Faculdade de Engenharia da Universidade do Porto

Processamento de Imagens Térmicas Para a Avaliação do Risco de Pé Diabético - *FrontOffice*

Joana Maria Vieira da Costa Tavares

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Ricardo Vardasca, PhD.
Co-orientador: Joaquim Gabriel Mendes, PhD.

junho, 2017

Resumo

A *Diabetes Mellitus* tem uma enorme prevalência, tanto a nível mundial como em Portugal, onde cerca de 14% da população possui a condição. As complicações do *pé diabético* estão associadas a uma enorme morbilidade, alta taxa de mortalidade e grandes custos para o sistema de saúde. Torna-se, assim, muito importante apostar na sua prevenção. Atualmente, não existe nenhum método de rastreio objetivo que permita avaliar o risco de *pé diabético*. O uso da termografia por infravermelhos é promissor como candidato a ferramenta de rastreio em fase muito precoce, nomeadamente pela monitorização da temperatura à superfície do pé nestes doentes. Assim sendo, é muito relevante o avanço no aperfeiçoamento tecnológico na obtenção, registo e análise destas imagens térmicas. Para se atingir essa premissa, é necessário validar as recentes câmaras térmicas de baixo custo acopladas a um dispositivo móvel, de forma a aumentar a sua portabilidade e adoção na prática clínica.

A não existência de uma aplicação disponível para dispositivos móveis com as características e funcionalidades requeridas para esta implementação criou a necessidade de desenvolver uma aplicação para o sistema operativo *Android*, cujo objetivo era facilitar a captura de imagens térmicas dos pés, dando a possibilidade de alterar alguns parâmetros importantes, como a emissividade, e armazená-las na galeria do dispositivo, associando-as aos dados clínicos de cada doente, pelo preenchimento do formulário concebido na aplicação. Inicialmente, a aplicação permitia guardar as imagens numa base de dados local e, posteriormente, permite o envio dos dados para um *webservice* remoto, para a sua avaliação e processamento.

A solução desenvolvida foi testada e validada em situação real, o que permitiu comprovar a sua funcionalidade. A utilização da aplicação desenvolvida em dispositivos móveis permite disponibilizar uma ferramenta de diagnóstico clínico mais adaptada e de maior portabilidade, podendo, no futuro, os dados recolhidos serem classificados por sistemas inteligentes, permitindo uma melhor caracterização do estado de saúde dos doentes, o que vai poder reduzir o impacto nos custos associados à patologia.

Palavras-chave: *Android,* FLIR ONE, *pé diabético*, termografia

Abstract

Diabetes Mellitus has an enormous prevalence, both around the world and in Portugal, where about 14% of the population has this condition. Diabetic foot's complications determine high morbidity and mortality and big costs to the health system. Thus, its prevention is of the maximum importance. Nowadays, there is not an objective diagnostic method that allows to evaluate the risk of having diabetic foot. The use of IR thermography is promising as a candidate for a screening tool in a very early stage, namely by monitoring these patients surface foot temperature. It is of extreme relevance the technologic perfecting in obtaining, registering and analyzing these thermal images. To do so, it is important to validate the use of the recent low-cost thermal cameras coupled to a mobile device, which would increase its portability and clinical use.

Because there is not an app for mobile devices with the characteristics or functionalities required for this implementation, an Android app was created with the purpose of capturing foot thermal images, giving the possibility of altering a few crucial parameters, like emissivity, and save them in the phone's gallery, associating them with each patient's clinical data, by filling a form conceived in the app. Initially, the app allowed saving images in a local database and, now it allows sending the data to a remote webservice for evaluation and processing.

The developed solution was tested and validated in a real scenario, which allowed to prove its functionality. The use of the app in mobile devices allows the provision of a clinical diagnostic tool better adapted and with better portability. In the future, the collected data can be classified by intelligent systems, allowing a better characterization of patients' health status, which will reduce the cost impact associated with the pathology.

***Keywords**: Android, diabetic foot, FLIR ONE, thermography*

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu Orientador, Professor Doutor Ricardo Vardasca, não apenas pela oportunidade de desenvolver este projeto, que considero um grande e extraordinário ensejo de investigação, mas também pelo enquadramento científico do mesmo, pelo apoio, críticas e sugestões e pela disponibilização dos meios necessários à sua realização.

Ao Professor Doutor Joaquim Gabriel, também agradeço pela simpatia e estímulo.

