve apresentar o *interface* de escolha do tipo de equipamento para catalogar (fase 1). É utilizado neste *form* uma lista do tipo *TreeView,* o qual permite a apresentação ordenada dos *items*, (ver Figura 65).

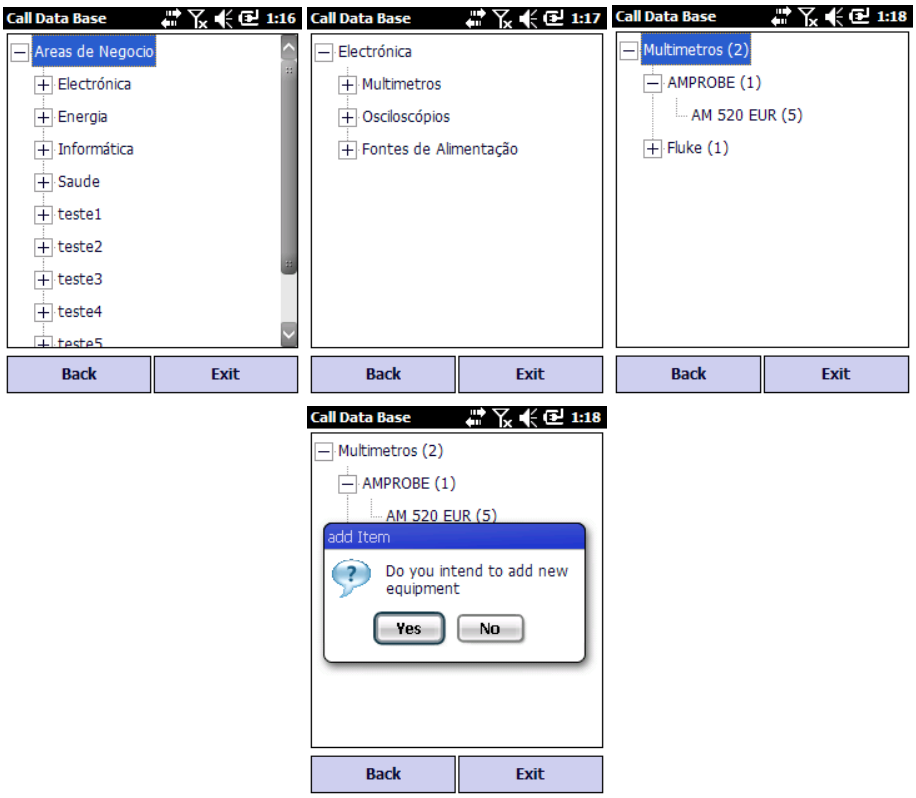


Figura 65 – Menu lista de equipamento em dispositivo *Lynx*.

O *GUI* da imagem seguinte (Figura 66) é o responsável pela recolha do código de barras. Este *GUI* autoriza o uso do leitor de código de barras, o qual pode ser chamado através do botão de

scan, (ver anexo A.1.7 Função setKey2ScannerState), de forma a iniciar a aquisição do ID do objeto a catalogar. Como alternativa, é possível introduzir manualmente o ID na caixa de texto (fase 2). Caso a leitura seja bem-sucedida, o evento "*decodeEventeScan*" será desencadeado, (evento interno responsável pela recolha do valor lido pelo leitor de código de barras). Este deve iniciar a verificação da validade do código obtido e caso este seja aprovado, desbloquear o botão "*Next*".

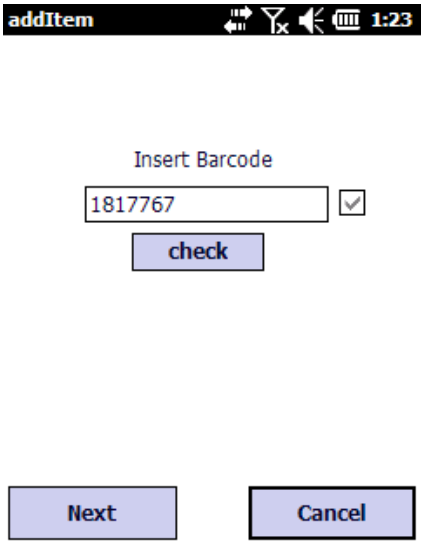


Figura 66 – *GUI* de introdução de códigos de barras dispositivo *Lynx.*

Em seguida a aplicação deve entrar na fase 3 de funcionamento, permitindo a captura de uma foto e uma assinatura, como pode ser visto na Figura 67.

De referir que a classe utilizada para a recolha de fotografias é a mesma desenvolvida no capítulo 5.2.3, classe *shortCamara*.

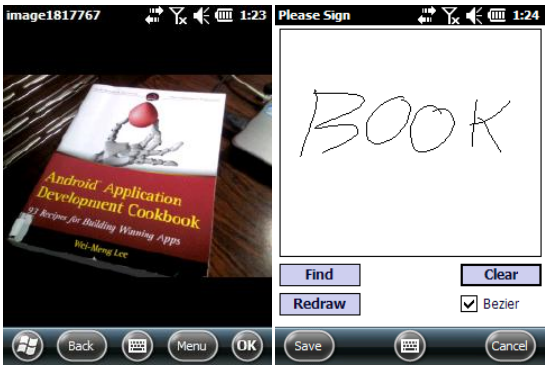


Figura 67 – *GUI* da captura de imagens e captura de assinaturas.

Por fim, a aplicação deve enviar todas as informações necessarias para a base de dados via HTTPs em quanto as imagens e assinaturas recolhidas são enviadas via *FTP* para o servidor, finalizando assim o processo de catalogação de um objeto e retornando de volta para o *GUI* da Figura 66, (fase 2).

De mencionar que a escolha do método de envio FTP para as fotos e assinaturas resulta da simplicidade de implementação deste método através da classe Wininet, tal como pelo interesse de experimentação de um protocolo de comunicação diferente.

6.7 Desenvolvimento da Aplicação num Dispositivo *Google Android*

6.7.1 Implementação em *Google Android*

Como já tinha sido referido no capítulo 6.2, seria implementado em conjunto com a aplicação em *PDA*, uma aplicação similar em *smartphone*, neste caso um dispositivo com o sistema operativo *Android.* Esta escolha, resultou do facto de ser um dos sistemas operativos mais comuns no mercado à data, da escrita deste texto.

Todo o processo de desenvolvimento desta aplicação foi realizado através IDE *Eclipse,* enquanto os testes de *debug* foram realizados no dispositivo *Samsung Galaxy Fresh* (*Samsung GT-S7390*) com a versão de *Android* 4.1.2. Esta aplicação foi desenvolvida em *Java*, a linguagem nativa de *Android.* De referir que, tal como nas aplicações desenvolvidas em *C#* para o dispositivo *Lynx*, não existia qualquer conhecimento prévio de *Java* para *Android*.

Esta aplicação segue o mesmo comportamento que a sua aplicação irmã (comportamento descrito nos fluxogramas das Figura 59, Figura 60, Figura 61 e Figura 62), mas com a distinta diferença de que o dispositivo *Samsung* não dispõe de um *scanner* a *laser* para ler códigos de barras, o que obrigou a implementação de um método alternativo através da câmara. Esta implementação foi possível através do uso da aplicação externa *Barcode Scanner* da *Zxing*, mais informação anexo D.2.1, a qual utiliza *software* de reconhecimento ótico como forma de desempenhar o papel de *scanner* e devolver os códigos lidos para a aplicação principal solução encontrada no livro *Android Application Development Cookbook* [30], mais especificamente no capítulo 9.7 – *Capturing Barcode.* Este processo de comunicação entre aplicações é o que se designa por sistemas de serviços em *Android.* Baseia-se no princípio de que a aplicação *Barcode Scanner* é concebida de forma a indicar ao sistema operativo que está a oferecer um determinado serviço, neste caso, ler o respetivo código de barras e assim qualquer outras aplicação pode requerer ao OS que pretende utilizar esse serviço.

Tal como na aplicação para *WEH 6.5*, esta necessitou de implementar os protocolos de comunicação *HTTPs* e *FTP* e ao contrário do sucedido na aplicação para *WEH 6.5*, a qual requereu o desenvolvimento de uma biblioteca para efetuar estes protocolos, a implementação destes protocolos nesta aplicação decorreu sem grandes incidentes.

A implementação dos protocolos *HTTP* [61] e *HTTPs* [62] foi direta, visto estes protocolos já existirem nas bibliotecas do sistema *Android*. Por outro lado, para implementar o protocolo *FTP* e posteriormente o protocolo *FTPs,* foi necessário utilizar a biblioteca *Apache Commons Net* versão 3.3 [63] da *Apache*.

6.7.2 Funcionamento do *GUI*

Inicialmente o desenvolvimento do *GUI* desta aplicação não se diferenciava muito do criado para o dispositivo *Lynx*. Mas devido ao crescente interesse profissional deste sistema operativo, foi decidido que seria útil explorar melhor as suas capacidades de forma a

desenvolver melhores *interfaces GUI*, com o objetivo de oferecer uma melhor experiência de utilização desta aplicação.

A aplicação ao iniciar deveria apresentar o seguinte menu inicial, (ver Figura 68). A partir destes, é possível iniciar o funcionamento da aplicação através do botão "*Start Work*" ou ir para o menu opções, botão "*Options*".

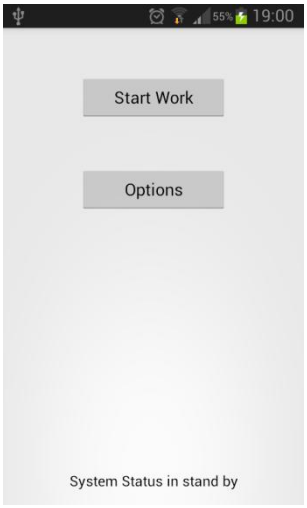


Figura 68 – Menu inicial da aplicação para *Android*.

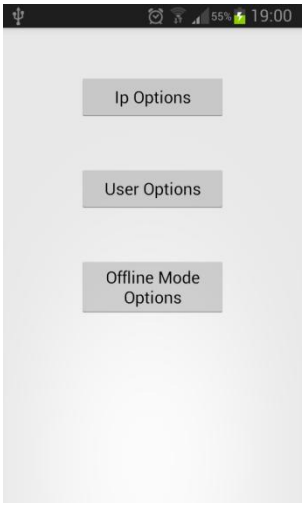


Figura 69 – Menu opções

Através do menu opções, (Figura 69), é possível definir todas as configurações necessárias para estabelecer a ligação ao servidor:

- Para definir o *IP* do servidor, tanto para as comunicações *HTTPs* como *FTPs* → Menu "*IP Options*" (Figura 70).
- Para definir conta de utilizador do serviço *FTP* → Menu *"User Options"* (ver Figura 71).

Infelizmente o modo *Offline* desta aplicação nunca chegou a ser implementado, mas caso seja necessário, deve-se ativar o botão "*Offline Mode Options*".

De referir que este modo *offline,* deveria consistir, na aplicação operar sem conexão direta com o servidor, operando a partir de uma base de dados interna. Esta base de dados deve ser obtida do servidor no instante em que a aplicação pede para entrar em modo *offline*, e a qual deve sincronizar com o servidor no fim de todas as operações estarem completas.

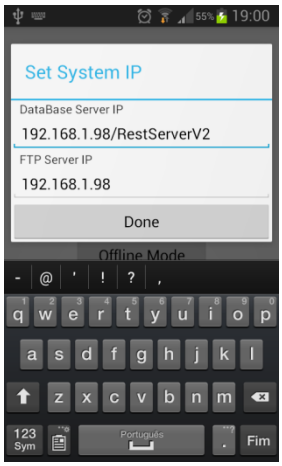


Figura 70 – Menu *IP Options*.

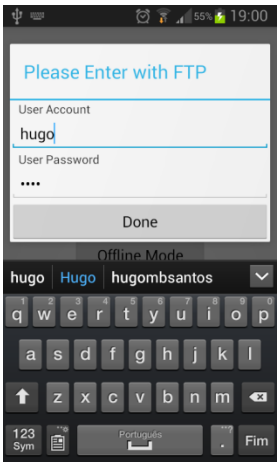


Figura 71 – Menu *User Options*.

A aplicação ao entrar em *Start Work,* ou por outras palavras em funcionamento, deve apresentar o *GUI* Lista de Equipamento. Esta lista baseia-se na utilização de caixas de textos que apresentam todos os *Items* da lista*,* as quais podem ser filtradas, (Figura 72). Exemplo da sequência: Eletrónica, Multímetros, *Fluke* modelo 77; Por fim, é possível passar para a fase seguinte deslizando para a direta o *seekbar*[28]*,* localizada no fundo do *GUI,* (verFigura 73).

[28] SeekBar Tutorial: http://webtutsdepot.com/2011/12/03/android-sdk-tutorial-seekbar-example/ (15/08/2014)

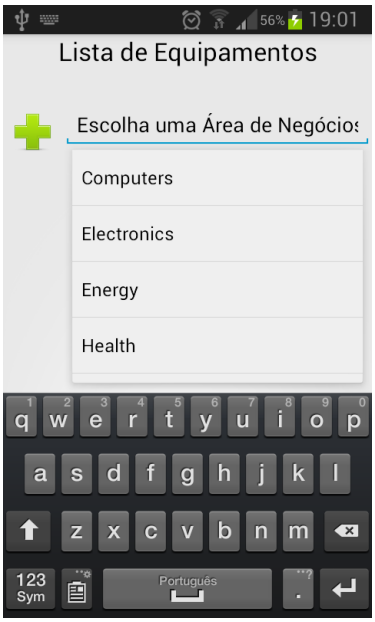


Figura 72 – Listas de equipamento, escolha da área de trabalho.

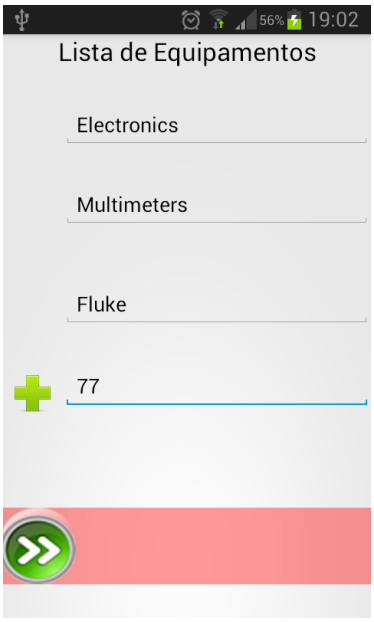


Figura 73 – Listas de equipamento, escolhas completas.

Durante o processo de desenvolvimento desta aplicação, foram testadas outras formas de visualizar a lista de equipamento. Originalmente esta escolha era realizada através de uma *ListView*[29], disposta de uma forma muito similar a aplicação desenvolvida para WM, (Figura 74). Exemplo da sequência: Área de Negócios, Eletrónica, Multímetros, *Fluke* modelo 77; Onde o utilizador ao escolher um elemento do *item* da lista passava para o ecrã seguinte.

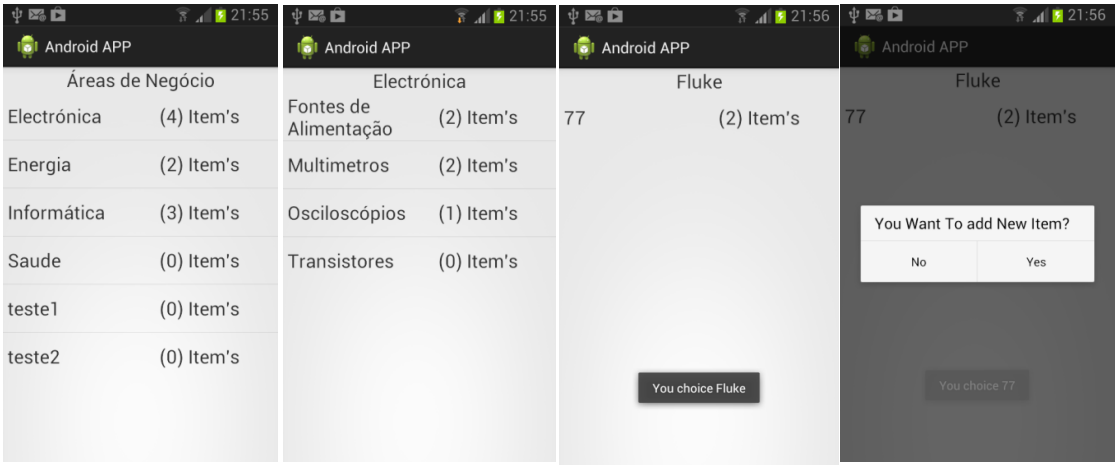


Figura 74 – Menu *ListView* de equipamento em *Android.*

Após a escolha do *Item* a cadastrar, o *interface* deve mudar para o *GUI* da Figura 75, (fase 2). Tal como o *GUI* da aplicação WM descrito na Figura 66, este permite a introdução manual do código identificativo do objeto, ou a sua obtenção através da aplicação *Barcode Scanner,* (ver Figura 76), a qual é ativada ao pressionar o botão ”*Start Scan*”.

[29] *ListView* Tutorial: http://www.vogella.com/tutorials/AndroidListView/article.html (16/08/2014)

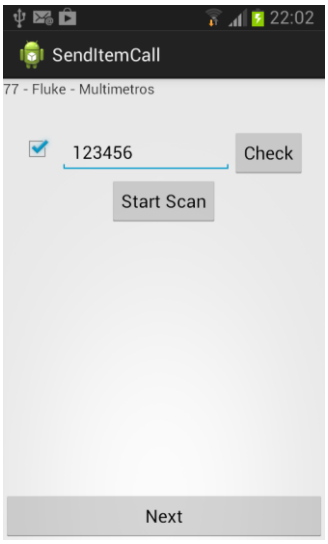


Figura 75 – *GUI* de aquisição de código de barras.

Figura 76 – *Barcode Scanner* de *Zxing*.

Após a base de dados verificar se o código de barras é valido, é possível passar para a terceira e ultima fase desta aplicação, (*GUI* Recolha), carregando no botão "*Next*".

Este *GUI,* permite a recolha de todos os dados relevantes para a catalogação do *Item*, captura de fotos, (Figura 77), assinaturas, (Figura 78) e registo de notas, (Figura 79), a respeito do equipamento como por exemplo a posição ou o tipo de alimentação do dispositivo, etc.

Figura 77 – *GUI* Recolha, (*Tab* "Fotos").

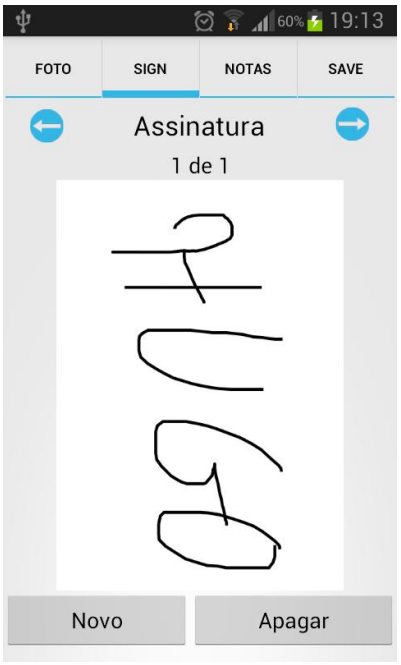


Figura 78 – *GUI* Recolha, (Tab "Sign").

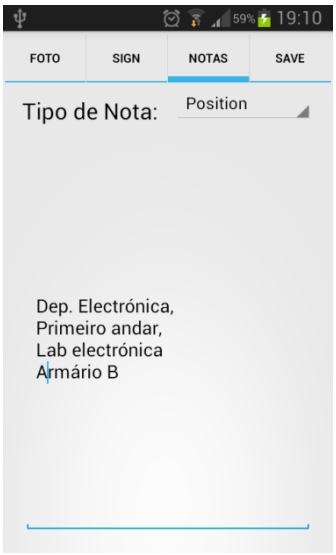


Figura 79 – *GUI* Recolha, (*Tab* "Notas").

A captura de fotos e assinatura é iniciada através do botão "Novo", o qual deve iniciar o *GUI* de captura. No caso das fotos, o *GUI* de câmara do dispositivo, enquanto no caso das assinaturas é um *GUI* personalizado para o efeito, (ver Figura 80). De referir que esta aplicação suporta a captura de múltiplas fotos e assinaturas, sendo possível selecioná-las através das setas nas Figura 77 e Figura 78 ou apagá-las através do botão "Apagar".

Figura 80 – *GUI* de captura de assinaturas.

Ao ser pressionado o *tab* "*Save*", conforme foi descrito anteriormente, a aplicação deve carregar todos os dados recolhidos no *GUI* Recolha para o lado do servidor e alerta para o sucesso ou falha deste procedimento e retornar para o *GUI* de aquisição de código de barras, (fase 2), terminando assim o processo.

O processo de envio dos dados recolhidos nem sempre é bem-sucedido. Isto pode ser atribuído a inúmeros fatores, como falta de sinal *Wireless,* congestionamento da rede, falha do servidor, entre outros. Idealmente, deve existir na aplicação um sistema de salvaguarda desses

dados de forma e evitar a perda total ou parcial da informação não enviada. Infelizmente, devido a uma questão de tempo, este sistema nunca chegou a ser possível implementar na aplicação Listagem de Material, mas em teoria deve operar na seguinte forma:

- A aplicação ao ser iniciada deve arrancar em paralelo com este sistema, de preferência por detrás da aplicação e de forma a ficar fora do olhar do utilizador;

- No caso de ocorrer uma falha de envio dos dados, o utilizar deve ser alertado do facto, podendo este optar por cancelar a operação por completo, (todos os dados do item não enviado serão perdidos), ou continuar, (todos os dados do item não enviado serão enviados para o sistema de salvaguarda);

- Todos os dados recebidos pelo sistema devem ser armazenados em memória não volátil, de forma a permitir o restauro, mesmo que a aplicação sofra uma falha catastrófica ou o dispositivo fique sem bateria. A aplicação ao iniciar, deve verificar se existem dados por enviar da sessão anterior e continuar o processo periódico de reenvio;

- Periodicamente o sistema deve tentar verificar se é possível reenviar todos os pedidos ainda não realizados;

- Caso o utilizador pretenda encerrar a aplicação e o sistema ainda tenha dados por enviar, a aplicação deve alertar para o facto, tal como deve permitir a escolha entre:

  - Desligar a aplicação e apagar todos os dados por enviar;
  - Desligar a aplicação e guardar todos os dados para a próxima sessão de funcionamento;
  - Esperar pelo envio de todos os dados e então desligar a aplicação.

