FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Controlo de Acessos Multimodular com Dispositivo Móvel Android (CAMDMA)

Filipe Manuel Magalhães Pereira

PARA APRECIAÇÃO POR JÚRI

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Ricardo Vardasca

Co-orientador: Joaquim Gabriel

20 de Setembro de 2015

Resumo

Atendendo à crescente utilização dos dipositivos móveis que estão a mudar o paradigma de acesso a informação, sendo algumas dessa informações sensíveis ou confidenciais, podendo estes dispositivos substituir sistemas computacionais de maior porte e menos ubíquos, é necessário implementar novas formas de controlar esses acessos.

Tirando partido do conjunto de sensores que equipam os dispositivos móveis e da sua capacidade de processamento pretende-se implementar formas confiáveis no acesso a recursos e a registo de quem acedeu.

Para implementar a solução desejada, desenvolveram-se aplicações para o sistema operativo Android de reconhecimento facial, leitura de QRCodes e de tags NFC, criando uma solução de suporte de backoffice através de um webservice RESTful.

A solução desenvolvida mostrou uma boa fiabilidade e performance, apresentando o reconhecimento facial desenvolvido uma sensibilidade de 70%, enquanto os métodos complementares tem um valor para o mesmo parâmetro de 100%. Há a destacar o fato de a solução de reconhecimento facial desenvolvida não necessitar de imagens numa base de dados, o que vem reforçar a segurança e garantia de confidencialidade dos utilizadores.

Utilizando a solução desenvolvida nos dispositivos móveis permite uma maior mobilidade no acesso a informações sensíveis e soluções de controlo de acessos e registos de assiduidade com custos mais reduzidos que as soluções convencionais para, além de constituir um desenvolvimento tecnológico no sentido de oferecer soluções de segurança baseadas na integração das várias formas de reconhecimento de indivíduos.

Palavras-chave: Android; Assiduidade; Biometria; Controlo de Acessos

Abstract

With the growing usage of mobile devices, which are changing the access to information paradigma. Being most of that information sensitive or confidential, in order to these devices can replace larger and less ubiquitous computational systems, it is necessary to implement new ways to control these accesses.

Taking advantage on the several sensors existente in the mobile devices and its processing capacity it is intended to implement reliable forms of access to resources and a record of who had accessed.

To implement the desired solution, applications were developed for the Android operating system for facial recognition, QRCodes and NFC tags reading, creating a back-office support solution through a RESTful webservice.

The developed solution showed good reliability and performance, presenting the facial recognition developed a sensitivity of 70%, while the complementary methods have a value for the same parameter of 100%. It should be highlight the fact that the developed face recognition solution does not require images in a database, which will strengthen the confidentiality of users and overallsafety and security of the system.

Using the developed solution on mobile devices allows greater mobility on access to sensitive informatiom, stronger access control solutions and attendance records with lower costs than conventional solutions. In addition, it is a technological development towards offering security solutions based on the integration of various forms of individual recognition.

Key words: Access Control; Android; Assiduity; Biometry; Facial recognition

Agradecimentos

Ao Professor Doutor Ricardo Ângelo Rosa Vardasca, orientador da FEUP, pelos conhecimentos transmitidos, participação no trabalho realizado, apoio e disponibilidade sempre demonstrados e pela orientação e sugestões importantes ao longo de toda a dissertação.

Ao Professor Doutor Joaquim Gabriel Magalhães Mendes, co-orientador da FEUP, pela oportunidade de realização deste trabalho em ambiente de investigação no laboratório L003 e nomeadamente ao Intelab pelo acolhimento e excelentes condições de trabalho providenciadas.

A todos os meus colegas do laboratório L003

Filipe Pereira

"Whether you think you can or you think you can't, you're right"

Henry Ford

Conteúdo

1 Introdução **1**
- 1.1 Enquadramento . . . 1
- 1.2 Motivação . . . 1
  - 1.2.1 Sistemas de autenticação . . . 2
- 1.3 Finalidade . . . 2
- 1.4 Objetivos Gerais . . . 2
- 1.5 Resultados Esperados . . . 3
- 1.6 Estrutura da Dissertação . . . 3
- 1.7 Diagrama de organização da dissertação . . . 5

2 Estado da Arte **7**
- 2.1 Sistemas Operativos Móveis . . . 7
  - 2.1.1 iOS . . . 7
  - 2.1.2 Android . . . 8
  - 2.1.3 Windows Phone . . . 9
- 2.2 Biometria em dispositivos móveis . . . 10
  - 2.2.1 Técnicas . . . 10
  - 2.2.2 Panorama das técnicas biométricas mais usadas . . . 12
  - 2.2.3 Impressão da palma da mão . . . 25
  - 2.2.4 Padrão vascular da mão . . . 26
  - 2.2.5 ADN . . . 27
  - 2.2.6 Reconhecimento de escrita . . . 28
  - 2.2.7 Reconhecimento do padrão de teclar . . . 28
  - 2.2.8 Reconhecimento de equilíbrio . . . 29
  - 2.2.9 Reconhecimento de gestos . . . 29
  - 2.2.10 Reconhecimento de batimento cardíaco . . . 29
  - 2.2.11 Odor . . . 30
  - 2.2.12 Padrão termográfico da face . . . 31
  - 2.2.13 Padrão termográfico do dorso da mão . . . 32
  - 2.2.14 Desempenho das técnicas de biometria . . . 33
- 2.3 Métodos alternativos de identificação em dispositivos móveis . . . 36
  - 2.3.1 QR Code . . . 36
  - 2.3.2 Passado, Presente . . . 36
  - 2.3.3 RFID . . . 38
  - 2.3.4 Smart Cards . . . 40
- 2.4 Bibliotecas de Suporte a processamento de imagens para Android . . . 42
  - 2.4.1 OpenCV . . . 42
  - 2.4.2 JavaCV . . . 43

2.4.3 Zxing . . . 43
2.5 HTTP e conceito Restfull . . . 44
2.5.1 Porquê um Serviço REST ("*REpresentational State Transfer*")? . . . 44
2.5.2 HTTP . . . 44
2.5.3 URLS . . . 45
2.5.4 Métodos HTTP . . . 45
2.5.5 Classificação dos Métodos HTTP . . . 46
2.5.6 Representações . . . 47
2.5.7 Códigos de Resposta . . . 47
2.6 Bibliotecas REST . . . 48
2.6.1 A maneira antiga: AsyncTasks . . . 48
2.6.2 Volley e Retrofit . . . 48
2.7 Sumário . . . 51

3 Metodologia **53**
3.0.1 Arquitetura da Solução . . . 54
3.0.2 Interface gráfica . . . 55
3.0.3 Funcionamento da Aplicação . . . 57
3.1 Reconhecimento Facial . . . 59
3.1.1 Criação da aplicação para a detecção das Faces . . . 60
3.1.2 Método de deteção das Faces - Viola-Jones . . . 63
3.1.3 Reconhecimento Baseado na Localização de Características da Face . . . 66
3.1.4 Métodos de Comparação . . . 76
3.2 Reconhecimento por QRCode . . . 79
3.3 Reconhecimento por NFC . . . 81
3.4 Serviço Web RestFull . . . 83
3.4.1 Funcionalidades . . . 83

4 Resultados **91**
4.1 Reconhecimento facial . . . 91
4.1.1 Eigenfaces . . . 91
4.2 Reconhecimento baseado na Localização de Características da Face . . . 94
4.2.1 Resultados: Detecção e Localização das Características da Face . . . 101
4.2.2 Resultados: Contornos da face . . . 101
4.2.3 Resultados: Detecção e Localização das Características da Face + Contornos da face . . . 101
4.3 Reconhecimento com QRcode . . . 101
4.4 Reconhecimento com NFC . . . 106
4.5 Webservice . . . 111

5 Discussão **113**

6 Conclusão **115**
6.1 Conclusão Final . . . 115
6.2 Proposta de Trabalho Futuro . . . 115

Referências **117**

Lista de Figuras

1.1 Diagrama de navegação do conteúdo da dissertação 5

2.1 Gráfico do crescimento do número de aplicações disponíveis para iOS 8
2.2 Quotas de mercado dos sistemas operativos móveis 9
2.3 Previsão das quotas de mercado para 2016 10
2.4 Requesitos de um sistema Biométrico 12
2.5 Sensor para a geometria da mão 13
2.6 Exemplo Impressão digital 14
2.7 Eficácia dos Métodos por número de imagens 15
2.8 Transformada de Fourier 16
2.9 Imagens da base de dados 17
2.10 Face média Y do conjunto de treino 18
2.11 Faces subtraídas da média 18
2.12 Eigenfaces 19
2.13 PBLH de cada píxel 21
2.14 Descritores PBLH 22
2.15 Exemplo Retina 23
2.16 Exemplos Irís 24
2.17 Análise de geometria da orelha 25
2.18 Exemplo de impressão digital da palma da mão 26
2.19 Exemplo do padrão vascular da mão 27
2.20 Digitalização da vasculatura da mão 27
2.21 Exemplo de proteção por padrão de teclar em dispositivo móvel 28
2.22 Exemplos de gestos passíveis de ser reconhecidos 29
2.23 Exemplo de um sinal ECG 30
2.24 Exemplo de uma imagem térmica da face 32
2.25 Exemplo de uma imagem térmica do dorso das mãos 32
2.26 Taxas de erro em Sistemas Biométricos 33
2.27 Receiver operating characteristic (ROC) 34
2.28 Técnicas de biometria e o seu desempenho em cada requisito 35
2.29 Exemplo QR Code 36
2.30 Interpretação de um código QR 38
2.31 Exemplo RFID 39
2.32 Exemplo de um *Smart Card* de contato 40
2.33 Exemplo de *Smart Card* sem contacto 41

3.1 Interface Normal para o utilizador da solução 55
3.2 Interface após Registo entrada/saída para o utilizador da solução 55

3.3 Interface de Registo para o utilizador da solução . . . 56
3.4 Diagrama de fluxo de dados para o registo . . . 57
3.5 Diagrama de fluxo de dados do reconhecimento (entradas/saídas) . . . 58
3.6 Classe Preview . . . 60
3.7 Função para a Câmara frontal . . . 61
3.8 Classe *Preview* . . . 61
3.9 Converter *Frame* para escala cinza para reconhecimento . . . 62
3.10 Caracteristicas haar . . . 63
3.11 Características Faciais . . . 64
3.12 Cascade of Classifiers . . . 65
3.13 Interface RBLCF . . . 66
3.14 Regiões de Interesse . . . 67
3.15 Detecção das Características . . . 67
3.16 Funções ROI . . . 68
3.17 Suavização . . . 69
3.18 Equalização do Histograma . . . 69
3.19 Binarizada . . . 70
3.20 Contornos . . . 70
3.21 Método Reconhecimento Completo . . . 71
3.22 Entidade Face . . . 72
3.23 Entidade-Relação Face-Picagem . . . 74
3.24 Exemplo da aplicação QR Code . . . 79
3.25 Entidade e Relacionamento QR Code . . . 80
3.26 Entidade-Relação QR Code-Picagem . . . 80
3.27 Exemplo da aplicação NFC . . . 81
3.28 Entidade NFC . . . 81
3.29 Entidade-Relação NFC-Picagem . . . 82
3.30 Volley vs RetroFit . . . 83
3.31 Dependências . . . 84
3.32 POJO para resposta ao Login . . . 84
3.33 Interface RetroFit . . . 85
3.34 Classe RestClient . . . 86
3.35 Classe SessionRequestInterceptor . . . 87
3.36 Função getCookieString . . . 87
3.37 Formato de um erro . . . 88
3.38 POJO para um erro . . . 88
3.39 Classe RestCallback . . . 89
3.40 Mensagem de Login em caso de sucesso . . . 90
3.41 JSON Picagem . . . 90
3.42 Entidade-Relação picagem.XML-Picagem.JSON . . . 90

4.1 Interface pessoa reconhecida . . . 93
4.2 Interface de registo de face . . . 95
4.3 Interface de registo de informação de indivíduo após a deteção da face . . . 95
4.4 Interface com mensagem de sucesso no registo de uma face . . . 96
4.5 Mensagem de tentativa de registar uma face com um id já existente . . . 96
4.6 Notificação da não existência de faces registadas . . . 98
4.7 Notificação de face desconhecida . . . 98
4.8 Notificação de entrada . . . 99

4.9 Notificação de saída 99
4.10 Exemplo de QRcode 102
4.11 Interface de registo de QRCode 102
4.12 Interface de introdução de informação do indivíduo no registo de QRCode 103
4.13 Interface com mensagem de sucesso no registo de QRCode 103
4.14 Notificação de identificação de QRCode desconhecido 104
4.15 Notificação de entrada de um indivíduo com um QRCode reconhecido no sistema 105
4.16 Notificação de saída de um indivíduo com um QRode reconhecido no sistema . . 105
4.17 Interface de registo de tag NFC 107
4.18 Interface de introdução de informção do indivíduo no registo da tag NFC 107
4.19 Notificação do registo de tag NFC com sucesso 108
4.20 Notificação que a NFC lida não está registada no sistema 109
4.21 Notificação de entrada de um indivíduo com uma NFC reconhecida no sistema . 109
4.22 Notificação de saída de um indivíduo com uma NFC reconhecida no sistema . . . 110
4.23 Sistema com ligação à internet 111
4.24 Sistema sem ligação à internet 111
4.25 Sistema sem ligação à internet 112

Lista de Tabelas

3.1 Equipamento utilizado 54
3.2 Software Utilizado 54
3.3 face.XML 73
3.4 picagem.XML 75
3.5 qrcode.XML 80
3.6 nfc.XML 82
3.7 config.XML 89

4.1 Base de Dados Eigenfaces 92
4.2 Resultado Eigenfaces para 20 indivíduos 94
4.3 Ficheiro XML com face 97
4.4 Ficheiro XML com registos de entrada/saída 100
4.5 Resultado: Detecção e Localização das Características da Face para 15 indivíduos 101
4.6 Resultado: Contornos da face 101
4.7 Resultado: Detecção e Localização das Características da Face + Contornos da face 101
4.8 Ficheiro XML com QRCode 104
4.9 Ficheiro XML com picagens 106
4.10 Ficheiro XML com tag NFC 108
4.11 Ficheiro XML com registos de entrada saída 110

Abreviaturas e Símbolos

ADT	Abstract Data Type
ANDF	Architecture-Neutral Distribution Format
API	Application Programming Interface
CAD	Computer-Aided Design
CASE	Computer-Aided Software Engineering
CORBA	Common Object Request Broker Architecture
UNCOL	UNiversal COmpiler-oriented Language
Loren	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed vehicula lorem commodo dui
WWW	*World Wide Web*

Capítulo 1

Introdução

1.1 Enquadramento

Este trabalho de dissertação surge no âmbito do Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MIEEC), major de Automação, da Faculdade de Engenharia da Universidade do Porto (FEUP). Foi realizado parcialmente em ambiente laboratorial no Departamento de Engenharia Mecânica, com a supervisão do Professor Doutor Ricardo Ângelo Rosa Vardasca e co-supervisão do Professor Doutor Joaquim Gabriel Magalhães Mendes.

1.2 Motivação

Nos últimos anos temos testemunhado uma adopção generalizada dos dispositivos móveis. Os serviços oferecidos servem para aceder a uma gama cada vez maior de informações, tornando-se uma necessidade a existência de métodos cada vez mais confiáveis de autenticação [1].

Quando precisamos de usar dispositivos móveis para nos ligar a redes empresariais, os conceitos de utilizador e de sistemas de autenticação ganham um papel fundamental para prevenir más utilizações, abusos e ataques [2].

Esta investigação começa com um estudo das técnicas de autenticação existentes, sendo identificadas as abordagens biométricas como as de maior potencial e já com um desenvolvimento em grande escala [3]. É seguro admitir que a autenticação por biometria está ao nosso redor há muito tempo e os dispositivos de hoje estão equipados com mais sensores do que nunca, mas não é razoável pensar que esses sensores serão optimizados ao ponto de serem capazes de reconhecer a nossa identidade nos próximos anos com 100% eficácia. Porém é seguro assumir que a biometria será vista como uma camada de segurança.

Pode-se igualmente esperar que a autenticação multi-plataforma (dois métodos complementares) irá desempenhar um papel maior, seja através de um serviço que forneça um único segundo código para um segundo dispositivo para que o usuário digite, ou o segundo dispositivo em si, ser a verificação.

Irão ser introduzidas as várias estratégias e na Revisão Bibliográfica será efectuado um estudo aprofundado das tecnologias.

1.2.1 Sistemas de autenticação

A autenticação verifica que os utilizadores ou sistemas são quem dizem ser, baseados em identidades (por exemplo, nome de utilizador) e credenciais (por exemplo, senha). A maioria das falhas de segurança conhecidas são atribuídas a autenticações fracas ou ausência das mesmas (por exemplo de computadores portáteis a redes sem fio desprotegidas). Muitos incidentes dispendiosos e embaraçosos poderiam ser evitados exigindo-se uma autenticação robusta para os dispositivos móveis e as redes que utilizam.

Os dispositivos móveis são facilmente perdidos ou roubados, sendo necessário uma proteção extra contra o acesso não autorizado de dados, aplicações e conexões. No entanto, normalmente os utilizadores de dispositivos móveis exigem acessos frequentes e por breves períodos de tempo, tornando inconveniente as repetidas reposições das credenciais. Enquanto a maioria dos computadores portáteis são definidos para exigir logins, na maioria dos dispositivos móveis isso não acontece, apesar de ser um recurso disponível e amplamente diversicado que pode ser facilmente integrado em aplicações para o mesmo.

Ao programar-se uma estratégia de autenticação móvel, os esforços são dirigidos para se combinar a força/segurança com a usabilidade. Considerando ambos, o dispositivo móvel e as credenciais de acesso à rede, cada método deve satisfazer as necessidades requeridas da plataforma, de segurança e do utilizador.

1.3 Finalidade

Após uma reflexão sobre o projeto, defeniu-se como premissa principal a implementação de um sistema de controlo de acessos automático usando dispositivos móveis e tirando partindo das tecnologias biométricas e com recurso a métodos complementares de autenticação.

1.4 Objetivos Gerais

De forma a atingir a finalidade anteriormente referida defeniram-se os seguintes objetivos:

- Definição de uma interface simples e a requerer a minima interecção do utilizador;
- Implementação das soluções de reconhecimento facial convencionais num dispositivo móvel;
- Desenvolvimento de uma nova forma de reconhecimento facial sem recurso a uma base de dados de imagens;
- Desenvolvimento de uma forma de autenticação com recurso a QRCodes;

- Desenvolvimento de uma forma de autenticação com recurso a tag NFC.

- Implementação de uma interface de *back-office* com recurso a webservices Restfull;

- Comparação de performance entre o método de reconhecimento facial desenvolvido e os métodos implementados.

1.5 Resultados Esperados

Pretende-se desenvolver uma solução com boa fiabilidade e performance apresentando os métodos de autênticação uma boa sensibilidade reforçando a segurança no controlo de acessos e autênticação.

1.6 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 7 capítulos. No capítulo 2, é descrito o estado da arte. Aí é explicada a arquitectura e o funcionamento de um sistema típico de reconhecimento biométrico. Bem como identificados e expostos os métodos e algoritmos mais relevantes nesta matéria. Será também alvo de estudo vários sistemas de reconhecimento auxiliares.

No capítulo 3, os métodos e estratégias utlizados durante o desenvolvimento da dissertação.

Na secção 3.1 são descritos os vários métodos de reconhecimento facial, onde inclui um conjunto de algoritmos desenvolvidos.

Após isso, nas secções 3.2 e 3.3 são dedicados aos métodos de reconhecimento auxiliar desenvolvidos, nomeadamente por QRCode e NFC.

Na secção 3.4 é apresentado o trabalho desenvolvido para que as aplicações de reconhecimento enviassem os detalhes para uma base de dados externa através de um webservice e seguindo um conceito Restfull,

No capítulo 4 são apresentados os resultados do trabalho desenvolvido com base nos testes realizados e analisados em tabelas e gráficos demostrativos.

No capítulo 6 uma reflexão sobre todo o trabalho desenvolvido, se foi de acordo com as expectativas e possibilidades de trabalho futuro.

No capítulo 5 são discutidos os resultados obtidos e as dificuldades encontradas.

1.7 Diagrama de organização da dissertação

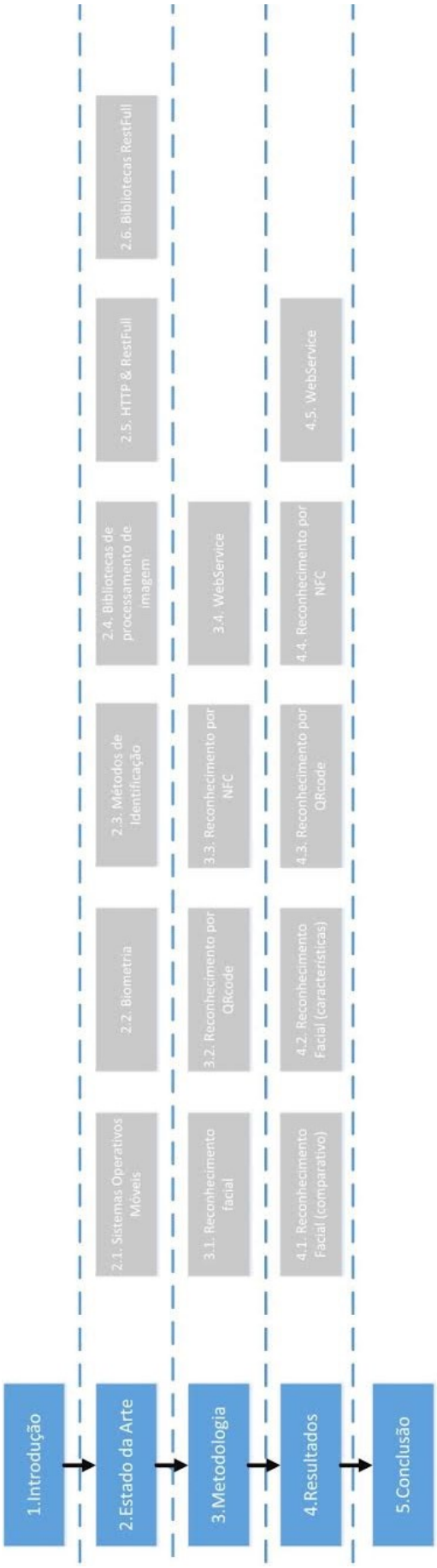


Figura 1.1: Diagrama de navegação do conteúdo da dissertação

Capítulo 2

Estado da Arte

Nesta secção do relatório é feito enquadramento teórico do tema do projeto, indicando o estado-da-arte da tecnologia envolvida.

2.1 Sistemas Operativos Móveis

2.1.1 iOS

O sistema operativo iOS é um sistema desenvolvido e criado inteiramente pela Apple Inc para dispositivos móvies [4]. Este sistema operativo tem a particularidade de ser desenvolvido de forma a apenas ser utilizado em dispositivos Apple, como iPhones, iPads e iPods.

Este sistema operativo surgiu em 2007 e, desde então, teve um grande crescimento . Através da figura 2.1 podemos ver o crescimento de novas aplicações para este sistema operativo tornando assim evidente o crescimento exponencial destas desde o seu aparecimento [5].

Tal como outros sistemas operativos para dispositivos móveis, o iOS baseia a sua navegação através de uma interface multi toque tentando manter a interface entre o dispositivo e o utilizador o mais simples possivel. Parte da interação com o utilizador é baseada no movimento através de acelerometros e giroscopios embutidos no dispositivo de forma a tornar a usabilidade do mesmo mais atrativa (um exemplo é a passagem do modo de retrato para o modo de paisagem apenas rodando o aparelho).

O desenvolvimento de aplicações para este sistema operativo baseia-se na linguagem C, sendo o seu desenvolvimento em Objective-C.

A Apple disponibiliza aos seus utilizadores a possibilidade de produzir aplicações próprias

através de um Software Development Kit (SDK) proprietário, contudo, de forma a disponibilizar as suas aplicações o programador tem de pagar uma taxa, a "iPhone Developer Program fee"e a aplicação desenvolvida passar pela certificação da Apple Store, sendo estes os principais inconvenientes no que toca ao desenvolvimento de aplicações para iOS.

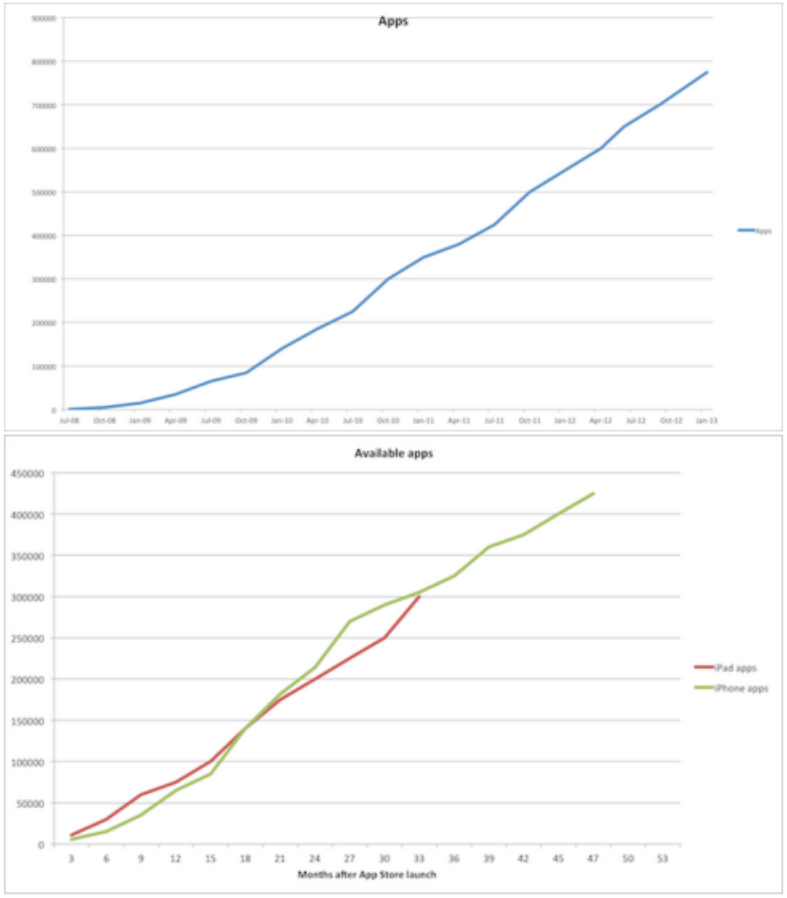


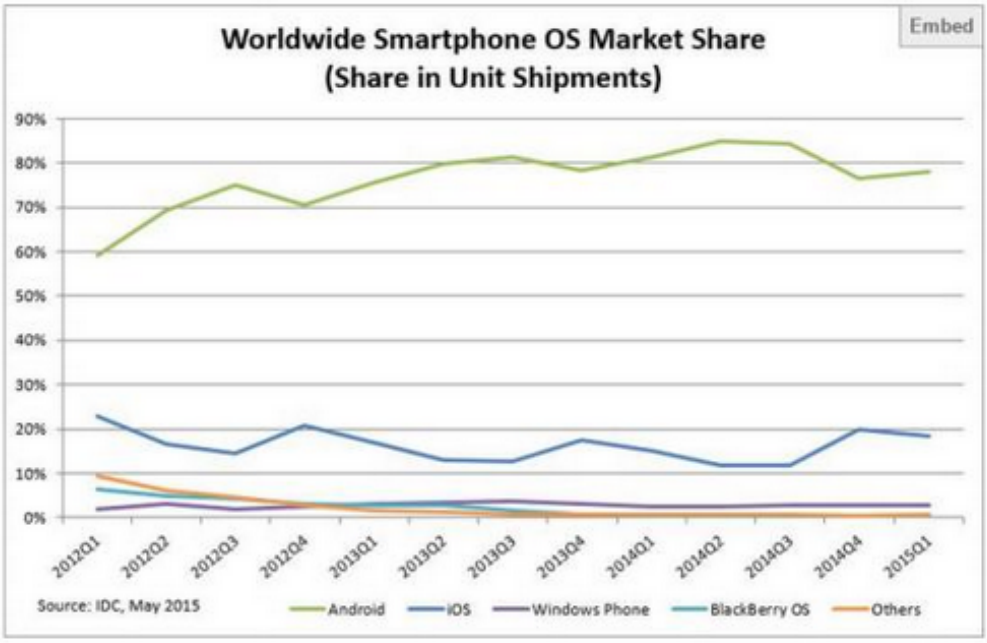
Figura 2.1: Gráfico do crescimento do número de aplicações disponíveis para iOS

2.1.2 Android

O sistema operativo Android é baseado no núcleo do Linux (sendo dessa forma também open source) e é desenvolvido e suportado pela Google Inc [6].

À semelhança do que acontece com o sistema operativo iOS, o foco de desenvolvimento Android são os dispositivos móveis tácteis, ou seja, os smartphones e os tablets. Contudo, actualmente verificamos um alargar de horizontes e aparecem interfaces para televisões (AndroidTV), automóveis (Android Auto) e para relógios de pulso (Android Wear). Os movimentos também são baseados na manipulação de objectos no ecrã táctil.

A principal característica deste sistema operativo é o facto do seu desenvolvimento ser aberto a qualquer utilizador, sendo a subsequente comercialização do produto desenvolvido também gratuito. Actualmente este sistema operativo é o que tem maior quota de mercado, Figura 2.2, possui várias ferramenta de desenvolvimento, no entanto a recomendada "Android Studio"com a utilização do Android SDK [7].



Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q1 2015	78.0%	18.3%	2.7%	0.3%	0.7%
Q1 2014	81.2%	15.2%	2.5%	0.5%	0.7%
Q1 2013	75.5%	16.9%	3.2%	2.9%	1.5%
Q1 2012	59.2%	22.9%	2.0%	6.3%	9.5%

Source: IDC, May 2015

Figura 2.2: Quotas de mercado dos sistemas operativos móveis

2.1.3 Windows Phone

O sistema operativo Windows Phone é a opção da Microsoft Inc para dispositivos móveis. Apareceu de forma a substituir o Windows Mobile (WM) e o Zune, aliando o melhor que cada um destes sistemas operativos tinha para oferecer. Ou seja, aliando as capacidades executivas do WM às capacidades multimédia do Zune, apareceu o Windows Phone [8].

O seu primeiro lançamento foi em 2010 tornando-o o último, dos maiores players, a aparecer neste mercado e, por essa mesma razão, o que tem a menor quota de mercado quando comparado com os principais concorrentes [9], Figura 2.3.

A versão mais recente é a denominada 10 e a sua grande vantagem são os vários poderes de sincronização com os vários computadores que utilizam sistemas operativos Windows, permitindo uma melhor gestão de ficheiros e aplicações quando comparados com os outros dois concorrentes.