A todos os que partilharam comigo o laboratório, um muito obrigada pelo bom ambiente de trabalho e boa disposição com que fui sempre recebida. Em particular, agradeço à Lúcia, minha companheira de Mestrado, a sua amizade e por ter estado sempre disponível para mim.

Gostava de demonstrar o meu apreço pelo projeto NORTE-01-0145-FEDER-000022-*SciTech – Science and Technology for Competitive and Sustainable Insdustries*, co-financiado pelo Programa Operacional Regional do Norte (NORTE2020), financiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER) e pela FCT – Fundação para a Ciência e Tecnologia sobre o projeto (PEst-OE/EME/LA0022/2013).

Agradeço a todos os meus amigos pelo encorajamento e por compreenderem que nesta fase não pudesse estar tão presente.

Ao João e à Diana, um obrigada especial pelo apoio incondicional e pelo incentivo.

Queria, também, mostrar o meu reconhecimento à Margarida pelo afeto e carinho.

Ao Professor Doutor Luís Teixeira, pela amizade sempre demonstrada e pelo estímulo, um agradecimento muito especial e sentido.

Agradeço à minha familía e aos meus avós em particular, pelo apoio, ânimo e carinho.

Aos meus pais e irmã por serem o meu porto de abrigo, a minha gratidão sem limites. Sem eles nada disto teria sido possível.

“Great things are done by a series of small things brought together.”

Vincent Van Gogh

Índice

Lista de figuras

Lista de tabelas

Abreviaturas e Símbolos

Lista de abreviaturas (ordenadas por ordem alfabética)

ACES	Agrupamento de Centros de Saúde
CRUD	Criar, Ler, Atualizar e Apagar
DM	*Diabetes Mellitus*
DSP	Processamento de Sinal Digital
FEUP	Faculdade de Engenharia da Universidade do Porto
HTTP	*Hypertext Transfer Protocol*
IMC	Índice de massa corporal
IR	Infravermelhos
IRT	Imagem Térmica de Infravermelhos
JPG	*Joint Photographic Experts Group*
MIEEC	Mestrado Integrado em Engenharia Electrotécnica e de Computadores
MS	*Microsoft*
REST	*Representational State Transfer*
SDK	*Software Development Kit*
WBS	*Work Breakdown Structure*

Lista de símbolos

ε	Emissividade
α	Energia Absorvida
ρ	Reflexão
τ	Transmissão
σ	Constante de *Stefan-Boltzmann*
P	Energia Irradiada
A	Área
T	Temperatura

Capítulo 1

Introdução

Neste documento, apresenta-se uma descrição detalhada e fundamentada do trabalho desenvolvido no contexto desta dissertação de mestrado. Esta surge no âmbito do Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MIEEC), da Faculdade de Engenharia da Universidade do Porto (FEUP).

O trabalho foi realizado, maioritariamente, no laboratório L003, no Departamento de Engenharia Mecânica da FEUP, sob orientação do Professor Doutor Ricardo Vardasca e co-orientação do Professor Doutor Joaquim Gabriel Mendes.

Este capítulo serve para fazer o enquadramento do trabalho, nomeadamente explicando a sua pertinência e importância, bem como a motivação para a sua idealização e realização. Descreve, ainda, os principais objetivos deste projeto e a estrutura e organização deste documento.

1.1 - Contexto e Motivação

Diabetes Mellitus (DM) é um grupo de doenças metabólicas que se caracterizam pela presença de hiperglicemia. Associa-se a alterações gerais do metabolismo dos carboidratos, lipídico e proteico e esta desregulação metabólica causa alterações fisiopatológicas secundárias em múltiplos órgãos [1],[2].

A DM (sobretudo a do tipo 2) tem tido uma importante e crescente prevalência a nível mundial, sendo uma causa importante de morbilidade, podendo conduzir a morte precoce quando não diagnosticada e tratada de forma correta. O aumento do número de casos de DM é dependente do envelhecimento das populações, do desenvolvimento económico, da urbanização e da alteração de comportamentos daí decorrentes, tais como a escolha de dietas hipercalóricas e pouco saudáveis, a atividade física reduzida e a epidemia concomitante de obesidade e síndrome metabólico.

Além das consequências devastadoras na saúde global das populações, a DM tem um impacto económico importante, quer para os doentes e familiares, quer para a Sociedade e os Sistemas de Saúde [3].