Por fim, fica a ideia que com mais algumas alterações, este sistema de salvaguarda poderia passar a ser um "serviço" responsável por todo o processo de envio dos dados recolhidos. Ao fazer isto seria possível remover por completo o tempo de espera resultante do processo de envio, tornando, assim, a aplicação mais prática e eficiente.

6.8 *Paper*

Tendo em conta a temática deste trabalho, foi proposto pelos orientadores o desenvolvimento de um artigo académico a respeito deste sistema de catalogação de equipamentos para o evento *Maintenance Performance Measurement and Management Conference 2014*. De mencionar que os melhores artigos selecionados pelo Júri do evento seriam publicados numa edição especial dos jornais *International Journal of Strategic Engineering Asset Management* (publicado por *Inderscience*), e *Journal of Industrial Engineering International* (publicado por *Springer*) [64].

Este artigo encontra-se atualmente publicado no livro *Proceedings of Maintenance Performance Measurement and Management (MPMM) Conference 2014* [65]. É possível consultar este artigo no anexo C.

Capítulo 7 – Desenvolvimento de *Sites Web*

7.1 Enquadramento

No campo de desenvolvimento de soluções informáticas é essencial nos dias de hoje ter alguns conhecimentos na área de desenvolvimento *web*. O domínio sobre as atuais linguagens *web* como *HTML5*, *JavaScript*, *PHP* e *CSS*, não só são essenciais no desenvolver de página *web* "tradicionais", como atualmente oferecem a possibilidade de desenvolver aplicações móveis através de *frameworks,* como *PhoneGap* e *Icenium*, sendo assim uma grande valia em termos de aprendizagem.

Foi então pedido durante o período de estágio a construção e manutenção de dois *sites*, com o intuito de consolidar conhecimentos nesta área.

Posteriormente foi requerido a implementação de outro serviço *web,* mais especificamente uma aplicação no lado do servidor em *PHP*. Toda a informação a respeito desta tarefa encontra-se disponível no Anexo B.

7.2 Eventos Associados aos *Sites*

7.2.1 MPMM 2014

O primeiro *site web* desenvolvido no decorrer do estágio, foi o do Congresso internacional *Maintenance Performance Measurement and Management Conference* 2014, ou também conhecido por MPMM 2014 (Figura 81).

Figura 81 – Logotipo do evento MPMM 2014.

Trata-se da sua quarta edição e anteriormente foi realizado na Suécia em 2011, no Reino Unido em 2012, na Finlândia em 2013. Este ano realizar-se-á em Coimbra, Portugal, com o apoio da Universidade de Coimbra.

Esta página *web* foi realizada com o apoio da comissão local do evento, que forneceu todos os conteúdos do *site*, tal como idealizaram o aspeto gráfico pretendido nas páginas.

7.2.2 ENEGI 2014

O segundo *site* desenvolvido no âmbito do estágio, foi o do Encontro Nacional de Engenharia e Gestão Industrial de 2014 (ENEGI Coimbra 2014), (Figura 82). Esta edição congrega o 3º Encontro Nacional de Engenharia e Gestão Industrial e o 7º Encontro Nacional de estudantes de Engenharia e Gestão Industrial num só evento.

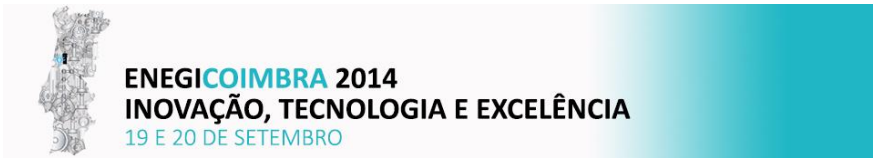


Figura 82 – Logotipo ENEGI Coimbra 2014.

Este congresso foi fruto da organização conjunta do Departamento de Mecânica da Faculdade de Ciências e Tecnologia da Universidade de Coimbra (DEM-FCTUC) e da Associação de Engenharia e Gestão Industrial da Universidade de Coimbra (AEGI-UC), tendo sido realizado em setembro deste ano em Coimbra.

7.3 Ferramentas Utilizadas

Dada a simultânea versatilidade e complexidade do desenvolvimento *web*, não existe uma ferramenta universal para todo o processo de desenvolvimento *web*. Algumas ferramentas, como *Dreamweaver* e *NotePad++,* são optimas para o desenvolvimento do código do *site*, outras, como o *Photoshop*, facilitam a edição de imagens para o site e por fim ferramentas, como o *Xampp*, permitem a rápida implementação de serviços *web* no computador de forma a visualizar e testar o funcionamento do site antes de o implementar finalmente no servidor.

7.3.1 *Dreamweaver*

Lançado em 1997, o *Dreamweaver*, (ver Figura 83), é uma ferramenta IDE de desenvolvimento *web*. Foi desenvolvido originalmente a partir das ideias de *Kevin Lynch*, arquiteto-chefe de *software* da *Macromedia*.

Esta ferramenta é atualmente capaz de desenvolver soluções em linguagens tão diversificadas com *HTML5, PHP, JavaScript, ASP.NET, C#,* etc..

Atualmente a *Adobe Systems*[30], (ver Figura 84), é a detentora de todos os direitos comerciais desta ferramenta.

[30] Adobe System Home Page: http://www.adobe.com (23/07/14)

Figura 83 – Logotipo da *Dreamweaver.*

Figura 84 – Logotipo do *Adobe System.*

Infelizmente esta ferramenta não é gratuita, sendo requerido um investimento inicial para a sua aquisição profissional[31]. Por outro lado é uma boa ferramenta de desenvolvimento, principalmente para iniciantes, devido à quantidade de tutoriais disponíveis na web a respeito desta ferramenta.

O processo de desenvolvimento dos *sites* foi maioritariamente realizado através de uma versão demo de trinta dias deste IDE[32], posteriores alterações foram realizadas através do IDE gratuito *NotePad++*[33].

7.3.2 *Xampp*

Xampp[34] é um pacote de protocolo de *web services*, gratuito e multiplataformas.

Esta ferramenta foi desenvolvida com o intuído de facilitar o desenvolvimento de aplicações e serviços *web*, oferecendo assim uma forma simples, rápida e dinâmica de instalar os produtos desenvolvidos num servidor virtual, dentro de um computador "tradicional".

Para os dispositivos *Windows*, esta ferramenta instala os seguintes serviços:

- *Apache 2.4.9;*
- *MySQL 5.6.16;*
- *PHP 5.5.11;*
- *phpMyAdmin 4.1.12;*
- *FileZilla FTP Server 0.9.41;*
- *Tomcat 7.0.42;*
- *Strawberry Perl 5.16.3.1 Portable;*
- *XAMPP Control Panel 3.2.1.*

Para dispositivos *Linux*, esta ferramenta instala:

- *Apache 2.4.9;*
- *MySQL 5.6.16;*
- *PHP 5.5.11;*
- *phpMyAdmin 4.1.12;*
- *OpenSSL 1.0.1g.*

[31] Loja *online* da *Adobe*: https://creative.adobe.com/pt/plans?store_code=pt (14/08/2014).
[32] *Dreamweaver demo*: https://creative.adobe.com/products/download/dreamweaver (18/08/2014).
[33] *NotePad++ home page*: http://notepad-plus-plus.org/ (18/08/2014).
[34] *Xampp home page*: https://www.apachefriends.org/pt_br/index.html (23/07/14).

Este programa foi usado extensivamente para visualizar e testar ambos os *sites* antes destes entrarem em funcionamento no servidor principal, de forma a combater possíveis *bugs* e erros.

7.4 Composição dos *Sites*

Ambos os *sites* foram desenvolvidos em simultâneo, tendo daí resultado uma enorme partilha de recursos, conhecimentos e soluções a problemas durante o processo de desenvolvimento, tornado assim a estrutura interna dos dois sites bastante similar, mesmo tendo aspetos gráficos relativamente diferentes, tal como é possível verificar na Figura 85.

Figura 85 – A esquerda o *site* MPMM 2014 e a direita o *site* ENEGI Coimbra 2014.

Isto permitiu uma rápida e eficiente implementação dos *sites,* sem comprometer a sua individualidade e mantendo a satisfação de ambas as organizações dos eventos.

7.4.1 Constituição de uma Página em Geral

Como foi referido anteriormente, tanto o *site* MPMM 2014 como o ENEGI Coimbra 2014 partilham elementos muito similares entre si.

Em ambos os *sites*, uma página pode ser dividida em cinco elementos diferentes:

- *Header* ou Cabeçalho – Contem logotipos do evento e parceiros da organização;
- *Left Field* ou Campo do Menu – Menu das páginas do *site*;
- *Right Field* – Campo com publicidades dos patrocinadores e informações importantes;
- *Bottom* ou Fundo da Página – Campo com informações e hiperligações extra;
- *Body* ou Corpo da página – Conteúdo concreto da página, conforme se pode verificar no diagrama da Figura 86.

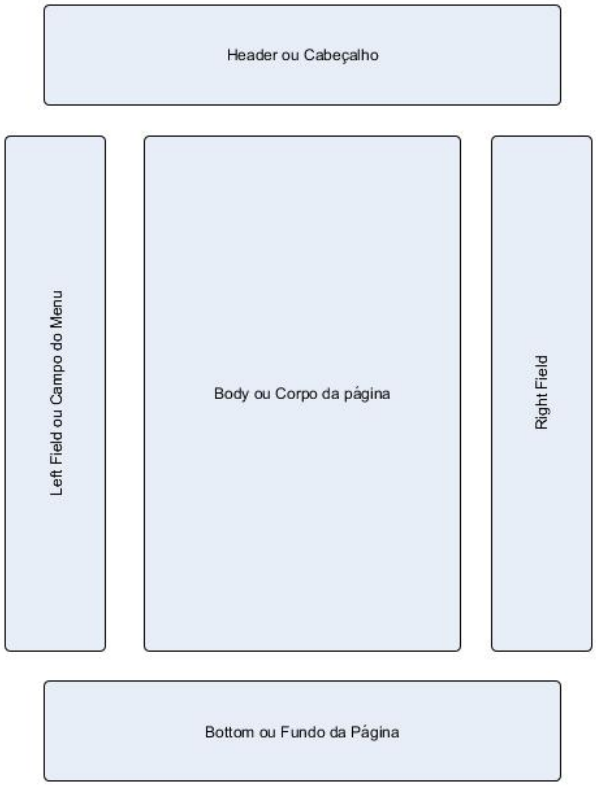


Figura 86 – Diagrama da composição de uma página.

O *Header*, o *Left Field*, o *Right Field* e o *Bottom* são constantes em todo o *site*, o que permite a reutilização destas partes ao longo do mesmo e por isso foram desenvolvidas em *scripts* separados. Isto simplifica o processo de desenvolvimento de uma página, uma vez que só é necessário desenvolver o conteúdo do *body* especifico da página.

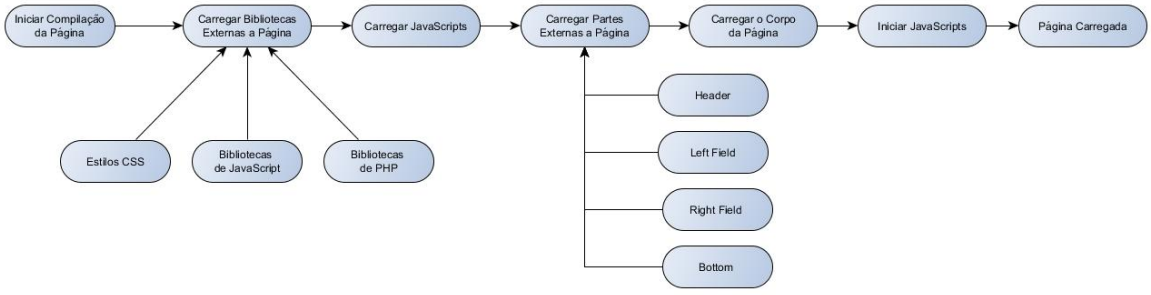


Figura 87 – Diagrama de compilação de uma página.

Durante o processo de construção da página, (Figura 87), só é necessário apontar para localização dos *scripts* responsáveis pelo *Header*, o *Left Field*, o *Right Field* e o *Bottom* e desenvolver o *body* a partir daí.

7.4.2 Estrutura dos *Sites*

A estrutura de ligação entre páginas em ambos os *sites* é bastante simples. Estas são todas mutuamente acessíveis entre si, como é possível ver pelos diagramas descritos nas Figura 88

e Figura 89. Isto fornece uma ótima experiência ao utilizador, pois não necessita de mais do que um ou dois cliques para chegar à página pretendida.

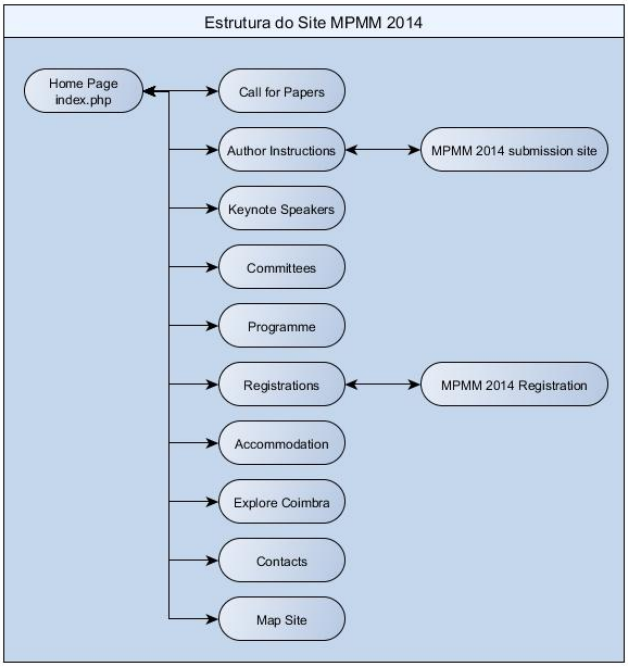


Figura 88 – Diagrama da estrutura do *site* MPMM 2014.

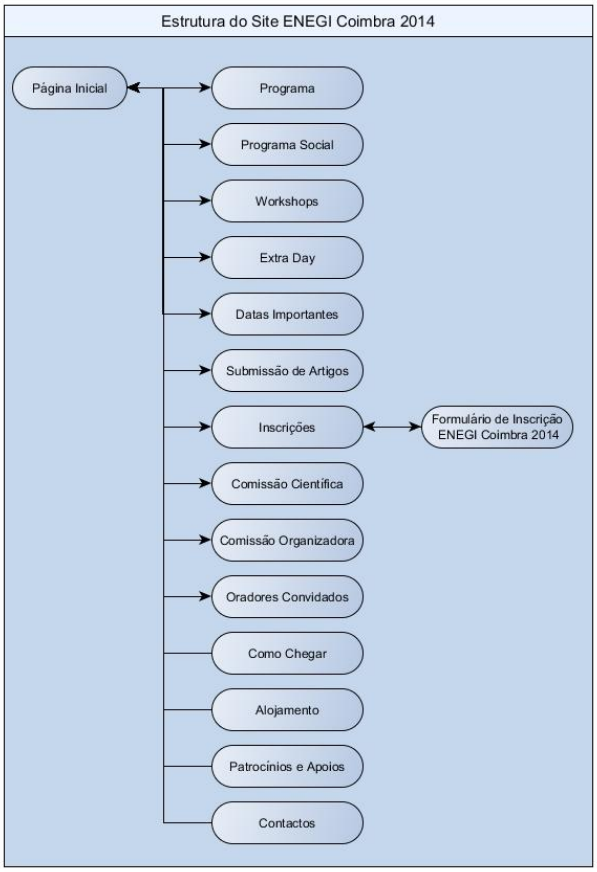


Figura 89 – Diagrama da estrutura do *site* ENEGI Coimbra 2014.

7.4.3 Implementação de uma Galeria de Imagens

Durante o processo de desenvolvimento do *site* MPMM 2014 foi pedido por parte da organização do evento, se seria possível implementar uma forma de apresentar dinamicamente imagens, de fotos de Coimbra. Esta apresentação necessitava não só de ser automática, (de x em x segundos mudar de imagem), mas também de ser interativa, (o usuário poderia manualmente interagir com a galeria).

Após algumas horas de pesquisa, foi possível resolver este desafio através da biblioteca *jQuery* e de um *plugin jQuery Cycle*. Implementando a seguinte função em *JavaScript*, é possível definir uma classe divisão com imagens, neste caso o nome da classe é "*slide*".

```
//---------------JavaScript---------------//
$(document).ready(function() {
        $('.slide').cycle({
                fx: 'fade',
                speed: 1000,
                next:   '#next',
                        prev: '#back'
});
//-----------Fim de JavaScript---------------//
```

Introduzindo de seguida o seguinte código em *HTML* definem-se as imagens a apresentar na galeria, tal como os botões dinâmicos para interagir com esta.

```
//------------Código HTML apresentação de imagens-----------------------//

  <div class="slide">
            <img src="images/slide_accom_1.png" width="500" height="317" />
            <img src="images/slide_accom_2.png" width="500" height="317" />
            <img src="images/slide_accom_3.png" width="500" height="317"  />
            <img src="images/slide_accom_4.png" width="500" height="317"  />
            <img src="images/slide_accom_5.png" width="500" height="317"  />
            <img src="images/slide_accom_6.png" width="500" height="317"  />
            <img src="images/slide_accom_7.png" width="500" height="317"  />
  </div>
//--------------Botões Back e Next-------------------//
  <div id=barslider style="z-index:10; margin-top:1000px">
       <div id="next">
          <table width="100%" height="100%" align="center" valign="center">
           <tr><td>
           <img style="alignment-adjust:central;margin-top:100px"
src="images/ic_arrow_circle_left50X50_03.png" />
           </td></tr>
       </table>
       </div>
       <div id="back">
         <table width="100%" height="100%" align="center" valign="center">
          <tr><td>
          <img style="alignment-adjust:central; margin-top:100px"
src="images/ic_arrow_circle_left50X50_03 - direita.png" />
          </td></tr>
```

```
            </table>
          </div>
        </div>
      //-----------------------------------------------//
```

Por fim, obtém-se uma galeria similar à apresentada nas Figura 90.

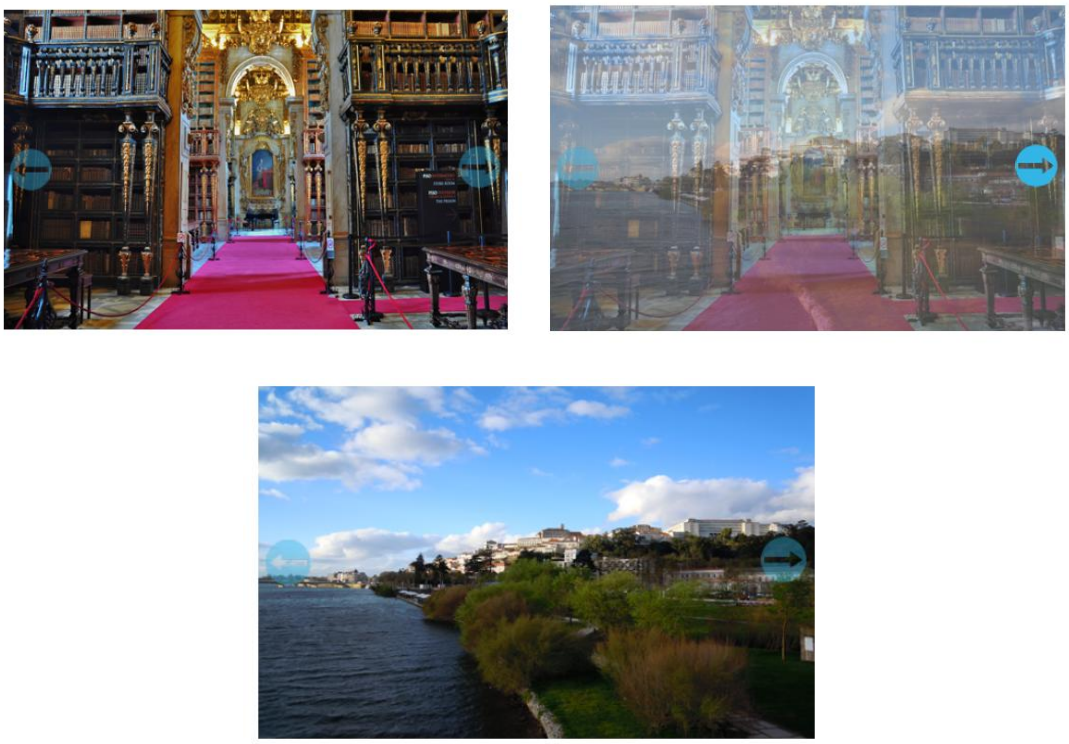


Figura 90 – *Slideshow* na página inicial do *site* MPMM 2014.

Este sistema foi posteriormente reutilizado em ambos os *sites*, para apresentar as condições de alojamento dos hotéis patrocinadores destes eventos.