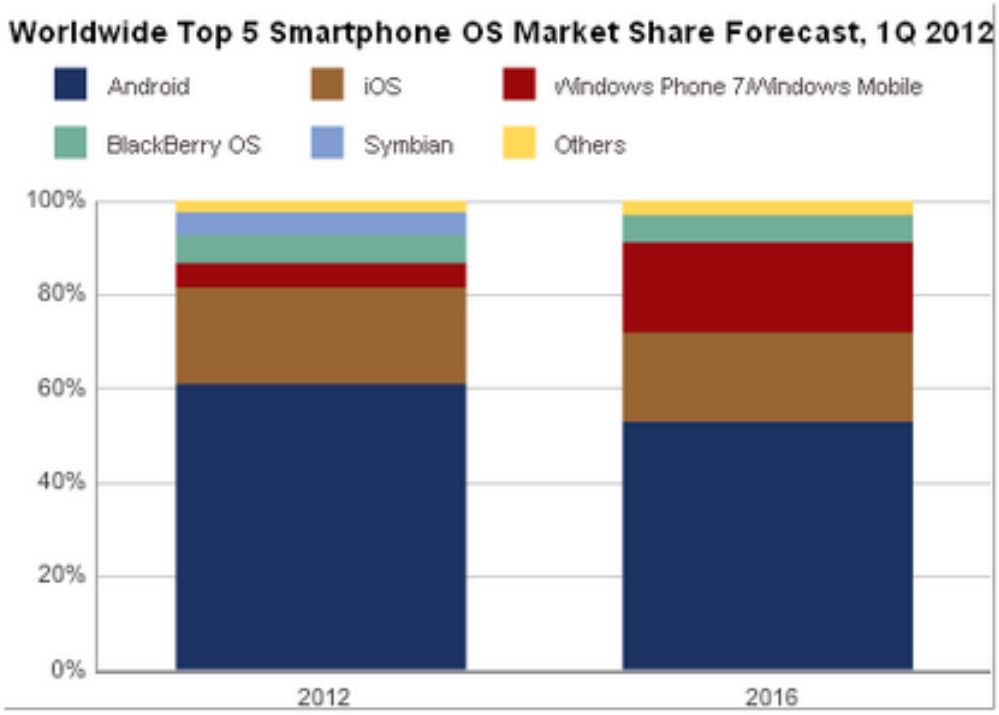


Figura 2.3: Previsão das quotas de mercado para 2016

2.2 Biometria em dispositivos móveis

A palavra Biometria tem origens nas palavras gregas bios(vida) e metrikos(medida) [10]. É do senso comum que os seres humanos usam características como a face e a voz como forma de reconhecimento. Tal como anteriormente referido, hoje em dia muitas aplicações requerem esquemas confiáveis de identificação de um individuo e reconhecer características humanas começa a ser cada vez mais interessante devido à tecnologia emergente nas novas aplicações. Tradicionalmente, senhas e cartões de identificação são usados para restringir o acesso e proteger os sistemas, mas as senhas e cartões de identificação podem ser roubados/adulterados ou até perdidos. Ganhando por este motivo a Biometria destaque, uma vez que é impossível, ou praticamente impossível, perder ou roubar uma característica biométrica.

2.2.1 Técnicas

Biometria é uma técnica utilizada para reconhecer características e comportamentos de forma eletrónica [11]. Identificam-se vários atributos do corpo humano, tais como: a voz, impressões digitais, retina, íris e parâmetros da mão. Uma técnica biométrica baseada em características fisiológicas é geralmente mais confiável do que uma que adopte características comportamentais, mesmo que esta possa ser mais fácil de integrar em determinadas aplicações. Nas técnicas biométricas, um indivíduo pode ser reconhecido através da utilização dos métodos de identificação e verificação. A verificação confirma ou nega a identidade de um indivíduo que diz ser quem

é, com uma correspondência padrão de 1-1. A identificação é uma comparação entre os vários modelos armazenados numa base de dados para um indivíduo em particular. Contêm uma correspondência de 1:N. Isto implica que a identificação e verificação são dois problemas que devem ser tratados separadamente.

Uma técnica biométrica tem quatro partes:

- **1) Sensor** aquisição dos dados biométricos,
- **2) Extração de recursos** com os dados adquiridos, extração dos vetores de características,
- **3) Comparação** os vetores de características estão prontos para comparação com os dados adquiridos,
- **4) A tomada de decisão** se um usuários faz um login com um ID válido ou não.

Para as técnicas biométricas são necessários avaliar seguintes requisitos [12], Figura 2.4:

- **Universalidade** - todas as pessoa que usem o sistema biométrico devem possuir a característica medida;
- **Singularidade** - medida de como a característica consegue discriminar indivíduos;
- **Permanência** - como a característica resiste ao envelhecimento;
- **Coletável** - facilidade de ser medida quantitativamente, sem causar inconvenientes para o usuário;
- **Desempenho** - precisão, velocidade e robustez da tecnologia utilizada;
- **Admissibilidade** - grau de aprovação da tecnologia biométrica pelos utlizadores;
- **Permeabilidade** - grau de sujeição a fraude que pode comprometer o sistema.

Nenhuma técnica é perfeita. Nenhuma vai satisfazer em 100% as características referidas em cima. Dependendo da aplicação, a tomada de decisão deve analisar as características e determinar qual/quais são a melhor escolha para a sua organização.

Figura 2.4: Requesitos de um sistema Biométrico

2.2.2 Panorama das técnicas biométricas mais usadas

Existem vários métodos biométricos em utilização (livres e comerciais) atualmente, em seguida é apresentado um resumo das características biométricas mais usadas [13]:

Biológicas:

- Impressões digitais;
- Reconhecimento facial;
- Reconhecimento da retina;
- Reconhecimento da íris;
- Geometria da orelha;
- Geometria da mão;
- Impressão da palma da mão;
- Padrão vascular da mão;
- ADN;

Comportamentais:

- Reconhecimento de voz;
- Reconhecimento de escrita;
- Reconhecimento de padrão de teclar;
- Reconhecimento do equilíbrio;
- Reconhecimento de gestos;

- Reconhecimento do batimento cardíaco;

Químicas:

- Odor
- Padrão termográfico da face
- Padrão termográfico do dorso da mão

Em seguida apresentam-se algumas das características biométricas normalmente utilizadas e cujas tecnologias apresentam um estado avançado de implementação, indicando as suas particularidades.

2.2.2.1 Geometria da mão

O reconhecimento da geometria da mão resulta de uma análise de características como a forma, o comprimento dos dedos e as suas linhas características [10]. Podemos ter diferentes níveis de segurança neste sistema consoante se utilizem as características em si, a sua posição relativamente a um ponto fixo, ou a fixação de vários pontos e as suas distâncias relativas (Figura 2.5).

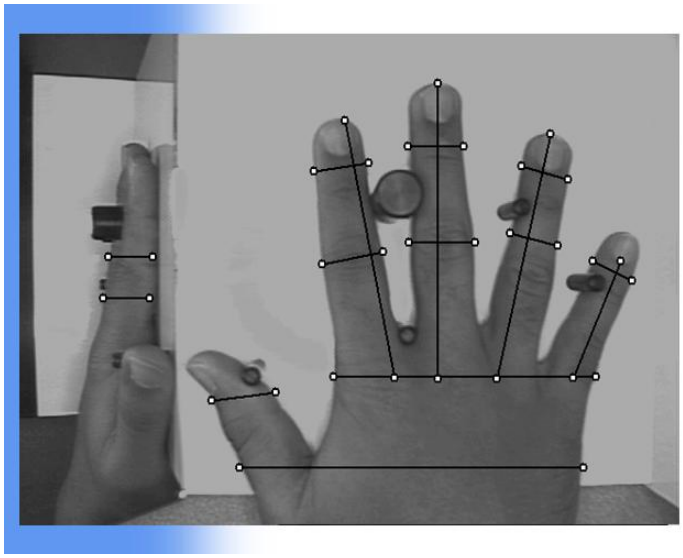


Figura 2.5: Sensor para a geometria da mão

De realçar que a geometria da mão não é uma característica própria de cada indivíduo, mas tem a vantagem de facilmente ser combinada com outras biometrias como, por exemplo, a impressão digital.

A geometria da mão, comparada com outras biometrias, produz uma quantidade reduzida de informação. Portanto, se um grande número de registos for usado, a geometria da mão pode não ser capaz de distinguir um indivíduo de outro com características da mão semelhantes. Esta propriedade pode ser analisada usando a mão toda, ou apenas parte da mão, como por exemplo

analisar apenas dois dedos. Tem a vantagem de ser invariante com a idade e de ser bastante útil para ambientes exteriores. O uso de templates facilita o peso computacional de comparação entre os dados extraídos da mão de forma a poder identificar um individuo. São poucos os dados técnicos relativos a sistemas biométricos baseados na geometria da mão. Quando combinada com outros fatores inerentes à mão, como as linhas da palma, os valores melhoram consideravelmente, não pela precisão da representação mas por fatores ligados aos algoritmos de decisão.

2.2.2.2 Impressões Digitais

As Impressões Digitais são formadas por dobras na pele encontradas nos dedos das mãos e dos pés que resultam numa representação geométrica formada pelas papilas (elevações da pele), presentes na região distal dos dedos, passíveis de serem marcadas numa superfície lisa. As impressões digitais são únicas em cada indivíduo, sendo distintas inclusive entre gémeos monozigóticos. A sua formação depende do desenvolvimento embrionário e de fatores genéticos e ambientais [10].

São datadas desde o período pré-histórico, mas foi em 1667, no período científico, que se iniciaram os primeiros estudos das suas características, altura em que apareceu a Dactilosscopia (Médicos de Portugal: Dactiloscopia). O primeiro sistema científico de identificação foi o sistema antropométrico, lançado em Paris por Alfonse Bertillon (1853-1914), em 1882. Em 1888, o inglês Francis Galton (1822-1911) estabeleceu as bases científicas da impressão digital. Foram usadas para identificação pessoal por muitos séculos e a precisão de correspondência é muito alta. Primariamente eram extraídas criando uma versão de tinta num papel, hoje em dia, há tecnologias compactas que as tornam passíveis de ser adquiridas por digitalização(Figura 2.6).

Figura 2.6: Exemplo Impressão digital

Há vários benefícios da utilização de sistemas de reconhecimento de impressões digitais, entre eles: é fácil de usar e de instalar, e é normalmente passível de ser usado com equipamentos baratos e com baixo consumo de energia.

Contudo existem sempre questões que se colocam em sistemas baseados nesta propriedade biométrica como: que dedo se deve usar ou em que posição deve o dedo estar. Problemas como artrites, unhas compridas, presença de cremes, disfunção circulatória e exposição a radiação condicionam bastante a aquisição de impressões digitais.

2.2.2.3 Reconhecimento Facial

O reconhecimento facial é um método de identificação simples, rápido, não intrusivo e facilmente aceite pelos indivíduos, sendo o modo natural de reconhecimento humano. Baseia-se na recolha e análise de uma imagem da face do individuo para estabelecimento da identidade. Esse reconhecimento pode ser feito de duas formas, aplicando uma transformada ou baseada na localização dos atributos que caracterizam a face. Existem uma série de factores que influenciam o reconhecimento facial tais como: o disfarce facial, as expressões faciais, as condições de iluminação, a distância à câmara e a pose. Este tipo de identificação biométrica está sujeito a ataques simples (uso de uma foto, óculos ou boné), ou complexos (com o auxilio de uma máscara) [10].

Sendo que existem três soluções que se destacam pela sua utilização ser mais comum e que já são amplamente estudadas: PCA - Principal Component Analysis, LDA - Linear Discriminant Analysis e PBLH - Local Binary Pattern Histogram que irão ser descritas em seguida.

Para reconhecer rostos quando o conjunto de imagens de treino é pequeno, os métodos *Eigenfaces* (PCA) e *Fisherfaces* (LDA) não devem ser usados. Para o melhor funcionamento destes modelos, quantas mais imagens, melhor. Precisa-se de dados para trabalhar com estes modelos uma vez que são baseados na estimativa das variâncias nos dados, por isso quantos mais dados, mais completo o modelo fica. Uma análise muito detalhada dos métodos PCA (*Eigenfaces*) e LDA (*Fisherfaces*) para pequenos conjuntos de dados, pode ser consultada no artigo [14], Foi feita a comparação destes dois métodos usando a *AT&T Facedatabase*, usando a *framework facerec*, que mostra o desempenho destes métodos com um número variável de imagens por pessoa, Figura 2.7:

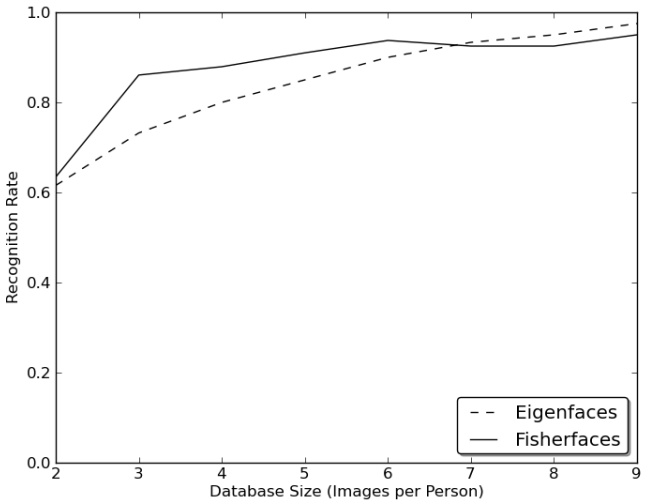


Figura 2.7: Eficácia dos Métodos por número de imagens

Eigenfaces (PCA)

Esta técnica de utilizar *eigenfaces* para reconhecimento de faces foi desenvolvido por Sirovich e Kirby (1987) [15]. Foi considerado o primeiro exemplo bem-sucedido da tecnologia de reconhecimento facial. Eigenfaces tem vantagens em relação às outras técnicas disponíveis, Fisherfaces e PBLH, tais como a velocidade e a eficiência do sistema. É um método rápido em comparação e capaz de funcionalmente operar em muitos rostos em pouco tempo.

Este método de reconhecimento facial, um dos mais intuitivos para classificar uma face, consiste na extração e codificação de toda a informação da imagem facial e posterior comparação da imagem codificada com a base de dados.

Parte de um príncipio semelhante á FT (Transformada de Fourier) [15], frequentemente usada em aplicações ligadas com o processamento de sinais, Figura 2.8. Esta operação matemática pega numa função e decompõe-a em várias funções oscilatórias com parâmetros bem conhecidos. De forma análoga, cada imagem facial é decomposta numa série de componentes, ou vetores próprios da matriz de covariância, definido por um conjunto de faces de referência. Basicamente, cada face pode ser representada como a combinação linear de diversas *Eigenfaces*.

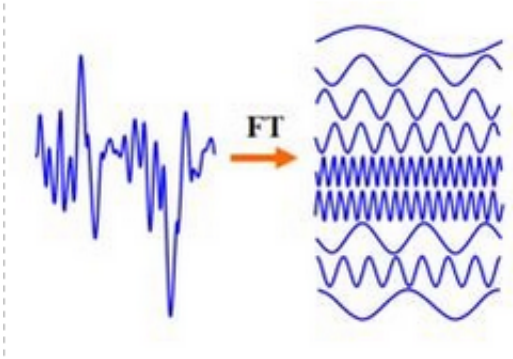


Figura 2.8: Transformada de Fourier

Para ilustração do método, usou-se a seguinte base de dados, Figura 2.9.

Figura 2.9: Imagens da base de dados

As eigenfaces podem ser calculadas nos seguintes 6 passos:

- **Passo 1:** Assume-se uma imagem facial que pode ser vista como uma matriz A de n*m pixéis, ou seja um espaço bidimensional. Uma imagem com 184*224 pixéis passa a ser considerada um vector de dimensão 41,216, ou então um ponto num espaço com 41,216 dimensões. Para relacionar um conjunto de imagens, necessita-se de um "espaço de faces" no qual se processa o cálculo de similaridade entre duas faces. Assim, ao invés de representar as imagens como matrizes, estas são representadas como pontos. Num espaço de grandes dimensões como este, a distribuição não é feita de forma aleatória. Sabendo que todas as faces partilham as mesmas caracteristicas (ex: dois olhos, nariz, boca) e aproveitando essa similiaridade, consegue-se descrever recorrendo a um subespaço relativamente pequeno. Procura-se encontrar os vectores que melhor representam a distribuição das faces estudadas em todo o espaço da imagem. Para isso, todas as linhas da matriz A são unidas num único vetor linha G com n*m coordenadas, desta forma obtêm-se um "espaço de faces" nxm. Uma vez que esses vectores são os vectores próprios (eigenvectors) da matriz covariância correspondente às imagens faciais originais, e uma vez que se assemelham a faces, são designados eigenfaces.

- **Passo 2:** Em seguida, a face média (Y) do conjunto é determinada através da expressão seguinte, que representa o centro de gravidade do conjunto de faces. Um exemplo do tipo de resultado obtido pode ser observado na Figura 2.10.

$Y = \frac{1}{M} \sum_{i=1}^{M} G_i$

Figura 2.10: Face média Y do conjunto de treino

- **Passo 3:** Para todo vetor G calcula-se o vetor F: $F = G_i - Y$

  A Figura 2.11 apresenta o resultado deste passo.

Figura 2.11: Faces subtraídas da média

- **Passo 4:** Constrói-se a matriz A tal que cada i-ésima coluna da matriz é a transposta do vetor linha Fi. Portanto, nesta experiência, a matriz A possui a dimensão de 41216x16.

- **Passo 5:** Calcula-se a matriz de covariância C: $C = AA^T$

- **Passo 6:** Para se obter as eigenfaces é preciso apenas processar os autovetores da matriz AA^T. Entretanto, a matriz AA^T possui a dimensão 41216x41216, o que faz com que o cálculo de seus autovetores seja computacionalmente inviável. Para contornar este problema considera-se a matriz A^TA de dimensão 20x20 e seus autovetores v tais que:

  $A^TAv_i = \lambda v_i$

  Multiplicando ambos os lados por A, tem-se:

  $AA^TAv_i = A\lambda v_i$

  Desta forma pode-se observar que $A\lambda v_i$ são os autovetores de $C = AA^T$ associados aos 20 maiores autovalores da matriz.

A Figura 2.12 apresenta as eigenfaces obtidas com as imagens usadas.

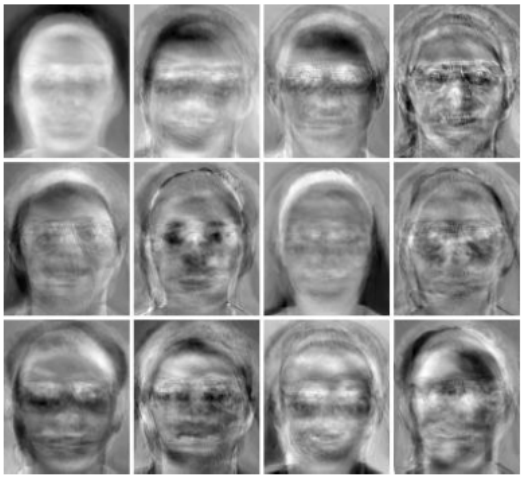


Figura 2.12: Eigenfaces

Para se calcular a proximidade entre duas faces, é necessário projetar a imagem do rosto no "espaço de faces", o que pode ser facilmente processado considerando as M eigenfaces u:

$\omega_k = \mu_k^T F$

Para k = 1,..., M. Desta forma os pesos ω formam o vector::

$\Omega^T = [\omega_1, \omega_2, ..., \omega_M]$

Que representa a imagem projetada no "espaço de faces". Após o cálculo do vetor Ω para todos os rostos armazenados no banco, inicia-se a etapa de reconhecimento facial.

A etapa de reconhecimento facial recebe como entrada uma imagem capturada e efetua-se nos seguintes 4 passos seguintes:

- **Passo 1** - Para a nova face calcula-se o vetor F.
- **Passo 2** - Projeta-se no "espaço de faces" o vetor obtido no passo anterior.
- **Passo 3** - Utilizando a projeção Ω obtida no passo 2 calcula-se a distância entre Ω e todas as projeções obtidas na passo 5 da construção das eigenfaces, selecionando a menor distância d entre as projeções:

  $d = min \|\Omega - \Omega_k\|$

- **Passo 4** - Se d for menor que um limiar pré-estabelecido, então a nova face pertence ao indivíduo cujo rosto armazenado no banco apresenta uma distância d da nova face. Caso contrário, trata-se de uma face desconhecida.

Fisherfaces (LDA)

Outro método de análise de faces é o de fischerfaces, do tipo discriminante linear de Fisher (FLD), também conhecido com análise de discriminantes linear (ADL), foi desenvolvido por R. A. Fisher na década de 1930, porém, apenas recentemente tem sido utilizado para o reconhecimento de pessoas e objetos. É um método específico à classe, pois, ele trabalha com o uso de "rótulos", uma vez identificado os rostos indicando qual a face que pertence a que individuo, os mesmos são agrupados por individuo, e cada agrupamento desses é conhecido como classe. O método tenta modelar a dispersão dos pontos visando maior confiabilidade para a classificação. O ADL busca otimizar a melhor linha em uma superfície que separa satisfatoriamente as classes [16].

Inicia-se o algoritmo obtendo as matrizes de dispersão entre classes, interclasse, e dentro das classes, intraclasse. A projeção é feita maximizando a dispersão interclasse e minimizando a intraclasse, formulado pela razão entre as determinantes de ambas as matrizes, com isso diferindo do ACP, que maximiza oespalhamento, dispersão, dos padrões no espaço de características, independente da classe a que esses pertencem [17]. As duas medidas citadas, matematicamente são definidas como:

- **1.** matriz de dispersão intraclasses, within class:

  $S_W = \sum_{j=1}^{c} \sum_{i=1}^{|T_j|} (x_i^j - \mu_j).(x_i^j - \mu_j)^T$

  em que x_i^j é o i-ésimo exemplo da classe j,μ_j é a média da classe j, c é o número de classes, e $|T_j|$ o número de exemplos na classe j

- **2.** matriz de dispersão interclasses, between class:

  $S_b = \sum_{j=1}^{c} (\mu_j - \mu).(\mu_j - \mu)^T$

  ,em que μ representa a média de todas as classes.

A maximização da medida inter-classes e a minimização da intra-classes são obtidas ao maximizar a taxa $\frac{det(Sb)}{det(Sx)}$ O espaço de projeção é então encontrado resolvendo a equação $S_bW = \lambda S_W W$, onde W é a matriz com autovetores generalizados associados com λ, que é a matriz diagonal com autovalores. Essas matrizes estão limitadas à ordem c-1, em que c é o i número de classes, limitação devido à comparação ser realizada entre duas classes diferentes.

Para identificar uma imagem de teste, o algoritmo funciona da mesma forma que o *Eigenfaces*. A imagem de teste é projetada e comparada com cada uma das faces de treino também projetadas, identificando-a com a de treino que mais se aproxima. A comparação, é de novo feita utilizando um classificador específico ou a combinação de dois ou mais.

Padrão Binário Local do Histograma (LBPH)

Método utilizado para a extração de características globais, descrito pela primeira vez por T. Ojala, M. Pietikäinen e D. Harwood [16]. Devido ao poder discriminativo que possui e à sua simplicidade computacional, este operador de texturas tem sido utilizado em diversas aplicações, principalmente como classificador de caracteristicas humanas [17].

Em oposição aos algoritmos descritos antes, o LBPH num problema de reconhecimento facial faz uma extração das características da imagem através de uma abordagem local, não necessitando de projetar as imagens num sub-espaço.

As imagens faciais podem ser vistas como uma composição de pequenas regiões que são bem descritas pelo LBP. Neste método cada pixel de uma imagem é substituído por um valor binário. Este valor é determinado pela comparação de uma matriz quadrada contendo os pixels vizinhos, onde cada vizinho é comparado com o valor central, conforme a seguinte condição:

$$b_{ij} = \{0, v_{ij} < v_c 1, v_{ij} \geq v_c\}$$

Os valores obtidos para cada vizinho são concatenados e o número binário gerado é convertido na base decimal para substituir o valor central vc. A figura 2.13 exemplifica este processo para cada píxel, multiplica-se os valores da vizinhança por uma máscara de pesos, e somando-se obtém-se o resultado (1+2+4+8+128 =143), resultado que se considera como sendo um número binário. Após se obter o resultado em binário de todos os pixéis, será calculado o histograma com 28 = 256 valores que vai ser usado como descritor. para uma matriz 3*3 de pixels vizinhos. Contudo, o tamanho e o formato da vizinhança podem variar.

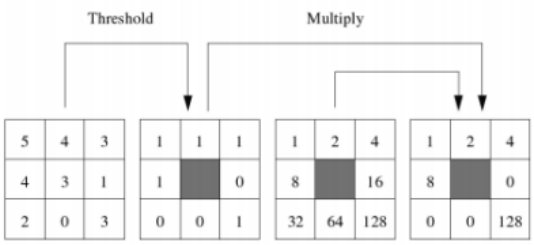


Figura 2.13: PBLH de cada píxel

A imagem gerada é dividida em regiões e são extraídos os descritores PBLH de cada região independentemente. Entretanto, é possível empregar outras formas com, ou sem, sobreposição. Os descritores de textura são extraídos de cada região isoladamente, calculando-se o histograma de intensidade dos pixels. Os descritores são depois concatenados e formam um descritor global da face, como mostrado na Figura 2.14:

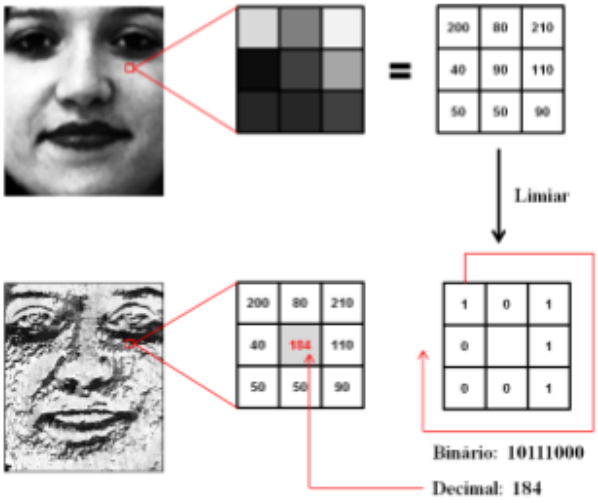


Figura 2.14: Descritores PBLH

Para calcular as semelhanças entre imagens diferentes vamos usar a distância euclidiana entre os diferentes descritores PBLH.

2.2.2.4 Reconhecimento da Retina

A retina é a camada mais interna do olho humano, é a parte do olho sensível a luz, sendo responsável pela formação de imagens, possibilitando a visão [10]. A sua vasculatura tem uma estrutura elaborada (Figura 2.15). O padrão de distribuição desses vasos é único para cada pessoa, sendo tão complexo que até gémeos monozigóticos possuem configurações diferentes. Esta não se altera durante toda vida do indivíduo, excepto em caso de mutilação ou condições médicas que alterem o olho de alguma forma. O fato de a retina possuir uma configuração única e ser muito estável ao longo da vida faz dela uma das biometrias mais confiáveis conhecidas.

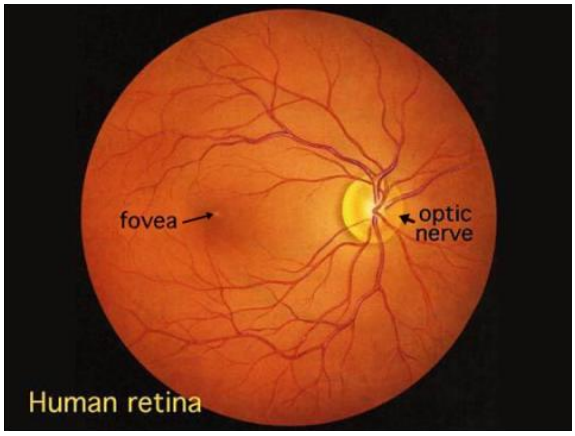


Figura 2.15: Exemplo Retina

A captura da imagem da retina requer a cooperação do individuo a ser identificado, o que por vezes não é bem aceite, pois requer que a pessoa se ajuste a um óculo e se concentre num ponto especifico. Uma parte pré-determinada da vascularização da retina é imaginada para simplificar o processo de captura de informação. Este método de reconhecimento biométrico pode ser revelador de condições médicas como hipertensão ou diabetes.

2.2.2.5 Reconhecimento da Irís

O reconhecimento da íris é outro sistema de reconhecimento biométrico capaz de reconhecer positivamente a identidade dos indivíduos sem contato físico [10].A íris é a parte colorida do olho humano que envolve a pupila e a sua leitura pode ser feita mesmo que hajam lentes de contacto ou óculos. A forma de funcionamento deste sistema baseia-se num feixe de luz na *gama near infrared* (resolução 320x480 pixéis posicionada a menos de 1 m) enviado através de uma câmara que incide sobre o olho e regista os contornos e padrões geométricos, criando o padrão biométrico. A leitura deve ser feita num ambiente bem iluminado. As imagens recolhidas são então segmentadas e codificadas.

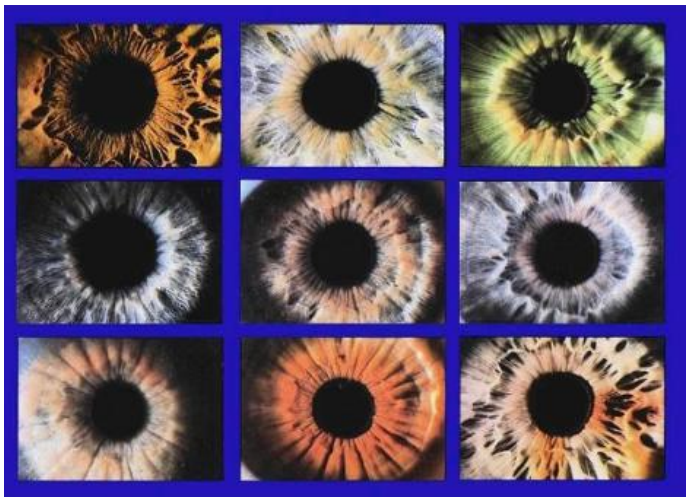


Figura 2.16: Exemplos Irís

O reconhecimento leva geralmente apenas um ou dois segundos para ser concluído. É uma das técnicas biométricas mais precisas de autenticação de utilizadores. Um exemplo mediático baseado no reconhecimento de íris são as *Indian Technologies, Inc* de Moorestown. No entanto nesta propriedade biométrica existem algumas nuances, tais como qual o olho a usar, os riscos inerentes à saúde (danificar os olhos ou despoletar epilepsia).

Pode também fornecer informação clínica de diagnóstico (hipertensão, diabetes, gravidez e uso de substâncias dopantes). É de alta precisão porque as íris é diferente de pessoa para pessoa, mesmo entre gémeos monozigóticos, e raramente muda (acidentes). Quanto a vulnerabilidades de segurança, este método é suscetível de ataques simples ou complexos (uso de lentes de contacto).

2.2.2.6 Reconhecimento de Voz

A voz é uma combinação de dados de várias origens, tais como: biométricas, fisiológicas e comportamentais. As características da voz de um indivíduo são baseados no tamanho e forma dos apêndices (trato vocal, boca, fossas nasais e lábios) que são usados para a síntese do som. Estas características fisiológicas da fala humana são invariantes para um indivíduo, mas a parte comportamental do discurso de uma pessoa muda ao longo do tempo devido à idade, condições médicas (constipação, tosse), estado emocional, entre outros. A voz também não é muito distinta, e pode não ser apropriada para a identificação em larga escala [10].

Um sistema de reconhecimento de texto dependente da voz é baseado na emissão de uma frase fixa pré-determinada, enquanto um sistema de reconhecimento de voz independente de texto reconhece a pessoa independente do que ela fala. Um sistema de texto-independente é mais difícil de conceber que um sistema de texto-dependente, mas oferece maior proteção contra a fraude. A grande desvantagem do reconhecimento de voz é este ser baseado nas características do discurso, e estas serem sensíveis a uma série de fatores como o ruído de fundo. O reconhecimento de voz é mais adequado para aplicações baseadas no sinal telefónico, mas o sinal de voz por telefone é também afetado por uma degradação na qualidade, pelo microfone e/ou pelo canal de comunicação.

A análise de sistemas biométricos baseados em reconhecimento de voz é feita com base na forma de onda e de pressão do ar quando o individuo fala, usando normalmente características individuais da voz ou algumas palavras padrão. Este parâmetro apresenta uma baixa fiabilidade causada pela baixa assertividade.

2.2.2.7 Geometria da orelha

A identificação de um individuo baseado na geometria da orelha, consiste no uso de uma câmara digital para a recolha de uma imagem da orelha a uma certa distância e ângulo. Este método tem como grande vantagem registar à distância particularidades do membro, como lóbulo e as formações dentro da concha auricular. Além de ser pouco intrusivo ao corpo humano, a biometria por reconhecimento da geometria de orelha apresenta como outra grande vantagem: a baixa alteração na dimensão da orelha no decorrer da vida do individuo [10].

A geometria da orelha é analisada com base em 2 vetores característicos do mesmo contorno (Figura 2.17), método que tem uma eficiência de cerca de 90% e é invariante à escala e rotação, apresenta ainda uma baixa complexidade computacional sendo um método recomendado para identificação passiva.