A International Diabetes Federation considera a DM uma das maiores emergências de Saúde globais do século XXI, quer pelas complicações importantes que se lhe associam, quer pela sua grande prevalência, em aumento exponencial. Estima-se que a sua prevalência mundial é de aproximadamente 8% da população adulta, entre os 20 e os 79 anos de idade [4]. Em 2015, a sua prevalência estimada era de 415 milhões de diabéticos em todo o mundo (382 milhões em 2013), 29.3 milhões nos Estados Unidos da América e 59.8 milhões na Europa. De igual modo, estima-se que o número de 318 milhões, em todo o mundo, de adultos com intolerância à glicose é igualmente alarmante, já que estão em risco de vir a desenvolver a doença no futuro. Importa ainda considerar que em cerca de 50% dos casos, a DM não é diagnosticada, devido à sua evolução e apresentação clínica silenciosas [3].

Por outro lado, através de estudos observacionais a longo prazo, prevê-se um aumento preocupante do número destes doentes em todo o mundo, até cerca de 591.9 milhões no ano de 2035 [3], [5] e de 642 milhões no ano de 2040 (um aumento aproximado de 55% e de 60%, respetivamente) [3]. Isto corresponde a um crescimento anual médio de cerca de 2.5%, o que é aproximadamente 1 a 2 vezes o crescimento anual da população adulta mundial [5].

Em Portugal, a prevalência de DM é elevada quando comparada com outros países, nomeadamente da Europa, ultrapassando em cerca de 4.5% a prevalência média europeia [3]. O Observatório Nacional da Diabetes estima que a prevalência de DM em 2015 foi de aproximadamente 13.3%, em adultos entre os 20 e 79 anos de idade, ou seja, mais de um milhão de portugueses deste escalão etário tinha diabetes. Verificou-se, de acordo com a tendência do resto do mundo, um aumento de cerca de 13.5% na prevalência desta doença desde 2009. Por outro lado, 40.7% da população portuguesa (20-79 anos) tem DM ou hiperglicemia intermédia, o que é uma prevalência alarmante em questão de Saúde Pública e de custos económicos para o Sistema Nacional de Saúde [6].

Uma das principais complicações a longo prazo da DM é a neuropatia periférica que, em conjunto com a doença dos pequenos vasos sanguíneos do pé, são os principais fatores que contribuem para o desenvolvimento das úlceras do *pé diabético*.

As complicações do *pé diabético* associam-se a gastos económicos substanciais e importante perda da qualidade de vida. Por vezes, as lesões evoluem para situações tão graves como infeção, o que pode pôr em risco a própria vida do doente, e amputação por vezes de grandes extensões do pé ou da quase totalidade do membro inferior. Na realidade, estima-se que mais de metade das amputações da perna sejam realizadas em doentes diabéticos e em mais de 70% destes casos as amputações são precipitadas pela progressão de úlceras do pé para infeções gangrenosas profundas [1], [7]. Os doentes diabéticos têm um risco de amputação de qualquer parte do membro inferior 25 vezes maior do que a população geral [3].

É muito importante a prevenção através do rastreio muito precoce de pequenas áreas de isquemia ou de lesão neuropática, mesmo antes do aparecimento de úlceras visíveis [1], [7].

Em Portugal, 1 em cada 4 doentes [8] está em risco de desenvolver *pé diabético* e, em 2015, o número de internamentos relacionados com complicações do *pé diabético* foi de 1643 e o número total de amputações do membro inferior foi de 1250 (545 amputações *major* e 705 amputações *minor*) [6], donde se infere que, apesar dos avanços no seu tratamento médico, estas complicações ainda contribuem para uma grande morbilidade e sofrimento destes doentes. É, por isso, muito importante o seu rastreio muito precoce, para prevenir a sua evolução.

Atualmente, não existe nenhum método objetivo de rastreio de doença periférica diabética (*pé diabético*). Os métodos atuais são subjetivos e dependentes do operador. No entanto, estudos recentes [5] mostram que a monitorização regular da temperatura do pé pode ajudar a diminuir a incidência de lesões ulcerosas deste e amputações.

Em doentes diabéticos, a perfusão vascular periférica, sobretudo dos pés, pode ser indicativa de doença do sistema nervoso autónomo, já que a função vasomotora é regulada pelas pequenas fibras nervosas periféricas do sistema nervoso simpático. A sua disfunção pode associar-se a diferentes padrões de temperatura nestes doentes [9].