7.4.4 Implementação do Formulário de Inscrição do *Site* MPMM 2014

Durante o desenvolvimento do *site* do MPMM 2014 verificou-se que seria necessário desenvolver um formulário de inscrição para este evento.

Este formulário, em primeiro lugar, deveria recolher todos os dados da inscrição. Em segundo lugar deveria reconhecer a presença (ou não) de todos os dados obrigatórios ao registro, como nome, correio eletrónico, telefone, etc. do candidato. Por fim, deveria enviar um *email* com os dados do candidato, tanto para este como para a organização, como forma de formalizar a inscrição.

7.4.4.1 Diagrama de Funcionamento do Formulário

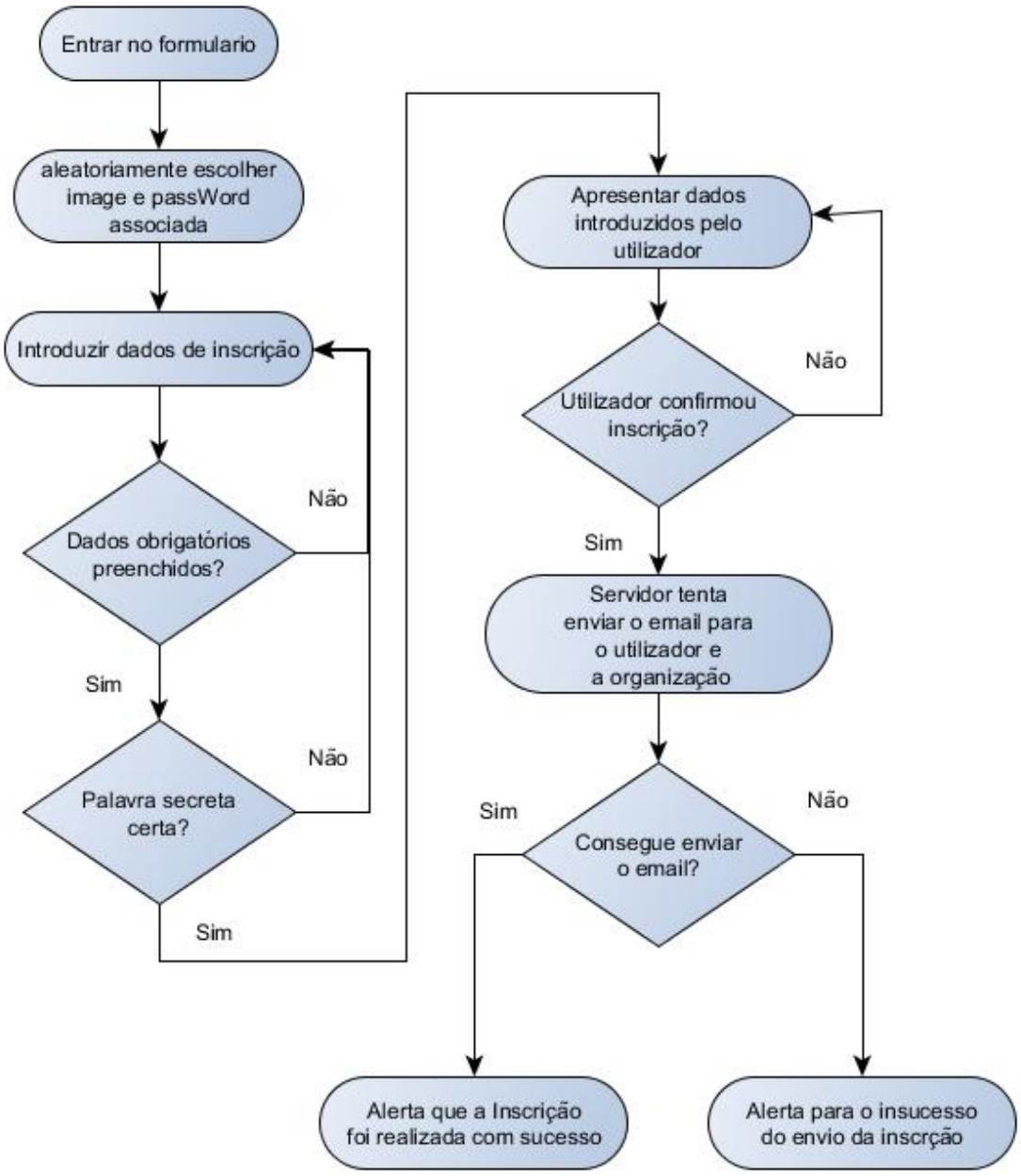


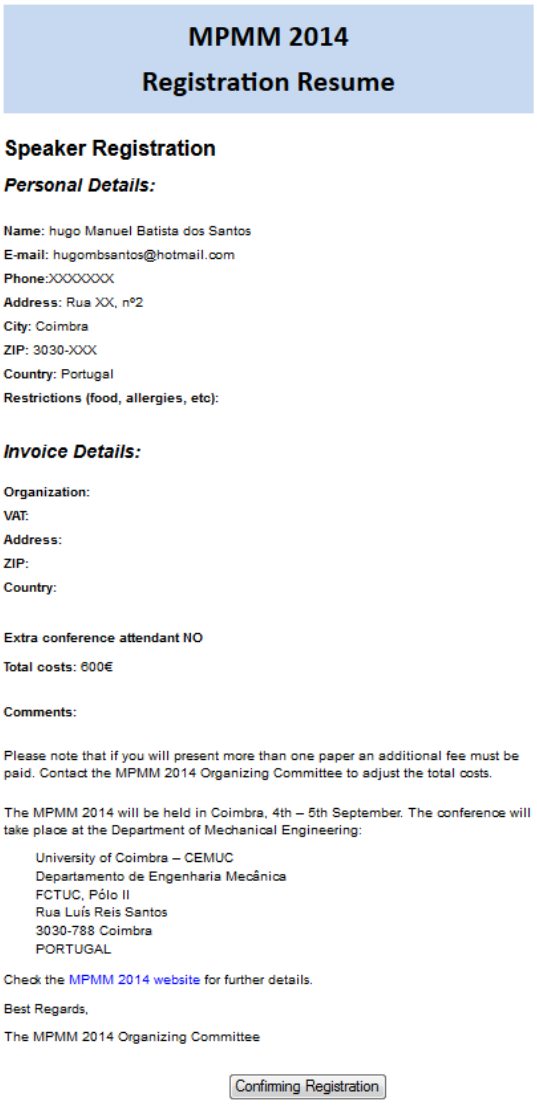
Figura 91 – Diagrama de funcionamento do formulário.

7.4.4.2 Descrição de Funcionamento Básico

Como é possível observar no diagrama descrito pela Figura 91, no momento em que o utilizador carrega na hiperligação do formulário, o servidor deve iniciar o algoritmo *anti-spam* e enviar a página ao *browser*[35].

Após o utilizador submeter a inscrição, (botão *"Submit Form"*), a, página deve verificar se todos os dados obrigatórios foram inseridos antes de autorizar o envio dos dados de volta para o servidor. Isto é possível através do uso de código *JavaScript*, o qual é executado dinamicamente no lado do cliente.

O servidor em seguida deve apresentar a listagem dos dados introduzidos no formulário, como é possível ver no exemplo da Figura 92 e esperar pela confirmação do registo, (botão "*Confirming Registration*"). Após confirmação, o servidor deve então construir e enviar o *email* para a organização, com o intuito de finalizar a inscrição.



MPMM 2014

Registration Resume

Speaker Registration

Personal Details:

Name: hugo Manuel Batista dos Santos
E-mail: hugombsantos@hotmail.com
Phone:XXXXXXX
Address: Rua XX, nº2
City: Coimbra
ZIP: 3030-XXX
Country: Portugal
Restrictions (food, allergies, etc):

Invoice Details:

Organization:
VAT:
Address:
ZIP:
Country:

Extra conference attendant NO

Total costs: 600€

Comments:

Please note that if you will present more than one paper an additional fee must be paid. Contact the MPMM 2014 Organizing Committee to adjust the total costs.

The MPMM 2014 will be held in Coimbra, 4th – 5th September. The conference will take place at the Department of Mechanical Engineering:

University of Coimbra – CEMUC
Departamento de Engenharia Mecânica
FCTUC, Pólo II
Rua Luís Reis Santos
3030-788 Coimbra
PORTUGAL

Check the MPMM 2014 website for further details.

Best Regards,

The MPMM 2014 Organizing Committee

Confirming Registration

Figura 92 – Exemplo de um resumo de inscrição.

[35]Formulário do *site* MPMM: https://cemuc.dem.uc.pt/MPMM2014/register_form_input.php (23/07/2014)

Caso o envio da inscrição decorra sem qualquer problema, o utilizador deve receber a seguinte mensagem de sucesso, *"Registration complete, you will receive an email confirming your registration. Do not forget to print this document, as proof."*, (ver Figura 93). Caso contrário deve visualizar a vermelho, *"We are sorry, but it was not possible to finalize your registration. Try again later and if the problem persists please contact us."*, (ver Figura 94).

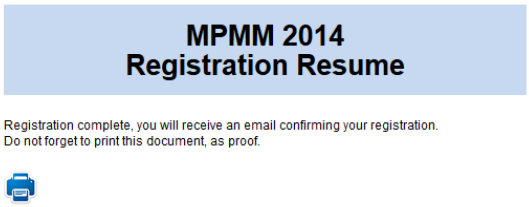


Figura 93 – Mensagem de envio bem-sucedido.

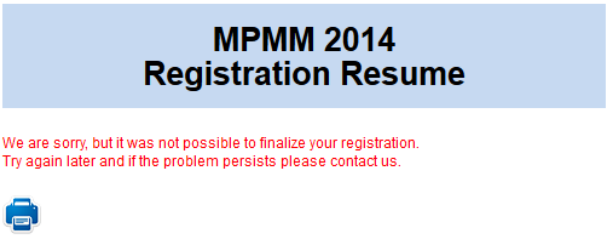


Figura 94 – Mensagem em caso de problemas de envio.

Em ambos os casos, existe a hipótese de imprimir a inscrição, deixando assim em aberto a possibilidade do utilizador imprimir um comprovativo de inscrição, (ver Figura 95) e reenviá-la, via correio pessoal (nos casos de falha de envio).

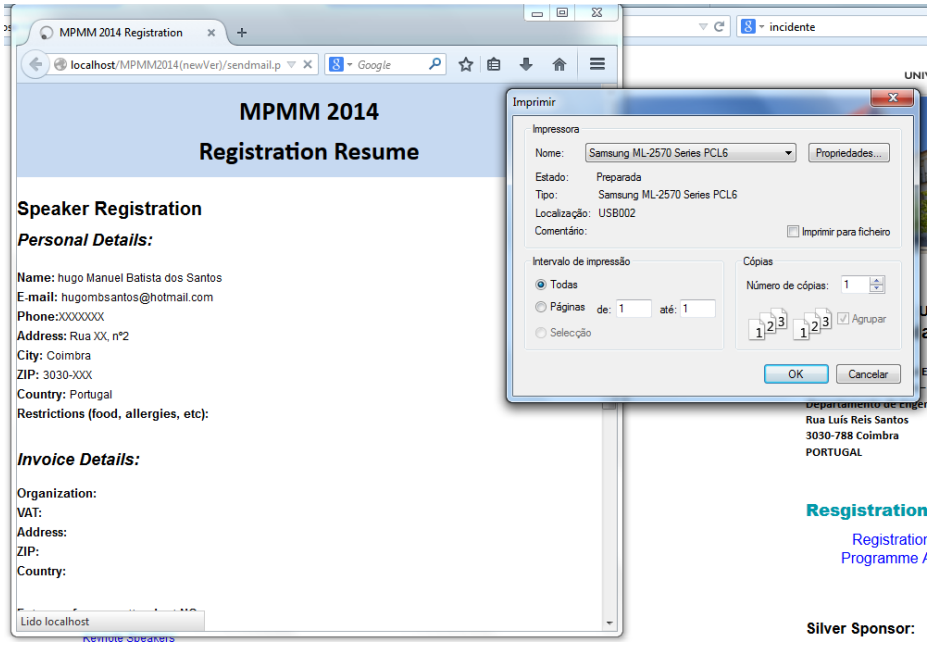


Figura 95 – Impressão da inscrição.

7.4.4.3 Implementação de Número Secreto via Imagem

A segurança é um tópico importante nos dias de hoje, principalmente em ambiente *online*, pois existem milhares de ferramentas e softwares maliciosos à disposição de quem os saiba usar.

Tendo em conta esta problemática, durante o processo de construção do formulário, foi implementado um sistema *anti-spam* simples mas eficiente.

Este sistema consiste na introdução de uma imagem inscrita com uma combinação de quatro dígitos no formulário, sendo obrigatório a reintrodução desta combinação numa caixa de texto antes de ser permitido a submissão da inscrição (Figura 96).

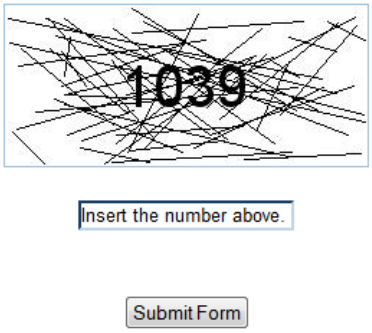


Figura 96 – Formulário MPMM, campo *anti-spam*.

Originalmente, era pretendido que este sistema fosse dinâmico. O servidor ao iniciar a compilação do formulário, a função *geneImage()* seria chamada, (Figura 97). Esta função teria o trabalho de obter um número aleatório de quatro dígitos e com a ajuda da biblioteca GD, gerar uma imagem com esse código em conjunto com ruído à mistura e finalmente retornar a imagem e o código de volta ao formulário.

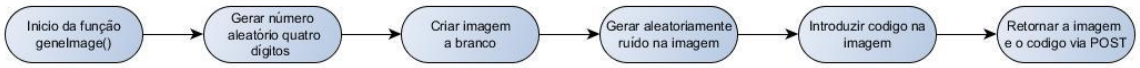


Figura 97 – Diagrama do funcionamento *geneImage().*

Infelizmente não foi possível instalar a biblioteca GD no servidor dos sites MPMM o que impossibilitou esta implementação.

Para resolver este problema, optou-se então por gerar à parte, um conjunto de 500 imagens e uma tabela em ficheiro de texto, que relaciona os nomes de cada uma destas com a sua respetiva palavra-chave, tendo sido instalado posteriormente no servidor as imagens e a tabela.

No início da compilação do formulário, a função *imagesistem()* é chamada (Figura 98). Esta, ao ser ativada gera um número aleatório entre 0 e 499, sendo este usado para verificar na tabela de texto em cima referida, (Figura 99), o respetivo código e imagem a serem apresentados no formulário.

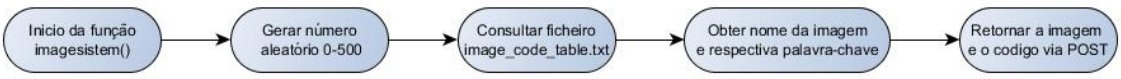


Figura 98 – Diagrama do funcionamento *imagesistem*().

```
key=000 | 5089 | Code000.jpg | <br>
key=001 | 6405 | Code001.jpg | <br>
key=002 | 3676 | Code002.jpg | <br>
key=003 | 7437 | Code003.jpg | <br>
key=004 | 8375 | Code004.jpg | <br>
key=005 | 9185 | Code005.jpg | <br>
key=006 | 6637 | Code006.jpg | <br>
key=007 | 1490 | Code007.jpg | <br>
key=008 | 1378 | Code008.jpg | <br>
key=009 | 4284 | Code009.jpg | <br>
key=010 | 8736 | Code010.jpg | <br>
key=011 | 2346 | Code011.jpg | <br>
key=012 | 1817 | Code012.jpg | <br>
key=013 | 6046 | Code013.jpg | <br>
key=014 | 7443 | Code014.jpg | <br>
key=015 | 4098 | Code015.jpg | <br>
key=016 | 4909 | Code016.jpg | <br>
key=017 | 6312 | Code017.jpg | <br>
key=018 | 4726 | Code018.jpg | <br>
key=019 | 5341 | Code019.jpg | <br>
key=020 | 1375 | Code020.jpg | <br>
key=021 | 1429 | Code021.jpg | <br>
key=022 | 7054 | Code022.jpg | <br>
key=023 | 9167 | Code023.jpg | <br>
key=024 | 4432 | Code024.jpg | <br>
key=025 | 1239 | Code025.jpg | <br>
```

Figura 99 – Parte da tabela (imagem, código).

7.4.5 Implementação de Envio de *Emails* em Ambos os *Sites*

Foi necessário em ambos os *sites* utilizar um sistema de envio de correio eletrónico via servidor, ou seja em linguagem *PHP*.

No caso do *site* MPMM 2014 como forma de enviar as inscrições no evento para a organização e para o inscrito, enquanto no *site* ENEGI Coimbra 2014, como forma de contacto com a organização para esclarecimento de dúvidas (Figura 100).

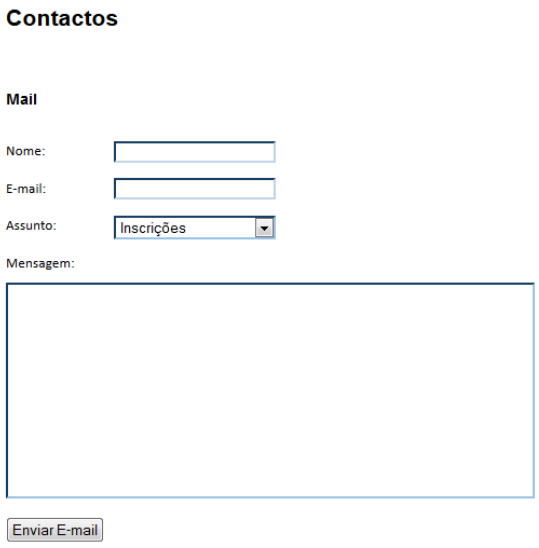


Figura 100 – Interface de envio *emails* no *site* ENEGI Coimbra 2014.

Para implementar este processo, foi utilizada a biblioteca *phpMailer_v2.3*. Esta biblioteca *open-source,* é uma das mais populares e fiáveis bibliotecas de envios de *emails* existentes à disposição, sendo ideal para implementar *Simple Mail Transfer Protocol* (smtp), *Post Office Protocol (POP)* e *qmail*. Esta já vem incluída com todos os protocolos de segurança, como

Secure Sockets Layer ou *Transport Layer Security* e é ideal para enviar *emails* em formato *HTML*, oferecendo assim uma grande versatilidade na personalização dos *emails* a enviar.

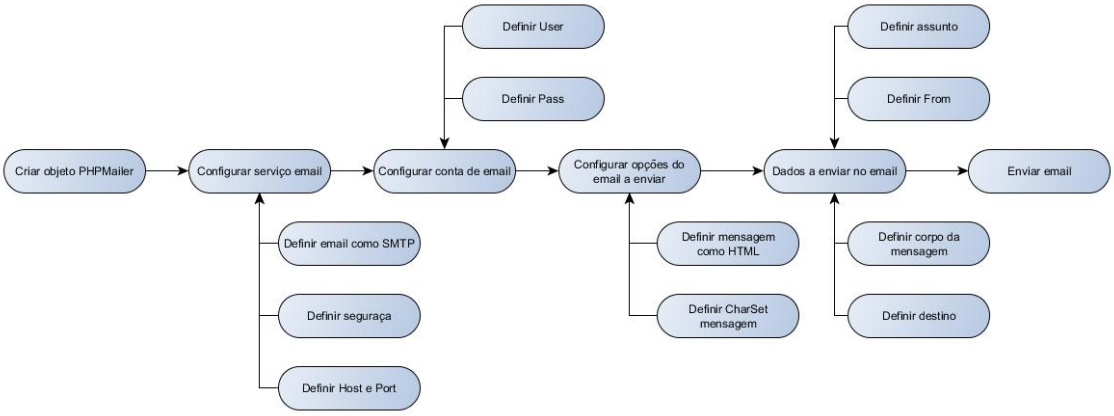


Figura 101 – Diagrama do processo de envio de *email* utilizando *phpMailer.*

Como é possível ver no diagrama de funcionamento descrito na Figura 101, o processo de configuração e envio deste *script*, através desta biblioteca é simples, direto e prático.

Capítulo 8 – Conclusão

8.1 Conclusão

A presente monografia teve como objetivo a descrição do trabalho realizado durante o período de estágio na empresa NE2000. Nessa medida, o presente documento reúne um levantamento de todos os estudos e trabalhos realizados para o efeito, como por exemplo a descrição de múltiplas ferramentas de desenvolvimento para dispositivos móveis, ou a diferença teórica entre diferentes tipos de aplicações, ou até a descrição do funcionamento de um sistema de listagem de material, entre outros assuntos.

Durante este período foi possível adquirir novos e inúmeros conhecimentos, ideias e conceitos, a respeito do *design* e desenvolvimento de aplicações para dispositivos móveis e sistemas *web* a nível profissional. Destes conhecimentos os mais importantes a referir são:

- O desenvolvimento de aplicações para o *Windows Mobile*:
  - Extenso contacto com a ferramenta de desenvolvimento *Windows Visual Studio*;
  - Aprendizagem da linguagem de programação *C#;*
  - Aprendizagem e desenvolvimento de classes e bibliotecas de forma a facilitar o acesso a recursos do sistema, como por exemplo a classe *Hardware* e a biblioteca *Wininet.*

- O desenvolvimento de aplicações para *Google Android*:
  - Contacto com a ferramenta de desenvolvimento *Eclipse*;
  - Aprendizagem dos conceitos principais da linguagem de *Java*;
  - Utilização dos vários recursos oferecidos pelos dispositivos *Android* (controlo recursos de *hardware* e *software*).