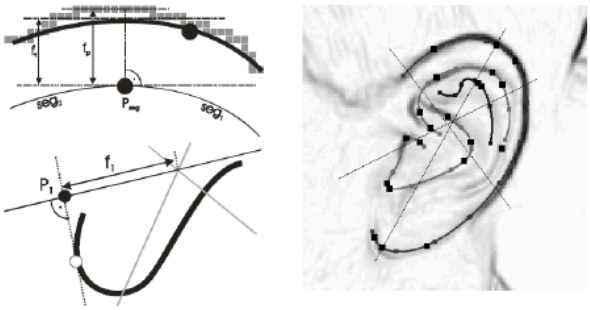


Figura 2.17: Análise de geometria da orelha

A distancia e o tamanho do individuo são condicionantes à suscetibilidade a erros, é um sistema facilmente comprometido por fraude através de um molde de orelha falsa. Esta falta de segurança poderia ser ultrapassada caso se pudesse adicionar um sistema que monitorizasse o padrão de circulação sanguínea da orelha.

2.2.3 Impressão da palma da mão

A palma da mão contêm um padrão de irregularidades (rasgos e vales) que são semelhantes à impressão digital (Figura 2.18). Também dependem de fatores genéticos e ambientais. Porém, a área da palma é muito maior, pelo que é esperado que sejam ainda mais distintas do que a impressão digital. Porém como os digitalizadores da palma necessitam abranger uma área maior, tem uma dimensão e custo superior que os de impressão digital. Ainda assim, como as palmas têm características distintas como as linhas principais e as rugas isso permite que a captura seja feita numa resolução menor o que torna esses sensores mais baratos [10].

Figura 2.18: Exemplo de impressão digital da palma da mão

Este método poderá ter problemas com feridas e cicatrizes, e também é suscetível de fraude por molde. A análise das características da palma da mão e comparação com os valores presentes numa base de dados para verificação é computacionalmente pesado.

2.2.4 Padrão vascular da mão

O padrão das veias da mão é uma característica única de cada individuo e invariante ao longo da vida. A digitalização desse padrão pode ser feita através da utilização de luz projetada contra a mão de modo a criar uma comparação de alto contraste dos padrões de veias nos dedos, ou na mão (Figura 2.19). A medição das características que estão por baixo da pele faz com que sejam mais difíceis de observar pelos outros, tornando assim a característica biométrica do padrão das veias, um método de verificação especialmente seguro [10].

Figura 2.19: Exemplo do padrão vascular da mão

O dispositivo de digitalização é composto por uma matriz de LEDs e uma câmara CCD, não pressupondo contacto (Figura 2.20). Trata-se de um parâmetro de alta aceitação, podendo ser utilizado em sistemas de verificação biométrica múltipla.

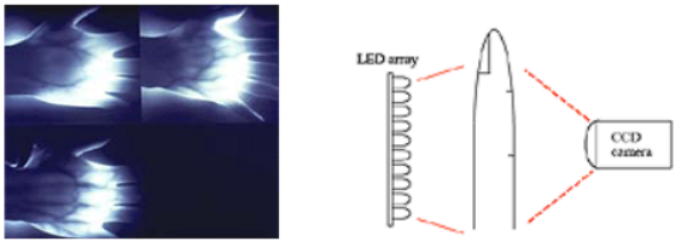


Figura 2.20: Digitalização da vasculatura da mão

2.2.5 ADN

O ADN é um composto orgânico, cujas moléculas contêm as instruções genéticas que coordenam o desenvolvimento e funcionamento de todos os seres vivos, transmitindo as características hereditárias de cada ser vivo. O seu principal papel é armazenar as informações necessárias para a construção das proteínas e ácido ribonucleico [10]. Trata-se de um código unidimensional único para indivíduos, exceção feita aos gémeos monozigóticos, em que o ADN é idêntico. Contudo, este parâmetro biométrico apresenta certas limitações, tais como:

- **Contaminação e sensibilidade** - com relativa facilidade pode-se furtar um pedaço de ADN de outra pessoa sem que esta se aperceba;
- **Problemas no que diz respeito ao reconhecimento automático em tempo real** - a tecnologia existente requer o uso de alguns produtos químicos, bem como competências de um perito, não estando orientada para o reconhecimento não invasivo em tempo real;
- **Problemas de privacidade** - a informação acerca da suscetibilidade da pessoa ter uma doença, pode ser obtida através do padrão de ADN e, essa é uma preocupação pois a informação do código genético pode ser usada com o intuito discriminatório, como por exemplo, em práticas de contratação.

2.2.6 Reconhecimento de escrita

A forma como um individuo assina o seu nome é conhecido por ser uma característica relativamente única. Embora nas assinaturas seja necessário o contacto com o instrumento de escrita e um esforço por parte do utilizador, existem soluções com auxílio de esferográficas digitais que digitalizam a assinatura, ou escrita do sujeito. As assinaturas são um comportamento biométrico que muda ao longo de um período de tempo e são influenciados pelas condições físicas e emocionais dos signatários. As assinaturas de algumas pessoas variam substancialmente: até mesmo

as impressões sucessivas de sua assinatura são significativamente diferentes. Estes métodos são permissivos à reprodução de assinaturas, o que consequentemente comprometeria o sistema [10].

2.2.7 Reconhecimento do padrão de teclar

Cada utilizador tem um perfil de interagir com um ecrã táctil. Essa característica pode ser entendida como um comportamento biométrico, contudo não deverá ser único para cada indivíduo, mas oferece informações discriminatórias suficientes para permitir a verificação de identidade. A dinâmica de digitação é um comportamento biométrico, podendo-se esperar grandes variações em padrões típicos (velocidade, pressão e latência) de digitação (Figura 2.21). Além disso, esta verificação poderia ser facilmente feita, de uma forma discreta quando alguém digita informações via teclado digital. Contudo é um método de fraca assertividade [10].

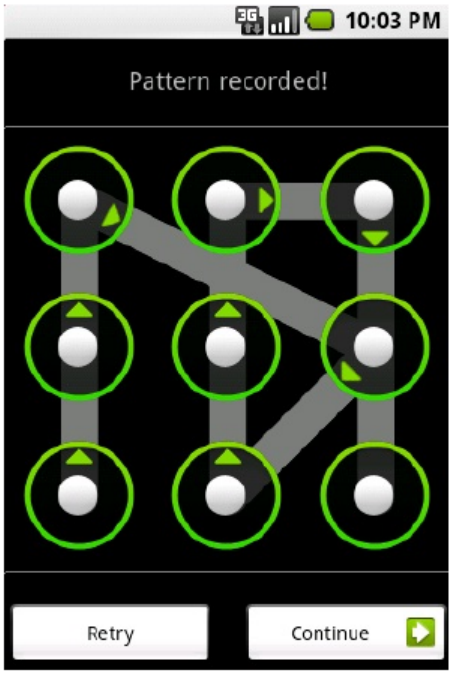


Figura 2.21: Exemplo de proteção por padrão de teclar em dispositivo móvel

2.2.8 Reconhecimento de equilíbrio

Trata-se uma forma peculiar de reconhecimento, pois é um dado biométrico complexo tanto no espaço, como no tempo, consiste em analisar a locomoção humana, que pode ser feita através de imagens vídeo (com recurso a câmara) ou por sensores, como acelerómetros e giroscópios que registam a velocidade e ângulos do movimento. O andar pode não ser notoriamente diferente, no entanto é diferente o suficiente para que seja discriminante num sistema que necessite uma segurança pouco complexa (menos confiável e possivelmente com mais falhas). Contudo, se a captura do andar for feita através da recolha de um vídeo, para captar diversos movimentos de cada articulação, irá tornar o sistema computacionalmente complexo [10]. Há ainda a salientar

que a forma de andar é temporalmente variável, depende de fatores externos, como o piso e estado de saúde do individuo e qualquer que seja a escolha para recolha de dados, a análise dos mesmos será sempre complexa.

2.2.9 Reconhecimento de gestos

O reconhecimento de gestos é realizado utilizando uma câmara de vídeo e técnicas da área de visão computacional na qual se utiliza um conjunto de técnicas de processamento de imagens e de análise de séries temporais para fazer com que o dispositivo com poder de processamento identifique o gesto capturado [10]. Apesar de ser possível o reconhecimento de gestos envolvendo mãos, braços, corpo ou cabeça, são mais comuns a deteção de gestos e padrões das mãos (Figura 2.22). Estes gestos podem ser estáticos (imagens captadas) ou dinâmicos (movimentos pré-definidos).

Figura 2.22: Exemplos de gestos passíveis de ser reconhecidos

Os padrões gestuais servem como uma chave gestual (token) para confirmar a identidade de uma pessoa. Cada sujeito será relacionado com um determinado padrão gestual, que irá servir como uma forma de poder aumentar a precisão e confiabilidade de outro sistema de reconhecimento biométrico. Tratando-se de um método de fácil aceitação por parte do público, o mesmo requer um sistema de suporte com alguma complexidade para análise e processamento de imagens com recurso a inteligência artificial, no entanto usando apenas reconhecimento gestual não garante só por si a identificação univocamente um individuo.

2.2.10 Reconhecimento de batimento cardíaco

O electrocardiograma (ECG) é o registo do sinal elétrico produzido pelo coração durante a sua atividade (Figura 2.23). Cada pessoa apresenta um sinal ECG distinto, que pode servir como característica biométrica. Existem fortes evidências de que este sinal é suficientemente discriminativo para identificar um indivíduo num vasto grupo populacional. A própria medição deste sinal possui inerentemente a verificação de que a pessoa está viva. Outras informações podem ser obtidas deste sinal, tais como, diferentes estados de emoção ou stress [11].

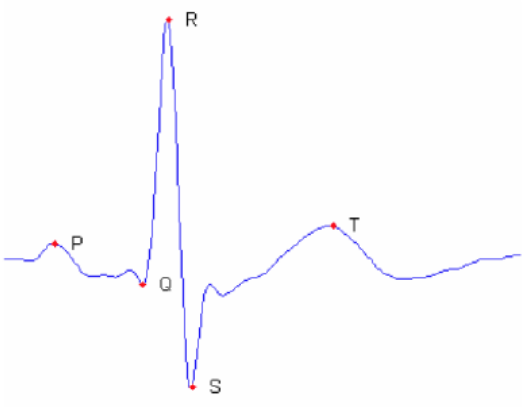


Figura 2.23: Exemplo de um sinal ECG

Na área do reconhecimento biométrico, o sinal ECG pode ser classificado de acordo com a abordagem seguida, em fiducial ou não-fiducial. Na abordagem fiducial usam-se pontos de referência a partir dos quais se mede o tempo ou a diferença de amplitude entre esses pontos para a criação de um modelo (template). Por outro lado, a abordagem não-fiducial trata o sinal ECG ou os vários segmentos do batimento cardíaco como um todo, obtendo características de base estatística relativas à morfologia geral do sinal. Ambas as abordagens têm vantagens e desvantagens. Enquanto que o risco da abordagem fiducial é a perda de informação escondida por de trás da morfologia da biometria, na abordagem não-fiducial lida-se com uma grande quantidade de informação redundante que precisa de ser eliminada. O desafio está em remover essa informação de forma a minimizar a variação entre amostras do mesmo sujeito e maximizar a variação entre sujeitos diferentes.

Uma das características fundamentais dos sistemas biométricos é a capacidade de não intrusão na aquisição dos sinais biométricos. Os sistemas biométricos baseados no sinal ECG tendem cada vez mais a seguir este objetivo, diminuindo o número de elétrodos utilizados e até mesmo o local onde estes são aplicados (junto ao peito, mãos, etc.).

As grandes condicionantes deste método são a necessidade de utilizar um sensor no corpo e numa posição específica, existência de alguma complexidade na análise dos dados recolhidos e a possibilidade do sinal poder variar com o tempo e com condições clínicas.

2.2.11 Odor

Cada individuo emana um odor que é característico da sua composição química e isso poderia ser utilizado para o identificar. Uma lufada de ar em torno de um indivíduo é soprado através de uma variedade de sensores químicos, cada um sensível a um determinado grupo (aromático) de compostos. Um componente do odor emitido pelo corpo de um ser humano é distinto de indivíduo para indivíduo.

Não está claro se a invariância no odor do corpo poderia ser detectada, apesar da existência de desodorizantes de cheiros, e da variância da composição química do meio ambiente [10].

Os químicos presentes no odor são conhecidos como compostos orgânicos voláteis e são obtidos através de sensores de forma não invasiva em regiões do corpo humano como a palma da mão, sendo depois transformados em templates, cada um desses templates corresponde a um único cheiro humano virtual. Este método tem como limitações o facto dos sensores estarem limitados a recolher uma pequena gama de odores, hábitos dos indivíduos como o uso de desodorizantes afetam o odor natural, a análise dos templates é complexa e computacionalmente exigente e pode vir a demonstrar informação sensível sobre o individuo, como o uso de substância proibidas.

2.2.12 Padrão termográfico da face

O padrão de energia irradiada pela superfície da pele do corpo humano é uma das características de um indivíduo que pode ser captada e registada por uma câmara de infravermelhos de uma forma discreta, muito parecida com uma foto normal (no espectro visível). Deste modo, esta tecnologia poderia ser usada sem que o alvo de reconhecimento tivesse conhecimento do fato. Um sistema baseado num termograma não requer contacto e é não-invasivo, mas a aquisição das imagens é um desafio em ambientes não controlados, onde pode haver reflexão e interferências térmicas que condicionariam a medição das temperaturas superficiais. Um sistema de avaliação térmica da face permitia identificar o padrão microvascular da mesma (Figura 2.24) , sendo este único para cada individuo. Este método tem como grande inconveniente o custo do equipamento necessário e a complexidade no processamento das imagens recolhidas, mas seria um bom método complementar ao reconhecimento facial em imagem digital pois permitia atestar que é uma pessoa que está presente e não uma foto ou máscara, evitando fraudes [10].

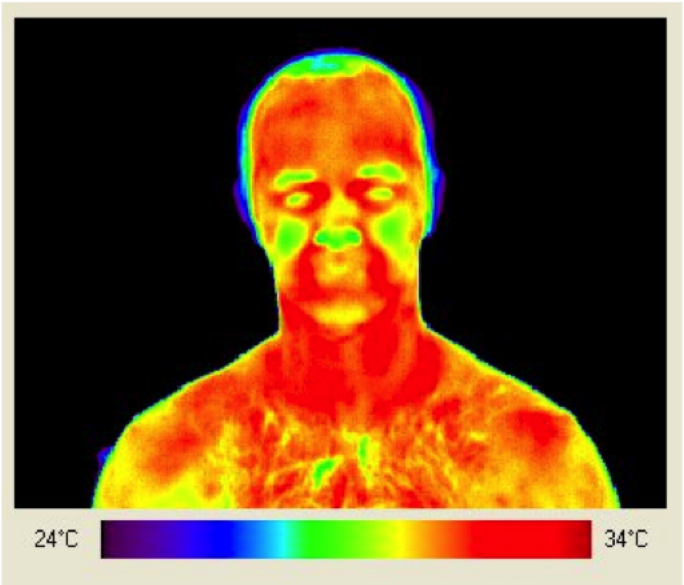


Figura 2.24: Exemplo de uma imagem térmica da face

2.2.13 Padrão termográfico do dorso da mão

À semelhança do método anterior, cada individuo tem um padrão único de vascularização da mão, fator que influencia a microcirculação cutânea da mão que é passível de ser registada com recurso a termografia (Figura 2.25) [10].

Contudo as condicionantes são as mesmas, o custo do equipamento, a complexidade da análise das imagens e a necessidade de ambiente de captura controlado (temperatura ambiente, humidade relativa e fluxo de ar). Seria um ótimo método complementar de biometria para situações onde a exigência de segurança seja crítica.

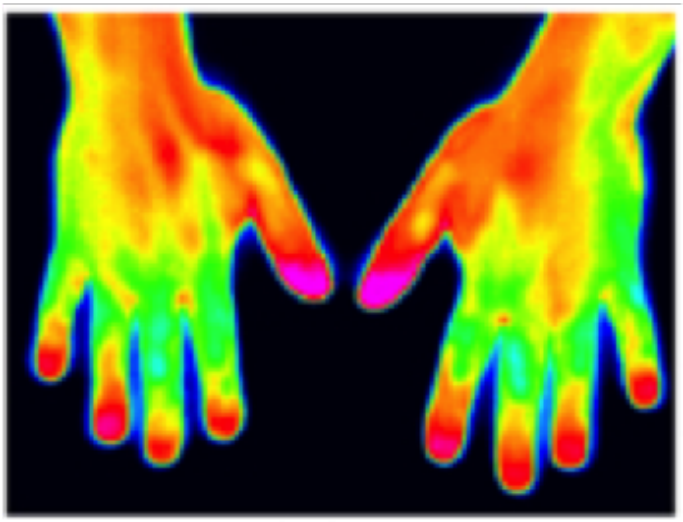


Figura 2.25: Exemplo de uma imagem térmica do dorso das mãos

2.2.14 Desempenho das técnicas de biometria

Devido às várias posições sobre o sensor de aquisição, condições de imagem imperfeitas, mudanças ambientais, deformações, ruído e má interação do utilizador com o sensor, os sistemas biométricos não conseguem ter dois padrões da mesma propriedade biométrica, adquiridos em diferentes sessões, exatamente colineares. Por esta razão, a resposta dos sistemas de reconhecimento é normalmente um resultado **s**, que é avaliado com os dados guardados numa base de dados. Uma pontuação de semelhança é comparada com um limiar **t** e caso seja maior ou igual as amostras comparadas pertencem à mesma pessoa. Caso **s** seja menor que **t** trata-se de um impostor 2.26.

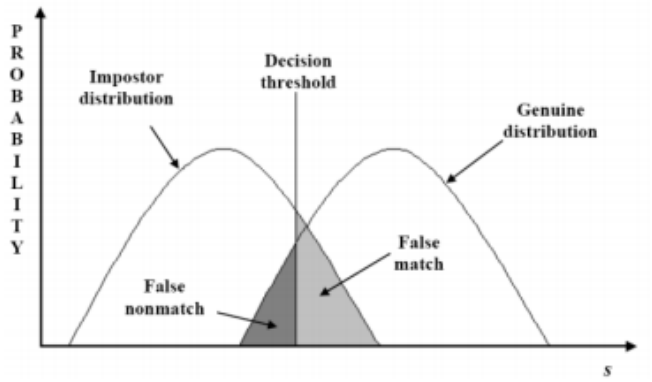


Figura 2.26: Taxas de erro em Sistemas Biométricos

As seguintes regras de desempenho são utilizadas para medir esses erros [10]:

- Falsos Positivos(FMR - *False match rate*) - mede a percentagem de entradas inválidas que incorretamente correspondem ao padrão.
- Falsos negativos (FNMR - *False non-match rate*) - mede a percentagem de entradas válidas que são incorretamente rejeitadas.

As funções básicas do limiar são o FNMR e FMR. Quando o arquitecto do sistema torna o sistema mais tolerante às entradas, o valor de **t** diminui, o que aumenta o FMR. Quando torna o sistema mais seguro, então o valor de **t** aumenta. FMR e FNMR são juntos numa curva característica de operação do receptor (ROC- receiver operating characteristic) que traça o FMR contra FNMR (ou 1-FNMR) em diferentes limiares 2.27

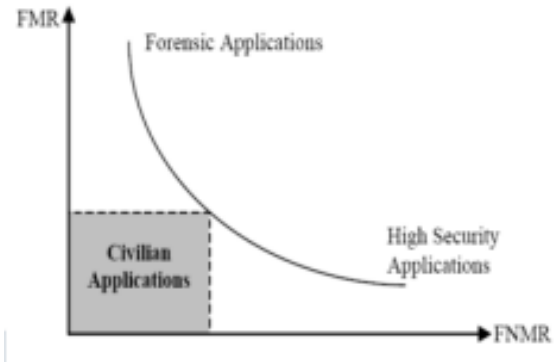


Figura 2.27: Receiver operating characteristic (ROC)

2.2.14.1 Técnicas Individuais

Considerando uma técnica de biometria como única forma de autenticação, estas têm as seguintes limitações [19]:

- **O ruído nos dados adquiridos**: por exemplo uma impressão digital com uma cicatriz. Dados com ruído podem também resultar da acumulação de sujidade num sensor ou das condições ambientais.

- **Variações internas**: estas variações são normalmente causadas por um utilizador que está a interagir com o sensor (por exemplo, postura facial incorrecta).

- **Distinção**: num sistema que compreende um grande número de utilizadores, pode haver semelhanças (sobreposição) nos vetores de característica de múltiplos utilizadores.

- **Não universal**: o sistema biométrico pode não ser capaz de realizar a aquisição de dados biométricos significativos num subconjunto de utilizadores.

Característica	Universalidade	Singularidade	Permanência	Coletável	Desempenho	Aceitabilidade	Permeabilidade
Impressão digital	Média	Alta	Alta	Média	Alta	Média	Média
Reconhecimento facial	Alta	Baixa	Média	Alta	Baixa	Alta	Alta
Reconhecimento de retina	Alta	Alta	Média	Baixa	Alta	Baixa	Baixa
Reconhecimento da íris	Alta	Alta	Alta	Média	Alta	Baixa	Baixa
Geometria da orelha	Média	Média	Alta	Média	Média	Alta	Média
Geometria da mão	Média	Média	Média	Alta	Média	Média	Média
Impressão da palma da mão	Média	Alta	Alta	Média	Alta	Média	Média
Padrão vascular da mão	Média	Média	Média	Média	Média	Média	Baixa
ADN	Alta	Alta	Alta	Baixa	Alta	Baixa	Baixa
Reconhecimento de voz	Média	Baixa	Baixa	Média	Baixa	Alta	Alta
Reconhecimento de escrita	Baixa	Baixa	Baixa	Alta	Baixa	Alta	Alta
Reconhecimento de padrão de teclar	Baixa	Baixa	Baixa	Média	Baixa	Média	Média
Reconhecimento de equilíbrio	Média	Baixa	Baixa	Alta	Baixa	Alta	Média
Reconhecimento de gestos	Baixa	Baixa	Baixa	Média	Baixa	Alta	Baixa
Reconhecimento de batimento cardíaco	Baixa	Baixa	Média	Média	Média	Média	Baixa
Padrão termográfico da face	Alta	Alta	Baixa	Alta	Alta	Alta	Baixa
Padrão termográfico da mão	Alta	Alta	Baixa	Alta	Alta	Alta	Baixa
Odor	Alta	Alta	Alta	Baixa	Baixa	Média	Baixa

Figura 2.28: Técnicas de biometria e o seu desempenho em cada requisito

2.2.14.2 Técnicas Multi-Plataforma

O próprio nome indica uma fusão de diferentes tipos de informações (por exemplo: impressões digitais e face da mesma pessoa, ou impressões digitais de dois dedos diferentes) [20]. Estas técnicas multi-plataforma foram feitas para combater alguns dos problemas das técnicas individuais [21] [19]:

- Não-universalidade ou uma insuficiente cobertura populacional.
- Torna-se cada vez mais difícil para um impostor para falsificar múltiplas características biométricas de uma pessoa.
- Sistemas multi-plataforma tratam eficazmente o problema dos dados ruidosos (doença que afeta a voz, cicatriz numa impressão digital).

Sistemas multi-plataforma podem oferecer uma melhoria substancial na correspondência e precisão de um sistema biométrico, dependendo da informação que está a ser combinada e a metodologia usada [22].

2.3 Métodos alternativos de identificação em dispositivos móveis

2.3.1 QR Code

O termo "QR Code"é uma abreviação para "*Quick Response Code*", e consiste num tipo de código de barras, mais complexo que os típicos que se vêem normalmente nos artigos de loja. Os códigos de barras padrão são usados para controlo de inventário, facilitando a tarefa de controlar o inventário que é vendido ou retornado. Um QR Code é geralmente usado para fins mais tecnológicos, pois podem armazenar muita mais informação que um código de barras padrão e podem também armazenar praticamente todo o tipo de informações. Normalmente usam símbolos alfabéticos, numéricos ou mesmo Kanjii.

O QR Code é um código de barras bidimensional, Figura 2.29, que pode ser facilmente digitalizado usando a câmara de um dispositivo móvel [23]. Esse código é convertido em texto, um endereço URI, um número de telefone, uma localização geográfica, um e-mail, um contato ou um SMS.

Figura 2.29: Exemplo QR Code

O padrão visual de um QR Code é preto e branco, com um padrão disposto num formato quadrado. Este é o aspecto mais básico, embora recentemente estejam a ganhar popularidade QR Codes personalizados. Tornando-se cada vez mais comum o uso de um logotipo de uma marca ou cores brilhantes para ajudar a agarrar a atenção e atrair potenciais clientes para digitalizar o código e receber mais informações.

2.3.2 Passado, Presente

Os códigos QR são um conceito relativamente novo no mundo da tecnologia, foram originalmente desenhados pela *Toyota Motor Corporation* em 1994 [24] para rapidamente identificar as partes do veículo durante o processo de montagem, de forma a garantir que todas estavam nos sítios corretos. Começaram portanto a ser usadas no Japão mas rapidamente ganharam popularidade pelo mundo. Expandiram-se naturalmente para fora da indústria automóvel devido à rapidez, capacidade de armazenar informação e pela facilidade de acesso à mesma. Como referido, qualquer smartphone pode ser preparado como um leitor de QR Codes, bastando fazer download de uma das muitas aplicações disponíveis. Podem ser colocados em qualquer sítio, t-shirts por exemplo, bastando tirar uma foto e instantaneamente a informação ligada ao QR Code é apresentada no ecrã.

2.3.2.1 Padrões

O padrão japonês para o código QR, JIS X 0510, foi lançado em janeiro de 1999 e corresponde ao padrão internacional ISO/IEC 18004, tendo sido aprovado em junho de 2000.

Segundo o site da Denso-Wave, o "código QR é aberto para uso e sua patente, pela Denso-Wave, não é praticada".

Existem diversos padrões de codificação QR [25]:

- Janeiro de 1999 — JIS X 0510
- Junho de 2000 — ISO/IEC 18004:2000 [26] (presentemente retirado) Define símbolos QR Code Model 1 e QR Code Model 2.

- 1 de setembro de 2006 — ISO/IEC 18004:2006 [23] (Define QR Code 2005 symbols, uma extenão do QR Code Model 2. Não especifica como ler QR Code Model 1 symbols, ou requer-o para aderência ao padrão.)

O projeto de código aberto "ZXing"mantém uma lista de tipo de QR Code [27]

2.3.2.2 Capacidade de armazenamento

Quanto à capacidade de armazenamento num código QR [28], este indica o tipo de conteúdo que foi armazenado, por exemplo, apenas um número, ou um conjunto de caracteres. Para tal, o padrão estabelece diferentes modos de entrada no armazenamento.

Possíveis caracteres conforme modo (e respectiva taxa de ocupação):

- Somente numérico ($3\frac{1}{3}$ bits/char): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Alfanumérico ($5\frac{1}{2}$ bits/char): 0–9, A–Z (maiúsculas apenas), espaço, $, %, *, +, -, ., /, :
- Binário (8 bits/char): padrão ISO 8859-1
- Kanji/kana (13 bits/char): padrão Shift JIS X 0208

A "capacidade em número de caracteres"depende das capacidades da versão ("resolução"em número de módulos), do modo (tipo de caracter), e do nível de correção de erro.

Capacidade da versão 1 (imagem com 21x21 módulos):

- Numérica - máx. 41 caracteres no nível L, 17 no nível H
- Alphanumérica - máx. 25 caracteres no nível L, 10 no nível H
- Binário (8 bits) - máx. 17 bytes no nivel L, 7 no nivel H
- Kanji/Kana - máx. 10 caracteres no nivel L, 4 no nivel H

Capacidade máxima, versão 40 nível L (imagem com 177x177 módulos) :

- Numérica - máx. 7089 caracteres
- Alfanumérica - máx. 4296 caracteres
- Binário (8 bits) - máx. 2953 bytes
- Kanji/Kana - máx. 1817 caracteres

De forma a ser possivel interpretar o código e efetuar correções, assim como verificar a posição do mesmo, é utilizada a análise apresentada pela Figura 2.30.

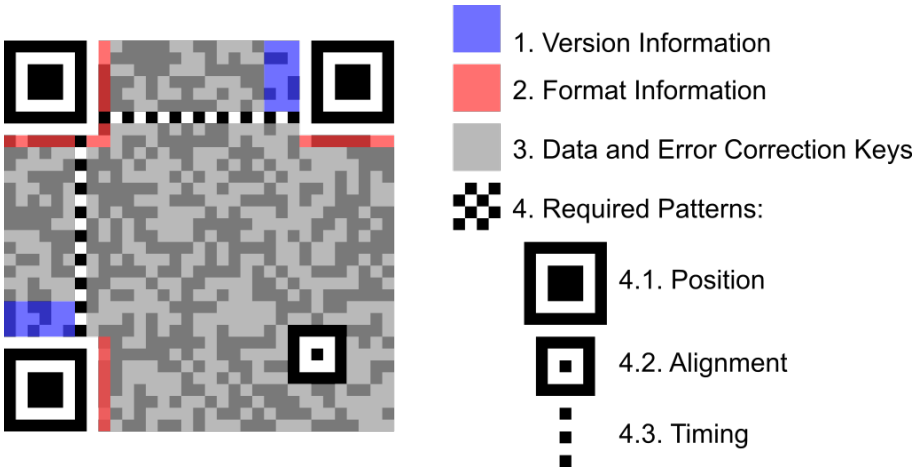


Figura 2.30: Interpretação de um código QR

2.3.3 RFID

A Identificação por radiofrequência, ou RFID, é um método de identificação automática através de sinais de rádio, recuperando e armazenando dados remotamente através de dispositivos denominados etiquetas RFID [29].

De acordo com Roy Want em [29] , "*Radio Frequency IdentificationTechnology* (RFID) passou da obscuridade para as principais aplicaçõe atuais que ajudam a manter o inventário de produtos manufaturados e materiais". O código de barras continua a ser a tecnologia dominante nas indústrias de cadeias de abastecimento e nos departamentos de lojas, no entanto o RFID está a substituir a tecnologia do código de barras e possui a grande vantagem de não ter problemas de leitura a certas distâncias. Baseia-se numa premissa de níveis reduzidos de trabalho, maior visibilidade e melhor gerenciamento de inventário. As etiquetas RFID têm uma capacidade de memória de 16 - 64 Kbytes, que é muito mais do que os códigos de barras (1 - 100 bytes) [30] e pode armazenar dados adicionais, como o nome do fabricante e as especificações do produto.

As primeiras utlizações da tecnologia RFID foram durante a Segunda Guerra Mundial, quando o exército britânico a usou para identificar os aviões, "amigos ou inimigos". Alguns problemas técnicos resultaram no abate de aviões aliados, e desde então o uso da tecnologia RFID foi limitado. Novos avanços na ciência e na tecnologia têm permitido o uso em aplicações comerciais. As grandes instituições, como o Departamento de Defesa dos EUA, desde que implementaram o RFID permitiram que se espalhasse para outras organizações e indústrias [29]. A Walmart tem sido um dos líderes na adoção em larga escala da tecnologia RFID [29] [31].

Os problemas de segurança ainda prevalecem sobre a tecnologia RFID, existe o medo que as pessoas possam facilmente construir leitores de RFID com custos mais baixos e que possam ler dados de um chip RFID sem conhecimento e talvez até mesmo alterar dados. Por exemplo, alguém poderia usar o leitor RFID num produto barato e fazer o upload dos dados para um chip que está num produto caro, ficando assim o último por um preço menor. Outro exemplo é sobre a recuperarão dos dados RFID num dispositivo móvel habilitado.

Os padrões NFC - *Near Field Communication* (tipo de RFID) inovaram o processo de descoberta, passando endereços de controlo de acesso ao meio sem fio e chaves de criptografia de canal entre rádios através do canal de uma união de campo-próximo, que, quando limitado a 20 cm, permite que os utilizadores reforcem a sua própria segurança física usando troca de chaves de criptografia. O padrão NFC, que opera na banda dos 125 kHz) é compatível com a etiqueta RFID ISO 15693 que opera na banda de 13.56 MHz [26].