Assim, a análise da temperatura do pé tem sido considerada recentemente como um método complementar de avaliação e diagnóstico médico nestes doentes. Vários estudos demonstraram que a monitorização das variações da temperatura dos pés dos doentes diabéticos pode ser uma ferramenta muito útil na identificação precoce das manifestações do *pé diabético*, podendo, também, conduzir a alterações do comportamento destes doentes, o que pode contribuir para reduzir a sua incidência [4], [10], [11].

A termografia infravermelha permite fazer o mapeamento da temperatura da superfície da pele de uma região do corpo humano, pela obtenção de imagens térmicas e pelo estudo da fisiologia da relação entre a temperatura e a perfusão vascular. Com o uso mais regular desta técnica, nomeadamente com a padronização dos valores de temperatura das várias regiões do corpo humano e com os avanços dos sistemas computorizados de obtenção e interpretação de imagens, esta pode vir a tornar-se uma ferramenta de avaliação de um valor indiscutível. A termografia permite obter informações funcionais em tempo real, de fácil acesso e interpretação, não sendo um método dispendioso. Tem a vantagem de ser um método sensível e preciso, não invasivo ou doloroso para o doente e de não emitir radiação, pelo que poderá permitir uma monitorização regular do doente, sem quaisquer riscos associados, através da captura repetida de imagens [12], [13], [14].

A termografia poderá permitir, então, a avaliação e monitorização dos sistemas nervoso autónomo e microvascular periféricos em doentes diabéticos em risco de desenvolver *pé diabético*, pela avaliação das imagens térmicas dos pés com complicações decorrentes da doença quando comparadas com imagens térmicas dos pés contralaterais ou com imagens térmicas padrão obtidas a partir de grupos de controle de indivíduos saudáveis.

Dada a grande prevalência da DM e das complicações do *pé diabético*, nomeadamente em Portugal e a gravidade destas complicações associadas a custos tão elevados quer humanos, quer económicos para o doente e para a sociedade, é de enorme relevância o avanço no aperfeiçoamento tecnológico na obtenção de imagens térmicas, no seu registo e na sua interpretação através de protocolos padronizados. É, ainda, extremamente importante a capacidade de captura dessas imagens a partir de dispositivos móveis e de fácil manuseio, de forma a conseguir uma aproximação entre o clínico e o doente e, de facto, conseguir a sua aplicação e a sua rentabilização nos protocolos clínicos, nomeadamente na instituição de terapêuticas preventivas, na mudança de comportamentos e na eventual elaboração de

normas de orientação clínica fundamentais para a uniformização, o que resultaria numa melhoria de cuidados médicos.

Presentemente, existem algumas aplicações (*Android* e *iOS*) para dispositivos móveis que utilizam a termografia e que permitem obter imagens térmicas, funcionando como uma câmara térmica. No entanto, ainda não existe nenhuma que tenha as características mínimas para uso clínico, nomeadamente que permitam ao utilizador escolher o valor da emissividade da pele (0.98).

Assim, considero este projeto uma enorme oportunidade de investigação e desenvolvimento e é um privilégio poder fazer parte dele.

1.2 - Objetivos

Sendo este um projeto pioneiro, existem várias componentes que podem ser desenvolvidas, havendo espaço significante para a investigação e o desenvolvimento.

Dado que o tempo designado para a realização desta dissertação de mestrado é limitado, os objetivos pretenderam ser realisticamente balizados. Assim, como finalidade, pretende-se definir uma combinação de dados, de modo a guardar as imagens térmicas obtidas de *pé diabético*, sendo para isso imprescindível o desenvolvimento de uma aplicação que capture essas imagens, com uma interface simples e atrativa, acessível aos profissionais de saúde, que poderão beneficiar da sua utilização na prática clínica. Esta ferramenta deve permitir o posicionamento do sujeito a ser examinado, a integração dos dados clínicos, nomeadamente a conjugação das imagens térmicas, na sua totalidade, no conjunto de dados previamente definido e, ainda, o seu envio para um *webservice* remoto.

Para isso, é imperativa a análise do SDK da FLIR ONE para os sistemas operativos *iOS* e *Android*.

Concluindo, é expectável que desta dissertação possa surgir uma ferramenta eletrónica que seja útil no rastreio e seguimento de doentes diabéticos em grande risco de desenvolver úlceras do *pé diabético*, nos postos de atendimento aos doentes, em Portugal.

1.3 - Estrutura do documento

Este documento está estruturado em 6 capítulos principais. O capítulo 1 refere-se à análise introdutória do problema a tratar e é constituído por 3 subcapítulos: Contexto e Motivação, Objetivos deste projeto e Estrutura do presente relatório.