- Aquisição dos conceitos básicos de programação orientada a objetos:
  - Esta forma de programação não só foi usada predominante em quase todas as linguagens em cima descritas e usadas durante este período de estágio, como também é atualmente a base para quase todas as novas linguagens de programação como *Python, ou Ruby.*

- Contacto com o processo de desenvolvimento profissional de soluções *web*:
  - Aprendizagem de múltiplas linguagens como *HTML*, *JavaScript* e *PHP;*
  - Desenvolvimento e manutenção de dois *web sites*.

- Conceção, desenvolvimento completo da aplicação Listagem de Material:
  - Aquisição de múltiplos conhecimentos teóricos a respeito de protocolos de comunicação e arquiteturas de funcionamentos de servidores, entre outros;
  - Utilização de todos os conhecimentos adquiridos ao longo do estágio de forma a possibilitar o desenvolvimento deste trabalho;
  - Escrita de um artigo académico a respeito deste trabalho.

De uma forma geral, é possível concluir que todas as experiências adquiridas ao longo deste período académico contribuíram para um excelente primeiro contato com o mundo profissional e deixou um desejo muito forte de continuar a trabalhar nesta área de trabalho.

8.2 Trabalhos futuros

A possibilidade de poder continuar a trabalhar dentro desta área é uma idea bastante apelativa, mas até a data deste documento só existe algum interesse por parte da empresa na continuação do desenvolvimento da aplicação Listagem de Material. No entanto, até ao momento nada foi decidido sobre o assunto.

Por outro lado, tenho em mente o desenvolvimento de três aplicações para *Android,* utilizando para isso muitos dos conhecimentos adquiridos ao longo do período de estágio, sendo estas as seguintes aplicações:

- A primeira consiste no desenvolvimento de uma aplicação com o objetivo de armazenar informação a respeito dos diversos terrenos que um individo possa possuir. Esta aplicação dependerá essencialmente da precisão do GPS ou A-GPS do dispositivo móvel.
- A segunda aplicação resume-se a utilização de múltiplos sistemas sensoriais do dispositivo móvel para analisar vibrações ou sons anómalos de um motor ou máquina em funcionamento. Esta aplicação utilizaria os acelerómetros e sensores de áudio para efectuar uma análise espectral com a ideia de identificar padrões anómalos nas frequências geradas pelos mecanismos em análise.
- Por fim, a terceira aplicação consiste na utilização de um dispositivo móvel para controlar uma rede de autómatos ou sistemas de domótica, utilizando para isso um serviço REST como forma de interface entre o dispositivo móvel e os autómatos.

Referências Bibliográficas

[1] M. Meeker's, "Mary Meeker's 2013 Internet Trends: Mobile Makes Up 15% Of All Internet Traffic, With 1.5B Users Worldwide," 2013. [Online]. Available: http://techcrunch.com/2013/05/29/mary-meeker-2013-internet-trends/. [Acedido em 19 1 2014].

[2] S. Litchfield, "The History of Psion," [Online]. Available: http://stevelitchfield.com/historyofpsion.htm. [Acedido em 16 08 2014].

[3] A. Orlowski, "Psion: the last computer," The Register, 26 01 2007. [Online]. Available: http://www.theregister.co.uk/2007/06/26/psion_special?page=2. [Acedido em 2014 08 2014].

[4] I. Sager, "Before IPhone and Android Came Simon, the First Smartphone," Bloomberg, 29 06 2012. [Online]. Available: http://www.businessweek.com/articles/2012-06-29/before-iphone-and-android-came-simon-the-first-smartphone. [Acedido em 16 08 2014].

[5] M. Mulligan e D. Card, "Sizeing The EU app economy," 25 2014. [Online]. Available: http://eurapp.eu/sites/default/files/Sizing%20the%20EU%20App%20Economy.pdf. [Acedido em 16 08 2014].

[6] Facebook , "Facebook to Acquire WhatsApp," Facebook , 19 02 2014. [Online]. Available: http://newsroom.fb.com/news/2014/02/facebook-to-acquire-whatsapp/. [Acedido em 18 08 2014].

[7] R. S. Griswold, Professional and Trade Organizations, Wiley Publishing, Inc., 2001.

[8] Symbian Freak, "The Supremely Addicting Angry Birds. Hits 42 Million Free and Paid Downloads," 22 10 2010. [Online]. Available: http://www.symbian-freak.com/news/010/12/angry_birds_hits_42_million_free_and_paid_downloads.htm. [Acedido em 18 08 2014].

[9] E. Gibson, "Games of 2013: Candy Crush Saga," Eurogamer, 24 12 2013. [Online]. Available: http://www.eurogamer.net/articles/2013-12-24-games-of-2013-candy-crush-saga. [Acedido em 18 08 2014].

[10] J. White, "Freemium App Candy Crush Saga Earns A Record-Breaking $633,000 Each Day," 9 7 2013. [Online]. Available: http://appadvice.com/appnn/2013/07/freemium-app-candy-crush-saga-earns-a-record-breaking-633000-each-day. [Acedido em 18 08 2014].

[11] IDC, "IDC Home Page," [Online]. Available: www.idc.com. [Acedido em 18 1 2014].

[12] WorklightInc, "Native, Web or Hybrid Mobile Apps?," 2011. [Online]. Available: https://www.youtube.com/watch?v=Ns-JS4amlTc&list=PL9lgpfnwzhU9imogTrjUiAzJUWHPw00Kj,. [Acedido em 19 1 2014].

[13] S. Olson, J. Hunter, B. Horgen e K. Goers, Professional Cross-Platform Mobile Development in C#, John Wiley & Son, Inc., 2012, pp. 9-10.

[14] wikipedia, “wikipedia,” wikipedia, [Online]. Available: http://www.wikipedia.org/. [Acedido em 30 7 2014].

[15] Dremel, “Dremel Home Page,” [Online]. Available: http://www.dremeleurope.com/index.html.

[16] N. Kolakowski, “Microsoft Explains Windows Phone 7 Lack of Compatibility,” 2010. [Online]. Available: http://www.eweek.com/c/a/Mobile-and-Wireless/Microsoft-Explains-Windows-Phone-7-Lack-of-Compatibility-588900/. [Acedido em 23 6 23].

[17] J. MARKOFF, “I, Robot: The Man Behind the Google Phone,” The New York Times, 4 11 2007. [Online]. Available: http://www.nytimes.com/2007/11/04/technology/04google.html?_r=3&hp=&pagewanted=all&. [Acedido em 11 08 2014].

[18] S. Kirsner, “Introducing the Google Phone,” The Boston Globe, 2 9 2007. [Online]. Available: https://web.archive.org/web/20091001171031/http://www.boston.com/business/technology/articles/2007/09/02/introducing_the_google_phone/. [Acedido em 11 08 2014].

[19] F. Vogelstein, “How the Android Ecosystem Threatens the iPhone,” WIRED, 14 04 2011. [Online]. Available: http://www.wired.com/2011/04/mf_android/all/1. [Acedido em 11 08 11].

[20] B. Elgin, “Google Buys Android for Its Mobile Arsenal,” webcite, 17 08 2005. [Online]. Available: http://www.webcitation.org/5wk7sIvVb. [Acedido em 11 08 2014].

[21] Wind River, “Android for Industrial devices,” 2012. [Online]. Available: http://www.windriver.com/seminars/7839-android/6_Android_industrial.pdf. [Acedido em 20 08 2014].

[22] J. Yarow, “Business Insider,” 2014. [Online]. Available: http://www.businessinsider.com/androids-share-of-the-computing-market-2014-3. [Acedido em 24 6 2014].

[23] M. Saylor, The Mobile Wave: How Mobile Intelligence Will Change Everything., Vanguard Press., 2012, p. 33.

[24] Microsoft, “Visual Studio Home Page,” [Online]. Available: http://www.visualstudio.com/. [Acedido em 10 12 2013].

[25] Apple Computer, Inc., “Apple Computer, Inc. Finalizes Acquisition of NeXT Software Inc.,” 1997. [Online]. Available: http://web.archive.org/web/19990117075346/http://product.info.apple.com/pr/press.releases/1997/q2/970207.pr.rel.next.html. [Acedido em 19 1 2014].

[26] Apple Dev. Page, “Apple Developer,” [Online]. Available: https://developer.apple.com/. [Acedido em 19 1 2012].

[27] Eclipse , “FAQ Where did Eclipse come from?,” 2006a. [Online]. Available: http://wiki.eclipse.org/FAQ_Where_did_Eclipse_come_from%3F. [Acedido em 19 1 2014].

[28] M. Milinkovich, “IBM and Eclipse: A Decade of Software Innovation,” 2011. [Online]. Available: http://asmarterplanet.com/blog/2011/11/ibm_and_eclipse_10_years.html. [Acedido em 19 1 2014].

[29] Eclipse, “About the Eclipse Foundation,” 2014b. [Online]. Available: http://www.eclipse.org/org/. [Acedido em 19 1 2014].

[30] W.-M. Lee, Android™ Application Development Cookbook”, John Wiley & Sons, Inc., 2013.

[31] D. Intersimone, “Borland History: Why the name "Delphi?",” [Online]. Available: http://edn.embarcadero.com/article/20396. [Acedido em 19 1 2014].

[32] “Press Release: Borland forming CodeGear to focus exclusively on developer productivity,” [Online]. Available: http://edn.embarcadero.com/article/33819. [Acedido em 19 1 2014].

[33] RAD Studio XE5, “RAD Studio XE5 Home Page,” [Online]. Available: http://www.embarcadero.com/products/rad-studio.

[34] N. Friedman, “Announcing Xamarin 2.0,” 2013. [Online]. Available: http://blog.xamarin.com/announcing-xamarin-2.0/. [Acedido em 19 1 2014].

[35] M. Icaza, “Mono early history.,” 2003. [Online]. Available: http://lists.ximian.com/pipermail/mono-list/2003-October/016345.html. [Acedido em 19 1 2014].

[36] B. Schooley, “Creating iOS Apps with C# Using Xamarin.iOS,” 2013. [Online]. Available: https://www.youtube.com/watch?v=RnW7m0acxg0&list=PL9lgpfnwzhU9imogTrjUiAzJUWHPw00Kj&index=3. [Acedido em 19 1 2014].

[37] S. J. Vaughan-Nichols, “Is Mono dead? Is Novell dying?,” 2011. [Online]. Available: http://www.zdnet.com/blog/open-source/is-mono-dead-is-novell-dying/8821. [Acedido em 19 1 2014].

[38] M. Riley, “Xamarin 2.0 Review,” 2013a. [Online]. Available: http://www.drdobbs.com/tools/xamarin-20-review/240150634?pgno=1. [Acedido em 19 1 2014].

[39] M. D. Kirstein, “RhoMobile Suite Introduction,” 2012. [Online]. Available: http://www.youtube.com/watch?v=ajyt-NkzlS4&list=PL9lgpfnwzhU9imogTrjUiAzJUWHPw00Kj&index=4. [Acedido em 19 1 2014].

[40] Motorola Solutions , “Rhomobile Suite,” 2014. [Online]. Available: http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite#. [Acedido em 19 1 2014].

[41] NewCircle Training, “RhoMobile Suite Introduction,” 14 08 2012. [Online]. Available: https://www.youtube.com/watch?v=ajyt-NkzlS4&index=4&list=PL9lgpfnwzhU9imogTrjUiAzJUWHPw00Kj. [Acedido em 20 08

2014].

[42] M. Riley, “Product Review: Rhomobile's RhoMBUS Smartphone App Development Product Suite,” 2011b. [Online]. Available: http://devproconnections.com/development/product-review-rhomobiles-rhombus-smartphone-app-development-product-suite. [Acedido em 19 1 2014].

[43] Adobe , “Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap,” 2011 . [Online]. Available: http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html. [Acedido em 19 1 2014].

[44] J. Fermoso, “PhoneGap Seeks to Bridge the Gap Between Mobile App Platforms,” 2009. [Online]. Available: http://gigaom.com/2009/04/05/phonegap-seeks-to-bridge-the-gap-between-mobile-app-platforms/. [Acedido em 19 1 2014].

[45] Icenium Blog, “Demystifying Apache Cordova and PhoneGap,” 2013. [Online]. Available: http://www.icenium.com/blog/icenium-team-blog/2013/03/26/demystifying-apache-cordova-and-phonegap. [Acedido em 19 1 2014].

[46] Intel, “The Development of Mobile Applications using HTML5 and PhoneGap* on Intel® Architecture-Based Platforms,” 2012. [Online]. Available: http://software.intel.com/en-us/articles/the-development-of-mobile-applications-using-html5-and-phonegap-on-intel-architecture-based. [Acedido em 19 1 2014].

[47] D. Berthiaume, “Adobe Rolls Out PhoneGap Build for Cloud-Based Mobile Apps,” 2012. [Online]. Available: http://www.cmswire.com/cms/customer-experience/adobe-rolls-out-phonegap-build-for-cloudbased-mobile-apps-017519.php. [Acedido em 19 1 2014].

[48] A. Wilhelm, “Telerik launches Icenium, a cloud-based development platform for hybrid iOS and Android apps,” 2012. [Online]. Available: http://thenextweb.com/apps/2012/10/22/telerik-launches-icenium-a-cloud-based-development-platform-for-mobile-apps/#!sEpnn. [Acedido em 19 1 2014].

[49] J. Cowart, “What's the Difference Between Icenium and PhoneGap Build?,” 2013. [Online]. Available: http://www.icenium.com/blog/icenium-team-blog/2013/05/01/what%27s-the-difference-between-icenium-and-phonegap-build-. [Acedido em 19 1 2014].

[50] Icenium, “Icenium Home Page,” [Online]. Available: http://www.icenium.com/. [Acedido em 19 1 2014].

[51] P. Affari, “Datalogic si conferma leader nelle soluzioni per il settore sanitario,” 2012. [Online]. Available: http://finanza.lastampa.it/notizie/0,475628/Datalogic_si_conferma_leader_nelle_soluzioni_per.aspx. [Acedido em 19 1 2014].

[52] The marconi society, “Romano Volta,” [Online]. Available: http://www.marconisociety.org/aboutus/board/romano_volta.html. [Acedido em 19 1 2014].

[53] C. H. Blickenstorfer, “Datalogic Lynx PDA,” 2012. [Online]. Available:

http://www.ruggedpcreview.com/3_handhelds_datalogic_lynx.html. [Acedido em 19 1 2014].

[54] D. Rubino, “GPS vs. aGPS: A Quick Tutorial,” 2009. [Online]. Available: http://www.wpcentral.com/gps-vs-agps-quick-tutorial. [Acedido em 19 1 2014].

[55] Microsoft, "Microsoft.WindowsMobile.Forms Namespace," Microsoft, 29 3 2009. [Online]. Available: http://msdn.microsoft.com/en-us/library/Microsoft.WindowsMobile.Forms.aspx. [Accessed 02 08 02].

[56] Microsoft, “Windows Embedded Handheld 6.5,” Microsoft, [Online]. Available: http://www.microsoft.com/windowsembedded/en-us/windows-embedded-handheld-6-5.aspx. [Acedido em 08 12 2014].

[57] J.-P. Gravel, "Signature Box that Makes the Signature Look Right," 22 2 2011. [Online]. [Accessed 2014http://www.codeproject.com/Articles/158339/Signature-Box-that-Makes-the-Signature-Look-Right 08 03].

[58] D. Buytaert, "The history of MySQL AB," 2010. [Online]. Available: http://buytaert.net/the-history-of-mysql-ab. [Accessed 27 4 2014].

[59] Database Friends, “History of MySQL,” 2014. [Online]. Available: http://www.databasefriends.co/2014/02/history-of-mysql.html. [Acedido em 27 04 2014].

[60] R. T. FIELDING, “Principled Design of the Modern,” 2002. [Online]. Available: http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf. [Acedido em 28 4 2014].

[61] Android, "HttpClient," android, [Online]. Available: http://developer.android.com/reference/org/apache/http/client/HttpClient.html. [Accessed 13 08 2014].

[62] Android, “Security with HTTPS and SSL,” Android, [Online]. Available: https://developer.android.com/training/articles/security-ssl.html. [Acedido em 13 08 2014].

[63] Apache, “Apache Commons Net,” Apache, 28 06 2013. [Online]. Available: http://commons.apache.org/proper/commons-net/. [Acedido em 13 08 2014].

[64] Maintenance Performance Measurement and Management Conference 2014, “Call for papers,” [Online]. Available: https://cemuc.dem.uc.pt/MPMM2014/callforpapers.php. [Acedido em 14 08 2014].

[65] H. Santos, A. Simões, I. Fonseca and T. Farinha, "Mobile Applications and its Potential to Maintenance," in *Proceedings of Maintenance Performance Measurement and Management (MPMM) Conference 2014*, Coimbra, Imprensa da Universidade de Coimbra, 2014, p. 103.

[66] “windows mobile 6.5 隐藏 左下角（左上角）的开始按钮 叉号关闭按钮,” 19 08 2013. [Online]. Available: http://blog.csdn.net/code_style/article/details/10070131. [Acedido em 01 08 2014].

[67] "Windows Mobile: Hide StartButton in WinMo 6.5.x," 11 10 2010. [Online]. [Accessed 01 08 2014].

[68] Microsoft, "Windows Internet Services (WinInet)," 4 8 2010. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa917828.aspx. [Accessed 08 08 2014].

[69] Boutell.Com, Inc, “GD Graphics Library Home Page,” [Online]. Available: http://www.boutell.com/gd/. [Acedido em 16 08 2014].

[70] F. T. v. De Ven, “Compact Framework Process class that supports fully specified file paths,” 2009. [Online]. Available: http://www.codeproject.com/Articles/36841/Compact-Framework-Process-class-that-supports-full. [Acedido em 28 1 2014].

[71] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. [Accessed 28 8 2014].

[72] A. S. Incorporated., “Adobe Systems Home Page,” [Online]. Available: http://www.adobe.com/. [Acedido em 23 07 2014].

[73] H. M. B. d. Santos, "Maintenance Performance Measurement and Management Conference 2014," NE2000, 2 2 2014. [Online]. Available: https://cemuc.dem.uc.pt/MPMM2014/index.php. [Accessed 23 07 2014].

[74] HttpWebRequest Class, [Online]. Available: http://msdn.microsoft.com/en-us/library/system.net.httpwebrequest%28v=vs.100%29.aspx.

[75] Microsoft, “System.Net Namespace,” Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/System.Net%28v=vs.100%29.aspx. [Acedido em 07 08 2014].

[76] Encontro Nacional de Engenharia e Gestão Industrial de 2014, “Encontro Nacional de Engenharia e Gestão Industrial de 2014,” 24 3 2014. [Online]. Available: https://cemuc.dem.uc.pt/ENEGI2014/. [Acedido em 23 07 2014].

Anexos

Anexo A – Principais Classes Desenvolvidas para o Uso do *Datalogic Lynx*

Devido a complexidade do SDK disponibilizado para o dispositivo *Lynx* e das diversas limitações da *framework .Net Compact*, mais informação consultar anexo D.1.1, foi necessário desenvolver um conjunto de classes e funções de forma a aceder mais facilmente aos vários componentes de *hardware* e *software* do dispositivo.

É pretendido então neste capítulo descrever o funcionamento de algumas dessas classes, de forma a simplificar a descrição das aplicações desenvolvidas para este dispositivo.

A.1 Classe *Hardware*

A.1.1 Enquadramento

A classe *Hardware* é provavelmente uma das mais complexas deste trabalho. Surgiu em primeiro lugar da necessidade de padronizar o acesso e controlo dos vários componentes de *hardware* específicos ao dispositivo *Lynx*, mas de forma a contemplar a possibilidade de simular esse acesso, caso a aplicação esteja a ser executada num ambiente simulado, como por exemplo num emulador.

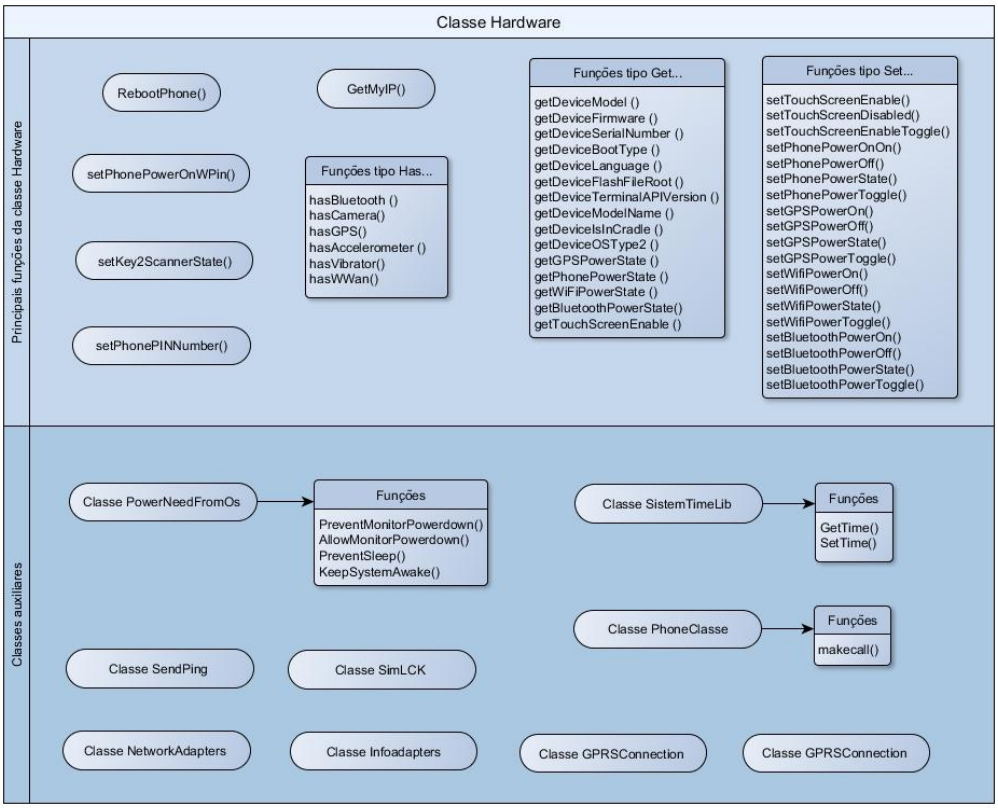


Figura 102 – Diagrama da classe *Hardware*

Como é possível ver pelo diagrama descrito na Figura 102, esta classe é composta por diversas funções e classes auxiliares dedicadas ao controlo dos diversos componentes dos dispositivos móveis e seu sistema operativo.