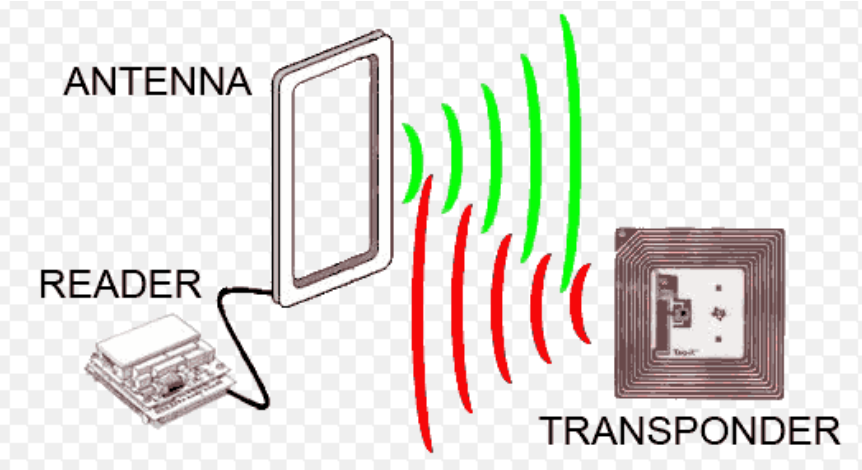


Figura 2.31: Exemplo RFID

As vantagens da tecnolgia RFID podem ser explicadas resumidamente como se segue:

- Um leitor pode ler e gravar dados em tags RFID sem contato direto e sem uma linha de visão livre.
- Os dados das várias tags RFID são acessados através de ondas de rádio.
- Sem custos de manutenção, o RFID pode trabalhar sob diferentes ambientes e pode ser utilizada de forma eficaz durante mais 10 anos.
- Lê e escreve rápido, alguns milissegundos.
- As etiquetas RFID modernas são feitas com boas capacidades que variam de 16 - 64 Kbytes, que é muitas vezes superior a um código de barras típico.
- Etiquetas RFID podem funcionar com GPRS e usadas para rastreamento.
- As etiquetas RFID também podem ser integradas com outras tecnologias. Por exemplo, é usada com redes de sensores sem fio para uma melhor conectividade.

2.3.4 Smart Cards

Um cartão de chip também conhecido como *smart card* é um cartão que geralmente assemelha-se em forma e tamanho a um cartão de crédito convencional de plástico com um chip dourado de

aproximadamente 1,27 cm embutido que possui capacidade de armazenamento de dados, Figura 2.32.

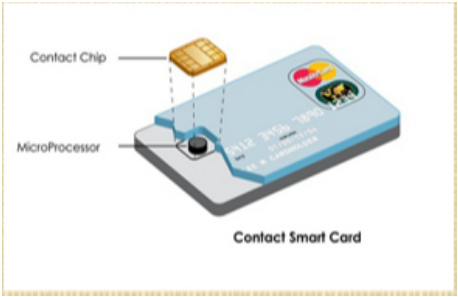


Figura 2.32: Exemplo de um *Smart Card* de contato

Os *smart cards* podem fornecer a documentação de identificação, autenticação, armazenamento de dados e processamento de aplicações [32]. Podem fornecer autenticações fortes para sistemas de segurança que utilizam single sign-on (SSO) [33] dentro de grandes organizações.

Os *smart cards* de contacto têm uma área de contacto de aproximadamente 1 centímetro quadrado, que possui várias almofadas de contacto banhadas a ouro. Estas almofadas proporcionam ligações eléctricas quando inserido num leitor, e que são utilizadas como um meio de comunicação entre o cartão e por exemplo um computador ou um dispositivo móvel. Estes cartões não possuem baterias, a alimentação é fornecida pelo leitor de cartão.

As normas ISO/IEC 7816 e ISO/IEC 7810 definem, para esta categoria de smart cards, os seguintes requisitos:

- O formato físico;
- A posição e o formato dos conectores elétricos;
- As características elétricas;
- Os protocolos de comunicação;
- O formato dos comandos enviados ao cartão e as respostas retornadas por ele;
- A robustez do cartão;
- A funcionalidade;

Um segundo tipo de cartão é o *smart card* sem contato, Figura 2.33. Este comunica e é alimentado pelo leitor através da tecnologia de indução RF (com taxas de dados de 106-848 kbit/s). Estes cartões exigem apenas uma certa proximidade com uma antena para conseguir comunicar. Como os *smart cards* com contactos, não possuem uma fonte de alimentação interna. Em vez disso, eles capturam uma parte do sinal de rádio-frequência incidente e usam-o para alimentar o cartão.

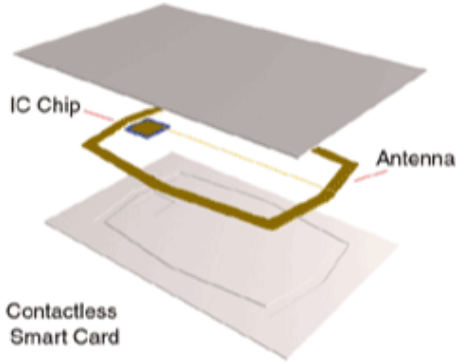


Figura 2.33: Exemplo de *Smart Card* sem contacto

O cartão sem contato mais utilizado hoje em dia é o MIFARE nas suas diversas especificações, baseado em vários níveis da ISO / IEC 14443 dependendo da sua finalidade, sendo os modelos existentes os seguintes [34]:

- Cartão MIFARE ISO 14443-A com chip EEPROM 1K de memória;
- Cartão MIFARE ISO 14443-A com chip EEPROM 4K de memória, usado em grandes operações;
- Cartão MIFARE DESFIRE ISO 14443-A com chip EEPROM 4K de memória, usado em grandes operações;
- Cartão MIFARE ISO 14443-A com chip Ultralight de 512 bites, usado para pequenas transações;

2.4 Bibliotecas de Suporte a processamento de imagens para Android

2.4.1 OpenCV

A Computer Vision Open Library (OpenCV) [35] é uma biblioteca originalmente criada no ano 1999, em laboratórios de pesquisa da Intel e foi escrito em C. Desde essa época, ele mudou de distribuidor várias vezes, tornou-se *open source* e chegou agora à terceira versão, sob a licença BSD e agora apoiado pela Willow Garage e Itseez [36]. É uma biblioteca de programação com funções de visão computacional em tempo real.

Esta biblioteca não está disponível apenas para os três principais sistemas operativos mais comuns (Windows®, Linux e Mac OS®), mas também para sistemas operacionais de dispositivos móveis como o Android® e iOS®. No presente o núcleo da biblioteca é escrito em C ++ com a interface pública em Python e um número de wrappers (invólucros) para outras linguagens de programação.

O OpenCV para Android® é composto por 10 módulos:

- **Android:** contém utilitários para a interação entre a plataforma Android® e estrutura do OpenCV. Possui as funções necessárias para a conversão entre o tipo de variável OpenCV Mat e Bitmap, o formato de imagem utilizado no Android®;
- **Camera Calibration and 3D Reconstruction:** contém as classes relacionadas com a câmara de calibração para corrigir os principais desvios do modelo *pinhole* simples que o uso de lentes impõe. Ele também compreende as funções relativas à reconstrução 3D;
- **Core:** abrange as funcionalidades fundamentais da estrutura do OpenCV, tais como as estruturas básicas, como o Mat e o tipo Point;
- **2D** features: cobre as classes e métodos relacionados com extratores descritor e correspondências e possui o detetor de ponto, incluindo algoritmos como SIFT ou SURF;
- **High-level GUI:** é o módulo que permite a interação entre OpenCV e o sistema operativo, o sistema de ficheiros e hardware como câmaras. Ele fornece uma interface para ler e gravar imagens de/para o disco ou memória e ler o vídeo de uma câmara ou gravá-lo em um ficheiro;
- **Image processing:** fornece os algoritmos fundamentais em visão computacional, desde filtragem de imagem, transformações de imagens geométricas, o cálculo e equalização do histograma, análise estrutural e descritores de forma e de deteção de características, como o algoritmo de extração de contornos de Canny. Também permite o cálculo de momentos de imagens;
- **Machine Learning:** a visão computacional e a aprendizagem de máquina são muitas vezes relacionadas, o OpenCV oferece um módulo de aprendizagem de máquina de propósito geral. Este módulo é focado no reconhecimento de padrões estatísticos e clustering;
- **Utilities:** fornece conversores para os diferentes tipos de variáveis e estruturas de dados;
- **Vídeo:** oferece um conjunto de algoritmos que proporciona métodos para a estimativa de movimento e controle de vídeo, por exemplo, por rastreamento de pontos-chave visualmente significativas. O OpenCV fornece dois métodos para a estimativa de movimento: o Lucas-Kanade e as técnicas de Horn-Schunck muitas vezes referido como fluxo óptico espaçado ou denso, respetivamente.

Para um sistema de reconhecimento biométrico, os módulos recomendados são: Core, Android, High-level GUI, Image Processing e Utilities.

2.4.2 JavaCV

JavaCV é um *wrapper* para a linguagem Java [37]. Assim, quando se executa um programa com JavaCV, na maioria dos casos vamos usar OpenCV também, basta chamá-la através de outra interface.

Mas a biblioteca JavaCV fornece mais do que apenas um invólucro em torno da OpenCV. Na realidade empacota várais bibliotecas de processamento de imagem, incluindo FFmpeg, OpenKinect entre outros (em C ++ pode-se igualmente ligar essas bibliotecas) e fornece classes de utilitários para tornar as funcionalidade mais fáceis de usar na plataforma Java, incluindo Android.

Além disso também vem com uma aceleração de hardware para exibição de imagem em fullscreen (CanvasFrame e GLCanvasFrame), métodos de fácil utilização para executar código em paralelo em múltiplos núcleos (Parallel), calibração geométrica e de cor *user-friendly* para câmeras e projetores (GeometricCalibrator, ProCamGeometricCalibrator , ProCamColorCalibrator), detecção e correspondência de pontos característicos (ObjectFinder), um conjunto de classes que implementam o alinhamento direto de imagem para sistemas de projetor-câmera (principalmente GNImageAligner, ProjectiveTransformer, ProjectiveColorTransformer, ProCamTransformer, e ReflectanceInitializer), um pacote de análise de blob (BLOBs), bem como funcionalidades diversas na classe JavaCV. Algumas destas classes também têm um OpenCL e OpenGL, os seus nomes terminam com CL ou começam com GL, como: JavaCVCL, GLCanvasFrame, etc.

2.4.3 Zxing

O ZXing ("*Zebra Crossing*"), é um *software open-source* [27] que implementa em Java uma biblioteca de processamento de código de barras em vários formatos 1D/2D, incluindo o QRcode, com funcionalidades para outras linguagens de programação [38]. O foco é sobre como usar a câmara embutida nos dispositivos para ler e descodificar códigos de barras no aparelho, sem ser necessário comunicar com um servidor. No entanto, o projeto pode ser usado para codificar e descodificar códigos de barras em dispositivos móveis, computadores desktops e servidores.

Esta biblioteca oferece suporte para sistema operativo Android®.

2.5 HTTP e conceito Restfull

Hypertext Transfer Protocol (HTTP) é o protocolo principal de comunicação utilizado na *web*. É usado todas as vezes que se transfere um documento ou se faz um pedido. Porém o HTTP é surpreendentemente desconhecido entre muito programadores web.

Existe um conjunto de princípios de design, conhecidos como REST [39], baseados em HTTP e permitem alcançar o seu poder máximo com a criação de interfaces, que podem ser utilizadas a partir de praticamente qualquer dispositivo ou sistema operacional.

2.5.1 Porquê um Serviço REST ("*REpresentational State Transfer*")?

REST é uma maneira simples de organizar as interações entre dois sistemas independentes. Tem-se tornado cada vez mais popular desde 2005 e serve de inspiração aos modelos de serviços como a API da textitTwitter.

Isto deve-se ao facto que um serviço REST permite interagir, com sobrecarga mínima, com clientes tão diversos como dipositivos móveis e outros websites. Teoricamente, REST não está

ligado à web, mas é quase sempre implementado como tal e foi inspirado por HTTP. Como resultado, REST pode ser utilizado sempre que o HTTP é invocado.

A alternativa é a construção de convenções relativamente complexas por cima do HTTP. O que muitas vezes se torna em linguagens baseadas em XML totalmente novas, o exemplo mais ilustre é o SOAP. Tem que se aprender um novo conjunto de convenções e nunca se usa o HTTP nas suas máxima funcionalidades. Como o REST foi inspirado em HTTP e foca-se nos pontos fortes, é a melhor maneira de aprender como funciona HTTP.

De seguida serão examinados cada um dos blocos de construção HTTP: URLs, métodos HTTP e códigos de resposta, todos usados de uma maneira RESTful.

2.5.2 HTTP

HTTP é o protocolo que permite o envio de documentos pela web. Um protocolo é um conjunto de regras que determina que mensagens podem ser trocadas e que mensagens são respostas adequadas para os outros. Outro protocolo comum é POP3, usado para obter o e-mail no disco rígido.

Em HTTP existem dois personagens diferentes: o servidor e o cliente. Em geral, o cliente inicia sempre a comunicação e o servidor responde. HTTP é baseado em texto, as mensagens são essencialmente bits de texto, embora o corpo da mensagem possa conter também outros meios de comunicação. Mas o uso de texto torna mais fácil a monotorização numa troca HTTP.

As mensagens HTTP são constituídas por um cabeçalho e um corpo. O corpo pode usualmente permanecer vazio, mas pode conter os dados que se querem transmitir através da rede e que serão usados de acordo com as instruções presentes no cabeçalho. O cabeçalho contém metadata, como codificação de informações, mas caso seja feito um pedido contém também métodos HTTP importantes. Num estilo REST descobre-se facilmente que o cabeçalho normalmente possui dados mais importantes que o corpo.

2.5.3 URLS

URLs é a forma de identificar os endereços a que se quer aceder. Cada URL identifica um recurso, e é exatamente assim que os URL são atribuídos ás paginas web, de fato uma página web é um tipo de recurso.

Não se deve usar um URL para descrever uma ação, por exemplo adicionar alguma coisa.

Este é um ponto bastante fundamental na distinção de sistemas RESTful e não-RESTful. Isso é feito através dos métodos HTTP.

Os URLs devem ser muito precisos de acordo com a necessidade, tudo o que seja necessário para identificar exclusivamente um recurso deve ser explícito no URL. Não se deve precisar de incluir dados de identificação do recurso no pedido. Assim os URLs funcionam como um mapa completo de todos os dados que uma aplicação manipula.

2.5.4 Métodos HTTP

Cada pedido especifica um determinado método HTTP no cabeçalho do pedido. Esta é a primeira palavra com letras maiúsculas no cabeçalho do pedido. Por exemplo,

GET / HTTP / 1.1 significa que o método GET é usado, enquanto

DELETE/ clientes / ana HTTP / 1.1 significa que o método delete é usado.

O pedido pode conter opcionalmente informação adicional no corpo, necessária para executar a operação, por exemplo, alguns dados que se desejam armazenar.

Normalmente está-se familiarizado com dois dos mais importantes métodos HTTP: GET e POST. Mas há muito mais métodos HTTP disponíveis. Os mais importantes para a construção de uma API RESTful são: GET, POST, PUT, PATCH e DELETE (a fonte oficial é IETF [40])

2.5.4.1 GET

Trata-se do método HTTP mais simples, usado pelo browser sempre que se carrega num link ou se acede a um URL através da barra de endereços. Dá instruções ao servidor para transmitir os dados identificados pelo URL para o cliente. Um pedido GET é apenas de leitura, mas uma vez que o cliente recebe os dados, é livre de fazer operações no seu lado. Os dados nunca devem ser modificados no lado do servidor como resultado de um pedido GET.

2.5.4.2 POST

Este método é usado quando se deseja enviar dados para o servidor, o processamento que acontece no lado do servidor é repetido, sempre que se repete o pedido POST (trata-se de um método não idempotente, descrição abaixo). Um pedido POST deve processar o corpo do pedido como um subordinado do URL.

Em palavras simples:

POST / clientes /

não deve fazer com que o recurso clientes , em si seja modificado, mas sim um recurso cujo URL começa com /clientes/. Por exemplo, pode acrescentar um novo cliente para a lista, com um ID gerado pelo servidor.

/ clientes / some-unique-id

Muitas vezes os pedidos POST são usados para desencadear as operações no servidor que não se encaixam no paradigma Create/Update/Delete, mas isto no entanto, está fora do âmbito REST.

2.5.4.3 PUT

Este método é usado quando se deseja adicionar ou atualizar um recurso identificado pelo URL, por exemplo:

PUT /clients/joao

pode criar um cliente chamado João no servidor. REST é agnóstico em relação ao back-end. Não há nada no pedido que informe o servidor sobre como os dados devem ser criados, apenas que

devem. Isso permite trocar facilmente a tecnologia back-end em caso de necessidade. Os pedidos PUT contém no corpo do pedido os dados para atualizar ou criar o recurso.

2.5.4.4 DELETE

Este método realiza o contrário do PUT, deve ser usado quando se pretende apagar um recurso identificado pelo URL.

DELETE /clients/joao

vai apagar todos os dados associados ao recurso.

2.5.5 Classificação dos Métodos HTTP

2.5.5.1 Métodos seguros e inseguros

Métodos seguros são aqueles que nunca modificam recursos. O único método seguro dos listados é o GET. Os outros não são seguros, uma vez que podem resultar em modificações dos recursos.

2.5.5.2 Métodos idempotentes

Estes métodos permitem alcançar o mesmo resultado, não importa quantas vezes se repetem. O único método não idempotente é o POST.

Destacam-se os seguintes: GET, PUT e DELETE. PUT e DELETE serem considerados idempotentes pode parecer surpreendente, mas na verdade é bastante fácil de explicar: repetindo um método PUT com exactamente o mesmo corpo, deve modificar um recurso da mesma forma que o pedido PUT anterior, ou seja nada vai mudar. Da mesma forma, não faz sentido excluir um recurso duas vezes. Não importa quantas vezes um pedido PUT ou DELETE é repetido, o resultado deve ser o mesmo que se tivesse sido feito apenas uma vez.

2.5.6 Representações

O pedido e a resposta contém uma representação do recurso, por representação entenda-se informação num certo formato sobre o estado do recurso ou como esse estado deve ser no futuro. Tanto o corpo como o cabeçalho são partes da representação.

Os cabeçalhos HTTP, que contêm metadata, são rigorosamente definidos pelas especificações HTTP, podem apenas conter texto simples e devem ser formatados de uma certa maneira.

O corpo pode conter dados em todo o tipo de formatos, e é neste aspeto onde todo o poder do HTTP é demonstrado. Sabe-se que se pode enviar texto simples, fotografias, HTML e XML.

Através da metadata do pedido ou por diferentes URLs, pode-se escolher entre diferentes representações para o mesmo recurso. Por exemplo pode-se mandar uma página web para um browser e objetos JSON para uma aplicação.

A resposta HTTP deve especificar o formato do corpo, isto é feito no cabeçalho no campo Content-Type, por exemplo:

Content/Type: application/json

2.5.7 Códigos de Resposta

Os cabeçalhos devem ser a primeira coisa na resposta, não se deve imprimir mais nada enquanto houver cabeçalhos para se processar. Às vezes, o servidor HTTP pode ser configurado para adicionar outros cabeçalhos, além daqueles especificados no código.

Os cabeçalhos contêm todo o tipo de informação meta, por exemplo, a codificação do texto usado na mensagem do corpo ou o tipo de MIME do conteúdo do corpo. Os códigos de resposta HTTP uniformizam uma maneira de informar o cliente sobre os resultados do pedido. Por defeito o servidor retorna um código de resposta 200, que significa que a resposta ao pedido é bem sucedida.

O servidor deve retornar o código de resposta mais apropriado, desta maneira o cliente pode tentar reparar os seus erros, assumindo que existem. Normalmente a maior parte das pessoas está familiarizado com o código de resposta "404 Not Found", mas existem muitos mais para corresponder a uma larga quantidade de situações.

O código de resposta não é muito preciso, isto resulta como consequência do próprio HTTP ser bastante genérico. Deve-se usar o código de resposta que mais se assemelha à situação em mão.

Aqui estão alguns códigos de resposta HTTP, que são muitas vezes utilizados com REST:

- **200 OK** - Indica que o pedido foi bem sucedido.
- **201 Created** - Indica que o pedido foi bem sucedido e o recurso criado. É usado para confirmar o sucesso num pedido PUT ou POST.
- **400 Bad Request** - Indica que o pedido foi mal formado. Acontece essencialmente com os pedidos PUT e POST quando os dados não passam na validação, ou estão no formato errado.
- **404 Not Found** - Indica que o recurso pedido não pode ser encontrado. É retornado este erro sempre que um pedido aponta para um URL sem recurso correspondente.
- **401 Unauthorized** - Indica que é necessário realizar uma autenticação antes de aceder ao recurso.
- **405 Method Not Allowed** - Indica que o método usado não é suportado pelo recurso.
- **409 Conflict** - Indica um conflito, por exemplo, usar um pedido PUT para criar o mesmo recurso duas vezes.
- **500 Internal Server Error** - Quando todos os outros falham, este código é usado normalmente quando o processamento falha devido a circunstâncias inesperadas no lado do servidor.

2.6 Bibliotecas REST

2.6.1 A maneira antiga: AsyncTasks

Uma rápida pesquisa mostra rapidamente os vários problemas com esta abordagem, entre eles: nenhum apoio em caso de mudanças na orientação, sem capacidade para cancelar chamadas à rede, bem como nenhuma maneira intuitiva de fazer chamadas API em paralelo. Com a exceção das versões Android Froyo e Gingerbread, as AsyncTasks (por padrão) executam de uma forma serializada. Por outras palavras, isto significa que apenas uma chamada AsyncTask está a correr em determinado período de tempo. Aplicações que requiram várias chamadas API (por exemplo o DashBoard precisa de pelo menos 7 [41]) correm de forma extremamente lenta, demorando por vezes vários segundos a carregar.

2.6.2 Volley e Retrofit

Existem atualmente algumas bibliotecas de suporte que fornecem suporte para várias threads simultâneas em segundo plano, network cashing, bem como outras funcionalidades que permitem limpar o código.

Destacando-se duas bibliotecas:

- **Volley**, uma biblioteca *open source* escrita pela Google [42]. Actualmente é usada em AOSP (Android Open Source Project) Android bem como na grande parte das aplicações desenvolvidas pela Google. Foi introduzida na Google IO 2013, ela oferece grandes características como: solicitações síncronas, solicitações assíncronas, priorização, várias solicitações simultâneas, ordenazão de pedido e cache. Mas um dos principais problemas enfrentados por desenvolvedores que usam esta biblioteca é que ela não possui documentação oficial detalhada.

- **Retrofit**, também uma biblioteca *open source* escrita pela Square, Inc [43]. Retrofit é um cliente REST para Android, através do qual você pode construír interfaces fáceis de usar que permitem tornar qualquer aplicação Android mais poderosa. A grande vantagem e diferença, é que com pode-se executar solicitações assíncronas e sincronas com análise automática do JSON sem qualquer esforço. Este recurso sozinho é uma razão suficiente para o tornar num forte candidato.

À primeira vista elas são em tudo semelhantes em termos de usabilidade. Ambas permitem "Callback", uma interface com dois métodos que se devem substituir: sucesso e falha. Um dos métodos deve ser chamado na Thread principal após a realização da chamada assíncrona à rede. A grande diferença está na forma de especificar o endpoint da API e no que é recebido como resposta.

Com a biblioteca Volley, especificamos a totalidade do *endpoint* dinamicamente (parâmetros e o resto) no momento de realizar a chamada API. O retorno depende do tipo de solicitação, um JSONObject ou um JSONArray, definido por padrão.

Por outro lado, na biblioteca RetroFit, especificamos um URL base para todas as chamadas à API, construindo depois interfaces estáticas para especificar o endpoint através de anotações em Java. Podemos de forma limpa e dinâmica substituir alguns segmentos, variáveis POST/GET, entre outros no endpoint na altura de realizar a chamada API. Para se realizar uma chamada API com a RetroFit, chama-se o método na interface, onde passamos todos os parâmetros, e recebemos como resposta um objeto padrão em java. Por defeito, RetroFit parcela o JSON automaticamente usando uma biblioteca de apoio GSON, que é muito rápida, mas caso se queira podemos ligar outro JSON parser.

Mesmo que a configuração seja um pouco diferente, as chamadas à API são feitas de uma forma similar.

Vamos apontar as principais diferenças:

2.6.2.1 Tipos de Solicitação incorporados

Os dados devolvidos por um serviço web complicam sempre a implementação, mas com o apoio destas bibliotecas quase todos os tipos de respostas podem ser facilmente capturados e analisados.

Com o Volley podemos capturar automaticamente 4 tipos de pedidos:

- **StringRequest** - A resposta é convertida e guardada como uma String.
- **JsonObjectRequest** - Converte a resposta num JSONObject.
- **JsonArrayRequest** - Converte a resposta num JSONArray.
- **ImageRequest** - Converte automaticamente a resposta num bitmap descodificado.

Por outro lado Retrofit pode analisar muitos outros tipos de respostas automaticamente como:

- **Boolean** - A resposta Web API tem de ser uma String booleana.
- **Integer** - A resposta Web API tem de ser um Inteiro.
- **Date** - A resposta Web API deve ser do tipo Long format date.
- **String** - A resposta Web API tem de ser no formato String.
- **Object** - A resposta Web API tem de ser um Json object.
- **Collections** - A resposta Web API tem de ser no formato String.

Volley possui análise de imagens ao contrário do Retrofit, mas não pode converter um objeto JSON diretamente em um POJO, como o RetroFit.

2.6.2.2 Mecanismo de repetição e Cache

Só o Volley possui estes dois mecanismos já incorporados, podendo ser incorporados facilmente através de terceiros no RetroFit.

Quanto ao mecanismo que suporta uma repetição de um pedido em resposta a um limite de tempo., podemos definir uma política de repetição usando o método setRetryPolicy, por defeito o limite de tempo está definido em 5 segundos, mas pode ser alterado. Pode-se especificar os seguintes parâmetros para melhor satisfazer as necessidades:

- Timeout
- Número de repetições
- Back Off Multiplier

Quando um pedido é feito através de volley é primeiro verificado no cache. Se uma resposta apropriada está já presente na cache, ela é analisada e enviada diretamente para thread principal. Toda a funcionalidade pode ser personalizada, para satisfazer as necessidades.

2.6.2.3 Complexidade do Código

Não há muito para personalizar no Retrofit, é uma biblioteca simples mas poderosa. No Volley, por outro lado, é altamente personalizável mas tem uma maior complexidade de código.

2.7 Sumário

Perante todos os métodos de autênticação descritos nesta secção decidiu-se optar pelo reconhecimento facial e utilização de métodos complementares tais como: o uso de QRCodes e de tags NFC. Alguns dos métodos descritos seriam dispendiosos em termos de hardware motivo pelo qual não foram considerados. Os métodos de reconhecimento facial convencionais requerem para além de grande poder computacional, uma base de dados de imagens da face o que, fato que pode trazer preocupações de segurança, nomeadamete na garantia de confidencialidade dos utilizadores. Sendo desta forma necessário investigar uma forma de implementar o reconhecimento facial sem recurso a imagens armazenadas.

Capítulo 3

Metodologia

A realização do projeto de investigação dividiu-se em três passos. Em primeiro lugar, a identificação dos problemas, em segundo lugar a selecção das soluções possíveis para cada problema e em terceiro lugar a implemetação das soluções adotadas e os motivos que levaram à escolha dessas soluções.

Como o tema se encontra diretamente ligado a tecnologias computacionais, toda a pesquisa foi realizada sobre recursos eletrónicos. O foco da investigação baseou-se sobretudo nas tecnologias a utilizar, em tutoriais sobre as regras dessas tecnologias e das bibliotecas auxiliares escolhidas. A escolha da ferramentas e tutoriais foi realizada através de fontes seguras e fidedignas de forma a obter informação relevante para o trabalho desenvolvido.

De forma resumida é possível definir três problemas centrais, sendo que o segundo ponto apresenta a maior grau de dificuldade:

- **1º** Escolha das tecnologias e linguagens de programação para o trabalho.
- **2º** Escolha dos métodos de identificação facial e de outros métodos de identificação.
- **3º** Envio da informação da autenticação por Web Service.

O passo mais importante foi a implementação dos aplicativos, quando os objetivos são bem definidos e as ferramentas completamente dominadas. Este passo é longo devido ao processo complexo de debugging.

Para a realização deste projeto usaram-se os seguintes equipamentos, Tabela 3.1 e aplicações, Tabela 3.2 :

Tabela 3.1: Equipamento utilizado

Quantidade	**Tipo**	**Modelo**
1	Computador Portátil	Toshiba Satellite A500-148 16,0"HD 4GB DDR2-800 320GB Intel® Core™2 Duo P8700 2,53GHz
1	Tablet	Asus Google Nexus 7 Wi-Fi - 32GB HDD 1GB RAM NVIDIA® Tegra® 3 quad-core 1.2 GHz
1	Mobile Phone	Motorola Moto G (2nd Gen.) (5", 8GB) 1GB RAM Quad-core 1.2 GHz Cortex-A7

Tabela 3.2: Software Utilizado

Software	**Tipo**	**Versão**
Android®	Sistema Operativo	>4.2 Jelly Bean
Windows	Sistema Operativo	>8
Android Studio	Ambiente Desenvolvimento	>1.0
OpenCV	Biblioteca	>2.4.8
JavaCV	Biblioteca	>0.8
ZXing	Biblioteca	2.2
RetroFit	Biblioteca	1.9

3.0.1 Arquitetura da Solução

O desenvolvimento da dissertação foi dividido na implementação das seguintes aplicações:

- Reconhecimento Facial, Secção 3.1;
- Reconhecimento por QRCode, Secção 3.2;
- Reconhecimento por NFC, Secção 3.3;
- Webservice, Secção 3.4.

Para cada uma das funcionalidades será apresentada uma base teórica e a especificação de requisitos (de acordo com a norma a IEEE Std 830-1998, "IEEE Recommended Practice for Software Requirements Specifications") tendo em conta o registo da forma de identificação e o futuro reconhecimento, no âmbito deste relatório denominado por registo de entrada/saída.

A arquitetura da solução apresentada foi desenvolvida e assente de acordo com as seguintes caracterísctcas:

- Duas aplicações por método, uma para registo de utilizadores e outra para registar entradas/saídas;
- Uma base de dados local em XML por método de identificação para o registo;
- Uma base de dados local em XML com as entradas/saídas;
- Integração dos métodos com o webservice para registo das entradas/saídas no servidor.

3.0.2 Interface gráfica

Desenvolveram-se 3 interfaces gráficas para contacto com o utilizador:: Normal 3.1, Registo entradas/saídas 3.2 e Registo 3.3. O ecrã para captura da imagem em *Live Feed* tem as dimensões de 683x645 pixéis.