O capítulo 2 diz respeito ao Estado da Arte. É composto por 11 subcapítulos, sendo eles o Estudo da temperatura e termografia de IR, a *Diabetes Mellitus* e *Pé Diabético*, o Uso de imagens térmicas na avaliação do *pé diabético*, os Sistemas operativos *iOS* e *Android*, a FLIR, o Levantamento de aplicações *iOS* e *Android* já existentes no mercado, o Estudo do SDK, o Levantamento das câmaras térmicas adaptáveis a dispositivos móveis, dando ênfase à Câmara da FLIR ONE de 2ª Geração, a base de dados SQLite e *Webservices*.

O capítulo seguinte relata a metodologia adotada para a realização deste trabalho, sendo formado por 5 subcapítulos. São eles a Solução proposta, o Estudo de câmaras térmicas, o Estudo do formato JPG radiométrico, o Desenvolvimento da aplicação *Android* e os Casos de Uso. A Solução proposta é composta por 4 outras secções: a Arquitetura funcional do sistema,

a Especificação do sistema, o Levantamento de Requisitos e a Interface inicial com o utilizador. Por seu lado, o Desenvolvimento da aplicação *Android* é constituído por 4 segmentos: *Work Breakdown Structure* (WBS), Cronologia do desenvolvimento, Processo de inicialização e Processo de desenvolvimento. Este último subdivide-se em cada uma das tarefas presentes no WBS.

O capítulo 4 é referente aos Testes e Resultados. Apresenta, então, os testes efetuados ao longo do projeto e quais os resultados obtidos. Divide-se em 5 subcapítulos: o Estudo de câmaras térmicas, o Estudo do formato JPG radiométrico, a Interface final com o utilizador, os Testes da aplicação *Android* desenvolvida e Aplicações finais (o que resultou de todo o trabalho desenvolvido).

No capítulo 5, apresenta-se a discussão dos resultados obtidos, bem como a resposta aos objetivos iniciais.

Finalmente, o último capítulo diz respeito à Conclusão final e ao Trabalho futuro proposto.

Capítulo 2

Estado da arte

O estudo desta temática iniciou-se pela revisão bibliográfica do tema em questão, utilizando alguns artigos fornecidos pelo seu Orientador e outros que se obtiveram através de motores de busca como o *PubMed* [15], *Scopus* [16] e *ISI of Knowledge* [17].

Fez-se o estudo dos sistemas operativos *iOS* e *Android*, da FLIR, da base de dados *SQLite* e de *webservices* e procedeu-se ao levantamento de aplicações *iOS* e *Android* já existentes no mercado que utilizam a câmara FLIR ONE. Realizou-se, ainda, o estudo do SDK, fornecido pelo fabricante, e a análise das câmaras térmicas adaptáveis a dispositivos móveis existentes, aprofundando o estudo da câmara FLIR ONE de 2ª geração.

Nesta secção procurou-se fazer uma breve súmula introdutória à problemática a tratar nesta dissertação, de modo a constituir as bases teóricas deste estudo.

2.1 - Estudo da temperatura e termografia de IR

A temperatura é uma medida relativa entre dois objetos, que tem por base o nível de agitação das partículas que formam as moléculas de um corpo. Quanto maior o nível de agitação das partículas, maior será a temperatura. Temperatura e calor são conceitos diferentes apesar de estarem interligados [18].

Entre dois corpos, a temperaturas diferentes, em contacto entre si, há uma transferência de energia até ao estabelecimento de um equilíbrio de temperaturas, conhecido por equilíbrio térmico. Assim, por definição, dois corpos têm a mesma temperatura se eles estão em equilíbrio térmico entre si (Lei 0 da termodinâmica), o que nos permite definir uma escala de temperatura. A diferença de temperatura entre dois corpos pode, desta forma, ser expressa pela taxa de transferência de calor [19], [20].

A temperatura de um objeto pode, também, ser estimada sem necessidade de haver contacto, baseando-se na temperatura de um corpo negro, cujos componentes microscópicos são fotões que constituem o campo eletromagnético na sua cavidade [19].

O espetro eletromagnético é uma representação das características de distribuição da radiação eletromagnética emitida ou absorvida por um corpo num determinado comprimento de onda. É constituído por muitos tipos de ondas, nomeadamente os raios gama, raios x, ultravioleta, a luz visível, as ondas infravermelhas, microondas e rádio [19], [21].