A.1.2 Princípio de Funções do Sistema de Emulação de *Hardware*

Esta classe ao ser inicializada, envia um pedido ao dispositivo móvel de forma a este lhe indicar os componentes de *hardware* e versão de *software* que possui. Caso o dispositivo seja incapaz de responder a este pedido ou os componentes necessários não existam no dispositivo, a classe *Hardware* deve assumir que o código está a correr num ambiente simulado, assumindo assim o controlo da virtualização do *hardware* não disponibilizado pelo emulador (Figura 103).

Figura 103 – Diagrama funcionamento da classe *Hardware* em modo emulador

Caso contrário, a classe associa os estados às variáveis "lc_has…", ao seu *hardware* respetivo e estes ficam a espera de serem ligados (Figura 104 e Figura 105).

Figura 104 – Diagrama funcionamento da classe *Hardware* em modo normal

É possível controlar através desta classe as seguintes ações:

- Obter informação acerca do modelo, *firmware*, número de série, linguagem, entre muitas outras informações a respeito do dispositivo;
- Controlar a alimentação do *GPS*, ecrã tátil, *Wifi*, *Bluetooth* e *Phone*;
- Ligar e desligar os dispositivos de *GPS*, ecrã tátil, *Wifi*, *Bluetooth* e *Phone*;
- Definir automaticamente o número de *PIN* do dispositivo;
- Definir os botões responsáveis pelo *scanner* a *laser*;
- Instruir o dispositivo a fazer *reboot*.

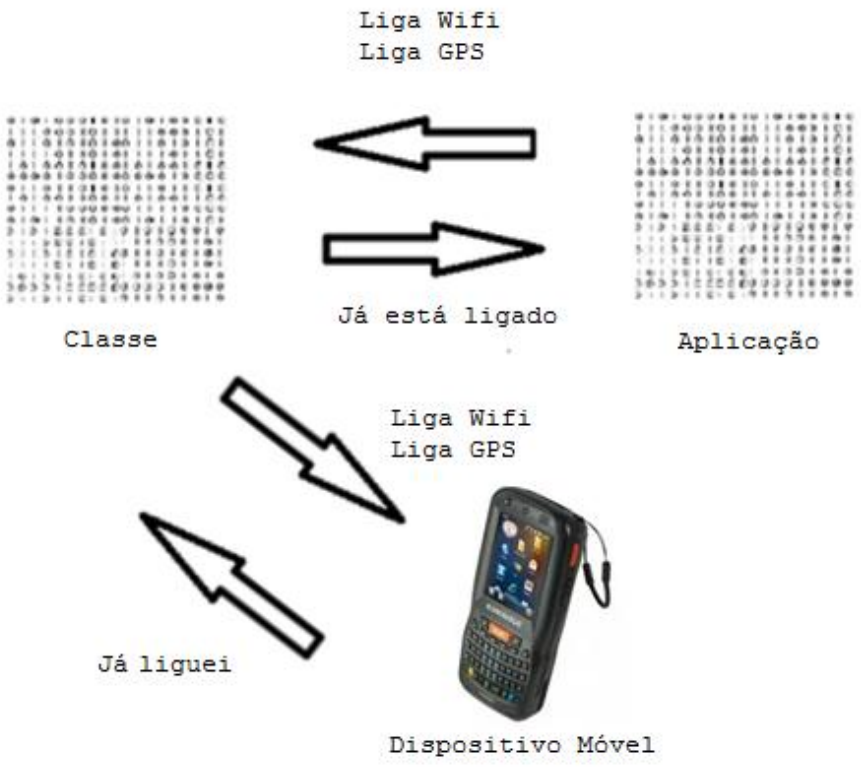


Figura 105 – Representação do pedido de ativação de *hardware*

A.1.3 Função *RebootPhone*

Com esta função é possível executar um pedido de *reboot* ao dispositivo, tal como definir o tipo de *reboot*. Isto é possível através do uso de algumas das funções fornecidas pela API da *Datalogic*.

É possível então escolher entre três tipos de *reboot*:

- *Clean boot* (opt 0) – Apaga toda a memória, incluído a memória persistente;
- *Cold boot* (opt 1) – Apaga toda a memória *RAM*;
- *Warm boot* (opt 2) – Apaga somente a memoria volátil *da RAM*.

```
public bool rebootphone(int opt) {
            try {
                bool b;
```

```
                    Device.BootType BootType = Device.BootType.Error;
                    switch(opt) {
                        case 0: BootType = Device.BootType.Clean; break;
                        case 1: BootType = Device.BootType.Cold; break;
                        default:
                        case 2: BootType = Device.BootType.Warm; break;
                    }
                    if (BootType != Device.BootType.Error) {
                        b = Device.Reset(BootType);
                        return true;
                    } else return false;
                } catch { return false;}
            }
```

A.1.4 Função *setPhonePowerOnWPin* e *setPhonePINNumber*

Esta função permite configurar o dispositivo de forma a este inserir automaticamente o valor de *PIN* do cartão *SIM*.

```
Bool isSet = setPhonePINNumber(String pin_number);
```

A função *setPhonePowerOnWPin* ao ser chamada, verifica se o sistema de rádio (transmissor de *GSM* ou *HSDPA*) já se encontra ligado. Caso não esteja ligado, a função tenta estabelecer a ligação ao sistema através do comando ”setPhonePowerOnOn();”, se mesmo assim o dispositivo de rádio não se iniciar, a função é abortada.

```
setPhonePowerOnOn();
//df.TopMost = true;
Application.DoEvents();
if (getPhonePowerState() == false)
{
    setPhonePowerOnOn();
}
if (getPhonePowerState() == false) {
        //df.TopMost = state;
    return 0;
}
```

Por outro lado se o pedido ”*setPhonePowerOnOn*();”, for realizado com sucesso, a função *setPhonePINNumber* é chamada para introduzir automaticamente o número *PIN*.

```
public bool setPhonePINNumber(String pin_number) {
            SimLCK lv_sim;

            try
            {
                lv_sim = new SimLCK();
                if (!lv_sim.UnlockSIM(pin_number)) return false;
            }
            catch (Exception e)
            {
                String str = e.ToString();
            }
            return true;
}
```

Esta função por sua vez utiliza a classe *SimLCK,* a qual por fim comunica ao dispositivo, via *DllImport,* que valor este deve assumir como *PIN*.

A.1.5 Função *GetMyIP*()

É pretendido com esta função o acesso rápido ao endereço de *IP* do dispositivo.

```
String[] MyIP = GetMyIP(out uint cnt, out bool has_valid_ip);
```

Para isso, este faz um pedido ao *DNS* local através da função de sistema *System.Net.Dns.GetHostEntry*, a qual deve enviar o endereço de *IP* respetivo.

```
IPHostEntry HostEntry = System.Net.Dns.GetHostEntry((System.Net.Dns.GetHostName()));
```

No caso do dispositivo não receber qualquer tipo de endereço, (manual ou automático), o *system.Net.Dns.GetHostEntry*(), deve devolver o endereço 127.0.0.1 o qual deve ser prontamente ignorado.

```
if (addresses[cnt] == "127.0.0.1") passei = false;
```

A.1.6 *Has, Get, Set*

A classe *Hardware* é a responsável pela gestão dos vários componentes do dispositivo Lynx. Tendo isso em conta foram incorporadas várias funções de acesso ao estado dos diversos dispositivos.

Todas as funções começadas por "*has*" permitem confirmar a existência dos específicos componentes de *hardware*:

- *hasBluetooth* – existe *Bluetooth*;
- *hasCamera* – existe câmara;
- *hasGPS* – existe *GPS*;
- *hasAccelerometer* – existe acelerómetro;
- *hasVibrator* – existe vibrador;
- *hasWWan* – existe *Wireless*.

As funções começadas por "*get*" fornecem o estado em que se encontra o componente ou informação respetiva ao *software*:

- *getDeviceModel* – devolve o modelo do dispositivo;
- *getDeviceFirmware* – devolve os dados respetivos ao *firmeware*;
- *getDeviceSerialNumber* – devolve o numero de série do dispositivo;
- *getDeviceBootType* – devolve os tipos de *reboot* do dispositivo;
- *getDeviceLanguage* – devolve linguagem do dispositivo;
- *getDeviceFlashFileRoot* – devolve o nome do *flash disk drive*;
- *getDeviceTerminalAPIVersion* – devolve a versão API do dispositivo;
- *getDeviceModelName* – devolve o nome do modelo de *PDA*;
- *getDeviceIsInCradle* – indica se o dispositivo se encontra em modo *cradle*;
- *getDeviceOSType2* – devolve o tipo de sistema operativo do dispositivo;
- *getGPSPowerState* – devolve o estado de alimentação do *GPS*;
- *getPhonePowerState* – devolve o estado de alimentação do sistema de comunicação;
- *getWiFiPowerState* – devolve o estado de alimentação do *Wireless*;
- *getBluetoothPowerState* – devolve o estado de alimentação do *Bluetooth*;
- *getTouchScreenEnable* – devolve o estado de alimentação do *TouchScreen*;

As funções começadas por "*set*" permitem definir o estado do componente:

- Alimentação do *TouchScreen*:
  - *setTouchScreenEnable*;
  - *setTouchScreenDisabled*;
  - *setTouchScreenEnableToggle*.
- Alimentação do *GSM* ou *HSDPA*:
  - *setPhonePowerOnOn*;
  - *setPhonePowerOff*;
  - *setPhonePowerState*;
  - *setPhonePowerToggle*.
- Alimentação do *GPS*:
  - *setGPSPowerOn*;
  - *setGPSPowerOff*;
  - *setGPSPowerState*;
  - *setGPSPowerToggle*.
- Alimentação do *Wireless*:
  - *setWifiPowerOn*;
  - *setWifiPowerOff*;
  - *setWifiPowerState*;
  - *setWifiPowerToggle*.
- Alimentação do *Bluetooth*:
  - *setBluetoothPowerOn*;
  - *setBluetoothPowerOff*;
  - *setBluetoothPowerState*;
  - *setBluetoothPowerToggle*;

A.1.7 Função *setKey2ScannerState*

Como foi referido anteriormente nesta obra, o dispositivo *Lynx* tem incorporado um leitor de código de barras. O processo de leitura do *scanner* é iniciado no momento em que é pressionado algum dos três botões de *scan* marcados a vermelho na Figura 106.

Figura 106 – Botões de *scan*.

Este acontecimento nem sempre é desejável. Então, tendo em conta este problema foi desenvolvido a função *setKey2ScannerState*, com o intuito de obter o controlo sobre o uso do *scanner* a *laser*.

```
        public void setKey2ScannerState(bool liga)
        {
            Device.TriggerInputType
scanOnAll=Device.TriggerInputType.BarcodeOrCameraShutter;
          Device.TriggerInputType scanOffAll= Device.TriggerInputType.None;
            try {
                if (!liga) {
                    Device.SetTriggerType(Device.TriggerId.Scan, scanOffAll);
                    Device.SetTriggerType(Device.TriggerId.LeftSide, scanOffAll);
                    Device.SetTriggerType(Device.TriggerId.RightSide, scanOffAll);
                } else {
                    Device.SetTriggerType(Device.TriggerId.Scan, scanOnAll);
                    Device.SetTriggerType(Device.TriggerId.LeftSide, scanOnAll);
                    Device.SetTriggerType(Device.TriggerId.RightSide, scanOnAll);
                }
            } catch {}
        }
```

Se for pretendido desativar os botões de scan utiliza-se "*setKey2ScannerState(true);",* caso contrário "*setKey2ScannerState(false);".*

A.1.8 Classe *PowerNeedFromOs*

Esta classe permite definir os vários estados de hibernação do dispositivo enquanto uma aplicação desempenha a sua atividade, mais especificamente, controlo do modo de hibernação (*Sleep Mode*) e o controlo do ecrã do dispositivo. Isto é possível através das seguintes funções:

- *PreventMonitorPowerdown()* → Impede que o monitor seja desligado automaticamente;
- *AllowMonitorPowerdown()* → Autoriza que o monitor seja desligado automaticamente;
- *PreventSleep()* → Impede que o sistema entre em modo hibernação de forma permanente;
- *KeepSystemAwake()* → Acorda o sistema ou reinicia o temporizador que define a entrada no modo hibernação;

Todos os pedidos desta classe são enviados para o *dll* "*kernel32.dll*" através do método de *Dll Import*

```
        [DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]
         static extern EXECUTION_STATE SetThreadExecutionState(EXECUTION_STATE esFlags);
```

A.1.9 Classe *SistemTimeLib*

Esta classe foi desenvolvida com o intuito de aceder ao registo do relógio interno do sistema operativo, utilizando para isso uma chamada ao *dll "coredll.dll".*

```
        [DllImport("coredll.dll")]
          private extern static void GetSystemTime(ref SYSTEMTIME lpSystemTime);
```

```
[DllImport("coredll.dll")]
 private extern static uint SetSystemTime(ref SYSTEMTIME lpSystemTime);
```

Obter o tempo atual do sistema operativo do dispositivo:

```
DateTime correntTime = GetTime();
```

Atualizar o tempo do sistema operativo do dispositivo:

```
SetTime(DateTime dt);
```

A.1.10 Classe *SendPing*

Esta classe permite o envio e receção de um pedido *ping* a um dado endereço web. Isto é possível através das classes *Ping* e *PingReply*, fornecidas pela biblioteca *OpenNETCF.Net.NetworkInformation*.

Ao utilizar a função *public int SendPing(string Ip)* é esperado um de três resultados:

- Resultado 1 – O endereço respondeu com sucesso ao pedido *ping*;
- Resultado 0 – O endereço não respondeu com sucesso ao pedido *ping*;
- Resultado -1 – Ocorreu um erro durante a execução do pedido de *ping*.

```
try
{
    pingReply = ping.Send(Ip, 60);
    if (pingReply.Status == IPStatus.Success)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
catch (Exception)
{
    return -1;
}
```

A.1.11 Classe *SimLCK*

Esta classe foi baseada na *Api Accessing Phone* da *Microsoft*, e tem como intuito, a gestão de informação relevante ao cartão *SIM*. Um bom exemplo disto, é o pedido de desbloqueio do cartão *SIM* feito pela função *setPhonePINNumber* o qual usa a função *UnlockSIM* desta classe para introduzir o código *PIN* automaticamente sem a intervenção do utilizador.

```
public bool setPhonePINNumber(String pin_number) {
    SimLCK lv_sim;

    try
    {
        lv_sim = new SimLCK();
        if (!lv_sim.UnlockSIM(pin_number)) return false;
    }
    catch (Exception e)
```

```
            {
                String str = e.ToString();
            }
            return true;
        }

       ////////////////////////////////////////////////////

       public bool UnlockSIM(string Password)
               {
               // check if phone is on, if not, turn it on
               // for the sim handle
               int hSim = 0;               // for the results
               bool pfEnabled = false;     // the PIN
               uint lpdwLockedState = 0;   // for function return code
               int result = 0;             // for the method result
               bool method = false;        // init the sim card
               if (SimInitialize(0, 0, 0, ref hSim) != 0) return false;

               const uint SIM_LOCKFACILITY_SIM = 8;
               const uint SIM_LOCKEDSTATE_SIM_PIN = 2;

               SimGetPhoneLockedState(hSim, out lpdwLockedState);
               if (lpdwLockedState!=1) {
                   if (lpdwLockedState == 2) {
                       result = SimUnlockPhone(hSim, Password, Password);
                       if (result == 0) method = true;
                   }
               }
               // ok, so sim is open, get the lock status
               result = SimGetLockingStatus(hSim, SIM_LOCKFACILITY_SIM, Password, ref
pfEnabled);
               if (result == 0) method=true;
               // all done, so close the sim
               SimDeinitialize(hSim);
               // return the result of the method
               return method;
               }
           }
```

A.1.12 Classe *GPRSConnection*

Devido ao sistema não suportar a gestão automática de comunicações *TcpClient* ao nível do *socket* por via *GPRS*, foi necessário implementar esta classe para facilitar a inicialização e finalização de todas as comunicações *GPRS* que o programa possa necessitar de fazer.

A.1.13 Classes *NetworkAdapters* e *Infoadapters*

Estas duas classes ao trabalharem em conjunto, permitem obter várias informações a respeito da rede local e na forma como o dispositivo se encontra ligado a esta, obtendo assim, entre outras coisas, o endereço *IP* do dispositivo, tal como o da *Gateway* da rede e até o estado do serviço *DHCP* na rede.

A.1.14 Classe *PhoneClasse*

Foi necessário desenvolver uma forma de iniciar chamadas telefónicas automáticas através do dispositivo móvel. Para isso, utilizaram-se algumas das funções fornecidas pela biblioteca *OpenNETCF.Telephony* para desenvolver a classe *PhoneClass*. Esta classe ao ser chamada,

analisa as capacidades do dispositivo e cria uma ligação com a rede telefónica, passando de seguida a gerir todas as comunicações realizadas por essa linha sendo assim possível efetuar chamadas através da função *makecall().*

```
call = line.MakeCall(phonenumber, 1, asprivatenumber);
```

A.2 Classe *ThreadStatus*

Quase todos, se não todos, os sistemas operativos atuais oferecem algum tipo de processamento de tarefas ou *tasks* em paralelo, permitindo assim a execução de várias atividades em simultâneo. Este fator torna-se importante ao considerar que com o aumento do grau de complexidade de uma aplicação, aumenta a necessidade de efetuar múltiplas operações em simultâneo. Este conceito é geralmente chamado *multithread*.

Com esse intuito em mente, desenvolveu-se esta classe para gerir um conjunto de recursos distintos, tais como, o sistema de *Wireless*, *GPS*, câmara, envio de *email* e de ficheiro via *FTP*, entre outros em simultâneo.

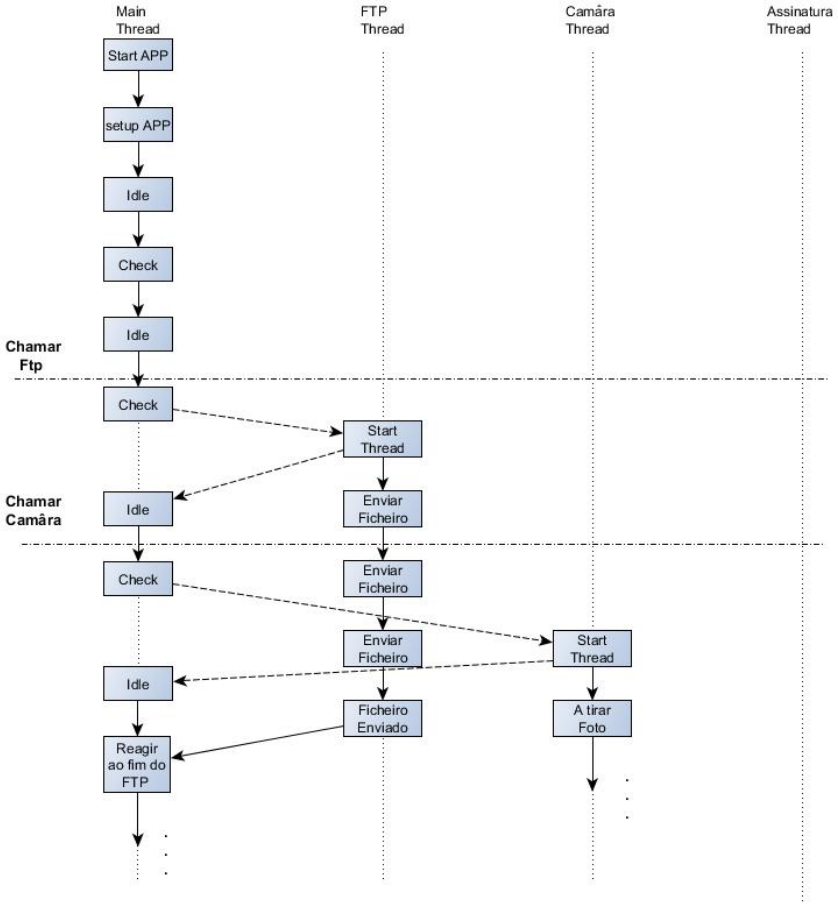


Figura 107 – Exemplo do funcionamento desta classe

Como é possível ver pelo exemplo descrito na Figura 107, a classe ao iniciar gera uma *thread* principal, a qual periodicamente verifica se foi requerida alguma das aplicações geridas por esta e voltando ao estado inerte ou *Idle* enquanto espera pelo período seguinte.

No caso de ocorrer pedido, a classe *ThreadStatus*, verifica qual foi o pedido realizado e inicia-o numa *thread* à parte, enquanto a *thread* principal, continua o seu ciclo periódico à espera de novo pedido, permitindo assim por exemplo o envio de uma imagem via *FTP* ao mesmo tempo que o utilizador está a enviar um *correio eletrónico* ou capturar uma foto através da câmara. Assim que o recurso termine a sua operação, envia mensagens via *handler* para a classe *ThreadStatus* a informar a sua conclusão.