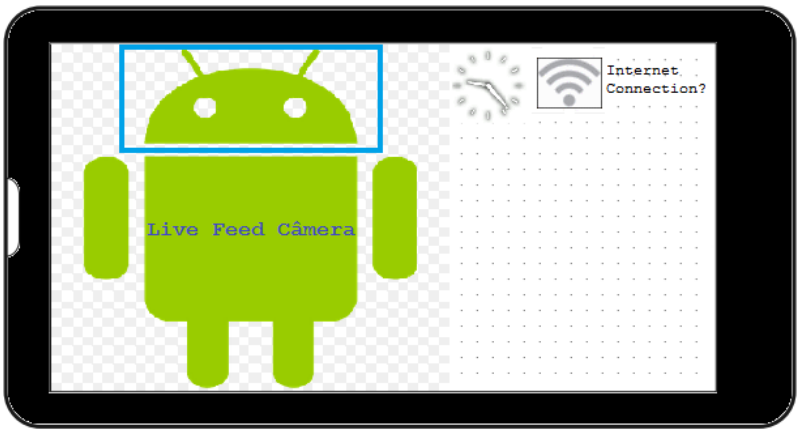


Figura 3.1: Interface Normal para o utilizador da solução

Figura 3.2: Interface após Registo entrada/saída para o utilizador da solução

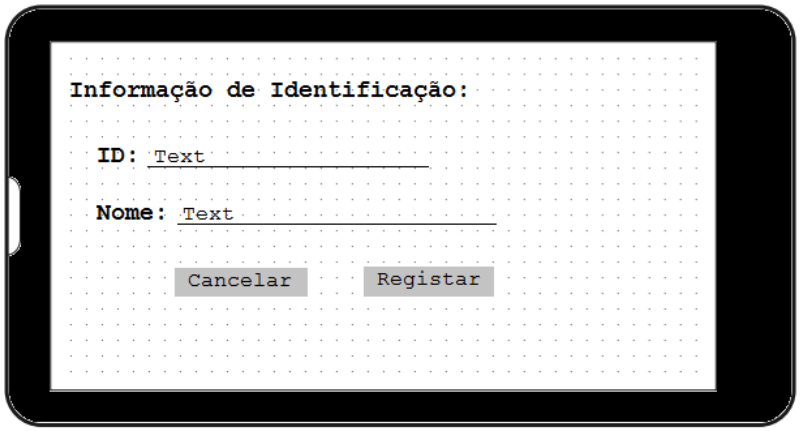


Figura 3.3: Interface de Registo para o utilizador da solução

Apesar de uma das aplicações não precisar da Câmara mas sim do Leitor de NFC, o layout é identico nas três, pois pretende-se integrar todas as aplicações numa só.

3.0.3 Funcionamento da Aplicação

Na Figura 3.4 é apresentado o fluxo de dados do registo para os três métodos. Todos os métodos vão seguir o mesmo fluxo, estando as diferenças assinaladas no diagrama. Sendo identificado um dos três métodos, o seu conteúdo é lido e analisado, seguidamente são pedidos ao utilizador a identificação e nome do individuo, sendo estes registados na classe correspondente e armazenados numa base de dados dinâmica .XML de acordo com o esquema definido pelo fluxo seguinte.

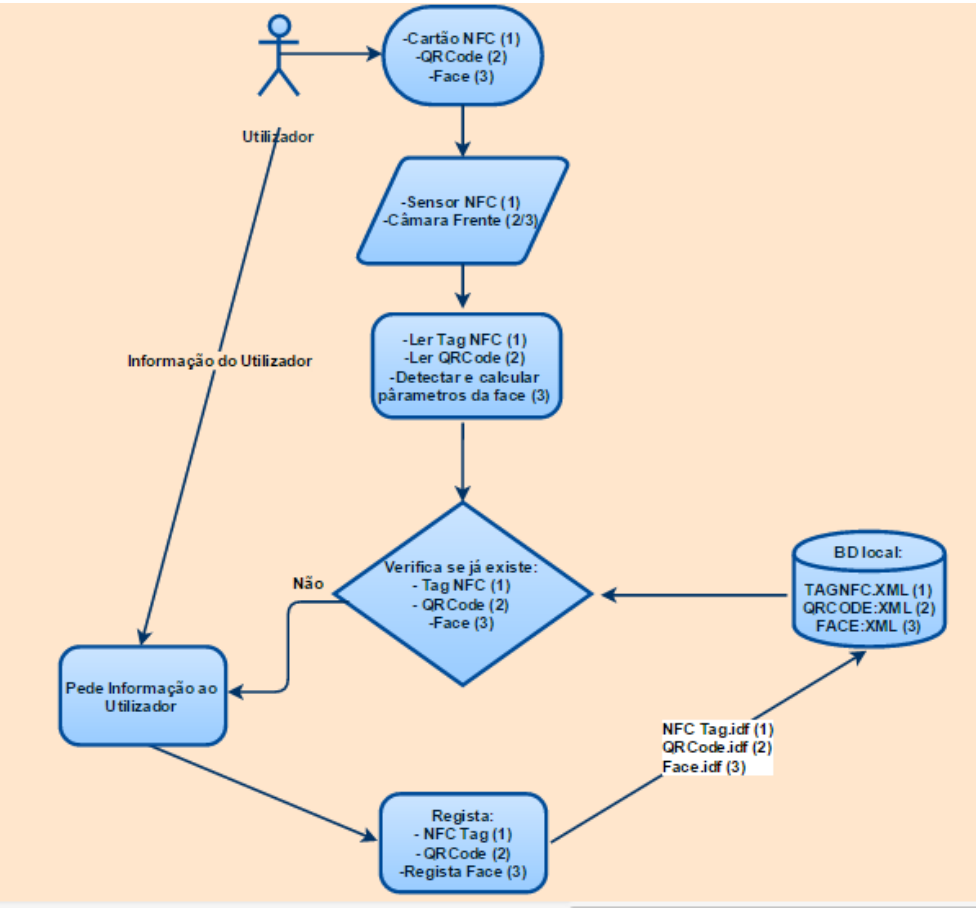


Figura 3.4: Diagrama de fluxo de dados para o registo

O processo de reconhecimento (entradas/saídas), é definido pelo diagrama de fluxo de dados representado na 3.5, sendo semelhante ao do registo, o fator de decisão de reconhecimento depende do método (String QRCode, parâmetros da face, tag NFC). O resultado da operação é armazenado na base de dados dinâmica picagem.XML definido pelo fluxo:

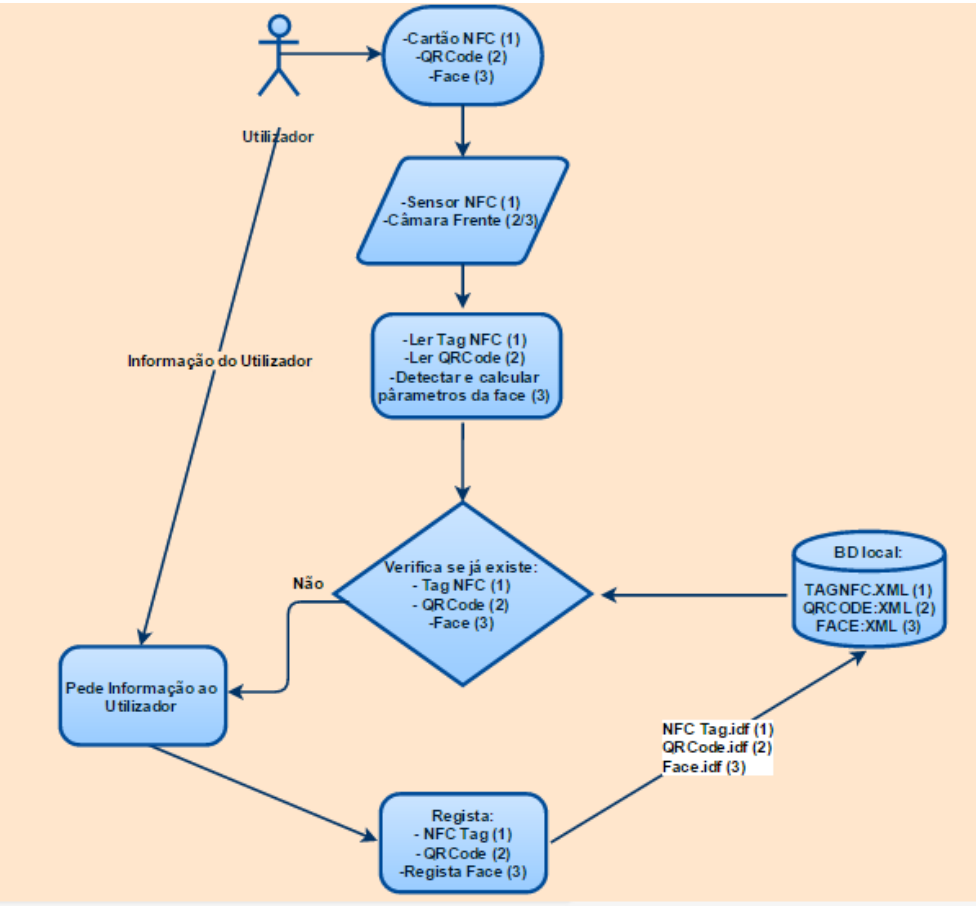


Figura 3.5: Diagrama de fluxo de dados do reconhecimento (entradas/saídas)

Os modelos entidades-relação bem como detalhes mais especifícos de cada método, vao ser discutidos nas secções seguintes referentes aos métodos de identificação.

3.1 Reconhecimento Facial

Como referido na secção 3.0.2, começou-se com o desenvolvimento do sistema de reconhecimento facial, tendo sido objetos de estudo, quatro métodos de reconhecimento facial em Android® usando as funcionalidades do OpenCV através do JavaCV.

- Reconhecimento Baseado na Localização de Caraterísticas da Face (RBLCF)
- Análise de Componentes Principais (ACP)
- Análise Discriminante Linear (ADL)
- Padrão Binário Local do Histograma (PBLH)

Sendo que as três últimas soluções já são amplamente estudadas (PCA, LDA e PBLH), existindo vários trabalhos e dissertações sobre esses métodos. Fez-se um estudo aprofundado de cada um desses métodos e adaptaram-se à nossa interface e requisitos pretendidos, porém todos necessitam de imagens guardadas na base de dados para efetuar o reconhecimento, sendo que vão ser usados apenas para comparação de performance com o método RBLCF.

Esta secção vai ser organizada da seguinte forma:

- Criação da aplicação para a detecção das Faces;
- Método de detecção das Faces - Viola-Jones;
- Reconhecimento Baseado na Localização de Características da Face;
- Métodos de Comparação;

3.1.1 Criação da aplicação para a detecção das Faces

No início foi usado como base o código de uma aplicação básica fornecida gratuitamente para uso livre pela *bytedeco* [37], que consiste numa fusão entre a função *facedetect* da *OpenCV* e as funções *CameraPreview* da *Android*, usando *JavaCV+JavaCpp* para realizar a interface entre elas.

Apresenta-se de seguida o código com as alterações afectuadas no código inicial e muitas funcionalidades implementadas.

Preview é uma classe que permite exibir no ecrã o que a câmera está a visualizar. Estende uma *SurfaceView*, que é a ferramenta básica que permite exibir as coisas no ecrã do dispositivo. Também implementa um *SurfaceHolder.Callback*. Isso permite que a classe receba todas as informações sobre as mudanças da superfície (rotação do dispositivo, final da aplicação, etc ..). A classe Preview é apresentada parcialmente de seguida:

```
public class Preview extends SurfaceView implements SurfaceHolder.Callback {
    SurfaceHolder mHolder;
    //the instance of this class permit us to change the parameters of the SurfaceView .
    Camera mCamera;
    //our camera
    Camera.PreviewCallback previewCallback;

    Preview(Context context, Camera.PreviewCallback previewCallback) {
        super(context);
        this.previewCallback = previewCallback;

        // Install a SurfaceHolder.Callback so we get notified when the
        // underlying surface is created and destroyed.
        mHolder = getHolder();
        mHolder.addCallback(this);
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    public void surfaceCreated(SurfaceHolder holder) {
        // The Surface has been created, acquire the camera and tell it where
        // to draw.
        mCamera = openFrontFacingCamera();

        try {
            mCamera.setPreviewDisplay(holder);
        } catch (IOException exception) {
            mCamera.release();
            mCamera = null;
            // TODO: add more exception handling logic here
        }
    }
```

Figura 3.6: Classe Preview

O método *surfaceCreated()* é automaticamente chamado quando a classe *Preview* é criada. A função *openFrontFacingCamera()* foi definida para retornar a câmara da frente, que é um dos requisitos da interface do programa proposto. Existem mais funções na classe *Preview* de menor relevo, mas que podem ser consultadas nos anexos.

```
private Camera openFrontFacingCamera() {
    int cameraCount = 0;
    Camera cam = null;
    Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
    cameraCount = Camera.getNumberOfCameras();
    for (int camIdx = 0; camIdx < cameraCount; camIdx++) {
        Camera.getCameraInfo(camIdx, cameraInfo);
        if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
            try {
                cam = Camera.open(camIdx);
            } catch (RuntimeException e) {
            }
        }
    }

    return cam;
}
```

Figura 3.7: Função para a Câmara frontal

É importante saber como esta classe está ligada com a classe principal *FacePreview*.

A classe *FacePreview* inicializa tanto a classe *Preview* como a classe *FaceView* (onde todo o processamento é feito) e é a responsável pela construção do Layout.

```
public class FacePreview extends Activity {

    TextView tvIsConnected;
    FaceView faceView;
    Preview mPreview;
    ImageView im;


    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);
        im= (ImageView) findViewById(R.id.photo);
        tvIsConnected = (TextView) findViewById(R.id.tvIsConnected);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        // Hide the window title.

        try {
            faceView = new FaceView(this, im);
            mPreview = new Preview(this, faceView);
            FrameLayout frame = (FrameLayout) findViewById(R.id.preview);
            frame.addView(mPreview);
            frame.addView(faceView);

        } catch (IOException e) {
            e.printStackTrace();
            new AlertDialog.Builder(this).setMessage(e.getMessage()).create().show();
        }

    }
```

Figura 3.8: Classe *Preview*

Sabendo como a aplicação é criada e a câmara iniciada, passa-se ao processamento que se faz ao longo da classe *FaceView*. Como dito anteriormente, a detecção da face é feita por *HaarCascades*, a aquisição da imagem é feita em escala de cinza, pois a *HaarCascade* foi assim preparada, neste caso: "haarcascade_frontalface_alt.xml".

Anteriormente definiu-se usar a câmara da frente, que vem por omissão com um efeito espelho integrado no software da câmera, sendo por isso necessário contrariar esse efeito espelho na imagem que se vai trabalhar. Começa-se por extrair a imagem da câmera e convertendo-a numa *IplImage* de escala cinza com o mesmo tamanho da original.

```
protected void processImage(byte[] data, int width, int height) {
    // First, downsample our image and convert it into a grayscale IplImage
    // We need a grayscale image in order to do the recognition, so we
    // create a new image of the same size as the original one.
    int f = SUBSAMPLING_FACTOR;
    if (grayImage == null || grayImage.width() != width/f || grayImage.height() != height/f) {
        grayImage = opencv_core.IplImage.create(width / f, height / f, IPL_DEPTH_8U, 1);

    }
    int imageWidth  = grayImage.width();
    int imageHeight = grayImage.height();
    int dataStride = f*width;
    int imageStride = grayImage.widthStep();
    ByteBuffer imageBuffer = grayImage.getByteBuffer();
    for (int y = 0; y < imageHeight; y++) {
        int dataLine = y*dataStride;
        int imageLine = y*imageStride;
        for (int x = 0; x < imageWidth; x++) {
            imageBuffer.put(imageLine + x, data[dataLine + f*x]);
        }
    }

    cvClearMemStorage(storage);

   cvFlip(grayImage, grayImage, 1); // CAMARA DA FRENTE VEM COM ESPELHO INTEGRADO

    // We detect the faces.
    faces = cvHaarDetectObjects(grayImage, classifier, storage, 1.1, 3, CV_HAAR_DO_CANNY_PRUNING);
    postInvalidate();

}
```

Figura 3.9: Converter *Frame* para escala cinza para reconhecimento

3.1.2 Método de deteção das Faces - Viola-Jones

A existência de elementos de fundo numa imagem produz um impacto significativo nos resultados obtidos por sistemas de reconhecimento facial automático, pelo que a deteção da zona de uma imagem onde se encontra representada a face a identificar é fundamental para um bom desempenho do sistema criado.

A deteção facial é efetuada com recurso a um classificador em cascata, designado de *boosted cascade classifier*, treinado especificamente para a deteção de faces.

É um método eficaz de deteção de objetos proposto por Paul Viola e Michael Jones no artigo, "Rapid Object Detection using a Boosted Cascade of Simple Features"em 2001 [44]. É uma abordagem baseada em *machine learning* onde uma função cascata é treinada a partir de muitas imagens positivas (imagens de faces) e negativas (imagens sem faces). Em seguida, é utilizada para detectar objetos em outras imagens.

O método de Viola-Jones é utilizado como a base de detecção de faces. Inicialmente, o algoritmo é alimentado com imagens positivas e imagens negativas para treinar o classificador. Posteriormente são retiradas as características resultantes das imagens de treino. Para isto, as carateristicas *haar* mostrados na Figura 3.10 são usadas. Que são semelhantes ao *kernel* convolutional. Cada recurso é um valor único obtido subtraindo a soma de pixels sob retângulo branco da soma de pixels sob retângulo preto.

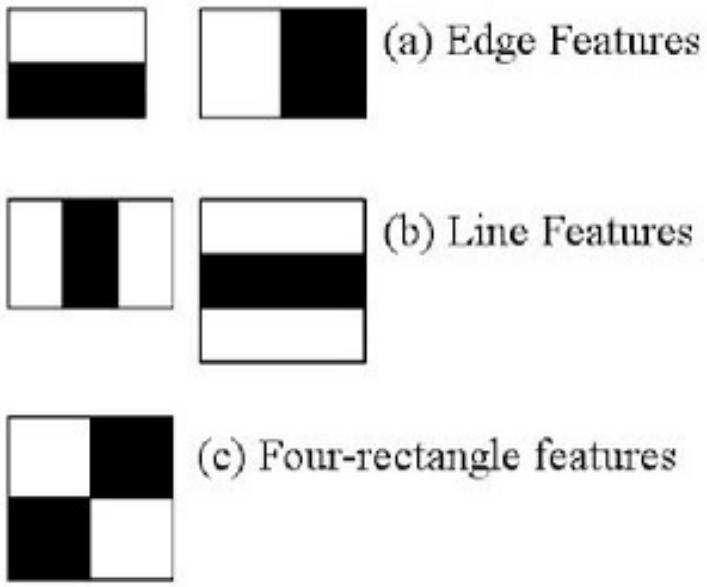


Figura 3.10: Caracteristicas haar

Todos os tamanhos e locais possíveis de cada núcleo são usado para calcular as muitas caracteristicas (mesmo uma janela 24x24 resulta em 160.000 características). Para o cálculo de cada caracteristica, precisamos encontrar a soma de pixels sob retângulos brancos e pretos.

A solução abordada implementa imagens integrais que simplificam o cálculo da soma de pixels independente do número de pixels, para uma operação que envolve apenas quatro pixels.

A maior parte das caracteristicas calculadas são irrelevantes, considerando a Figura 3.11, A linha de cima mostra duas boas caracteristicas. A primeira característica selecionada selecionada centrar-se na propriedade que a região dos olhos é muitas vezes mais escura do que a região do nariz e bochechas. A segunda característica selecionada baseia-se na propriedade que os olhos são mais escuros do que a ponta do nariz. Porém aplicando a mesma janela nas bochechas ou em qualquer outra parte é irrelevante, usando-se o *Adaboost* para selecionar as melhores caracteristicas.

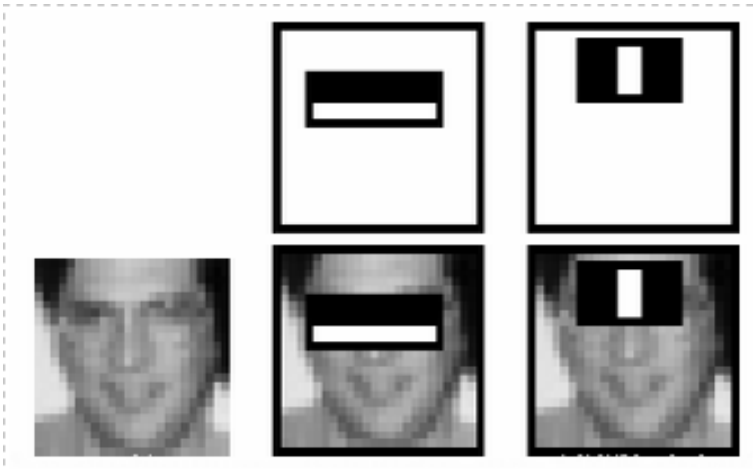


Figura 3.11: Características Faciais

Aplicando-se todas as características em todas as imagens de treino. Para cada característca, ele encontra o melhor *treshold* que irá classificar as faces de positivas ou negativas. Isto acarreta erros de classificação, seleccionando-se os recursos que apresentam uma taxa mínima de erro. A todas as imagens é dado o mesmo peso de início. Depois de cada classificação, os pesos são aumentados nas imagens erroneamente classificadas . Repete-se o processo, novas taxas de erro são calculadas, também novos pesos. O processo continua até se chegar à precisão necessária, ou a taxa de erro é alcançada ou são encontradas as cacteristicas requiridas.

O classificador final é uma soma ponderada desses classificadores fracos(sozinhos não podem classificar uma imagem, mas em conjunto constituiem um classificador forte. No artigo [44] refere que mesmo 200 caracterisitcas permitem uma detecção com precisão de 95%, mas na sua configuração final tinham cerca de 6000 recursos. (Redução de 160000+ características para 6000 características). Porém aplicar numa janela de 24*24 cerca de 6000 caracteristicas, continua a consumir muito tempo e recursos.

A solução reside no facto de numa imagem a maior região de imagem não faz parte da face. Com um método simples, verifica se uma janela é ou não uma região da face. Se não for, descarta-o imediatamente, não se processando novamente. Em vez disso, tenta focar uma região onde pode haver uma face. Desta forma, podemos encontrar mais tempo para verificar uma possível região com face.

Introduziu-se então o conceito de *Cascade of Classifiers*, que consiste em formar grupos classificadores de características em diferentes fases e aplicar um por um (normalmente as primeiras fases contêm um número muito menor de características). Se uma janela falhar na primeira fase, é descartada imediatamente, não se aplicando as restantes fases naquela região. Se passar, aplicar a segunda fase de características e continuar o processo. A janela que passa todas as fases é uma região de face. O detector dos autores do artigo [44] tinham 6000+ características em 38 etapas (com 1, 10, 25, 25 e 50 características nas cinco primeiras etapas).

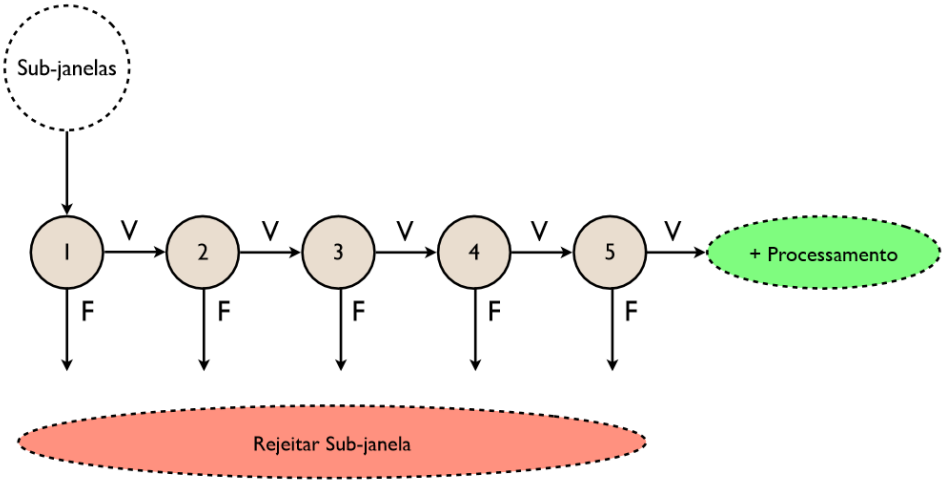


Figura 3.12: Cascade of Classifiers

3.1.3 Reconhecimento Baseado na Localização de Características da Face

Este método pretende localizar numa face já identificada as suas características (localização dos olhos, nariz e boca). Assim que a face é identificada, é passível de ser redimensionada ou normalizada a um tamanho fixo de 160*160 pixéis. Será nessa face de tamanho normalizado que são identificadas as características definidas anteriormente. A interface encontrasse modificada de forma a avaliar de forma mais clara o processo de caracterização, Figura 3.13

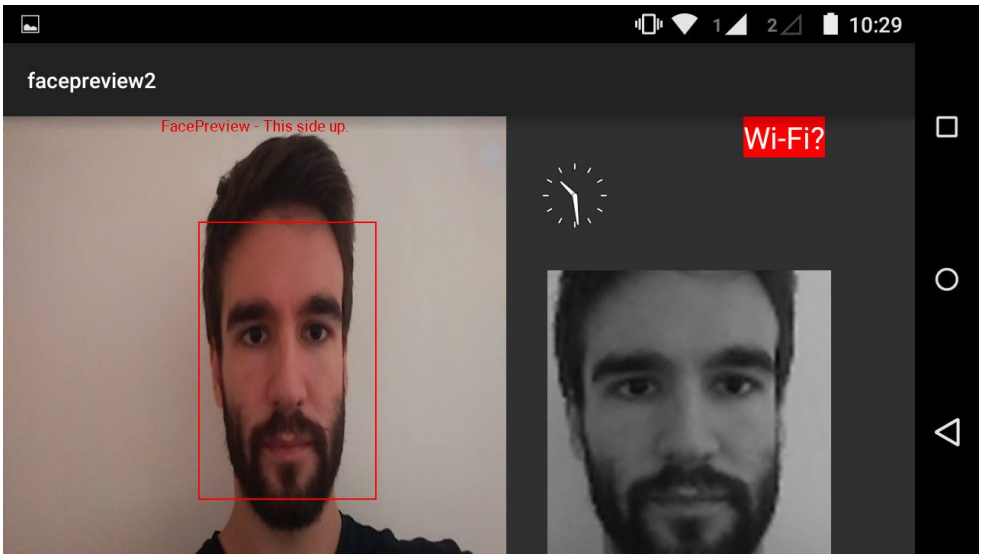


Figura 3.13: Interface RBLCF

A fase de detecção de uma face fica assim concluída, temos neste momento uma imagem em escala cinza que contém apenas a face, fazemos uma cópia dessa imagem uma vez que vamos trabalhar dois métodos que vão efetuar alterações nas imagens. Ilustraremos primeiro a detecção das características da face (localização dos olhos, nariz e boca) e de seguida o método da métrica dos contornos.

3.1.3.1 Detecção e localização das caracteristicas da face

Para conseguirmos automaticamente a detecção e localização, utilizamos de novo o método descrito por Viola-Jones 3.1.2, devido à sua elevada fiabilidade, tanto em termos de processamento como de taxas de erro. De forma a otimizar o algoritmo o olho esquerdo será apenas procurado no quadrante superior direito da imagem da face normalizada, da mesma forma o olho direito será procurado no quadrante superior esquerdo, o nariz será procurado numa região central de cerca de 70% do tamanho da imagem da face e a boca na metade inferior da mesma, como ilustrado na Figura 3.14.

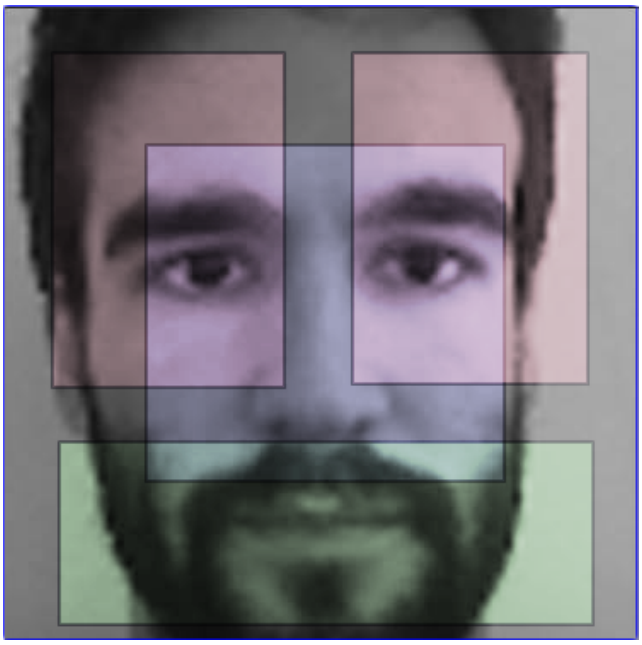


Figura 3.14: Regiões de Interesse

Existe a limitação de apenas se considerar a face detetada como válida quando são encontradas todas as caracteristicas, Figura 3.15. Pois só assim se consegue guardar todos os valores necessários.

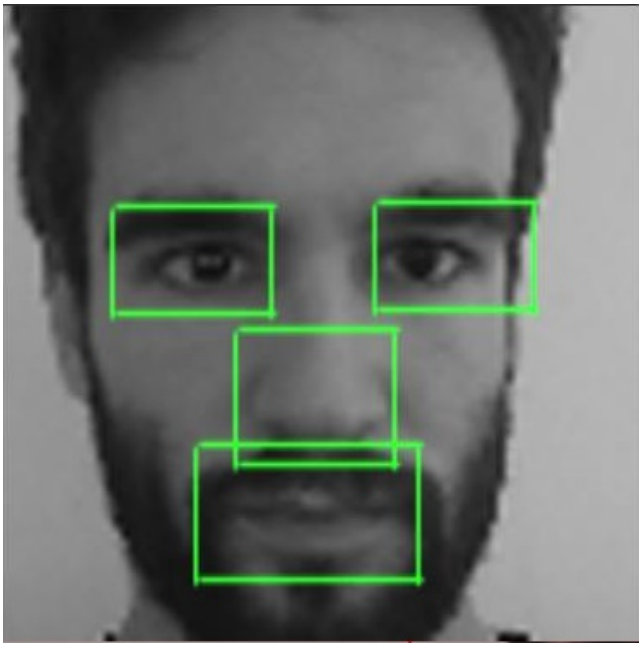


Figura 3.15: Detecção das Características

Para a detecção de cada uma dessas características usa-se um ficheiro adaptado de *Haar Cascade* começando pelo olho direito, depois o olho esquerdo, o nariz, e em seguida a boca. Foi usada apenas uma função que recebia como parâmetros: a imagem da face em escala cinza, a *HaarCascade* da característica a ser encontrada e as coordenadas das regiões de interese (todas as faces são normalizadas para 160*160), Figura 3.16. Sobre cada uma das características é desenhado um quadrado envolvente e retorna os valores das coordenadas do centro (x,y) e os valores de desvio no eixo xx e yy.

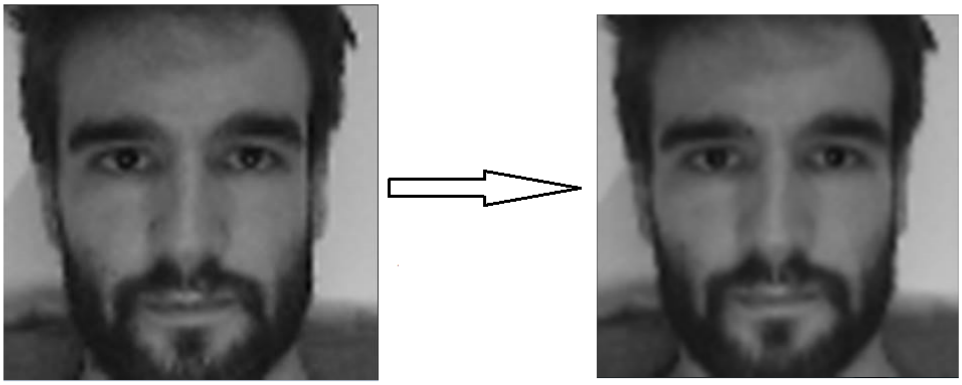
```
//Olho esquerdo
ROI(frame2, XML_FILE_LE, 14, 14, 67, 67);
//Olho direito
ROI(frame2, XML_FILE_RE, 93, 14, 67, 67);
//Nariz
ROI(frame2, XML_FILE_N, 40, 40, 80, 80);
//Boca
ROI(frame2, XML_FILE_M, 14, 80, 134, 76);
```

Figura 3.16: Funções ROI

3.1.3.2 Métrica dos Contornos

Neste método a imagem sofre alterações ao longo de quatro métodos aplicados na seguinte sequência:

- **1º** A imagem em escala cinza é suavizada com um filtro gaussiano (redução do ruído), Figura 3.17;



```
cvSmooth(cvImage, cvImage,
CV_GAUSSIAN, 3, 0, 0, 0);
//redução de ruído
```

Figura 3.17: Suavização

- **2º** De seguida a imagem é equalizada segundo o método de equalização do histograma (contraste), Figura 3.18;

```
cvEqualizeHist( cvImage,
cvImage ); //equalização do
histograma - contraste
```

Figura 3.18: Equalização do Histograma

- **3º** A imagem é binarizada usando o limiar do valor médio, Figura 3.19;

```
CvScalar avg = cvAvg(cvImage,
null);                          ,
int threshold_value = (int)
avg.val(0); //detecção do valor
médio
cvThreshold(cvImage, binImage,
threshold_value,
max_BINARY_value,
threshold_type); //binarização
```

Figura 3.19: Binarizada

- **4º** Finalmente são encontrados os contornos da face usando o operador *canny* com limiares 127 e 130 (valores em escala cinza) e tamanho de abertura 5, Figura 3.20.

```
cvCanny(binImage, edgeImage, 127, 130, 5);
//extração de contornos
```

Figura 3.20: Contornos

Como se pode ver na Figura 3.20 a luminosidade e o fundo são dois aspetos a ter em conta para a coerência dos resultados. Existe a obrigação de haver um fundo branco e luminosidade

constante. No fundo da imagem, da parte superior em branco os contornos da face são bem feitos, na parte inferior o fundo existe um elemento perturbador, visivelmente mais escuro que resulta em contornos mal feitos, influenciando o *score*.