A luz infravermelha (IR) situa-se à direita da luz visível, entre esta e as microondas no espetro eletromagnético. Como podemos verificar na figura 2.1, tem uma gama de comprimentos de onda que variam entre *near-IR*, que é a mais próxima do comprimento de onda da luz visível, e o *far-IR* que se aproxima da região das microondas do espetro.

As emissões do corpo humano, quando avaliadas para efeito de diagnóstico clínico, têm comprimentos de onda entre 8 e 13.5 μm, na área das radiações IR [22].

Na figura 2.1 [23], apresenta-se o espetro da radiação eletromagnética, dando ênfase às radiações IR.

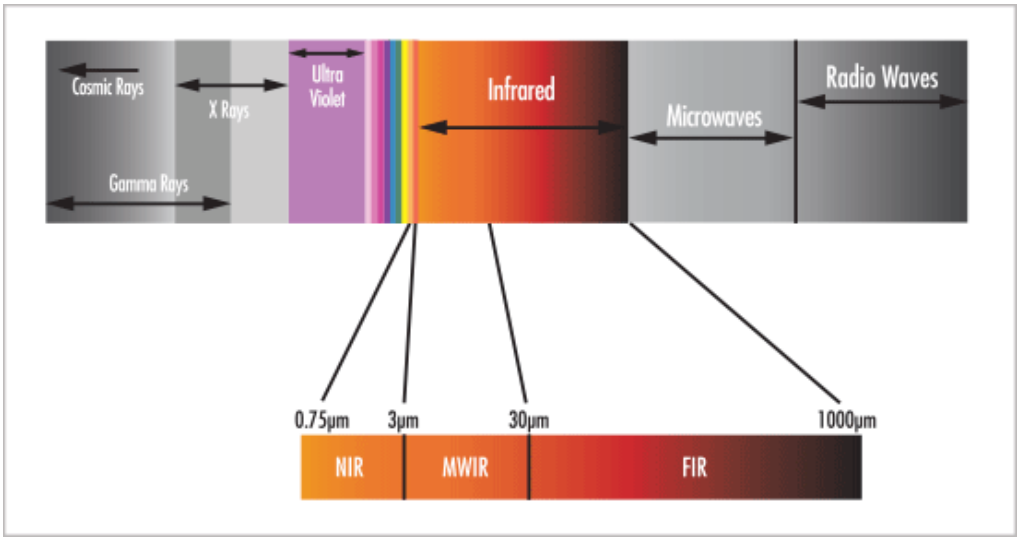

Figura 2.1 - Espetro de radiação eletromagnética - radiações IR

Qualquer corpo com uma temperatura superior ao zero absoluto (0 *Kelvin*) tem a capacidade de emitir radiação térmica. Esta é invisível ao olho humano, mas é sentida sob a forma de calor, sendo, então, uma das formas de transferência de calor [24] , [25].

A radiação representa o mecanismo mais importante de transferência de calor do corpo humano, sendo responsável por aproximadamente 60% das perdas totais de calor, numa situação de repouso. A energia térmica é convertida em energia radiante eletromagnética, que é emitida pelo corpo na banda IR do espetro [26], [27].

Um corpo capaz de absorver toda a radiação eletromagnética incidente e apenas emitir radiação relativa a uma temperatura específica é denominado um corpo negro, ou seja, a emissividade (ε) é igual a 1. Segundo a lei de *Kirchhoff*, a quantidade de energia absorvida é igual à emitida:

$$\varepsilon = a \tag{1.1}$$

Por outro lado, corpos reais não emitem nem absorvem como os corpos negros, que são objetos idealizados. Para além de absorver e emitir, são também capazes de transmitir e refletir energia:

$$\varepsilon + \rho + \tau = 1 \tag{1.2}$$

em que ε, como já foi referido, é a emissividade, ρ é a reflexão e τ é a transmissão, variando os seus valores entre zero e um.

Existem três equações que descrevem a radiação emitida de um corpo negro: a lei de *Planck*, a lei de *Wien* e a lei de *Stefan-Boltzmann*. Em relação à lei de *Planck*, esta informa-nos sobre a quantidade de energia emitida por um corpo negro na radiação de um determinado comprimento de onda ou frequência. Um corpo negro tem a capacidade de absorver toda a radiação sobre ele incidida.