A.3 Classe *GUIControl*

No decorrer do desenvolvimento de algumas das aplicações descritas no capítulo 5, verificou-se a necessidade de remover alguns dos botões *GUI* do *Windows Mobile 6.5*, mais especificamente o botão *Start* e *Done* (X), Figura 108, pois estes mantêm-se ativos durante o decorrer de uma aplicação, o que pode não interessar.

Figura 108 – A esquerda com os botões Start e Done a direita sem eles.

Esta classe foi desenvolvida a partir da documentação encontrada em artigos *online* [66] [67], como por exemplo *Windows CE Programming*[36] e numa forma muito simplificada, altera alguns dos registos *LocalMachine,* forçando assim as alterações pretendidas no *GUI*.

A.4 Classe *SignatureControl*

Esta classe foi concebida durante o desenvolvimento da aplicação *HandSignature* com o intuito de criar, gerir e manipular uma dada área do *GUI* da aplicação de forma a captar uma parte da rúbrica introduzida pelo utilizador e por fim inseri-la numa imagem de forma a autenticar, (Figura 109).

[36] *Site*: http://www.hjgode.de/wp/ (01/08/2014)

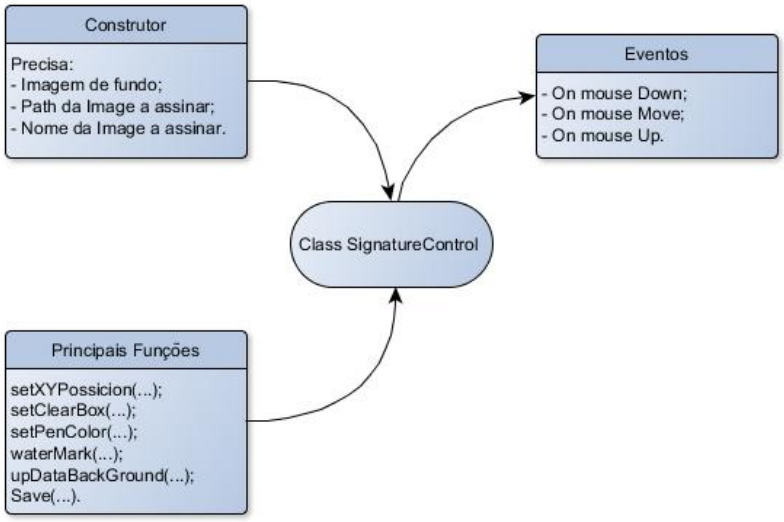


Figura 109 – Representação gráfica da class *SignatureControl*

A.4.1 Descrição das Principais Funções

Devido à complexidade desta classe, não é viável fazer uma descrição sistemática de todas as funções desenvolvidas, mas destas as mais significativas são:

- *setXYPossicion*(…) – Definir a posição da assinatura relativamente à imagem que é pretendida assinar;
- *setPenColor*(…) – Definir a cor da assinatura;
- *setClearBox*(…) – Instrui a classe a apagar a assinatura;
- *waterMark*(…) – Informa a classe que ao salvar a assinatura na imagem deve esbater as cores à volta desta de forma a ser mais visível o fundo;
- *upDateBackGround*(…) – Permiter alterar a imagem a ser certificada pela rúbrica;
- *save*(…) – Instruir a classe a introduzir a assinatura na imagem.

A.4.2 Eventos da Classe

Esta classe é capaz de detetar e atuar sobre os eventos resultantes da inscrição no ecrã do dispositivo, (*On Mouse Down*, *On Mouse Move* e *On Mouse UP*). Isto é possível porque esta classe é uma extensão da classe *Control*, (classe do sistema), a qual já tem esses eventos pré-programados. Sendo assim, é só necessário a classe *SignatureControl* executar um *override* sobre esses eventos de forma reagir a estes.

- ***On Mouse Down*** – Este evento é executado quando um objeto prime o ecrã do dispositivo, por exemplo uma *pen*. A classe começa a recolher o ponto X e Y inicial e a desenhar esse ponto no ecrã.
- ***On Mouse Move*** – Neste evento o objeto desloca-se no ecrã sem deixar de o premir. A classe continua a recolher o ponto X e Y atual do objeto e a desenhar a assinatura no ecrã.

- ***On Mouse Up*** – Finamente este evento ocorre quando o objeto liberta a pressão sobre o ecrã, finalizando assim a captação das coordenadas X e Y e o desenho da assinatura por parte da classe.

A.5 Biblioteca *WinINet*

A.5.1 Enquadramento

Durante o desenvolvimento da aplicação Listagem de Material, (capítulo 6), foi decidido que seria necessário implementar nesta os protocolos *HTTPs* e *FTP* como forma de comunicação com o servidor.

Infelizmente, chegou-se rapidamente à conclusão que não seria possível utilizar as funcionalidades disponibilizadas pela *framework .Net Compact*, visto esta não oferecer suporte aos protocolos pretendidos.

Este facto obrigou ao desenvolvimento da biblioteca *WinINet*. Esta biblioteca estabelece uma comunicação direta com o *dll* do sistema *WinINet*, a qual é o responsável pela gestão dos protocolos *HTTP*, *HTTPs* e *FTP*, permitindo assim o acesso a estes recursos.

Esta biblioteca teve como base toda a documentação disponível pela *Microsoft* [68], e pelo *site pinvoke*[37].

A.5.2 Composição da Classe *WinINet*.

Como é possível ver pela Figura 110, esta biblioteca é composta por cinco classes parciais, as quais em conjunto formam a classe *WinINet* e três classes do tipo extinção, que simplificam os processos de configuração para os seus sistemas de comunicação específicos.

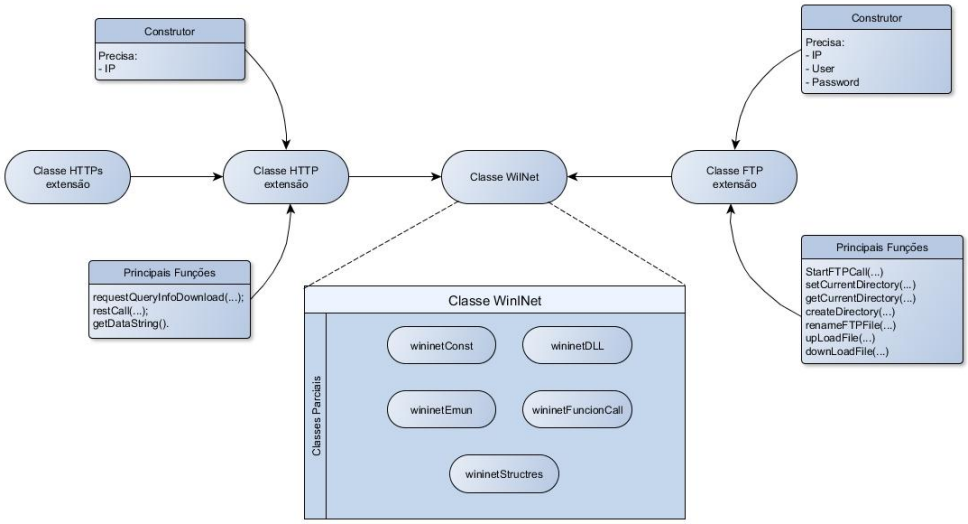


Figura 110 – Partes constituintes da biblioteca *WinINet API*

[37] *Home Page:* http://www.pinvoke.net (08/08/2014)

O intuito das classes parciais é obter uma forma de otimizar o processo de organização e desenvolvimento da classe *WinINet*, separando-a em cinco componentes distintas, cada uma responsável por uma área de trabalho:

- *WininetConst* – É nesta classe que são armazenadas todas as variáveis constantes da classe *WinINet*;
- *WininetEmun* – É nesta classe que são armazenadas as listas de todos os tipos *Enum* da classe *WinINet*;
- *WininetStructres* – São armazenadas nesta classe todas as variáveis do tipo estrutura (*StructLayout*) da classe *WinINet*;
- *WininetFuncionCall* – É nesta classe parcial que se encontram todas as funções necessárias ao funcionamento da classe *WinINet*.
- *WininetDll* – É nesta classe que se realizam todas as chamadas ao *dll Wininet*.

Devido a complexidade da classe *WinINet*, optou-se por desenvolver três classes extensão como forma de isolar os principais protocolos geridos por esta classe. Esta separação, permitiu um processo de desenvolvimento mais coerente e um acesso mais intuitivo a esta biblioteca.

Descrição das classes do tipo extinção:

- Classe *HTTP* – Extensão da classe *WinINet*, configurar classe em modo *HTTP*;
    - Classe *HTTPs* – Extensão da classe *HTTP*, configurar classe em modo *HTTPs*;
- Classe *FTP* – Extensão da classe *WinINet*, configurar classe em modo FTP;

Principais funções das classes *HTTP* e *HTTPs*:

- *requestQueryInfoDownload*(…) – Permite obter informação a respeito da página acedida;
- *restCall*(…) – Define e envia um requisito ao servidor;
- *getDataString*() – Devolve os dados de resposta do servidor.

Principais funções da classe *FTP*:

- *StartFTPCall*(…) – Inicia a comunicação *FTP* entre a aplicação e o servidor;
- *getCurrentDirectory*(…) – Obter o diretório atual;
- *setCurrentDirectory*(…) – Definir o diretório atual;
- *createDirectory*(…) – Criar novo diretório no servidor;

- *renameFTPFile*(…) – Alterar o nome de um ficheiro no servidor;
- *upLoadFile*(…) – Enviar para o servidor um ficheiro;
- *downLoadFile*(…) – Descarregar do servidor um ficheiro.

Anexo B – Serviço *Web*: Gerador de Etiquetas

É pretendido implementar neste serviço *web* um sistema que receba uma "*array"* de dados, efetue o seu processamento de forma a gerar uma etiqueta e identificação, por exemplo uma etiqueta de pescas, ou de transporte e no formato imagem (*JPG*, *PNG* ou *PDF*).

B.1 Conversor de *Array* de Dados em Imagem

B.1.1 Enquadramento da Aplicação

Como foi referido anteriormente é pretendido desenvolver uma função em *PHP* capaz de gerar uma etiqueta, a partir de uma array de dados contendo toda a informação relevante, exemplo: nº de identificação, data em que a mesma foi lançada, nome do recetor, etc. Para isso, foi usada a biblioteca gráfica *GD*.

B.1.2 Biblioteca *GD*

A biblioteca *GD* foi desenvolvida em 1994 pelo *web designer Thomas Boutell*, com o intuito de ser um *software* multiplataformas capaz de gerar imagens em *GIF*, *JPEG* e *PNGs*. A linguagem de programação nativa desta biblioteca é o *ANSI C*, mas esta é capaz de fazer interface com várias outras linguagens de programação, como por exemplo *PHP* [69]. Encontra-se, de momento, coberta pela licença *BSD* o que permite um uso gratuito e é a biblioteca gráfica recomendada no *site* oficial de *PHP*. O site oficial do *PHP*[38], dispõe gratuitamente toda a documentação destra biblioteca no seu *site*[39].

De referir que esta biblioteca já fora anteriormente usada no processo de inscrição do *site* MPMM 2014, mais especificamente na página de inscrição do evento, para apresentar uma imagem com código *anti-spam*, capítulo 7.4.4.3.

[38] *Site PHP*: https://php.net/ (16/08/2014).
[39] Documentação da biblioteca GD: https://php.net/manual/pt_BR/book.image.php (16/08/2014).

B.1.3 Diagrama do Sistema

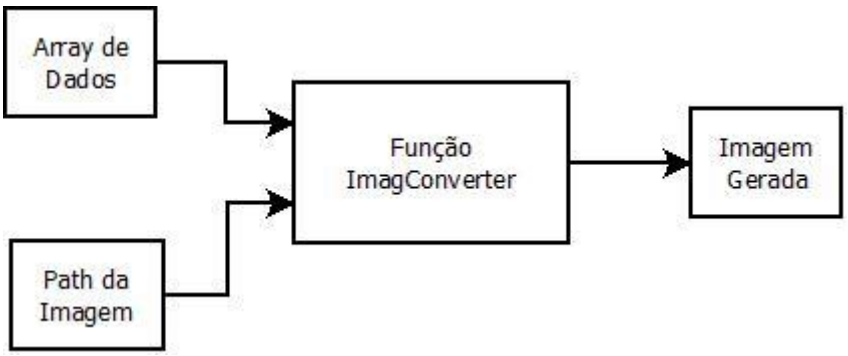


Figura 111 – Diagrama Base da função *ImagConverter.*

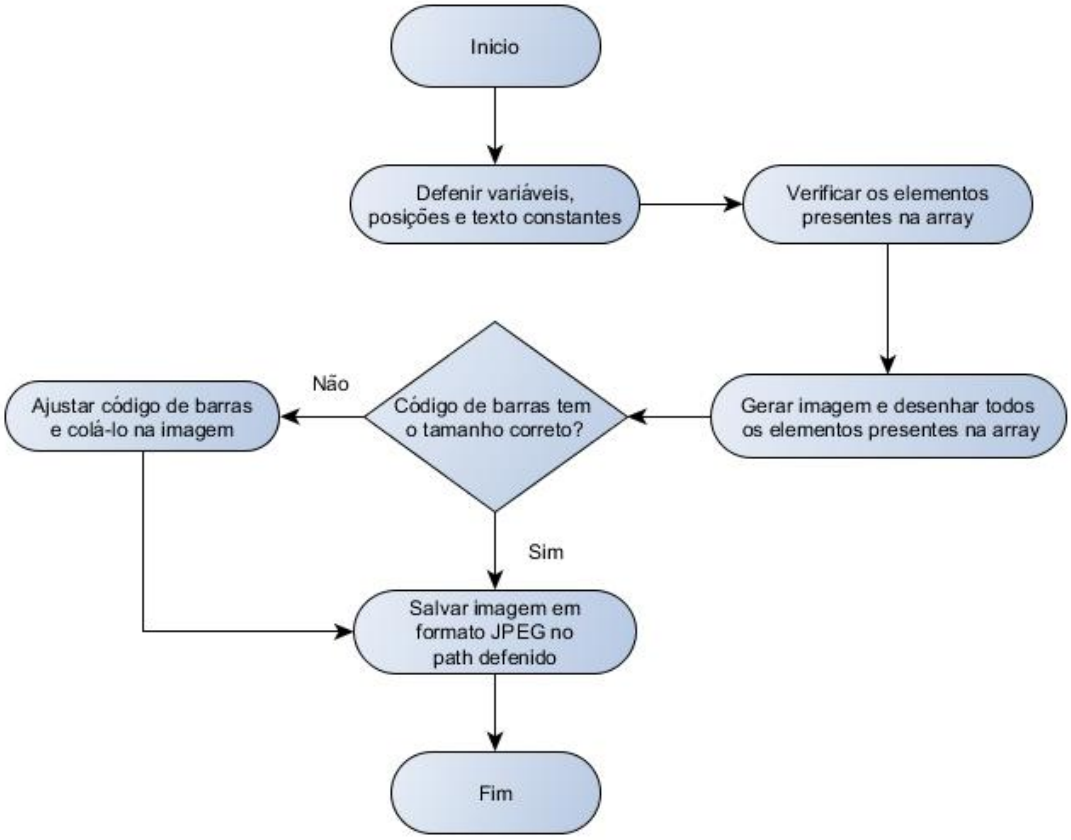


Figura 112 – Fluxograma da função *ImagConverter.*

B.1.4 Função *ImagConverter*

Como é possível verificar no diagrama descrito nas Figura 111 e Figura 112, esta função necessita de receber uma "*array*" de dados (exemplo em baixo) e de um *path* ou caminho a indicar a função onde salvar a imagem resultante.

Exemplo da *array* de dados, neste caso para uma etiqueta de transporte:

```
$dados=array("nome_empresa"=>"Tool's For Master Mind's Inc.",
"Descrição_empresa"=>"Empresa Transportadora",
"Morada_transportadora"=>"XXXXXXXXXXXXX",
"nif"=>"126128182, conservatoria do Porto",
"expedidor"=>"Hugo Manuel Batista dos Santos",
"destino"=> array("dest_1"=>"Manuel Antonio da Costa",
       "dest_2"=>"R Antero Esquental, 28",
       "dest_3"=>"Mesmo la",
       "dest_4"=>"3000-5000    Coimbra"),
"observ"=>array("obs1"=>"Material Electronico",
                     "obs2"=>"teste",
                     "obs3"=>"Fragil"),
"serviço"=>"Ilhas Aereo",
"valor_ref_gas"=>"1,316€/lt 2014/03/30",
"volta"=>"12345",
"nguia" => "GR454560",
"dt_guia"=>"2014/03/31",
"intermed"=>"XXXXXX",
"nguia_intermed" => "12355",
"ref_client"=>"XP76244555455",
"portes"=>"999.99",
"reembolso"=>"2€",
"nvolumes"=>"1",
"guia"=>"99999999",
"entregues"=>"3",
"kgs"=>"30Kg",
"cubico"=>"1000",
"quemrecebeu"=>"Pink & Brain",
"bi"=>"245455848",
"dt__hora_entreg"=>"2014/03/27  23:50:00",
"coordgps"=>array("lat"=>-10.473320,
              "long"=>40.896136),
"img"=>"inputImag/S1331279.sign.ng.fot.001.jpg");
```

Ao iniciar a função, esta deve em primeiro lugar definir os valores das coordenadas de cada elemento que vão ser desenhados, isto inclui a posição de todas as linhas, textos e imagens que compõem a etiqueta.
Em segundo lugar, verificar quais são os elementos presentes na "*array*", caso algum dos valores esperados não existir, ao desenhar a etiqueta, a função vai ignorar todos os elementos associados a esses valores.

```
$hasnome_empresa=false;
…
…
…
if(isset($dataArry["nome_empresa"]))
{
       if($dataArry["nome_empresa"]!=null || $dataArry["nome_empresa"] !=""){
              $hasnome_empresa=true;
       }
}
```

De seguida, caso exista uma imagem de código de barras para inserir na guia (variável na *array*: "img"=>"inputImag/S1331279.sign.ng.fot.001.jpg"), verifica se esta imagem tem menos de 320 *pixéis* de comprimento ou 240 *pixéis* de altura. Se o comprimento ou a altura excedessem estes paramentos, a imagem é enviada para a função *generate_image_thumbnail* a qual ajusta a imagem de forma a obter um tamanho apropriado.

```
$imag_parame= getimagesize($dataArry["img"]);
if(!$imag_parame) break; //erro sai
list($source_image_width,$source_image_height,$source_image_type) = $imag_parame;
```

```
if($source_image_width>320 || $source_image_height>240){
                $foto=generate_image_thumbnail($dataArry["img"]);
        …
```

Por fim, a função gere uma imagem em branco, desenha todos os elementos existentes nas coordenadas correspondentes e finalmente grava-a em formato *JPEG*, na localização definida no início da função.

```
//criar imagem em branco
$image=imagecreate($temp_imag_width,$temp_imag_height);
//Definir fundo da imagem
imagecolorallocate($image,255,255,255);
…
…
…
//salvar imagem em jpeg
if(!imagejpeg($image,$path_save))return false;
return true ;
```

B.2 Conversor de Imagem para *PDF*

B.2.1 Enquadramento da Aplicação

É pretendido com esta aplicação desenvolver uma forma de converter as imagens obtidas na aplicação anterior em ficheiros *PDF*, para isso foi utilizado a biblioteca *FPDF*.

B.2.2 Biblioteca *FPDF*

A biblioteca *FPDF*[40] foi desenvolvida em 2001 por *Olivier Plathey* como sendo uma classe *PHP* que permite gerar arquivos *PDF* através de linguagem *PHP*, isto é, sem o uso da biblioteca *PDFlib*[41]. Esta oferece várias funções de alto nível que permitem efetuar:

- Escolha da unidade de medida, o formato de página e margens;
- Editar cabeçalho e rodapé da página de gerenciamento;
- Quebra de linha automática e justificação do texto:
- Quebra automática de página;
- Suporte de imagem (*JPEG, PNG* e *GIF*);
- Compactação de página.

Juntando a isto, esta biblioteca é completamente gratuita e não sofre qualquer tipo de restrição no seu uso, tornando-a assim uma ótima ferramenta de trabalho para esta aplicação.