São contabilizados todos os pixéis dos contornos assim como a soma destes por linha e coluna, um valor de *score* é calculado pela razão da soma das multiplicações do valor de pixéis de contorno da linha i com a coluna i pelo número total de pixéis do contorno.

Cálculo do *Score*:

$$score = \frac{(npe(X_i) * npe(Y_i)) * 10}{npe}$$

De forma a garantir as funcionalidades especificadas, usaram-se as interfaces presentes na secção 3.0.2, nomeadamente as Figura 3.1 e Figura 3.3 apresentam as interfaces de registo de uma face e a Figura 3.2, a interface de reconhecimento (entrada/saída) com face.

Na secção 3.0.3 são apresentados os fluxos de dados do registo de entrada/saída da face. Sendo identificada uma face pela câmara do dispositivo, esta é processada de acordo com o algoritmo apresentado no gráfico da Figura 3.21, onde são calculados os parâmetros de identificação da face, seguidamente são pedidos ao utilizador a identificação e nome do individuo, sendo estes registados na classe Face (Figura 3.4) e armazenados na base de dados dinâmica faces.XML de acordo com o esquema definido pelo DTD seguinte, Figura 3.22 e Tabela **??**.

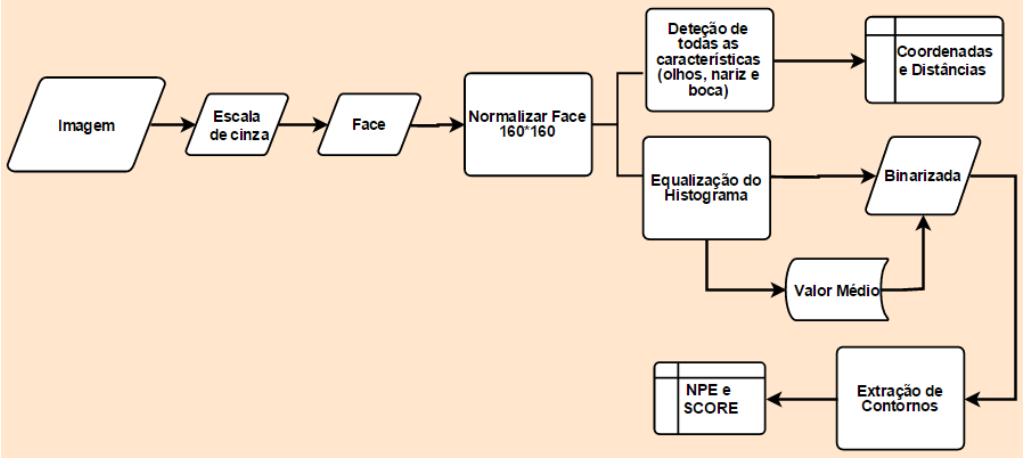


Figura 3.21: Método Reconhecimento Completo

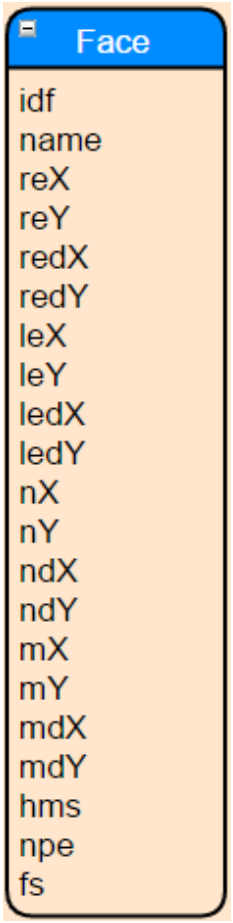


Figura 3.22: Entidade Face

Tabela 3.3: face.XML

```
<?xml version="1.0"encoding="UTF-8"?>
<!ELEMENT faces (reX, reY, redx, redy, leX, leY, ledx, ledy, nX,
nY, ndx, ndy, mX, mY, mdx, mdy, hms, npe, fs, id)>
<!ELEMENT reX (#PCDATA)>
<!ELEMENT reY (#PCDATA)>
<!ELEMENT redx (#PCDATA)>
<!ELEMENT redy (#PCDATA)>
<!ELEMENT leX (#PCDATA)>
<!ELEMENT leY (#PCDATA)>
<!ELEMENT ledx (#PCDATA)>
<!ELEMENT ledy (#PCDATA)>
<!ELEMENT nX (#PCDATA)>
<!ELEMENT nY (#PCDATA)>
<!ELEMENT ndx (#PCDATA)>
<!ELEMENT ndy (#PCDATA)>
<!ELEMENT mX (#PCDATA)>
<!ELEMENT mY (#PCDATA)>
<!ELEMENT mdx (#PCDATA)>
<!ELEMENT mdy (#PCDATA)>
<!ELEMENT hms (#PCDATA)>
<!ELEMENT npe (#PCDATA)>
<!ELEMENT fs (#PCDATA)>
<!ELEMENT id (#PCDATA)>
```

reX - coordenada x do olho direito;

reY - coordenada y do olho direito;

redx - distância em x do centro do quadrado ao limite do mesmo para o olho direito;

redy - distância em y do centro do quadrado ao limite do mesmo para o olho direito;

leX - coordenada x do olho esquerdo;

leY - coordenada y do olho esquerdo;

ledx - distância em x do centro do quadrado ao limite do mesmo para o olho esquerdo;

ledy - distância em y do centro do quadrado ao limite do mesmo para o olho esquerdo;

nX - coordenada x do centro detetado do nariz;

nY - coordenada y do centro detetado do nariz;

ndx - distância em x do centro do quadrado ao limite do nariz;

ndy - distância em y do centro do quadrado ao limite do nariz;

mX - coordenada x do centro detetado da boca;

mY - coordenada y do centro detetado da boca;

mdx - distância em x do centro do quadrado ao limite da boca;

mdy - distância do centro do quadrado ao limite da boca;

hms - soma das multiplicações dos pixéis de contorno da linha i pela coluna j;

npe - número total de pixéis do contorno da face normalizado;

fs - score da face calculado pela divisão inteira do hms pelo npe;

id - identificador do individuo no ficheiro XML.

De forma a poder efetuar o reconhecimento (entrada/saída) por face, o processo é definido pelo diagrama de fluxo de dados da Figura 3.5, semelhante ao do registo.

Usando como fator de decisão de reconhecimento o valor do score da face com a tolerância mínima de 5%.

Outro fator consiste na procura na face normalizada das caracteríticas, guardando-se as posições centrais dos olhos, nariz e boca, assim como o deslocamento em x (dx) e y (dy). À boca é dada uma tolerância de 10%, por causa das expressões.

Sendo estes pontos coincidentes e o score dos contornos coincidente em 95%, pode indicar-se que é a pessoa que existe na base de dados.

Esta forma de reconhecer não necessita de guardar imagens da face na base de dados, ao contrário dos 3 métodos posteriormente enunciados, reduzindo o espaço necessário de armazenamento, aumentando a rapidez e reforçando a segurança (por não existirem imagens de faces na base de dados Sendo representado pelo diagrama de entidades e relacionamento da Figura 53 e o seu resultado armazenado na base de dados dinâmica picagem.XML de acordo com o esquema definido pelo DTD seguinte, Figura 3.23 e Tabela 3.4.

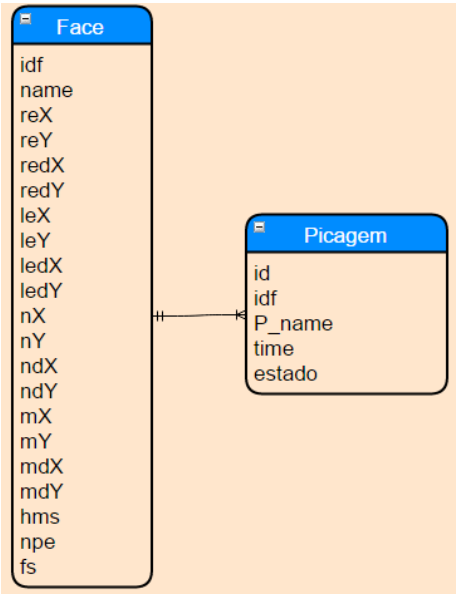


Figura 3.23: Entidade-Relação Face-Picagem

Tabela 3.4: picagem.XML

```
<?xml version="1.0"encoding="UTF-8"?>
<!ELEMENT picagem (id, idf, name, time, estado)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT idf (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT estado (#PCDATA)>
```

3.2 Reconhecimento por QRCode

O QRcode utiliza também a câmara frontal do dispositivo móvel, porém ao contrário dos métodos de reconhecimento facial, a câmara não é iniciada em conjunto com a biblioteca JavaCV, sendo normalmente pelas funções CameraPreview da Android. O QRcode já foi descrito na secção anterior (Estado da Arte), para a sua implementação usou-se a biblioteca ZXing, criando-se uma extensão para evitar o *ident* que impedia o reconhecimento contínuo em tempo real e de acordo com a interface definida para o projeto.

Após a inicialização da aplicação, a câmara é também imediatamente iniciada. A cada frame captada será verificado se existe um QRcode, caso exista é descodificado e verifica-se se a string já existe na base de dados, identificando se se trata de uma entrada/saída e regista a hora da ocorrência.

De forma a garantir as funcionalidades especificadas, usaram-se as interfaces presentes na secção 3.0.2, nomeadamente as Figuras 3.1 3.3 apresentam as interfaces de registo de um QRcode e a Figura 3.2 a interface de reconhecimento (picagem) com QRCode. Em baixo podemos ver um exemplo do funcionamento previsto, Figura 3.24

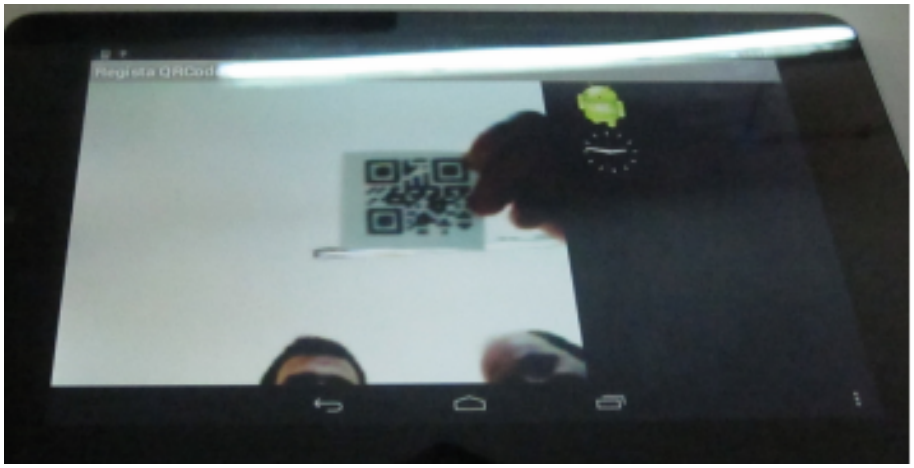


Figura 3.24: Exemplo da aplicação QR Code

Na secção 3.0.3 são apresentados os fluxos de dados do registo e registo entradas/saídas de um QRCode. Para o registo Figura 3.4, sendo identificado um QRCode pela câmara da frente do dispositivo, é lido o seu conteúdo, seguidamente são pedidos ao utilizador a identificação e nome do indivíduo, sendo estes registados na classe QRCode, Figura 3.25, e armazenados na base de dados dinâmica qrcodes.XML, Tabela **??**, de acordo com o esquema definido pelo DTD e diagrama de entidade e relacionamento.

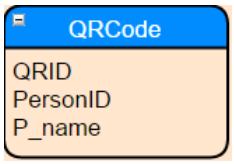


Figura 3.25: Entidade e Relacionamento QR Code

Tabela 3.5: qrcode.XML

```
<?xml version="1.0"encoding="UTF-8"?>
<!ELEMENT qrcodes (id, idf, name)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT idf (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

O processo de reconhecimento (entradas/saídas) por QRCode, é definido pelo diagrama de fluxo de dados representado na Figura 3.5, sendo semelhante ao do registo, usando como fator de decisão de reconhecimento a string do QRCode. O resultado da operação é armazenado na base de dados dinâmica picagem.XML definido pelo DTD e de acordo com o diagrama de entidades e relacionamento na Figura 3.26.

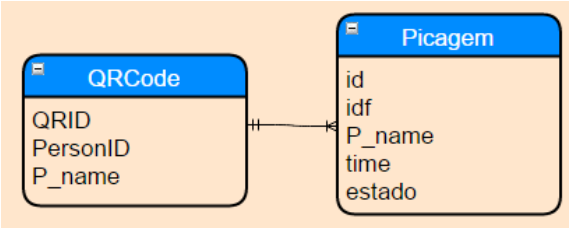


Figura 3.26: Entidade-Relação QR Code-Picagem

Na secção dos resultados será demostrado o funcionamento da aplicação desenvolvida.

3.3 Reconhecimento por NFC

O reconhecimento por NFC inicializa também a câmara frontal do dispositivo móvel apesar de não ser necessária, apenas por uma questão de interface. Para a leitura de cartões NFC, uma situação particular de RFID como discutido na revisão da literatura, é necessário que o dispositivo móvel tenha um sensor para essa tecnologia. Normalmente este está localizado na parte posterior do mesmo.

De forma a garantir as funcionalidades especificadas, usaram-se as interfaces presentes na secção 3.0.2, nomeadamente as Figuras 3.1 3.3 apresentam as interfaces de registo de uma tag NFC e a Figura 3.2, a interface de reconhecimento (entrada/saída) com NFC. Em baixo pode-se ver um exemplo do funcionamento previsto, Figura 3.27

Figura 3.27: Exemplo da aplicação NFC

Na secção 3.0.3 são apresentados os fluxos de dados do registo e registo entrada/saída de um NFC. Para o registo Figura 3.4, é apresentado o fluxo de dados do registo de uma tag NFC, sendo identificada uma tag NFC pelo sensor na parte posterior do dispositivo, é lido o seu conteúdo, seguidamente são pedidos ao utilizador a identificação e nome do individuo, sendo estes registados na classe NFCTag, Figura 3.28, e armazenados na base de dados dinâmica nfc.XML, Tabela **??** de acordo com o esquema definido pelo DTD seguinte.

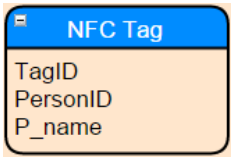


Figura 3.28: Entidade NFC

Tabela 3.6: nfc.XML

```
<?xml version="1.0"encoding="UTF-8"?>
<!ELEMENT tagsnfc (id, idf, name)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT idf (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

O processo de reconhecimento (entrada/saída) por NFC, é definido pelo diagrama de fluxo de dados representado na Figura 3.5, sendo semelhante ao do registo, usando como fator de decisão de reconhecimento a tag NFC. O resultado da operação é armazenado na base de dados dinâmica picagem.XML definido pelo DTD e de acordo com o diagrama de entidades e relacionamento na Figura 3.29. .

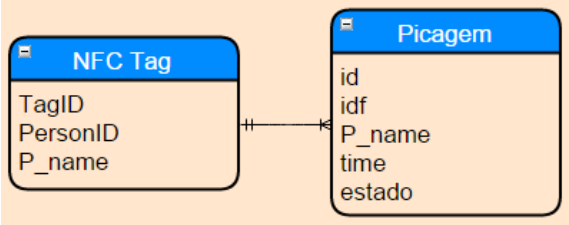


Figura 3.29: Entidade-Relação NFC-Picagem

Na secção dos resultados será demostrado o funcionamento da aplicação desenvolvida.

3.4 Serviço Web RestFull

Os conceitos de um serviço Web RestFull (HTTP e conceito Restfull) já foram descritas numa seção anterior (Estado da Arte). Para a sua implementação podia-se optar por duas bibliotecas auxiliares RetroFit ou Volley.

A escolha incidiu na biblioteca RetroFit, uma vez que, apesar de a biblioteca Volley oferecer mais recursos, todos os recursos essenciais estão presentes na RetroFit, exceptuando o mecanismo de cache que é facilmente adaptado. Revelou-se o meio mais fácil e eficaz para os objetivos da dissertação, além de ser mais leve em tamanho e complexidade, tem ainda a funcionalidade para análise automática das respostas, para os respectivos tipos de dados. O que resulta numa maior rapidez para pedidos simples como podemos ver pelos resultados dos testes efetuados pela *Instructure Tech Blog* [45], na Figura 3.30.

	One Discussion	Dashboard (7 requests)	25 Discussions
AsyncTask	941 ms	4,539 ms	13,957 ms
Volley	560 ms	2,202 ms	4,275 ms
Retrofit	312 ms	889 ms	1,059 ms

Figura 3.30: Volley vs RetroFit

A biblioteca Retrofit é um cliente REST seguro para aplicações Android/Java criado pela *Square Open Source* [43]. Através dela podemos solicitar webservices a uma API REST através de pedidos GET, POST, PATCH entre outros. É uma biblioteca muito útil, mas precisa de ser construída com uma boa arquitetura e boas práticas de programação para se poder usar da melhor forma possível.

As funcionalidades implementadas nesta dissertação vão ser exploradas em seguida.

3.4.1 Funcionalidades

Começamos então por adicionar esta biblioteca às dependencias do programa Android, Figura 3.31.

Há uma quantidade mínima de classes a escrever para solicitar uma API REST, sendo preciso pelo menos uma interface para escrever a consulta, um modelo para recuperar a resposta da API e um *restclient* para fazer as chamadas.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.1.1'
    compile 'com.squareup.retrofit:retrofit:1.9.0'
```

Figura 3.31: Dependências

3.4.1.1 Pojo's(Plain Old Java Object)

Foram necessários objectos preparados para receber os dados JSON retornados pelo servidor. Criou-se POJOs, Figura 3.32, os campos com os nomes devem ser os mesmos da resposta JSON, mas é possível utilizar nomes diferentes com a anotação *"SerializedName"*, onde se especifica o nome do campo JSON.

```
package com.pnfc.eb.fe.up.network.Response;

import ...

/**
 * Created by Filipe on 14/05/2015.
 */
public class LoginResponse {
    //TODO might have to make this public as private means introspection and it might be a performance hit.
    public String odatametadata;
    public String SessionId;
    public String Version;
    public Integer SessionTimeout;

    // default constructor, getters and setters
    public LoginResponse() {

    }

    public LoginResponse(String odatametadata, String SessionId, String Version, Integer SessionTimeout) {
        this.odatametadata = odatametadata;
        this.SessionId = SessionId;
        this.Version = Version;
        this.SessionTimeout = SessionTimeout;
    }
```

Figura 3.32: POJO para resposta ao Login

3.4.1.2 Serviços

Criou-se uma actividade de nome *Restapi*, onde se pode gerir as chamadas URL, Figura 3.33. Aqui especificou-se o tipo de pedido, como POST, GET, PATCH, entre outros, existe um URL base defenido na classe Restclient (URL comum para todas as solicitações,) e em cada método especificou-se o que adicionar ao fim do URL, os respectivos argumentos URI, *endpoint*.

```
public interface Restapi {

    @POST("/Login/")
    void login(@Body Person model,
               RestCallback<LoginResponse> callback);

    @PATCH("/EBR_UDOFUNC('{id}')")

    void picagem(@Body EBRUDOFUNC model, @Path("id") String userId,
                 RestCallback<PicagemResponse> callback);

}
```

Figura 3.33: Interface RetroFit

Para facilidade e boa prática de programação recomenda-se que o URL base termine sem a respetiva "/", sabendo-se então que o URI presente em cada método tem obrigatoriamente de começar com uma "/".

Manipulação do URL

Em algumas chamadas para o webservice pretende-se enviar alguns parâmetros, temos dois casos implementados como visto na 3.33 , @Path e @Body.

Uma chamada URL pode ser atualizada dinamicamente, com o uso de blocos de substituição e dos parâmetros dentro do método. Um bloco de substituição é uma sequência alfanumérica cercada por {}. O parâmetro correspondente deve ser anotado com @Path usando a mesma String.

Um objeto pode ser especificado para fazer parte do corpo da solicitação HTTP com a anotação @Body.

POST

O método POST envia os dados colocando-os no corpo da mensagem, neste caso os dados do LOGIN. Ele deixa a URI separada dos dados que são enviados e com isso podemos enviar qualquer tipo de dados por este método de forma segura.

PATCH

O método PATCH é usado para uma atualização parcial de um recurso. Ao contrário do método PUT, o método PATCH não exclui nenhuma propriedade do recurso local que não esteja incluída na solicitação. Usado para a inserção das picagens.

Sync. vs Async

Os métodos que retornarem *"DummyContent"* significam que as chamadas são sincronizadas e que a execução é bloqueada quando a chamada termina. Para se criar chamadas assíncronas tem que se adicionar um "Callback"nos métodos e retornar *void*.

Com os pedidos assíncronos tem que se adicionar uma chamada de retorno em cada um dos pedidos para lidar com o sucesso ou erro da resposta.

3.4.1.3 RestClient

A classe de nome *RestClient.java*, contém um objeto *restclient* com os seus serviços e um restAdapter, Figura 3.34. O *restAdapter* é usado para criar nossos serviços. O URL base é definido aqui também. Em vez de usar o cliente aconselhado para a biblioteca *Retrofit*, o *okhttp*, usou-se um cliente da Apache "HttpClient client = configureClient();"para lidar com os certificados.

```
public class RestClient {

    private static Restapi REST_CLIENT;
    private static String ROOT =
            "https://213.13.123.3:50000/b1s/v1";

    static {
        setupRestClient();
    }


    public static Restapi get() { return REST_CLIENT; }

    private static void setupRestClient() {

        HttpClient client = configureClient();

        RestAdapter restAdapter = new RestAdapter.Builder()
                .setLogLevel(RestAdapter.LogLevel.FULL)
                .setEndpoint(ROOT)
                .setClient(new ApacheClient(client)) //Chamar função
                .setRequestInterceptor(new SessionRequestInterceptor())
                .build();

        REST_CLIENT = restAdapter.create(Restapi.class);
    }
```

Figura 3.34: Classe RestClient

3.4.1.4 Header

Quando é necessário adicionar um cabeçalho para alguns pedidos, esta abordagem é útil caso seja preciso uma autenticação para enviar um pedido. Intercepta-se cada solicitação e adiciona-se um cabeçalho. Verifica-se se uma sessão de utilizador é aberta e adiciona-se neste caso um "Session ID", que é um parâmetro da resposta do servidor a um Login bem sucedido para a resposta ao login, Figura 3.32.

Adicionando uma classe chamada "SessionRequestInterceptor.java"que implementa "RequestInterceptor"onde se implementou o método de intercepção, Figura 3.35.

```java
package com.pnfc.eb.fe.up.network;

import retrofit.RequestInterceptor;
/**
 * Created by Filipe on 14/05/2015.
 */
public class SessionRequestInterceptor  implements RequestInterceptor {
    private static final String TAG = SessionRequestInterceptor.class.getSimpleName();

    private static String cookies;

    public static String getCookies() { return cookies; }


    public static void setCookies(String cookies) { SessionRequestInterceptor.cookies = cookies; }


    @Override
    public void intercept(RequestInterceptor.RequestFacade request) {

        request.addHeader("Content-Type", "application/json");//you can add header here if you need in your api
        request.addHeader("Accept","application/json");

        if (null != cookies && cookies.length() > 0) {
            request.addHeader("Cookie", cookies);
        }
    }
}
```

Figura 3.35: Classe SessionRequestInterceptor

Depois na atividade principal onde é chamado o Login adiciona-se a seguinte linha de código: *SessionRequestInterceptor.setCookies(getCookieString(response));* onde a função *getCookieString(response)*, Figura 3.36

```java
private String getCookieString(Response response) {
    for (Header header : response.getHeaders()) {
        if (null!= header.getName() && header.getName().equals("Set-Cookie")) {
            return header.getValue();
        }
    }
    return null;
}
```

Figura 3.36: Função getCookieString

3.4.1.5 Erros

A bilbioteca permite apenas respostas com códigos de erro 20*, caso a resposta seja um código 40* ou outro qualquer o Callback vai resultar num erro, não sendo fácil aceder ao corpo do pedido/chamada. Para se poder aceder e tratar dos dados em caso de erro, deve-se criar um POJO que consiga controlar esse erro. Os erros vão ser todos no seguinte formato, Figura 3.37, e o POJO correspondente, Figura 3.38:

```
{
	"error": {
		"code": -107,
		"message": {
			"lang": "en-us",
			"value": "Unable to connect with the specified username and or password"
		}
	}
}
```

Figura 3.37: Formato de um erro

```
package com.pnfc.eb.fe.up.network;

import com.google.gson.annotations.Expose;

/**
 * Created by Filipe on 14/05/2015.
 */
public class RestError {

    @Expose
    private Error error;

    /**...*/
    public Error getError() {
        return error;
    }

    /**...*/
    public void setError(Error error) { this.error = error; }

    public RestError withError(Error error) {...}


    public class Error {

        @Expose
        private Integer code;
        @Expose
        private Message message;
```

Figura 3.38: POJO para um erro

De seguida criou-se a classe RestCallback para lidar correctamente com as mensagens de erros. Que implementa uma Callback, Figura 3.39.

```java
package com.pnfc.eb.fe.up.network;

import ...

/**
 * Created by Filipe on 14/05/2015.
 */
public abstract class RestCallback<T> implements Callback<T> {
    public abstract void failure(RestError restError);

    @Override
    public void failure(RetrofitError error) {
        RestError restError = (RestError) error.getBodyAs(RestError.class); // create your own class as
        // how the error message gonna showup from server side if there is an error
        if (restError != null)
            failure(restError);
        else {
            failure(new RestError(error.getMessage()));
        }
    }
}
```

Figura 3.39: Classe RestCallback

3.4.1.6 Integração com os métodos

Após a autenticação dos métodos feita remotamente é necessário enviar os registos de entrada/saída para um webservice, pois os registos e as leituras serão feitas numa base de dados externa postgreSQL.

Para comunicar com um servidor HTTPS que exigia certificados de cliente, neste caso um certificado *self-signed*. A aplicação foi preparada para comunicar através de certificados.

Caso a autenticação do método (NFC, QRCode ou Facial) seja bem sucedida, é feito o Login no respetivo webservice, para que a aplicação seja portável e facilmente configurável, criamos um ficheiro "config.xml", Tabela 3.7 que tem de ser copiado para uma pasta /EB no dispositivo, esse ficheiro contem o URL do webservice, o utilizador, a password e o nome da base de dados.

Tabela 3.7: config.XML

```
<?xml version="1.0"encoding="UTF-8"standalone="yes"?>
<configs>
<config ID="1»
<URL>URL DA BASE DE DADOS</URL>
<username>USERNAME</username>
<password>PASSWORD</password>
<BD>BASE DE DADOS</BD>
</config>
</configs>
```

O login é feito através do método POST, que envia o username, a password e o BD num objeto JSON para o URL especificado. É recebida a seguinte mensagem de sucesso, Figura 3.40, que abre uma sessão de 30 segundos para o SessionID recebido.

```
{
        "odata.metadata": "https://213.13.123.3:50000/b1s/v1/$metadata#B1Sessions/@Element",
        "SessionId": "qSNA2WHP-U5TC-MhBc-qKhB-Xux6oJ5CrI2k",
        "Version": "910150",
        "SessionTimeout": 30
}
```


Figura 3.40: Mensagem de Login em caso de sucesso

De seguida é realizada a inserção do registo de entrada/saída através do método Patch, onde são enviados os seguintes dados num objeto JSON, Figura 3.41, de acordo com o esquema definido pela Figura 3.42.