Por outro lado, e segundo a lei de *Wien*, a emissão de radiação de um corpo negro tem um espetro de distribuição que depende apenas da temperatura, sendo que o comprimento de onda máximo emitido é inversamente proporcional à temperatura do corpo negro. Assim, sabendo a forma do espetro de uma temperatura, a forma de qualquer outra temperatura pode ser calculada.

A lei de *Stefan-Boltzmann* demonstra que a energia irradiada total por unidade de superfície de área de um corpo negro, em unidades de tempo (radiação de um corpo negro) é diretamente proporcional à sua temperatura elevada à sua quarta potência:

$$P = \varepsilon \sigma A T^4 \tag{1.3}$$

sendo P a energia irradiada, ε a emissividade, σ a constante de *Stefan-Boltzmann*, A a área e T a temperatura [19], [28].

Segundo esta lei, a energia irradiada pelo corpo humano é diretamente proporcional à temperatura da pele e do fluxo de sangue nos vasos sanguíneos periféricos e, à medida que a temperatura aumenta, ocorre um aumento exponencial da energia total irradiada sob a forma de IR [29].

O estudo da temperatura através da análise das radiações IR tem vindo a ser cada vez mais desenvolvido nos últimos anos e as suas aplicações práticas são múltiplas, nomeadamente nas áreas de medicina e veterinária, astronomia, agricultura, segurança e engenharia.

A informação clínica obtida por termografia baseia-se na distribuição da temperatura à superfície do corpo e na sua resposta térmica [30]. A pele funciona, assim, como interface entre o corpo e o ambiente envolvente.

A temperatura é um indicador fisiológico do corpo humano, podendo a sua variação demonstrar estados patológicos. A temperatura interna do corpo humano normal é 36.8 ± 0.6 °C e, na situação desta se apresentar superior ou inferior, isso pode ser indicativo de uma possível patologia, como, por exemplo, febre [19].

Os mecanismos fisiológicos da distribuição da temperatura na superfície da pele são cada vez mais conhecidos, o que resulta numa maior evidência clínica e uma maior precisão de caracterização na aplicação de imagens térmicas em doenças bem definidas. O valor do mapeamento da temperatura como medida de prognóstico foi também estabelecido. Por outro lado, a temperatura da pele é uma medida indireta do fluxo sanguíneo periférico [31].

Este tem um importante papel na termorregulação (função de manter a temperatura interna do corpo humano constante e dentro de valores normais) e qualquer alteração no seu estado conduz a mudanças na temperatura da pele que podem ser avaliadas.

A termografia médica digital ou imagem térmica de IR médica digital (IRT) é uma modalidade de imagem médica que não pressupõe contacto, sendo não invasiva, não ionizante, portanto segura, rápida e de baixo custo, que permite monitorizar a temperatura da superfície da pele [19], [28].

Nos últimos 20 anos foram feitos progressos notáveis no desenvolvimento de equipamentos de captura de imagem de IR, na padronização da técnica e na criação de protocolos clínicos de avaliação de imagens térmicas. Por outro lado, há cada vez uma maior precisão nesta avaliação e o tamanho e preço dos equipamentos tem vindo a ser cada vez menor, tornando-os mais portáteis e aplicáveis na prática clínica [19], [31] ,[32].

As suas aplicações na clínica são variadas, sendo particularmente útil na avaliação da microcirculação vascular, do sistema nervoso autónomo e em áreas tão amplas como a dermatologia, obstetrícia, reumatologia, oncologia, fisioterapia, cirurgia, saúde ocupacional e saúde pública, entre outras [19], [32]. Uma vez que infeção e inflamação podem causar alterações da temperatura superficial da pele, a termografia de IR pode ser uma ferramenta valiosa na monitorização destas situações clínicas, dado que é um método indolor e inócuo, podendo as imagens térmicas ser obtidas de forma repetida e sequencial [26], [33]. Um termograma IR é uma imagem da distribuição da temperatura da superfície da pele [31].

As imagens térmicas são obtidas a partir de dados radiométricos resultantes do valor de radiação térmica detetado pelos sensores térmicos da câmara, que são transformados em valores de tensão e codificados em valores numéricos de acordo com a resolução de *pixels* e a gama de temperaturas dos sensores. Os pacotes de *software* para análise de imagens térmicas oferecem diferentes escalas de cor falsa para uma melhor compreensão visual, sendo capazes de reproduzir interpretações diferentes, dado as diferentes perceções de cores entre os seres humanos. As escalas de cor falsa permitem uma avaliação subjetiva, enquanto os valores de temperatura média, máxima, mínima e desvio padrão possibilitam uma aferição objetiva [34].