[40] *FPDF Home Page*: http://www.fpdf.org/ (16/08/2014).
[41] *PDFlib Home Page: http://www.pdflib.com/ (16/08/2014).*

B.2.3 Diagrama do Sistema

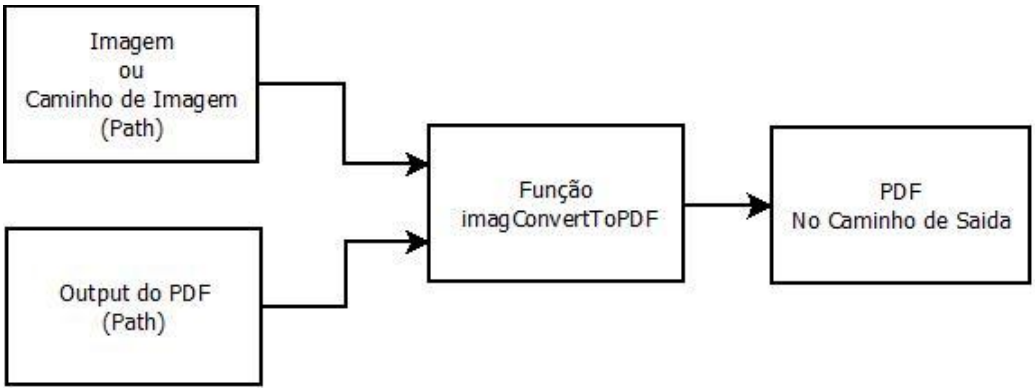


Figura 113 – Diagrama Base da função *imagConvertToPDF.*

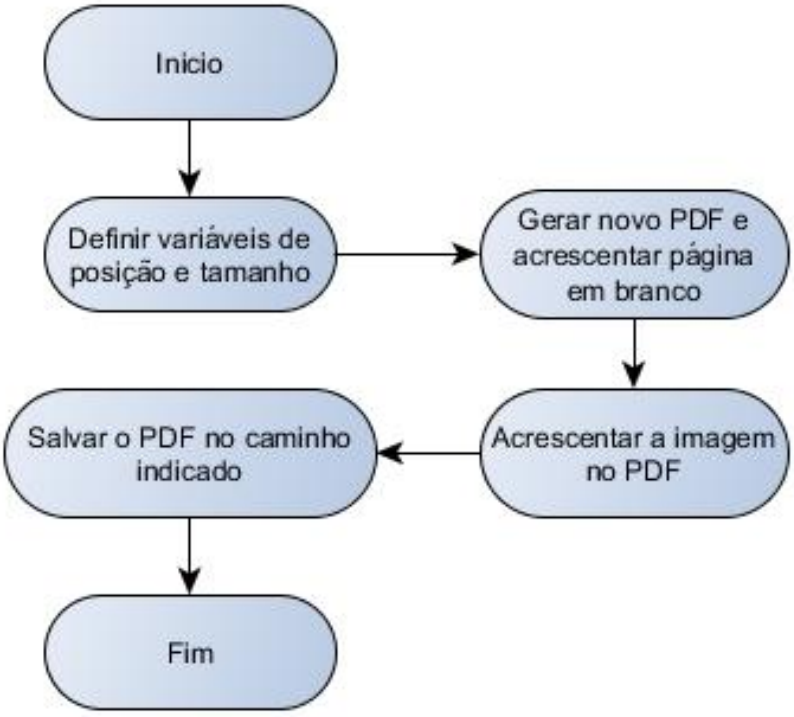


Figura 114 – Fluxograma da função *ImagConvertToPDF*.

B.2.4 Função *ImagConvertToPDF*

Esta função, ao iniciar necessita de receber o caminho da imagem a ser convertida, tal como o caminho de saída do *PDF* para poder trabalhar corretamente, como é possível verificar pelos diagramas da Figura 113 e o fluxograma da Figura 114. Opcionalmente, é possível também definir nesta função os parâmetros grau de inclinação, posição e tamanho da imagem no *PDF*, caso contrário a função utiliza os valores por defeito.

Como é possível ver pelo fluxograma, a função ao iniciar deve gerar um objeto *PDF* através das funções oferecidas pela biblioteca *FPDF* e adicionar uma nova página a este.

```
//gerar PDF
$pdf = new FPD();
//adicionar nova pagina
$pdf->AddPage();
```

De seguida, a imagem é inserida no *PDF*, através da função *RotatedImage*. Esta recebe todos os parâmetros da imagem e ajusta-a antes de a copiar no *PDF*.

```
$pdf->RotatedImage(
        $image,
        $start_point_X,
        $start_point_Y,
        $imag_w,$imag_h,
        $degrees,false);
```

Por fim, o *PDF* é gravado na posição definida na função terminando assim a sua operação.

```
$pdf->Output($path_for_pdf,'F');
```

Anexo C – *Paper* Realizado:

Mobile Applications and its Potential to Maintenance

[1]Hugo Santos, [2,3]António Simões, [1,3]Inácio Fonseca, [2,3]Torres Farinha
hugombsantos@hotmail.com;
[1] Electrical Engineering Department; [2]Mechanical Engineering Department; [3]CEMUC
Coimbra, Portugal

Abstract - Mobile technology is constantly evolving, most notably following the advent of Smartphone technology. As this equipment becomes faster, more powerful, and less expensive, it will become an excellent platform for the development of ergonomic and effective maintenance tools.

The present article describes an application that allows for an easy, organized and systematic way of improving the process of recording physical equipment data within the field of maintenance management, thus making the following contribution one of immense value to the field by increasing the accuracy of data storage.

The first, and arguably the most crucial step, necessary to elaborating the equipment dossier is to register its intrinsic data, planning data, and so on. Although, this phase is perceived as the easiest, it is usually one of the weakest points in equipment registering.

Having this issue in mind, an application was developed in order to facilitate equipment registering. The ultimate goal was to increase the efficiency of the registering process and equipment dossier, in addition to the processes involved during maintenance management. The present approach also aims to demonstrate that it is possible to develop effective maintenance via low-cost tools that do not require specialized equipment to function effectively. Mobile applications provide a huge opportunity to improve the working conditions experienced by maintenance teams.

From the initial process of regularly organizing workplace equipment, to operation issues when carrying out work orders, mobile technology can help improve the efficiency of various processes. The application allows effective and systematic way to improve the process of recording equipment, and is designed for mobile devices running Android (Google's system), or industrial equipment like PDAs running windows mobile – for example the Lynx model from Datalogic.

Keywords — Mobile applications; Maintence; Equipment dossier

I. INTRODUCTION

Mobile applications have a huge opportunity to improve the working conditions of maintenance teams, presently, there exists a void in available applications that use Smartphone devices. However, some companies have already begun to provide some products in this area, like Yardi [5] its mobile solutions allow completion of maintenance and inspection tasks, access to approve invoices and purchase orders, and also companies like RealPage [4] with its OneSite Facilities Mobile Service[9], which automates the whole process of maintenance performed by a technician through an easy management of all maintenance requests and Work Orders (WO). Thereby, reducing the time spent managing the entire documentation generated by the aforementioned process while also increasing technician productivity, by allowing for new WO to be received on site. In the field of wireless categorization of inventory it is possible to find solutions such as MobileAsset v7 and MobileAsset.EDU from the company Wasp Barcode Technologies [7]. These solutions offer a complete cataloging system that includes servers, an advanced management interface for standard computers, built-in communication with industrial equipment (capable of reading barcodes), and Smartphone applications for Android or iOS devices to access information from a database.

Using some of these ideas and concepts an application was developed to demonstrate various possibilities offered by Smartphone mobile systems in the maintenance field.

II. VALUE-ADDED TO THE MAINTENANCE FIELD

The main issue that exists in the maintenance field is that of data loss, following and in between scheduled maintenance completion and WO fulfillment. In order to counteract this loss, data may be more efficiently managed and stored, during and post-maintenance intervention, via the mobile application referred to in the present paper. The ultimate goal of this mobile application, thus, is to store all WO data immediately following each maintenance action as to avoid 'forgotten' data. Other contributions occur during the interventions, namely when these ones are not planned, because the technician can access on-line a fault diagnosis tool, if it exists.

But the interest of the technology under discussion begins at the moment that equipment is purchased, because it permits technicians to register *on-site*; including bar-code reading and so on. In fact, a correct registration of the equipment dossier in the working database may be one of the main determinants for a correct evaluation of Life Cycle Cost, including: WO, human resources, materials, and so on.

When an intervention request, in particular, urgent requests needing immediate attention, comes about a digital tool, such as a tablet or similar device may facilitate the completion of working orders received by a technician without loss of time. Furthermore, the introduction of new technologies (Augmented Reality, 3D models, Expert Systems, and to name a few) and applications creates synergies in order to minimize intervention time, increase quality, minimize risks, and maximize availability.

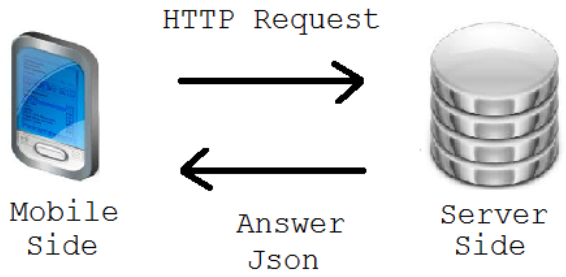


Fig.1. Interaction between the server side and the Mobile side.

III. GENERIC MODEL FOR A MOBILE APPLICATION

This application can be separated in two main categories, which include the server side, where all information is stored and processed, and the mobile side where all information is accessed or entered (Fig.1).

The server component runs a Rest server (Fig. 2) that receives and manages all requests to the server. This Rest server is a PHP script, which is able to receive HTTP requests, either GET or POST requests, and it is able to respond in XML, HTML or JSON. The Rest server is also used as a form of security because it sits between the database and all requests from the outside, not allowing direct access to the main database. The database should be sql type as it allows for easy development and integration with the Rest server.

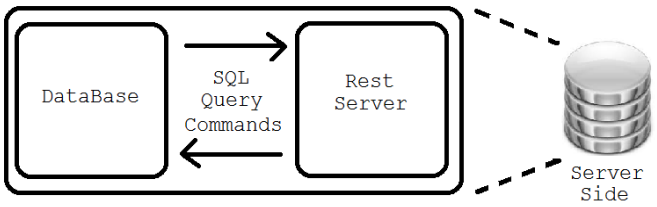


Fig. 2. Server Side Diagram

The mobile side must be able of connecting to the network via Wireless or via GPRS to obtain all the information needed. The device must also be capable of reading barcodes, this can be achieved through image capture or scanning, which requires the mobile to have a camera or a laser scan. The device must also have in his hardware list a touch screen sensitive enough to capture signatures as well as a camera to take photos and store them on the database. In some instances a GPS system may be useful to register the location of the device. In addition to all the listed prerequisites, the mobile device must be able to run an internal database to implement an offline mode. For several years all these features could only be found in expensive industrial devices, such as the PDA Datalogic Lynx, but at present, any Smartphone is capable of fulfilling these requirements. For this reason the author decided to use a PDA and Smartphone for the aforementioned research.

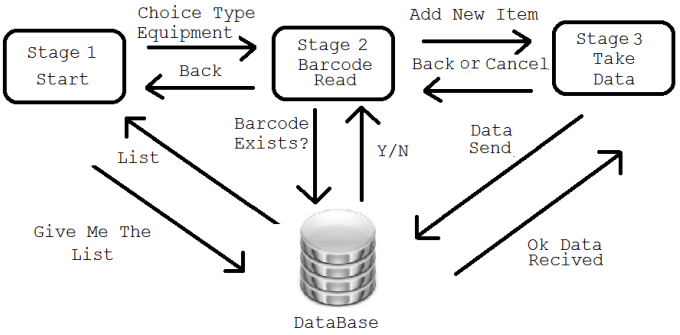


Fig. 3. Basic scheme of functioning of the application.

Once the mobile device is activated the application should check if the device is connected to the network, and that the server is functional. After this initial check, the application should ask the server for a complete list of all perishable equipment. After the list is displayed, the user needs to select the particular item that he wants to catalog.

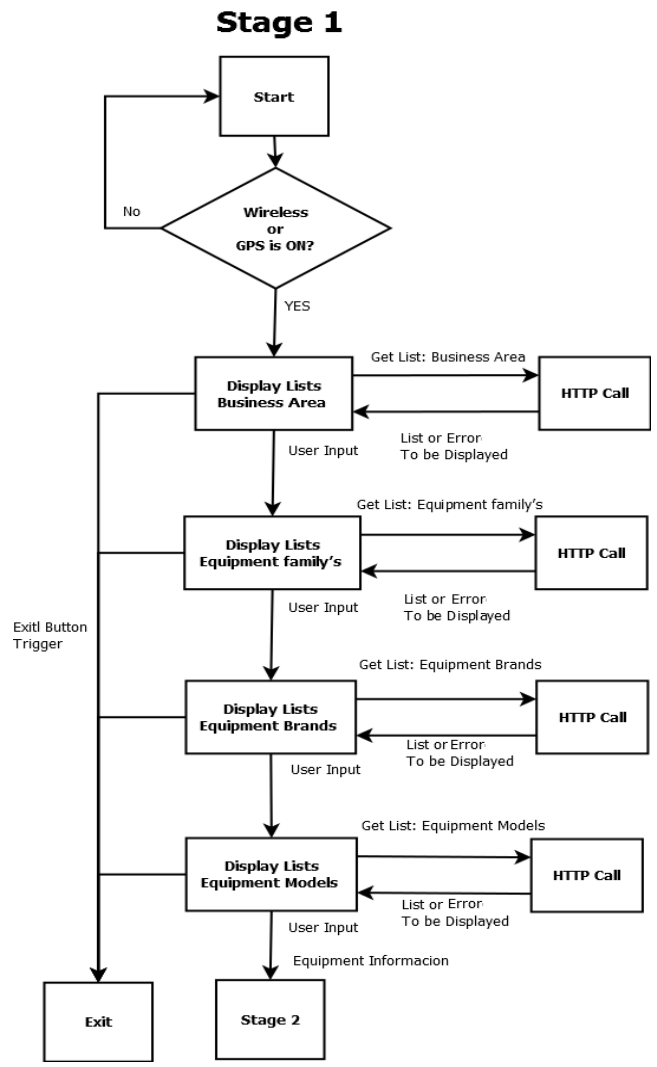


Fig. 4. Flowchart of Stage 1

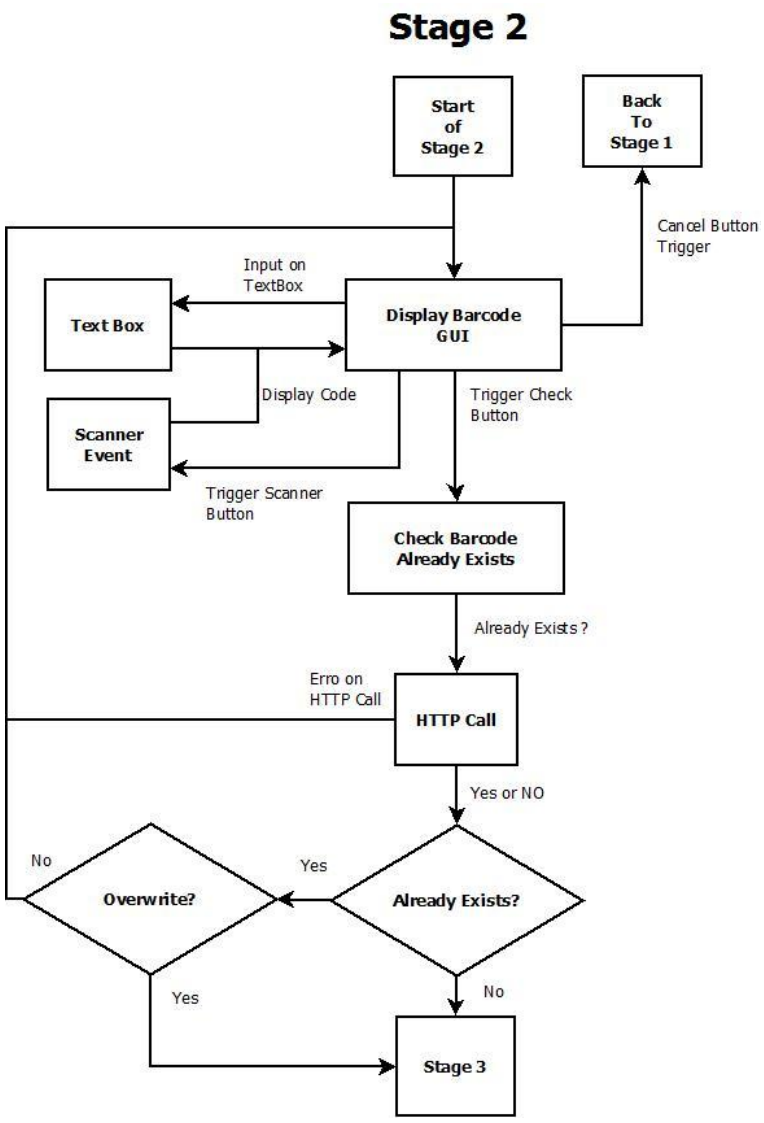


Fig. 5. Flowchart of Stage 2

In the next stage, the application can be used to scan barcodes, or input them manually, asking the database if the barcode has already been input. If so, the application will ask the user if he intends to overwrite the previous entry. In the event that the user wishes to overwrite the previous entry, or the input is a new entry, the application should jump to its third stage.

In the third stage, the user should collect photos, information about the location of equipment, comments regarding the state of the documented equipment, and etcetera, finally obtaining the user's signature. Note that in the case of indoor activity, the GPS function is not the ideal method to determine the device's location. So other means must be found, for example, through the use of a unique ID room system that can be either manually inputted or introduced as a barcode tag.

After all the aforementioned information has been collected, data should be sent to and then processed by the server. Following this action, the application should revert back to stage two – the barcode reading process. This behavior is stated on the following flowcharts (Fig. 3, Fig. 4, Fig. 5 and Fig. 6).

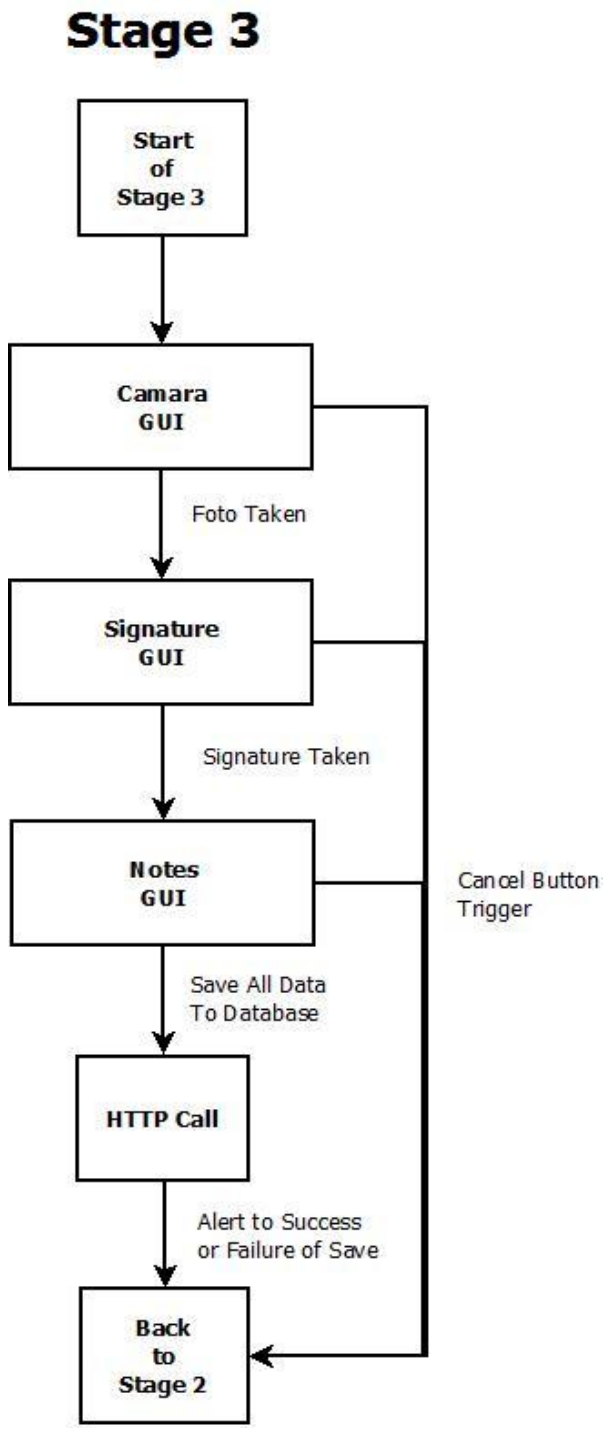


Fig. 6. Flowchart of Stage 3

IV. PRACTICAL IMPLEMENTATION

A. Database Implementation

The database MySQL was integrated with the development environment MySQL WorkBench. By doing so, time was saved, as the whole implementation of the database was developed graphically, a much more intuitive way of database development.

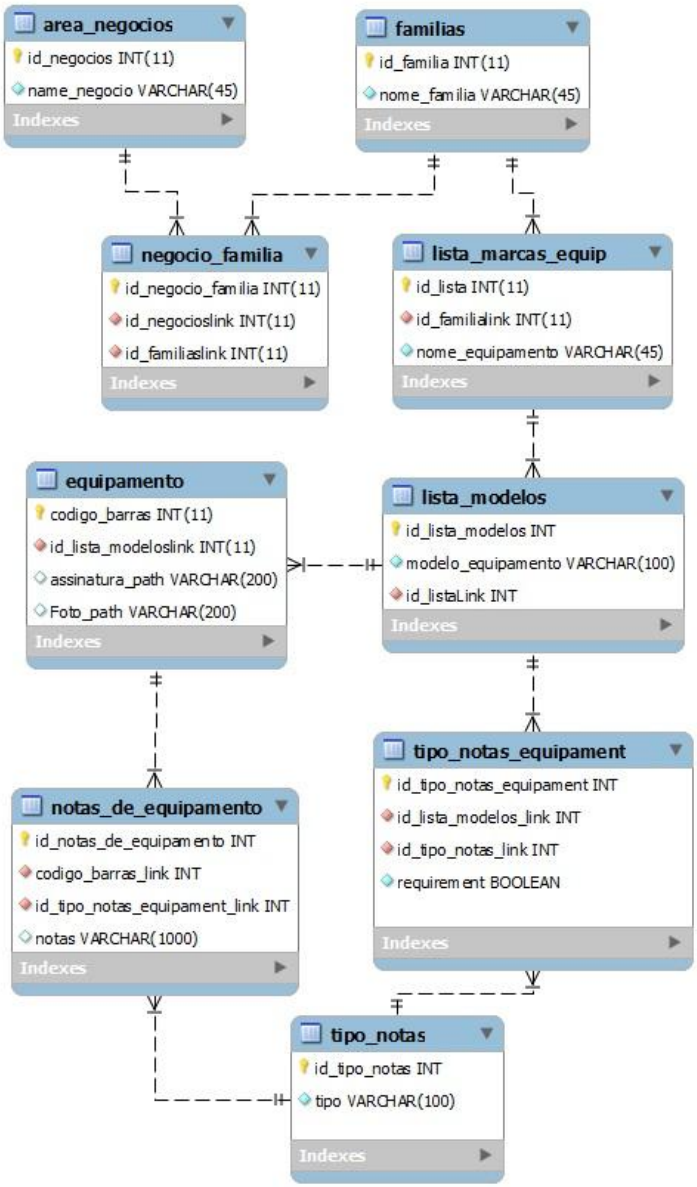


Fig. 7. Database Scheme.