```
{
"odata.metadata":"https://213.13.123.3:50000/b1s/v1/$metadata#EBR_UDOFUNC/@Element",
"Code":"Código do funcionário",
"EBR_FNC1Collection":[
        {"U_Data": "Data da Picagem",
        "U_Hora": "Hora da Picagem",
        "U_Tipo": "Entrada/Saída",
        "U_MacAddr":"Mac Adress do dispositivo",
        "U_SO":"N" para NFC, "Q" para QRCode e "F" para reconhecimento facial ]
}
```


Figura 3.41: JSON Picagem

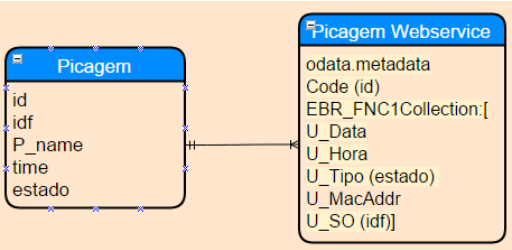


Figura 3.42: Entidade-Relação picagem.XML-Picagem.JSON

Que em caso de Sucesso não retorna nenhuma resposta.

Capítulo 4

Resultados

Neste capítulo são apresentados o desempenho dos métodos de reconhecimento expostos anteriormente, tal como do webservice e a sua integração conjunta.

4.1 Reconhecimento facial

Todos os métodos de reconhecimento facial usaram o mesmo conjunto de 20 indivíduos. Como dito anteriormente, os três primeiros métodos eram apenas para servir de comparação com o método de Reconhecimento Baseado na Localização de Caraterísticas da Face, porém apenas se conseguiu completar com sucesso o método Eigenfaces.

4.1.1 Eigenfaces

Após vários testes realizados, concluiu-se que para a eficiência deste método ser a melhor possível seria necessário cumprir as seguintes condições:

- São necessárias pelo menos 5 fotos por pessoa;
- Todas as fotos tem de ter a mesma dimensão, 160x160;
- As fotos devem ser tiradas com um fundo limpo, por exemplo usou-se uma parede clara;
- A luminosidade do local deve ser constante;
- As fotos são guardadas em escala cinza e equalizadas;
- O dispositivo móvel deve estar o mais estável possível durante o reconhecimento;

Para tirar as fotos, desenvolveu-se um programa que usa o mesmo método de detecção facial que o programa de reconhecimento. O seu funcionamento consiste em clicar-se num botão, sendo guardada a face que está a ser detectada, já em escala cinza, equalizada e com um tamanho de 160x160 num ficheiro de imagem "jpg". Pode-se consultar o código integral em anexo.

A base de dados usada é a seguinte:

Tabela 4.1: Base de Dados Eigenfaces

1 Filipe IMG_11.jpg	11 Rita IMG_121.jpgg
1 Filipe IMG_12.jpg	11 Rita IMG_122.jpgg
1 Filipe IMG_13.jpg	11 Rita IMG_123.jpgg
1 Filipe IMG_14.jpg	11 Rita IMG_124.jpg
1 Filipe IMG_15.jpg	11 Rita IMG_125.jpg
2 Rui IMG_21.jpg	12 Joana IMG_141.jpg
2 Rui IMG_22.jpg	12 Joana IMG_142.jpg
2 Rui IMG_23.jpg	12 Joana IMG_143.jpg
2 Rui IMG_24.jpg	12 Joana IMG_144.jpg
2 Rui IMG_25.jpg	12 Joana IMG_145.jpg
3 Fernando IMG_31.jpg	13 Diana IMG_151.jpg
3 Fernando IMG_32.jpg	13 Diana IMG_152.jpg
3 Fernando IMG_33.jpg	13 Diana IMG_1531.jpg
3 Fernando IMG_34.jpg	13 Diana IMG_154.jpg
3 Fernando IMG_35.jpg	13 Diana IMG_155.jpg
4 Monica IMG_41.jpg	14 Nelson IMG_161.jpg
4 Monica IMG_42.jpg	14 Nelson IMG_162.jpg
4 Monica IMG_43.jpg	14 Nelson IMG_163.jpg
4 Monica IMG_44.jpg	14 Nelson IMG_164.jpg
4 Monica IMG_45.jpg	14 Nelson IMG_165.jpg
5 Antonio IMG_51.jpg	15 Marco IMG_171.jpg
5 Antonio IMG_52.jpg	15 Marco IMG_172.jpg
5 Antonio IMG_53.jpg	15 Marco IMG_173.jpg
5 Antonio IMG_54.jpg	15 Marco IMG_174.jpg
5 Antonio IMG_55.jpg	15 Marco IMG_175.jpg
6 Jorge IMG_61.jpg	16 Brochado IMG_111.jpg
6 Jorge IMG_62.jpg	16 Brochado IMG_112.jpg
6 Jorge IMG_63.jpg	16 Brochado IMG_113.jpg
6 Jorge IMG_64.jpg	16 Brochado IMG_114.jpg
6 Jorge IMG_65.jpg	16 Brochado IMG_115.jpg
7 Fabio IMG_71.jpg	17 Ana IMG_131.jpg
7 Fabio IMG_72.jpg	17 Ana IMG_132.jpg
7 Fabio IMG_73.jpg	17 Ana IMG_133.jpg
7 Fabio IMG_74.jpg	17 Ana IMG_134.jpg
7 Fabio IMG_75.jpg	17 Ana IMG_135.jpg
8 Lopes IMG_81.jpg	18 Teixeira IMG_181.jpg
8 Lopes IMG_82.jpg	18 Teixeira IMG_182.jpg
8 Lopes IMG_83.jpg	18 Teixeira IMG_183.jpg
8 Lopes IMG_84.jpg	18 Teixeira IMG_184.jpg
8 Lopes IMG_85.jpg	18 Teixeira IMG_185.jpg
9 G3 IMG_91.jpg	19 Pacheco IMG_191.jpg
9 G3 IMG_92.jpg	19 Pacheco IMG_192.jpg
9 G3 IMG_93.jpg	19 Pacheco IMG_193.jpg
9 G3 IMG_94.jpg	19 Pacheco IMG_194.jpg
9 G3 IMG_95.jpg	19 Pacheco IMG_195.jpg
10 Luis IMG_101.jpg	20 Pintor IMG_201.jpg
10 Luis IMG_102.jpg	20 Pintor IMG_202.jpg
10 Luis IMG_103.jpg	20 Pintor IMG_203.jpg
10 Luis IMG_104.jpg	20 Pintor IMG_204.jpg
10 Luis IMG_105.jpg	20 Pintor IMG_205.jpg

Para um dispositivo móvel como o utilizado tratasse de uma operação muito pesada, 20 indivíduos resulta numa base de dados com 100 imagens, com mais indivíduos demoraria mais tempo, demonstra-se assim à partida não ser uma solução ideal para identificação em tempo real, sobretudo em dispositivos móveis.

As distâncias euclidianas entre as coordenadas da imagem de teste e as coordenadas das imagens de treino são calculadas, sendo escolhida a imagem de treino mais próximo . Um *threshold* é fixado de tal modo que se a distância mais próxima está acima do *threshold*, a face é considerada não reconhecida. Caso esteja abaixo, é reconhecida a identidade da face mais próxima. O resultado é assim dividido em quatro casos:

- **Falsa Rejeição (FR):** se a face está associada com a imagem correcta, mas a distância é maior do que o treshold.
- **Falsa Aceitação (FA):** se a face está associada com a imagem errada, mas a distância é menor do que o threshold.
- **Rejeição Correcta (CR):** se a face está associada com a imagem errada e a distância é maior do que o treshold.
- **Aceitação correcta (CA):** se a face está associada com a imagem correcta e a distância é menor do que o treshold.

Os resultados da tabela 4.2 foram conseguidos de acordo com os seguintes testes: um total de 15 indivíduos, em que 10 pertenciam à base de dados e 5 não pertenciam, foram identificados pelo programa de reconhecimento facial. Cada indivíduo foi testado duas vezes, perfacendo um total de 30 resultados, expostos na Tabela. Em caso de reconhecimento aparece a seguinte interface da Figura 4.1

Figura 4.1: Interface pessoa reconhecida

Tabela 4.2: Resultado Eigenfaces para 20 indivíduos

Método	**Treshold**	**CA**	**CR**	**FA**	**CR**	**Corretas**	**%**
Eigenfaces	125	15/20	7/10	4	4	22	73.33

Tempo de execução para a etapa de detecção da face num dispositivo móvel, neste caso o Moto G é em média 0.48s e o tempo de execução para a etapa de reconhecimento facial é 2.5s. Assim, o processo global de detecção de rosto e reconhecimento leva cerca de 3s.

4.2 Reconhecimento baseado na Localização de Características da Face

Este método difere no modo de reconhecimento, não necessitando de imagens na base de dados. A comparação é feita através da deteção das características da face (localização dos olhos, nariz e boca) guardando coordenadas de referência e a deteção normalizada dos contornos da face, a partir do qual o *score* é obtido. O funcionamento detalhado está na secção 3.1.3.

Os testes realizados com um grupo reduzido de pessoas foi possível identificar em cada uma delas todas estas características. O uso de técnicas de normalização como o redimensionamento da imagem da face a um tamanho fixo, o normalizar os contrastes da imagem em escala de cinza através da equalização de histograma, faz com que os efeitos de iluminação e variabilidade do mesmo individuo sejam reduzidos.

De forma a implementar a funcionalidade de reconhecimento facial em dispositivos móveis, foi necessário criar duas aplicações, uma para registar a face e outra para fazer o reconhecimento (picagem). A Figura 4.2 apresenta a interface da aplicação regista face onde, se uma for detetada, é ouvido um sinal sonoro e é mostrada a interface de registo (Figura 4.3) da mesma face pedindo ao utilizador o identificador e o nome do indivíduo, na parte superior podem-se ver os valores que vão ser guardados no ficheiro xml.

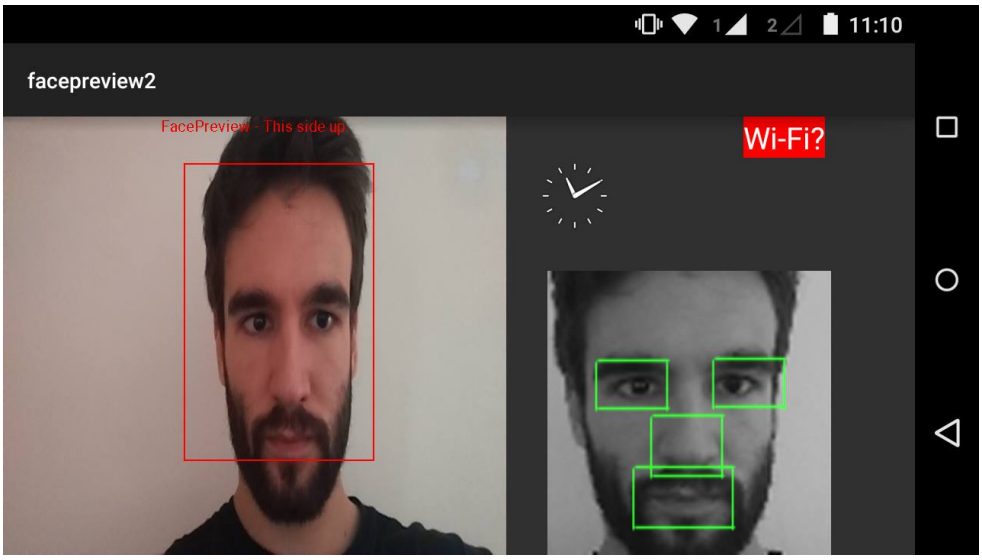


Figura 4.2: Interface de registo de face

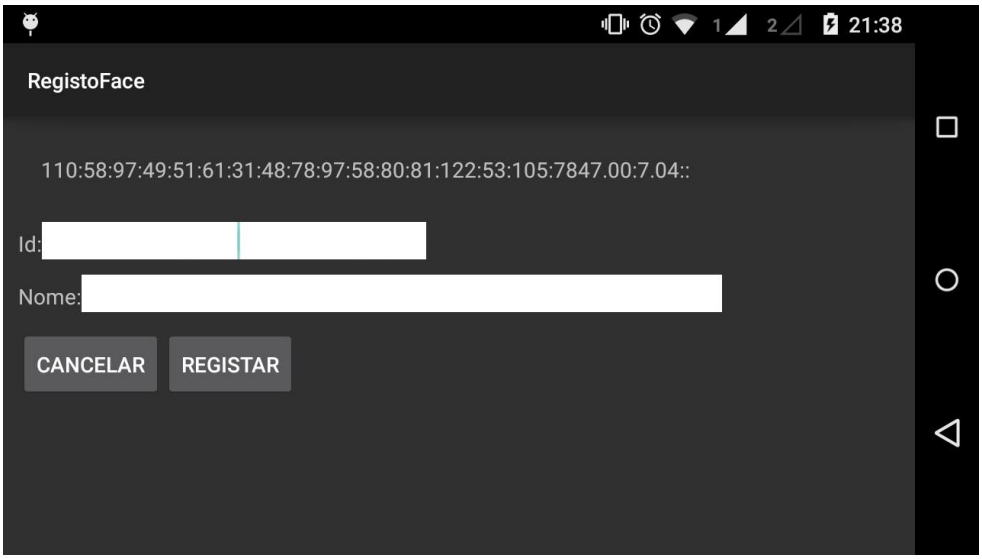


Figura 4.3: Interface de registo de informação de indivíduo após a deteção da face

No processo de determinação das características da face são usadas as operações de processamento de imagem: conversão para escala de cinza, equalização de histograma, binarização pelo valor médio e deteção de contornos usando um operador de gradiente Sobel (semelhante ao Canny mas de mais leve processamento para dispositivos móveis), secção 3.1.3.

Nesta interface, como a face registada anteriormente é mostrada numa vista de imagem no lado direito após se ouvir o sinal sonoro, ficando durante alguns segundos até a atividade de registo ser aberta, se essa imagem satisfizer o utilizador e estiver centrada pressionando o botão registar ela é inserida na base de dados local (faces.xml), numa pasta no dispositivo denominada ‘eb’. Caso

essa pasta não exista, é criada, e caso ainda não tenha havido nenhum registo o ficheiro da base de dados local também é criado.

A Figura 4.4 mostra a interface com a mensagem "Face registada com sucesso!", caso haja um erro de registo, essa mensagem de erro também é mostrada ao utilizador.

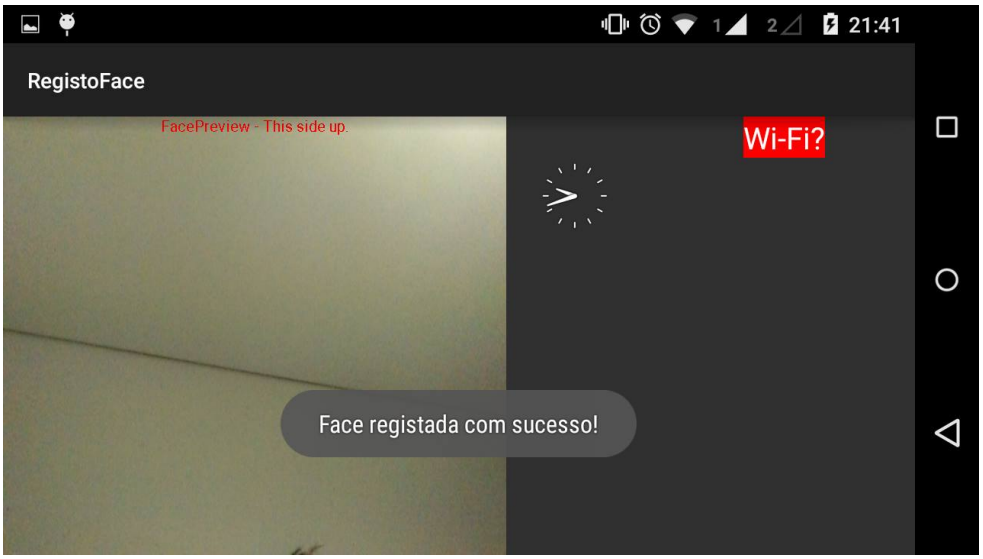


Figura 4.4: Interface com mensagem de sucesso no registo de uma face

Caso se pretenda registar um individuo já registado a mensagem presente na Figura 4.5 é apresentada.

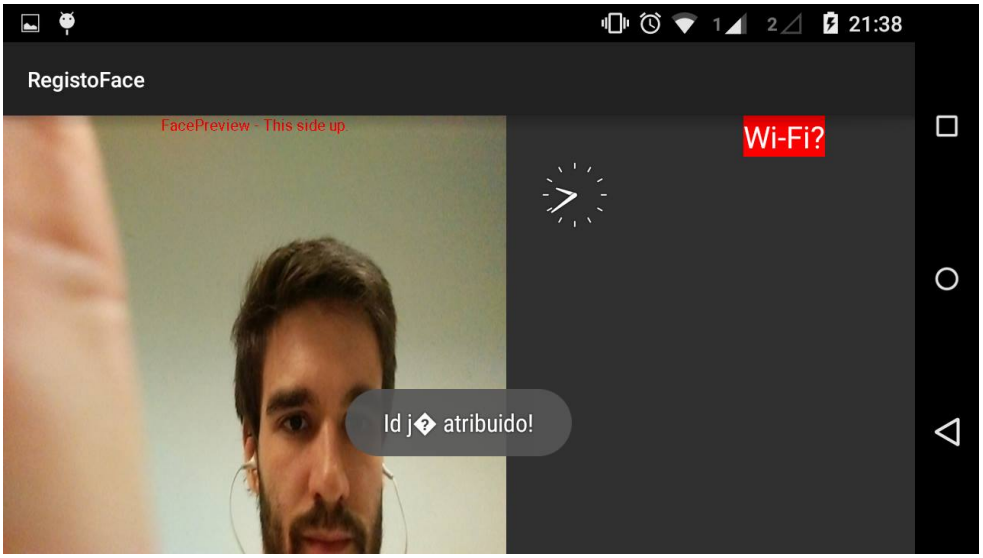


Figura 4.5: Mensagem de tentativa de registar uma face com um id já existente

Um exemplo de ficheiro com o registo de um indivíduo é apresentado em seguida, Tabela 4.3.

Tabela 4.3: Ficheiro XML com face

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<faces>
<face ID="1»
<reX>114</reX>
<reY>63</reY>
<redx>94</redx>
<redy>50</redy>
<leX>50</leX>
<leY>63</leY>
<ledx>30</ledx>
<ledy>50</ledy>
<nX>78</nX>
<nY>97</nY>
<ndx>58</ndx>
<ndy>80</ndy>
<mX>78</mX>
<mY>135</mY>
<mdx>50</mdx>
<mdy>118</mdy>
<npe>9149.00</npe>
<score>7.36</score>
<idf>fil</idf>
<name>filipe</name>
</face>
</faces>
```

Na aplicação de reconhecimento (entrada/saída) da face a interface é semelhante, o utilizador segura o dispositivo e aproxima-o da face até que ela seja detetada, sendo então acionada uma notificação sonora. Pode-se também ter o dispositivo móvel num local fixo. Primeiro é verificado se existem faces registadas, caso não existam é apresentada a mensagem presente na Figura 4.6. Caso não seja reconhecida a face, é apresentada a mensagem mostrada na Figura 4.7; na situação de ser reconhecida a face e identificado o individuo, caso não tenha ainda nenhuma entrada ou a última ocorrência tenha sido uma saída, é apresentada a interface da Figura 4.8; caso a última ocorrência tenha sido uma entrada, é apresentada a interface de saída (Figura 4.9).

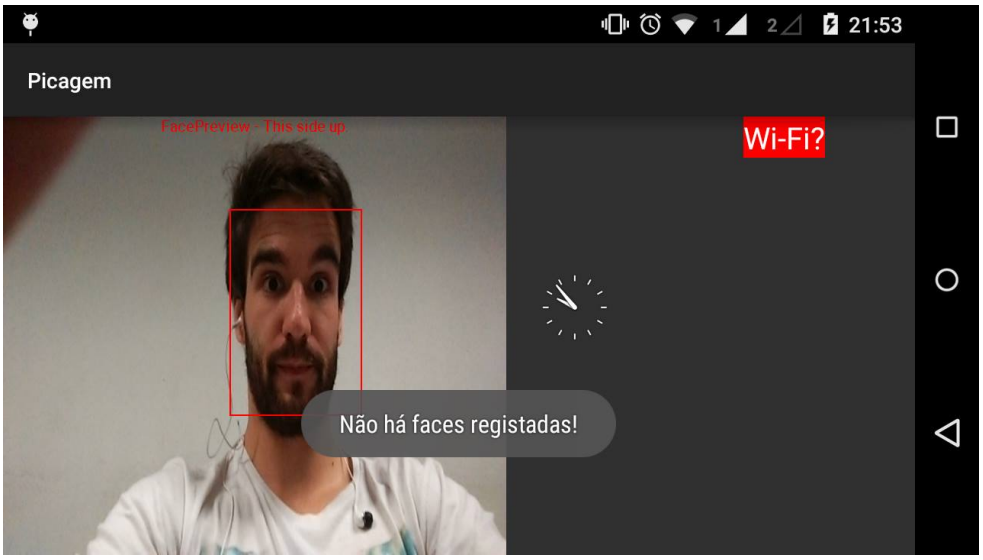


Figura 4.6: Notificação da não existência de faces registadas

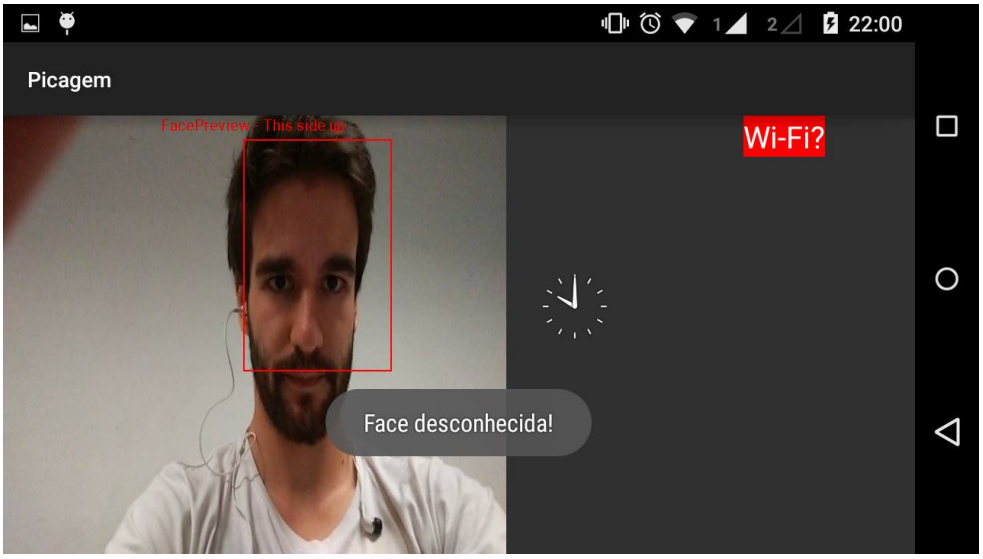


Figura 4.7: Notificação de face desconhecida

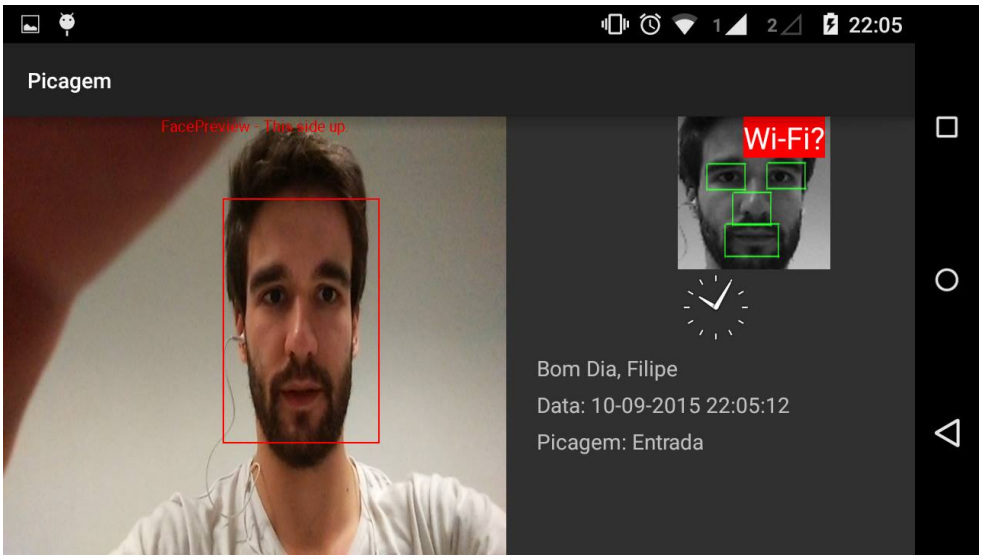


Figura 4.8: Notificação de entrada

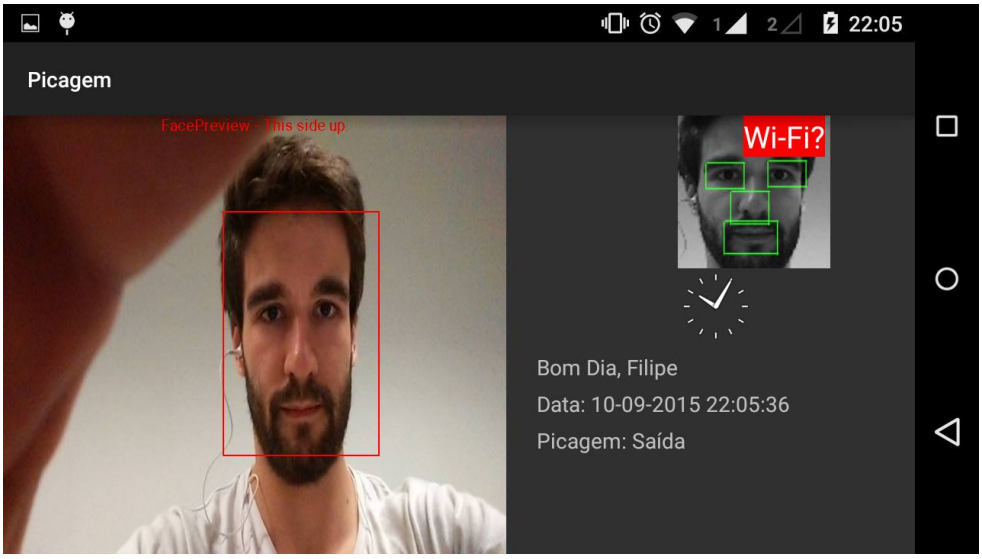


Figura 4.9: Notificação de saída

Um exemplo de um ficheiro resultante do registo de entrada/saída é apresentado em seguida.

Tabela 4.4: Ficheiro XML com registos de entrada/saída

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<picagens>
<picagem ID="1»
<idf>fil</idf>
<nome>Filipe</nome>
<tempo>02-09-2015 22:05:12</tempo>
<estado>1</estado>
</picagem>
<picagem ID="2»
<idf>fil</idf>
<nome>Filipe</nome>
<tempo>02-09-2015 22:05:36</tempo>
<estado>0</estado>
</picagem>
</picagens>
```

Contudo, existem ainda falhas no reconhecimento, ocorrências de não reconhecimento de uma face registada e de reconhecimento errado de faces registadas. Nesse sentido e após a realização de vários testes, concluiu-se que para a eficiência deste método ser a melhor possível seria necessário cumprir as seguintes condições:

- O dispositivo móvel deve estar fixo;
- O registo e reconhecimento deve ser feito com um fundo limpo, por exemplo usou-se uma parede clara;
- A luminosidade do local deve ser constante;
- Em caso de roupa com golas (camisas, pólos) o score é levemente influenciado;
- O registo e reconhecimento foram efetuados no mesmo dia;
- o registo e reconhecimento foram efetuados com o indivíduo à mesma distânica do dispositivo móvel, cerca de 50cm.

Foram testadas três versões do programa:

- Apenas usando a detecção e localização das características da face (dois olhos, nariz e boca), secção 4.2.1;
- Apenas usando o cálculo de um *score* através dos contornos, secção 4.6;
- A sua versão completa com a integração conjunta dos dois métodos citados anteriormente, secção 4.7.

A base de dados utilizada contém os mesmos indivíduos do que a usada no método Eigenfaces.

Os resultados para as 3 versões foram conseguidos de acordo com os seguintes testes: um total de 15 indivíduos, em que 10 pertenciam à base de dados e 5 não pertenciam, foram identificados pelo programa de reconhecimento facial. Cada indivíduo foi testado duas vezes, perfacendo um total de 30 resultados.

Os resultados são assim dividido em quatro casos:

- **Falsa Rejeição (FR):** se uma face está registada mas não é reconhecida.
- **Falsa Aceitação (FA):** se uma face não está registada mas é reconhecida ou está registada mas é reconhecida como outra.
- **Rejeição Correcta (CR):** se a face está não está registada e não é reconhecida.
- **Aceitação correcta (CA):** se a face está registada e é reconhecida.

4.2.1 Resultados: Detecção e Localização das Características da Face

Tabela 4.5: Resultado: Detecção e Localização das Características da Face para 15 indivíduos

Método	**Tolerância**	**CA**	**CR**	**FA**	**FR**	**Corretas**	**5%**
RBDLCF	5%	13/20	4/10	9	4	17	56.66

4.2.2 Resultados: Contornos da face

Tabela 4.6: Resultado: Contornos da face

Método	**Tolerância**	**CA**	**CR**	**FA**	**FR**	**Corretas**	**10%**
Score	5%	11/20	7/10	6	6	18	60.00

4.2.3 Resultados: Detecção e Localização das Características da Face + Contornos da face

Tabela 4.7: Resultado: Detecção e Localização das Características da Face + Contornos da face

Método	**Tolerância**	**CA**	**CR**	**FA**	**FR**	**Corretas**	**%**
RBDLCF + Score	5%/10%	17/20	8/10	4	8	20	83.33

4.3 Reconhecimento com QRcode

Para o reconhecimento por QRCode foram desenvolvidas duas aplicações, uma de registo e outra de reconhecimento (entrada/saída).

Na aplicação desenvolvida é lida uma imagem onde é verificado se existe um QRCode. No caso de existir, são mostrados os resultados de leitura independentemente da sua orientação e é ouvido um sinal sonoro para assinalar o reconhecimento, Figura 4.10.

Figura 4.10: Exemplo de QRcode

A aplicação utiliza a câmara da frente do dispositivo móvel, interface apresentada pela Figura 4.11, ficando a aguardar a identificação de um QRCode em qualquer uma das orientações. Assim que o QRCode é identificado, é lida a sua informação, um sinal sonoro é emitido e é mostrado na interface o registo (Figura 4.12), pedindo ao utilizador o identificador e o nome.

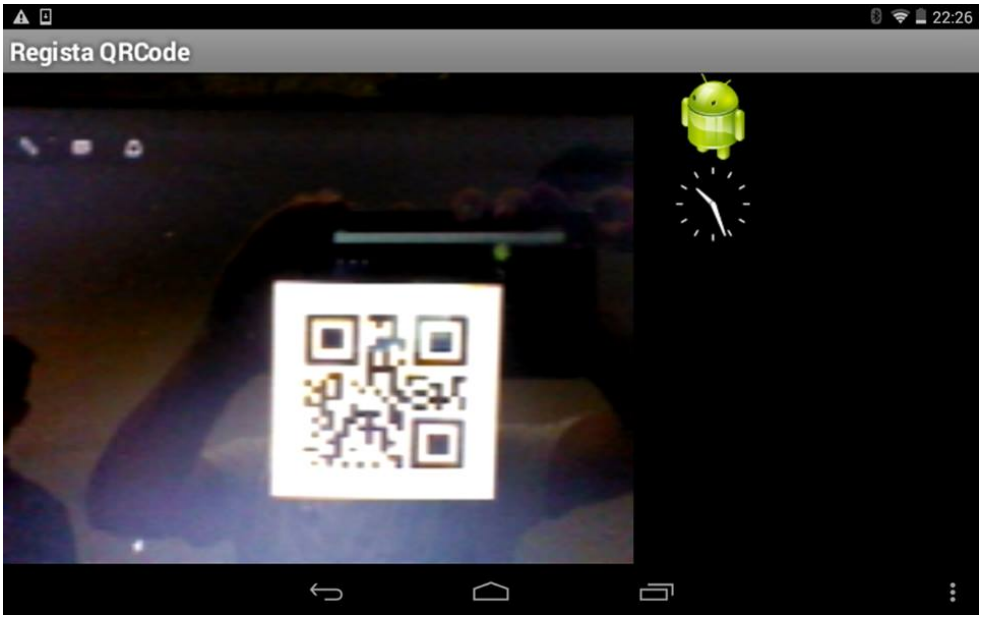


Figura 4.11: Interface de registo de QRCode

Nesta interface, é mostrado um texto superior com o conteúdo do QRCode, pressionando o botão registar o QRCode é inserido na base de dados local (qrcodes.xml), numa pasta no dispositivo denominada 'eb'. Caso essa pasta não exista, é criada, e caso ainda não tenha havido nenhum registo, o ficheiro da base de dados local também é criado.

Figura 4.12: Interface de introdução de informação do indivíduo no registo de QRCode

No caso de sucesso no registo é apresentada a notificação da Figura 4.13; caso ocorra algum erro, será apresentada a respetiva mensagem. Na situação do utilizador tentar adicionar um QR-Code com o mesmo identificador de outro já existente, isso não será possível e será apresentada uma notificação. O tempo de registo e deteção do QRCode é bastante rápido, demorando menos de 1 segundo.

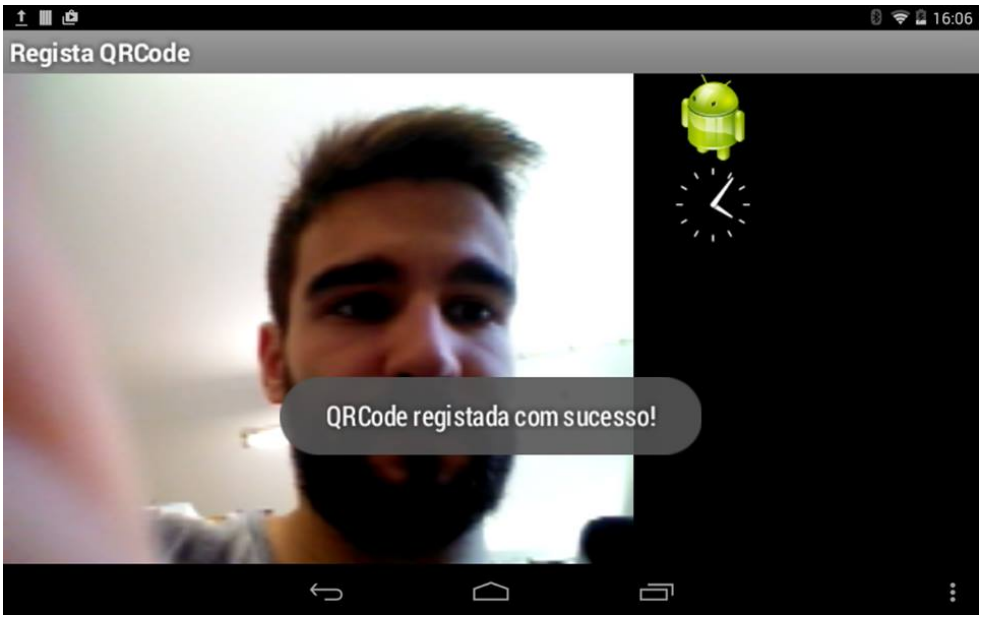


Figura 4.13: Interface com mensagem de sucesso no registo de QRCode

Um exemplo de ficheiro com o registo de um indivíduo por QRCode é apresentado em seguida.

Tabela 4.8: Ficheiro XML com QRCode

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<qrcodes>
<qrcode ID="1»
<qrcode>211256</qrcode>
<idf>fil</idf>
<name>Filipe</name>
</qrcode>
</qrcodes>
```

Na aplicação de reconhecimento (picagem) por QRCode, a interface é semelhante à do registo, o procedimento usado foi o mesmo, de apresentar um QRCode na frente da câmara frontal do dispositivo, sendo acionada uma notificação sonora assim que este é reconhecido, primeiro é verificado se existem QRCodes registados, caso não existam é apresentada uma mensagem a notificar o utilizador dessa ocorrência. Caso não seja reconhecida o QRCode é apresentada a mensagem mostrada na Figura 4.14, na situação de ser reconhecida o QRCode e identificado o individuo, caso não tenha ainda nenhuma entrada ou a ultima ocorrência fosse uma saída, é apresentada a interface da Figura 4.15, caso a última ocorrência fosse uma entrada, é apresentada a interface de saída (Figura 4.16).