As you can see from Fig. 7, the database is relatively small, containing only nine tables, each one devoted to a specific function.

The database information is organized in the following main areas:

- Business area – table area_negocios;
- Bridge between Business area and Equipment family's – table negocio_familia;
- Equipment family's – table familias;
- Equipment brands – table lista_marcas_equip;
- Equipment model – table lista_modeles;

- Bridge between Equipment model and Equipment type notes – table tipo_notas_equipament;
- Equipment type notes – table tipo_notas;
- Information cataloged – table equipamento;
- Equipment notes taken – table notas_de_ equipamento.

The first seven tables are responsible for the list and sub lists of all the equipment described in the first, initial stage of application. While the two remaining tables are responsible for the storage of all information regarding the cataloged equipment.

B. Rest Server Implementation

The Representational State Transfer (REST) style is an abstraction of the architectural elements within a distributed hypermedia system [2] [3]. It works by receiving HTTP requests from the mobile device, and acting based on them. Those requests are compose by a web address and the message, they are separated by the character "?" (example: http://192.168.252.207/PDA/rest.v2.php ?accao=autorizacao), as can be seen the message component functions by having a keyword, in this case "accao" and a value "autorizacao", it is also possible to add additional data by adding the character "&" between messages.

Upon receiving the request, the Rest server should execute the request action. In response, the server should send a message communicating to the device whether the message was a success or failure, in addition to data that was requested by the user. All this information uses the standard RFC 7159 format [1] (this is the JavaScript Object Notation for easy organization and information collection, allowing for friendly handling when receiving data at the mobile application terminal).

C. Android Implementation

This mobile application was developed on IDE Eclipse, and debugged on a Samsung GT-S7390, (Fig. 8), with Android version 4.1.2. It was written in Java, the android native language.

Fig. 8. Samsung GT-S7390

The aplicacion follows the sequence of information described in Fig. 3, but because the device does not have a laser scanner, it was necessary to find a way to read the barcode via camera. In order to achieve that, a separate application - the Barcode Scanner from Zxing, Fig. 11- was used in combination with the current one to obtain the barcode information. This technique was also recommended by Wei-Meng Lee in his book [6].

In Fig. 9 it is possible to see the interface that is available to the user during equipment selection from the catalog (stage 1 of the android application – Choose in sequence: Business Areas, Equipment Family, Marks, and Model). For example: Electronics, Multimeters, AMPROBE, and AM 520 EUR.

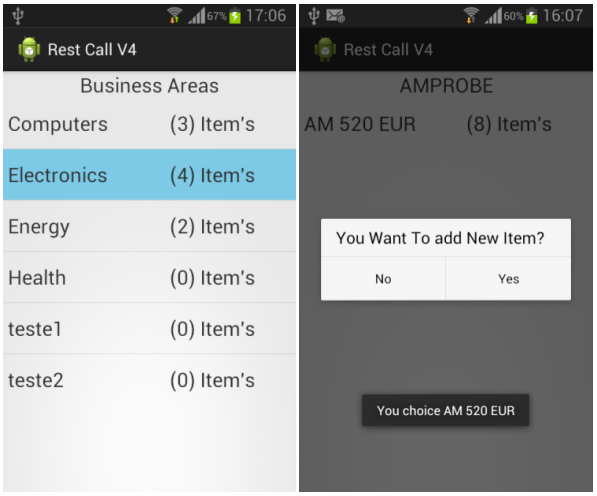


Fig. 9. Display of Equipment List on Android

The display in Fig. 10 and Fig. 11 are where the user can trigger the Barcode Scan activity by pushing the scan button "Start Scan" on the GUI, or manually introduce barcode data in textbox (stage 2).

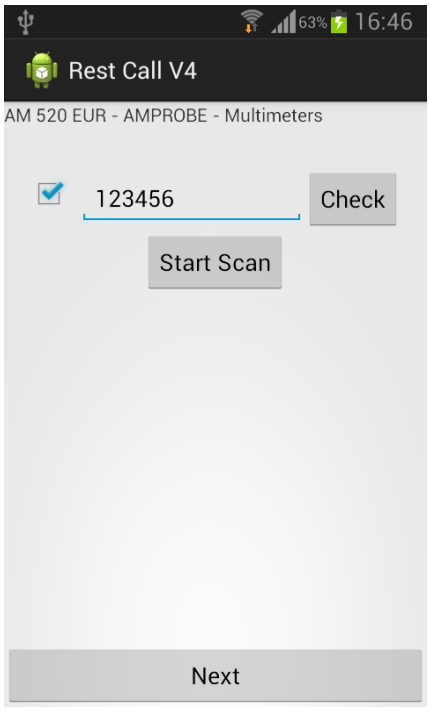


Fig. 10. Display to introduce the barcode on Android

Fig. 11. Barcode Scanner From Zxing.

After the database approves the barcode, and the user presses the "Next" button, the application should begin to gather information about the equipment; first, photos and signature, and then in the tab notes the user can insert other valuable information like location and power supply (see tab interface, Fig. 12 and Fig. 13).

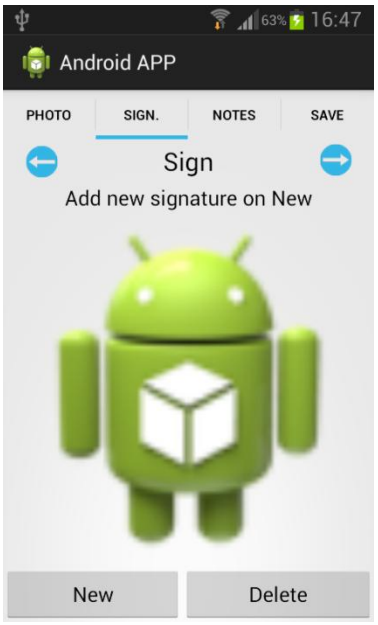


Fig. 12. GUI interface for user take photos and signature.

Here the user can choose to take photos, signatures or take notes about the item, as can be seen in the following pictures. By pressing the button entitled "New", the user should pass to the camera GUI, (default Android camera GUI from the device), or the signatures GUI and the Notes GUI (Fig. 14).

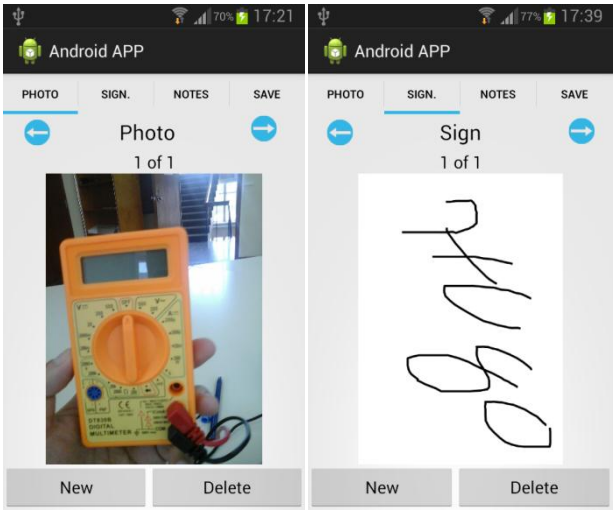


Fig. 13. Camara capture and signature capture GUI on Android.

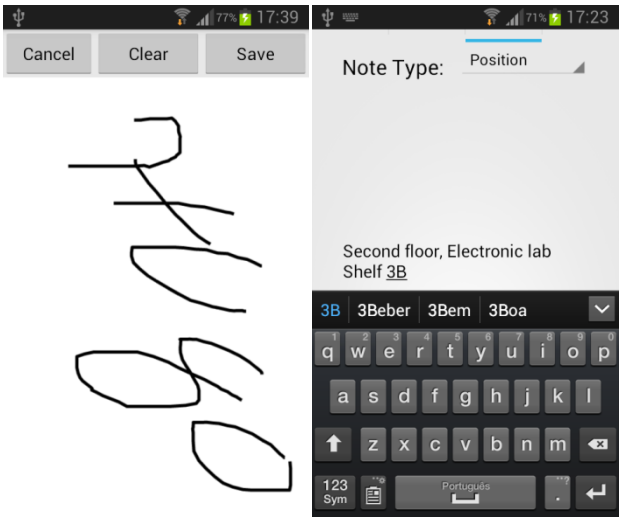


Fig. 14. Signture and Notes GUI

The user should press the "save" tab following data collection. In response, the application should upload all the registered information to the server, notifying the technician of the success or failure of this procedure before returning to stage 2.

D. Windows Mobile Implementation

To implement this application on Windows Mobile it was necessary to use the standard IDE of Microsoft, Microsoft Visual Studio, and all developed code was written in C#. In order to test and debug, this application used a Datalogic device, more precisely the Datalogic Lynx, (Fig. 15).

Fig. 15. Datalogic Lynx

The industrial PDA is a mobile device that is capable of withstanding manufacturing environments. Included in the device's hardware is a laser scanner, eliminating the need for a third party software, which is needed for the Android Implementation. Unfortunately, the industrial PDA requires the use of an application programming interface offered by Datalogic to access the scanner and is often much more expensive than conventional Smartphone devices.

In Fig. 16, it is possible to see the interface that the Windows mobile user would encounter when selecting the equipment catalog; it is similar to that of the Android (stage 1).

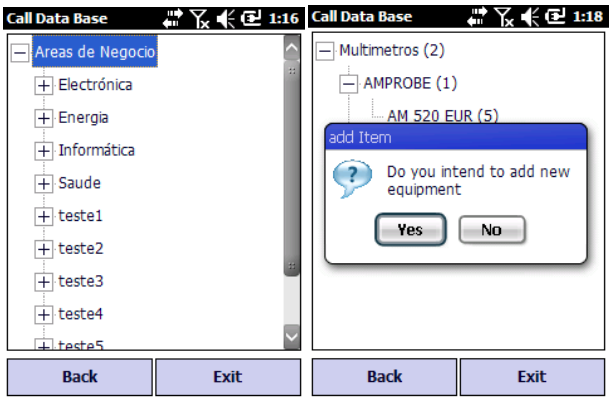


Fig. 16. Display of Equipment List on Lynx

The display Fig. 17 is where the user can trigger the laser scan by pushing the scan button on the device, or manually introducing the commend in the textbox (stage 2).

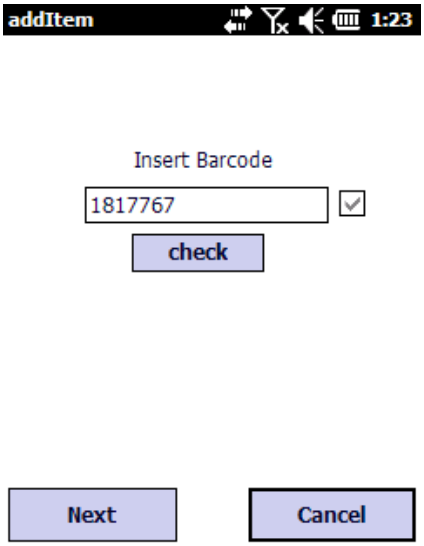


Fig. 17. Display to introduce the barcode on Lynx

After the database approves the barcode, and the user presses the "Next" button, the application should allow the user to take a photo and capture a signature to authenticate the job; as can be seen in Fig. 18(stage 3).

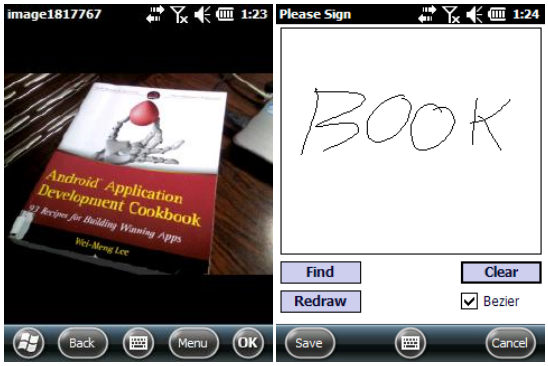


Fig. 18. Camara capture and signature capture GUI on Lynex.

As previously described, the application should upload all data taken to the server and alert whether the upload has been successfully, returning then to stage 2.

V. Improvements in Initial Maintenance Organization and Localization

Firstly, the outlined application is intended to improve the cataloging process of stocks and equipment through the automation of this process in an attempt to improve the overall maintenance process.

Secondly, there is the possibility of improving the management of assets since through the database it is possible to obtain several details about assets, conditions, images, location and additional details, ultimately effectively organizing day-to-day maintenance processes.

Finally, the present application demonstrates the endless possibilities of using generic equipment, instead of costly specialized equipment, in the maintenance process by ensuring a substantial reduction in costs.

VI. Conclusions

The present research demonstrates the potential and capabilities available through mobile and server applications within the maintenance field by enhancing automation, increasing data accuracy, lowering costs, and simplifying the overall process. The following application demonstrates an ever evolving process as new technological innovation may lead to greater optimization. For example, the GUI interface in both versions may be graphically improved upon to become more user friendly. It is also important to improve the running speed of the application in both the Android and Windows versions. Tests are taking place to interface the tables in the asset management application, and findings demonstrate that Object Maintenance with mobile devices improve the working time of human resources. Future applications may include data collection for hospital Object Maintenance catalogs.

VII. References

[1] Internet Engineering Task Force (IETF) , "The JavaScript Object Notation (JSON) Data Interchange Format," 3 2014. [Online]. Available: http://tools.ietf.org/html/rfc7159. [Acedido em 30 4 2014].

[2] R. T. Fielding, "CHAPTER 5 - Representational State Transfer (REST)," 2000. [Online]. Available: www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. [Accessed 2 5 2014].

[3] R. T. FIELDING and R. N. TAYLOR, "Principled Design of the Modern Web Architecture," 5 2002. [Online]. Available: www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf. [Accessed 4 5 2014].

[4] I. RealPage, "RealPage home page," RealPage, Inc, 2014. [Online]. Available: http://www.realpage.com/. [Acedido em 2 6 2014].

[5] I. Yardi Systems, "Yardi home page," Yardi Systems, Inc, 2014. [Online]. Available: http://www.yardi.com/. [Acedido em 2 6 2014].

[6] W.-M. Lee, "Android Application Development Cookbook," in *Capturing Barcodes*, Indianapolis, Indiana, USA, John Wiley & Sons, Inc., 2013, pp. 319 - 323.

[7] Wasp Barcode Technologies, "Introducing MobileAsset v7," Wasp Barcode Technologies, 2014. [Online]. Available: http://www.waspbarcode.com/asset-tracking/whats-new-v7. [Acedido em 30 7 2014].

[8] Wasp Barcode Technologies, "Wasp Barcode Technologies Announces MobileAsset v7, MobileAsset.EDU with iPhone, iPad, Android connectivity," Wasp Barcode Technologies, 2014. [Online]. Available: http://www.waspbarcode.com/about-us/press-release/2014-05-29-mobileasset-with-ios-android-connectivity. [Accessed 30 07 30].

[9] RealPageMedia, "Keep Your Service Technicians in the Field," RealPage, 12 4 2012. [Online]. Available: https://www.youtube.com/watch?v=8m3q0urO6uc. [Acedido em 2014 07 30].

Anexo D – Bibliotecas utilizadas nos Dispositivos Móveis

O processo de desenvolvimento de soluções informáticas, independentemente de estas serem ou não móveis, foi sempre um processo bastante interativo, baseando-se em trabalhos e experiências anteriormente desenvolvidas, por outras palavras, "não é necessário redescobrir a roda novamente" todas as vezes que se pretende desenvolver uma aplicação ou tarefa nova. Tendo isto em conta, começou a surgir o que se chama por bibliotecas de código ou também conhecidas por Software Development Kit ou SDK. Estas ficam responsáveis pelo desempenho de determinadas tarefas necessárias ao funcionamento da aplicação desenvolvida, mas sem requererem o conhecimento aprofundado por parte do programador a respeito da atividade interna desempenhada por estas, poupando assim bastante tempo ao processo de desenvolvimento.

Um bom exemplo disto, temos no SDK oferecido pela Datalogic para o dispositivo Lynx, o qual permite ao programador programar e chamar várias instruções para o equipamento de *hardware*, sem se ter de preocupar com o funcionamento interno da biblioteca, a qual, muito provavelmente, encontra-se a executar tarefas de baixo nível.

D.1 Bibliotecas utilizadas no Dispositivo *Lynx*

D.1.1 *Datalogic C/C++, dotNET, JAVA SDKs* para *Windows Embedded Handheld 6.5*

Este *Kit* de Desenvolvimento de *Software* fornecido pela própria *Datalogic* é desenvolvido explicitamente para os seus equipamentos móveis. Oferece ao programador diversos comandos e funções não *standard* do sistema operativo *Windows Embedded Handheld 6.5*, permitindo assim desenvolver aplicações que sejam capazes de usufruir de todas as funcionalidades do *Datalogic Lynx*, como por exemplo:

- Consulta e controlo do estado de funcionamento de todo o *hardware* do dispositivo, (*GPS, Wireless, Bluetooth, Display,* etc.);
- Acesso a informação do *software* e *firmware*, como versões do dispositivo, linguagem definida, nível de carga da bateria, entre outras;
- Inicialização e controlo do *scanner* a *laser* e os seus processos de descodificação;
- Gestão do retorno de erros e eventos específicos do dispositivo.

Numa forma abreviada, esta biblioteca serve de ponte entre a linguagem máquina específica deste dispositivo móvel e a linguagem de alto nível utilizada pelo programador durante o processo de desenvolvimento.

D.1.2 *OpenNetCF*

A *OpenNetCF* é uma empresa informática especializada em desenvolvimento de aplicações e soluções para sistemas embebidos. Ao longo dos anos esta empresa tem vindo a oferecer produtos comerciais, consultoria a clientes e *frameworks open-source*.

Durante o processo de desenvolvimento de algumas das aplicações para o dispositivo móvel *Lynx*, foram utilizadas as seguintes *Dll's*:

- *OpenNetCF.Net;*
- *OpenNetCF.Net.FTP;*
- *OpenNetCF.Net.Mail;*
- *OpenNetCF.Telephony.*

Estes *dll's* contêm funções e classes que possibilitam a implementação de vários sistemas de comunicação essenciais, como chamadas telefónicas, pedidos *ping* à rede ou envio de correio eletrónico, entre outros, permitindo assim a simplificação do processo de desenvolvimento.

D.1.3 *ProcessCE* ou *Terranova Api*

Esta biblioteca foi desenvolvida pelo *Sr. Frank T. van de Ven* e publicada no site *Code Project* [70], com o intuito de implementar o acesso e controlo de *tasks* a decorrer no *Windows Mobile*.

Por defeito o *Windows Moblie* só permite desligar um processo em execução, caso se saiba de antemão qual é o ID deste. Infelizmente o WM não oferece nenhuma forma prática de o obter ID do processo. A *Terranova API* pretende responder a esta necessidade, oferecendo:

- Enumeração dos processos a decorrer, tal como a informação dos seu *path*;

```
ProcessInfo[] list = ProcessCE.GetProcesses();
foreach (ProcessInfo item in list){
if (item.FullPath == @"\Windows\iexplore.exe"{
```

- A capacidade de verificar se um processo está em execução, especificando o caminho completo para o ficheiro *EXE*;

```
bool result = ProcessCE.IsRunning(@"\Windows\iexplore.exe");
```

- A capacidade de desligar um processo, dando o seu *full path* para o ficheiro *EXE*;

```
bool result = ProcessCE.FindAndKill(@"\Windows\iexplore.exe");
```

- Determinar o ID do processo, através *do full path* para o ficheiro *EXE*.

```
IntPtr pid = ProcessCE.FindProcessPID(@"\Windows\iexplore.exe");

if (pid == IntPtr.Zero)
   throw new Exception("Process not found.");
```

Sendo assim é possível consultar, identificar e desligar (se necessário) os processos a decorrer no processador.

D.2 Bibliotecas utilizadas no Dispositivo *Android*

D.2.1 Biblioteca *Zxing*

A biblioteca Zebra Crossing ou também conhecida por Zxing, Figura 115, é uma biblioteca open-source de processamento gráfico, especializada na leitura de códigos de barras 1-D e 2-D, suportando diversos tipos de formatos, Tabela 2, através dos recursos visuais de uma câmara.

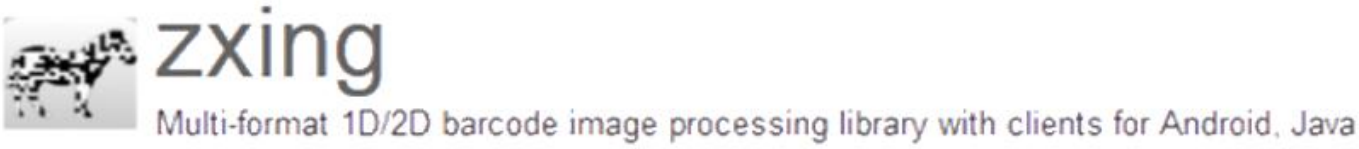


Figura 115 – Logotipo da Zxing

1D product	1D industrial	2D
UPC-A	Code 39	QR Code
UPC-E	Code 93	Data Matrix
EAN-8	Code 128	Aztec (beta)
EAN-13	Codabar	PDF 417 (beta)
	ITF	
	RSS-14	
	RSS-Expanded	

Tabela 2 – Formatos suportados pelo Zxing

Esta biblioteca foi implementada originalmente em *Java* para *Android* mas encontra-se à data deste documento em C# para plataformas Windows, em Objective C para IOS, em C++ para outras plataformas como Linux, entre outras linguagens, permitindo assim obter mecanismos de leitura de códigos de barras em varias plataformas distintas.