Figura 4.14: Notificação de identificação de QRCode desconhecido

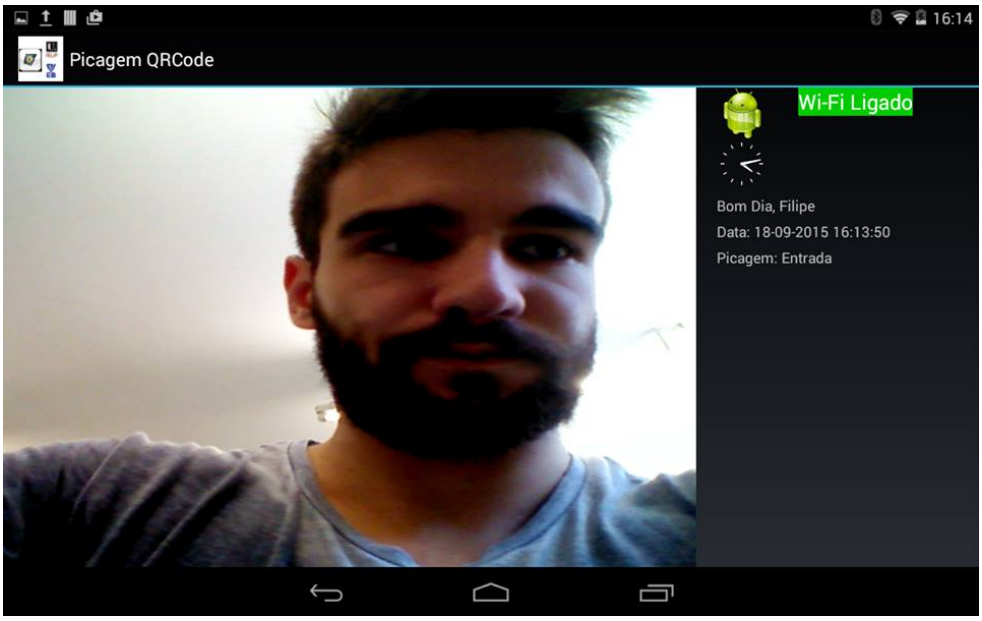


Figura 4.15: Notificação de entrada de um indivíduo com um QRCode reconhecido no sistema

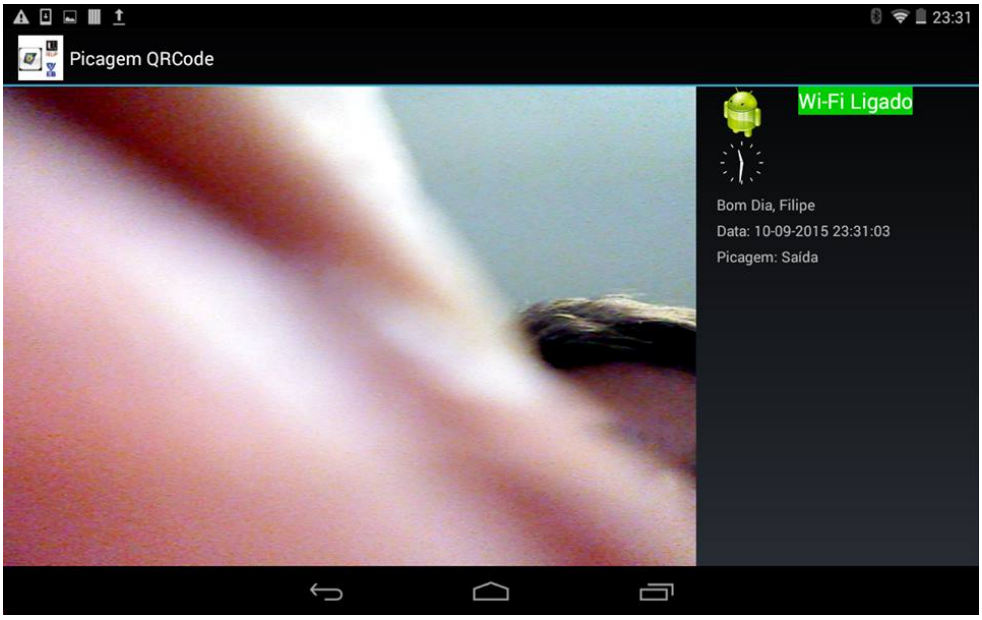


Figura 4.16: Notificação de saída de um indivíduo com um QRode reconhecido no sistema

Um exemplo de um ficheiro resultante do registo de entrada/saída é apresentado em seguida.

Tabela 4.9: Ficheiro XML com picagens

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<picagens>
<picagem ID="1»
<idf>fil</idf>
<nome>Filipe</nome>
<tempo>02-09-2015 22:05:12</tempo>
<estado>1</estado>
</picagem>
<picagem ID="2»
<idf>fil</idf>
<nome>Filipe</nome>
<tempo>02-09-2015 22:05:36</tempo>
<estado>0</estado>
</picagem>
</picagens>
```

O tempo de resposta é rápido, menos de 1 segundo, assim que o QRCode é detetado, desde que o QRCode esteja registado há uma taxa de sucesso no reconhecimento de 100%, caso não esteja a taxa de rejeição é de igual grandeza.

4.4 Reconhecimento com NFC

A aplicação de controlo de acesso por NFC foi desenvolvida para dispositivos móveis, esta divide-se em duas, à semelhança das anteriores, a de registo e a de reconhecimento (entrada/saída).

No dispositivo móvel é usado o sensor de leitura de NFC, localizado na parte posterior do dispositivo, a interface com o utilizador é a apresentada pela Figura 4.18, ficando a aguardar que o utilizador passe um cartão NFC na zona do sensor, para a tag ser lida por este. Assim que a tag NFC é identificado é lida, um sinal sonoro é emitido e é mostrada interface de registo (Figura 4.17), pedindo ao utilizador o identificador e o nome do indivíduo. Quando o botão de registo é selecionado, é inserido na base de dados local (nfc.xml), numa pasta no dispositivo denominada 'eb'. Caso essa pasta não exista, é criada, e caso ainda não tenha havido nenhum registo, o ficheiro da base de dados local também é criado.

Após o registo com sucesso é mostrada a notificação presente na Figura 4.19, caso ocorra alguma falha a mensagem sobre ela também seria apresentada.

Figura 4.17: Interface de registo de tag NFC

Figura 4.18: Interface de introdução de informção do indivíduo no registo da tag NFC

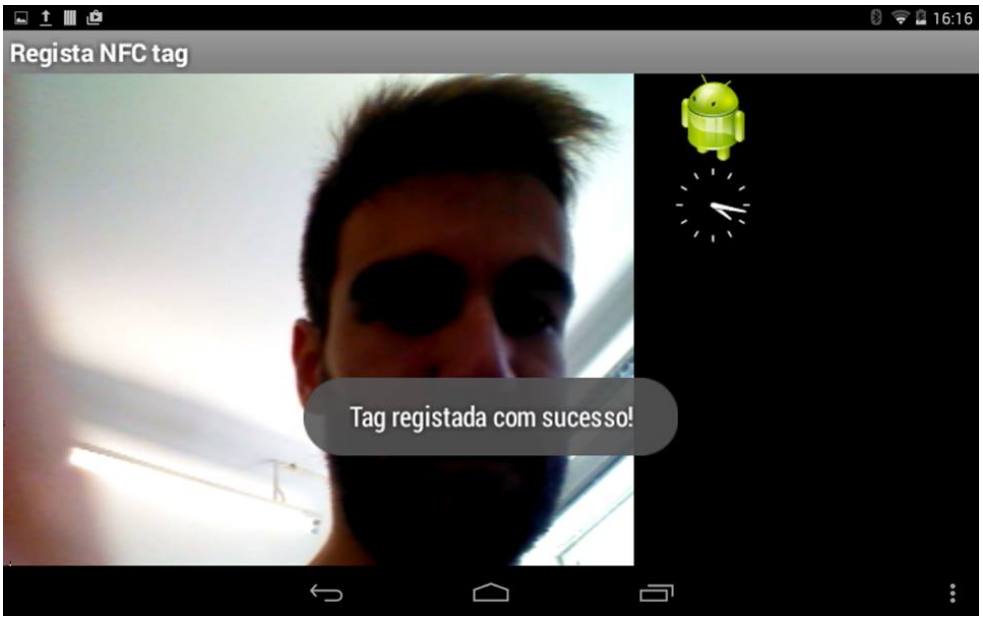


Figura 4.19: Notificação do registo de tag NFC com sucesso

Na situação do utilizador tentar adicionar uma tag NFC já existente, isso não será possível e será apresentada a notificação. O tempo de registo e detecção da tag NFC é bastante rápido, demorando menos de 1 segundo.

Um exemplo de ficheiro com o registo de um indivíduo por tag NFC é apresentado em seguida.

Tabela 4.10: Ficheiro XML com tag NFC

```
<?xml version=’1.0’ encoding=’ISO-8859-1’ ?>
<tagsnfc>
<tagnfc ID="1»
<tagnfc>AC42CB33</tagnfc>
<idf>fil</idf>
<name>Filipe</name>
</tagnfc>
</tagsnfc>
```

Na aplicação de reconhecimento (entrada/saída) por tag NFC, a interface é semelhante à do registo, o procedimento usado foi o mesmo, de apresentar um cartão NFC na zona posterior do dispositivo, onde se encontra o sensor NFC, sendo acionada uma notificação sonora assim que este é reconhecido, primeiro é verificado se existem tags NFC registadas, caso não existam é apresentada uma mensagem a notificar o utilizador dessa ocorrência. Caso não seja reconhecida a tag NFC é apresentada a mensagem mostrada na Figura 4.20, na situação em que é reconhecida uma tag NFC e identificado o indivíduo, caso não tenha ainda nenhuma entrada, ou a ultima ocorrência seja uma saída, é apresentada a interface da Figura 4.21, caso a última ocorrência fosse uma entrada, é apresentada a interface de saída (Figura 4.22).

Figura 4.20: Notificação que a NFC lida não está registada no sistema

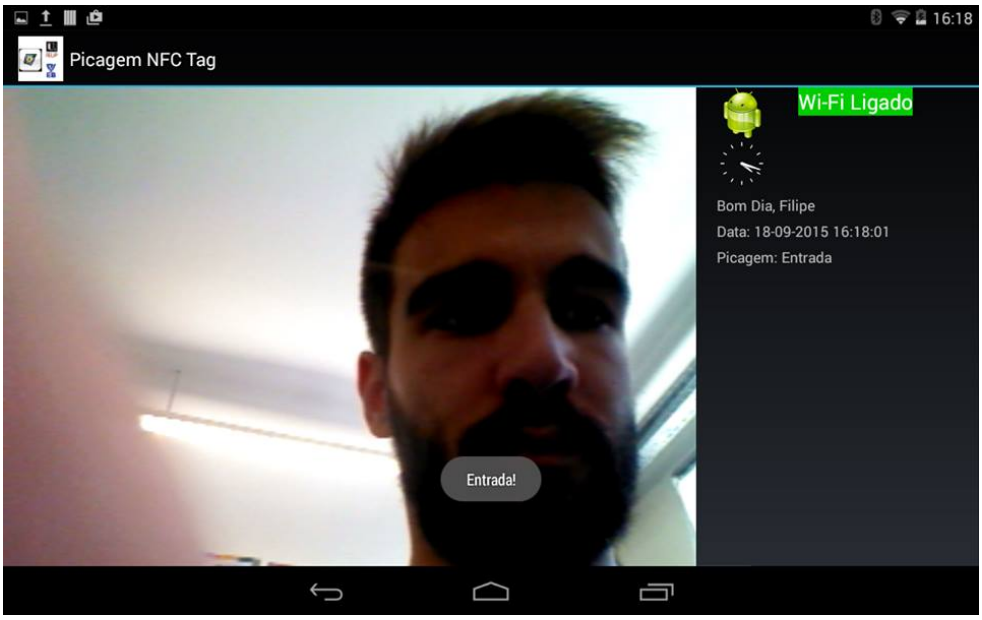


Figura 4.21: Notificação de entrada de um indivíduo com uma NFC reconhecida no sistema

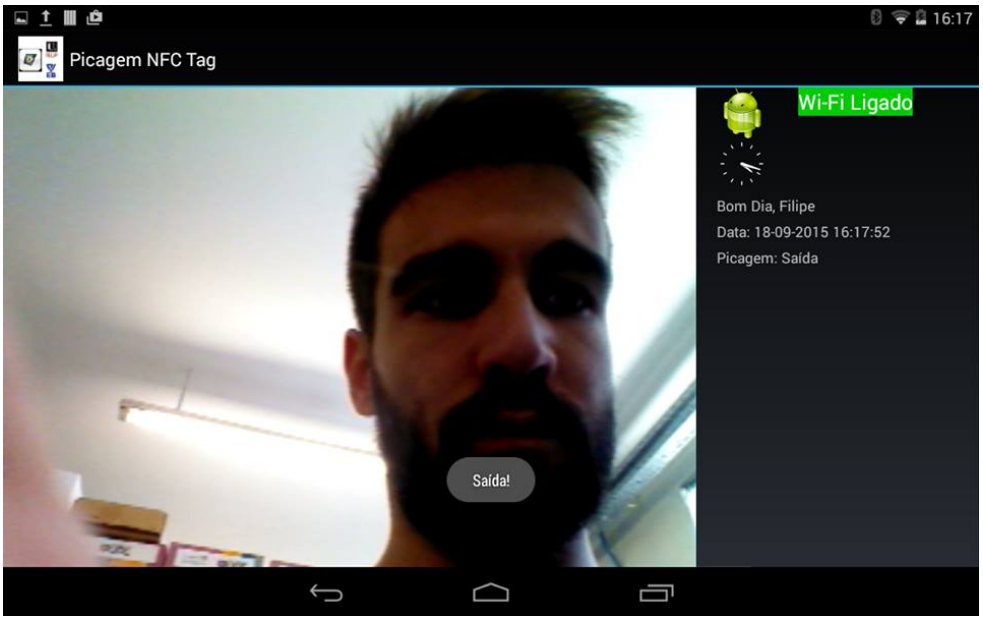


Figura 4.22: Notificação de saída de um indivíduo com uma NFC reconhecida no sistema

Um exemplo de um ficheiro resultante do registo de entrada/saída é apresentado em seguida.

Tabela 4.11: Ficheiro XML com registos de entrada saída

```
<?xml version=’1.0’ encoding=’ISO-8859-1’ ?>
<picagens>
<picagem ID="1»
<idf>fil</idf>
<nome>Filipe</nome>
<tempo>02-09-2015 22:05:12</tempo>
<estado>1</estado>
</picagem>
<picagem ID="2»
<idf>fil</idf>
<nome>Filipe</nome>
<tempo>02-09-2015 22:05:36</tempo>
<estado>0</estado>
</picagem>
</picagens>
```

O tempo de resposta é reduzido, menos de 1 segundo, assim que a tag NFC é detectada, desde que esteja registada no sistema há uma taxa de sucesso no reconhecimento de 100%; caso não esteja, a taxa de rejeição é total.

4.5 Webservice

Este serviço necessita de ligação à internet para efetuar as suas operações, sendo que só está operacional se aparecer como na figura 4.23, caso esteja como na figura 4.24 não é possível guardar as picagens na base de dados externa.

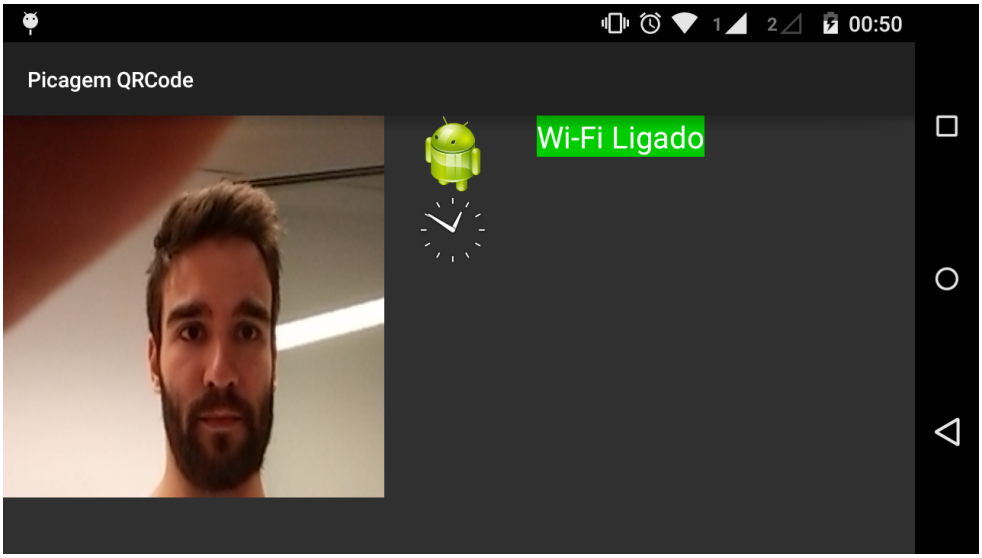


Figura 4.23: Sistema com ligação à internet

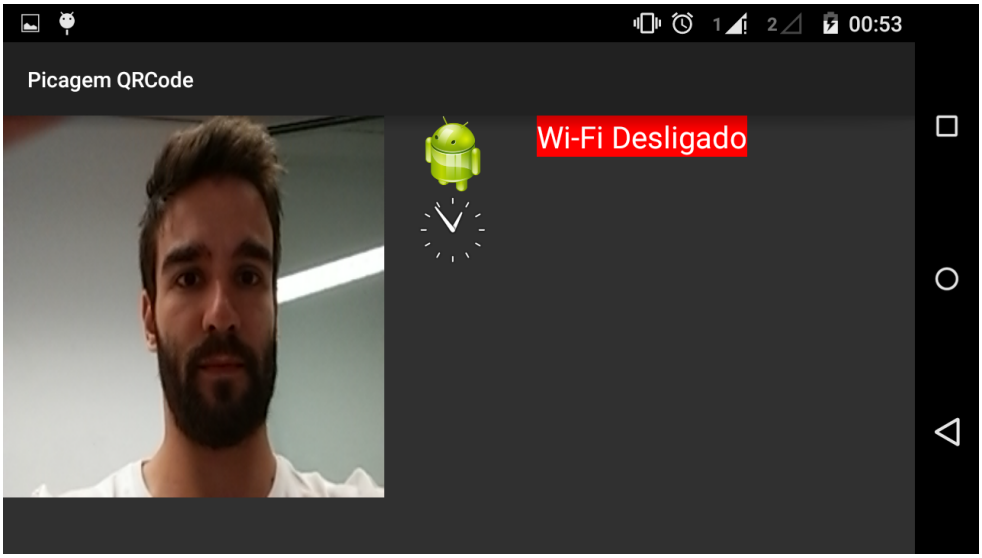


Figura 4.24: Sistema sem ligação à internet

Após o reconhecimento ser efetuado, o programa de forma automática, efetua o login e o registo de entrada/saída na base de dados externa através do webservice. Caso acha algum erro

aparece uma mensagem "Erro de Conexão", Figura 4.25 podendo dever-se ao login mal configurado ou caso o servidor esteja ocupado.

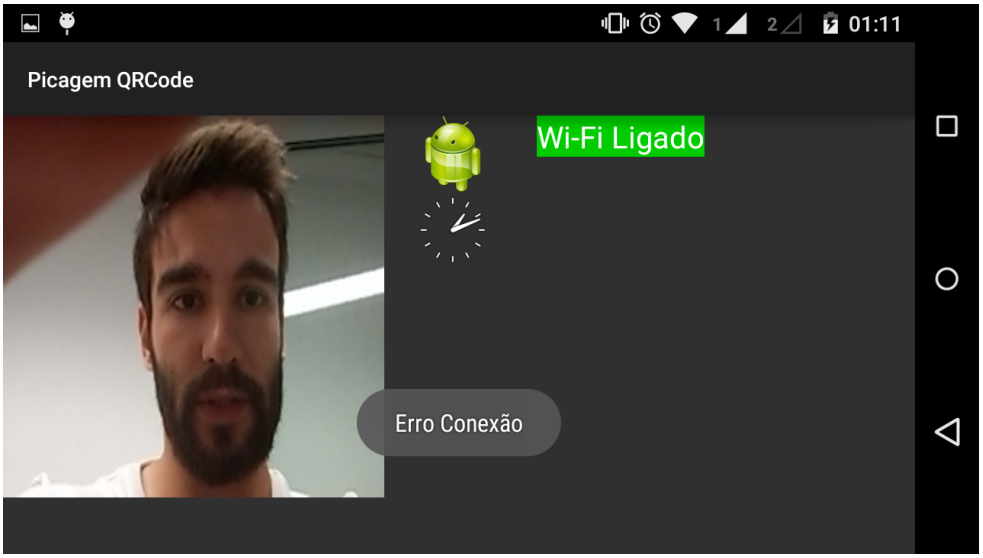


Figura 4.25: Sistema sem ligação à internet

O funcionamento do webservice ficou totalmente operacional, sendo que todas as picagens efetuadas foram guardadas na base de dados.

Capítulo 5

Discussão

Nesta secção do documento são apresentados os comentários sobre os resultados obtidos e se foram de acordo com os objetivos definidos inicialmente.

A interface definida para as aplicações de registo dos utilizadores e controlo de acesso e assiduidade é simples e requer a miníma interacção do utilizador, no entanto podem ser aperfeiçoadas em termos de aspeto visual. O mesmo optou-se por não se fazer de forma a garantir a simplicidade e a funcionalidade da mesma.

Dos métodos convencionais de reconhecimento facial com recurso a uma base de dados com imagens apenas se conseguiu implementar coma totalidade de sucesso as Eigenfaces.

Foi desenvolvida um método de reconhecimento facial baseado nas caracteristicas da face detectada, primeiro tendo em conta uma métrica dos contornos da face, sendo complementada em seguida coma posição do olho esquerdo, olho direito, nariz e boca. O método desenvolvido provou-se eficaz e garantiu o reconhecimento de indivíduos podendo futuramente a ser melhorado com a adição de mais características.

O desenvolvimento de autênticação com recurso à leitura de QRCode foi implementado na totalidade com uma taxa de sucesso de 100

A aplicação desenvolvida para a autênticação com tag NFC, apenas corre nos dispositivos móveis que possuem este sensor de leitura, foi também implementado na totalidade e a fiabilidade da sua utilização é também de 100

Foi desenvolvida uma interface de webservice restfull que permite o armazenamento do registo de acesso (entrada/saída) numa base de dados remota permitindo a gestão dos acessos numa aplicação de *back office*.

A sensibilidade do método Eigenfaces implementado é de 73%, apenas os contornos da face do método desenvolvido é de 60%, o método desenvolvido apenas com as características da face é de 57%, enquanto que o método desenvolvido combinando os contornos e as características da face apresentou um valor de 83%. Podemos então entender que o método desenvolvido tem uma descriminação superior ao método convencional revelando melhor performance e garantindo a segurança dos utilizadores protegendo a sua confidencialidade ao não usar uma base de dados de imagens.

Capítulo 6

Conclusão

No presente capítulo são apresentadas as conclusões finais da dissertação e são sugeridos alguns desenvolvimentos futuros que permitem adicionar valor ao trabalho realizado.

6.1 Conclusão Final

A finalidade de implementação do sistemas de controlo de acessos usando um dispositivo móvel usando características biométricas com recurso a métodos complementares de autenticação foi implementado com sucesso e o mesmo apresentou performance e fiabilidade superior aos métodos tipicamente convencionais de reconhecimento facial.

Perante isto pode dizer-se que os objetivos definidos para este projecto foram atingidos na totalidade. A execução deste projeto possibilitou a aprendizagem e o aumento de conhecimentos no domínio da biometria e dos dispositivos móveis, nomeadamente no que diz respeito ao reconhecimento facial, uso de bibliotecas de QRCode e NFC e no desenvolvimento de interfaces para webservices Restfull.

6.2 Proposta de Trabalho Futuro

Como desenvolvimentos futuros propõe-se o seguinte:

- Melhorar a interface com o utilizador sem reduzir a simplicidade e funcionalidade da mesma;
- Implementar os métodos convencionais de reconhecimento facial de fisherfaces e LBPH;
- Estabelezer uma comparação entre os vários métodos de reconhecimento facial numa população mais alargada;
- Adicionar mais características da face ao método desenvolvido no âmbito desta dissertação;
- Combinar numa única aplicação o reconhecimento facial com as restantes soluções de acesso já implementadas;

- Por o sistema operacional em modo offline, sendo o registo de entradas/saídas enviadas para a base de dados remota quando a ligação for restabelecida, permitindo assim uma tolerância a falhas de rede.

Referências

[1] Anne Adams e Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[2] Tech Target. Managing mobile authetication methods. URL: http://searchmobilecomputing.techtarget.com/feature/Managing-mobile-authentication-methods.

[3] Nathan Luke Clarke e SM Furnell. Advanced user authentication for mobile devices. *computers & security*, 26(2):109–119, 2007.

[4] date = 2015-05 url = http://www.apple.com/ios/what-is/ Apple inc, title = iOS.

[5] date = 2015-03 url = https://www.macstories.net/news/apple-announces-40-billion-app-store-downloads-almost-20-billion-in-2012-alone/ MacStories, title = Apple Announces 40 Billion App Store Downloads, Almost 20 Billion In 2012 Alone.

[6] date = 2015-03 url = http://www.androidcentral.com/what-android AndroidCentral, title = What is Android?

[7] date = 2015-05 url = http://www.idc.com/prodserv/smartphone-os-market-share.jsp IDC, title = Smartphone OS Market Share, 2015 Q2.

[8] date = 2015-03 url = https://en.wikipedia.org/wiki/Windows$_P honeWikipedia, title = WindowsPhone$.

[9] date = 2015-03 url = http://www.phonearena.com/news/IDC-predicts-Android-market-growth-to-end-and-Windows-Phone-to-beat-out-iOS$_id 30966IDC, title = IDCpredictsAndroidmarketgrowthtoend, andWindowsPhonetobeatoutiOS$.

[10] Garima Rani e Anshuman Saurabh. An overview of biometric techniques. Em *Journal of Emerging Technologies and Innovative Research*, volume 1. JETIR, 2014.

[11] Anil K Jain, Arun Ross, e Salil Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20, 2004.

[12] Adrian Pocovnicu. Biometric security for cell phones. *Informatica Economica*, 13(1):57–63, 2009.

[13] Sharath Pankanti, Ruud M Bolle, e Abhishek Jain. Biometrics: The future of identification. *Computer*, 33(2):46–49, 2000.

[14] Aleix M Martínez e Avinash C Kak. Pca versus lda. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(2):228–233, 2001.

[15] M. Almeida e H. Bento. Reconhecimento de padrões através de eigenfaces. URL: http://www.sinfic.pt/SinficWeb/displayconteudo.do2?numero=44666.

[16] Timo Ojala, Matti Pietikäinen, e David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[17] Caifeng Shan, Shaogang Gong, e Peter W McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 27(6):803–816, 2009.

[18] Ahmet Ozdil e Metin Mete Ozbilen. A survey on comparison of face recognition algorithms. Em *Application of Information and Communication Technologies (AICT), 2014 IEEE 8th International Conference on*, páginas 1–3. IEEE, 2014.

[19] Anil K Jain, Patrick Flynn, e Arun A Ross. *Handbook of biometrics*. Springer Science & Business Media, 2007.

[20] Harbi AlMahafzah e Maen Zaid AlRwashdeh. A survey of multibiometric systems. *arXiv preprint arXiv:1210.0829*, 2012.

[21] Anil Jain, Karthik Nandakumar, e Arun Ross. Score normalization in multimodal biometric systems. *Pattern recognition*, 38(12):2270–2285, 2005.

[22] Andrew Teoh, SA Samad, e A Hussain. Nearest neighbourhood classifiers in a bimodal biometric verification system fusion decision scheme. *Journal of Research and Practice in Information Technology*, 36(1):47–62, 2004.

[23] ISO/IEC 18004:2006. *Information technology – Automatic identification and data capture techniques – QR Code 2005 bar code symbology specification*. ISO, Geneva, Switzerland.

[24] Damri Samretwit e Toshihiko Wakahara. Measurement of reading characteristics of multiplexed image in qr code. Em *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on*, páginas 552–557. IEEE, 2011.

[25] Denso-wave.com. Qr code standardization | qr code.com. URL: http://www.qrcode.com/en/.

[26] ISO/IEC 18004:2000. *Information technology – Automatic identification and data capture techniques – Bar code symbology – QR Code*. ISO, Geneva, Switzerland.

[27] Sean Owen. Official zxing ("zebra crossing") project home. URL: https://github.com/zxing/zxing.

[28] Veldhuis R. N. Tao, Q. Biometric authentication for a mobile personal device. *In proceedings of the IEEE 3rd Annual International Conference on Mobile and Ubiquitous Systems-Workshops*, 1(03), 2006.

[29] Roy Want. An introduction to rfid technology. *Pervasive Computing, IEEE*, 5(1):25–33, 2006.

[30] Klaus Finkenzeller. Book tools. 2003.

[31] Ron Weinstein. Rfid: a technical overview and its application to the enterprise. *IT professional*, 7(3):27–33, 2005.

[32] Mike Hendry. *Multi-application smart cards: technology and applications*. Cambridge university press, 2007.

[33] Wikipedia. Single sign-on (sso). URL: https://en.wikipedia.org/wiki/Single_sign-on.

[34] Flavio D Garcia, Peter van Rossum, Roel Verdult, e Ronny Wichers Schreur. Wirelessly pickpocketing a mifare classic card. Em *Security and Privacy, 2009 30th IEEE Symposium on*, páginas 3–15. IEEE, 2009.

[35] OpenCV. Opencv. URL: http://opencv.org/.

[36] itseez. itseez. URL: http://itseez.com/.

[37] Samuel Audet. Java interface to opencv and more. URL: https://github.com/bytedeco/javacv.

[38] Robert Laganière. *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 recipes to master this library of programming functions for real-time computer vision*. Packt Publishing Ltd, 2011.

[39] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. Tese de doutoramento, University of California, Irvine, 2000.

[40] IEFT. Hypertext transfer protocol – http/1.1. URL: http://www.ietf.org/rfc/rfc2616.txt.

[41] Developers. Dashboards. URL: https://developer.android.com/about/dashboards/index.html.

[42] date = 2015-04 url = https://android.googlesource.com/platform/frameworks/volley Google, title = Volley.

[43] Square Open Source. Retrofit. URL: http://square.github.io/retrofit/.

[44] Paul Viola e Michael Jones. Rapid object detection using a boosted cascade of simple features. Em *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, páginas I–511. IEEE, 2001.

[45] INSTRUCTURE TECH BLOG. Android async http clients: Volley vs retrofit. URL: http://instructure.github.io/blog/2013/12/09/volley-vs-retrofit/.