Os principais componentes de uma câmara de IR são as óticas (lentes), que focam a energia IR para o detetor, o detetor IR que converte a energia IR num sinal elétrico proporcional, o obturador (*trigger*) e a unidade de processamento de sinal digital (DSP) que é a componente eletrónica que processa o sinal elétrico para produzir uma figura radiométrica e realiza os cálculos da temperatura.

Há principalmente dois tipos de detetores de IR: os refrigerados (*cooled*) e os não refrigerados (*uncooled*). Os primeiros requerem arrefecimento e normalmente funcionam a temperaturas criogénicas, de modo a reduzir o ruído induzido termicamente a um nível abaixo daquele do sinal do objeto que está a ser analisado. Estes detetores possuem um tempo de utilização limitado, são mais caros e difíceis de utilizar do que os sensores não refrigerados. No entanto, proporcionam uma qualidade de imagem superior e uma maior sensibilidade a pequenas diferenças nas temperaturas do objeto analisado.

Os detetores não refrigerados funcionam à temperatura ambiente ou são estabilizados a uma temperatura relativamente próxima, usando elementos de controlo de temperatura para reduzir o ruído da imagem. Funcionam com base em mudanças na resistência, tensão ou

corrente, quando expostos a radiação IR incidente. Estas alterações são medidas e comparadas com as alterações da temperatura de funcionamento do detetor. Estes detetores produzem imagens de menor qualidade e sensibilidade, quando comparados com os refrigerados. No entanto, a sua qualidade está cada vez mais a aproximar-se dos detetores refrigerados. As câmaras que usam os detetores não refrigerados são menores, mais fáceis de utilizar, com maior longevidade e mais baratas [19].

Não existem câmaras desenhadas especificamente para aplicação na prática clínica, existindo, também, uma lacuna numa aplicação completa de processamento e análise de imagens térmicas IR para os vários cenários clínicos que seja compatível com as várias câmaras de recolha de imagens disponíveis no mercado. No entanto, existem já dispositivos de captura de imagens térmicas, equipados com ligação sem fios, que satisfazem os requisitos mínimos para a prática clínica [32].

2.2 - *Diabetes Mellitus* e *Pé Diabético*

A *Diabetes Mellitus* é um distúrbio metabólico caracterizado pela presença de hiperglicemia. Há diversos tipos diferentes de DM, causados pela interação complexa de fatores genéticos, epigenéticos e de aspetos ambientais.

Vários processos patogénicos podem ser responsáveis pelo aparecimento da doença, desde alterações auto-imunes com a destruição das células ß-pancreáticas produtoras de insulina, como na DM tipo 1, a alterações metabólicas que resultam, em última análise, na resistência periférica à sua ação, como na DM tipo 2 e no síndrome metabólico que se lhe associa. Assim, os fatores que podem contribuir para a hiperglicemia, dependendo da etiologia de DM, incluem a redução da secreção da insulina, a resistência periférica à sua ação, ou a conjugação de ambos, resultando na diminuição da utilização da glucose pelas células e/ou no aumento da sua produção. A desregulação metabólica (metabolismo lipídico, proteico e de carbohidratos), associada a DM, causa alterações fisiopatológicas secundárias em múltiplos órgãos, o que acarreta uma enorme morbilidade nestes doentes [1], [2], [35].

A hiperglicemia crónica associa-se a lesão e disfunção de vários órgãos, nomeadamente olhos, rins, coração, vasos sanguíneos e sistema nervoso autónomo. São complicações clínicas da DM a longo prazo a retinopatia, que pode evoluir para cegueira, e a nefropatia, muitas vezes com falência renal terminal, com necessidade de tratamento dialítico ou transplante renal. A disfunção do sistema nervoso autónomo pode conduzir a doença cardiovascular, gastrointestinal, geniturinária e disfunção sexual. A DM pode associar-se também a maior risco aterosclerótico, de hipertensão e de doença vascular cerebral [1], [7].

O *pé diabético* é a complicação crónica mais comum em doentes diabéticos, e a sua presença relaciona-se com uma maior duração e com um pior controlo metabólico da doença. Em algumas séries de doentes diabéticos estão descritas lesões nos pés em 15 a 25% dos casos [36], [37], [38], [39] e em 2 a 15% noutras investigações publicadas [5], [4], [40].

Em Portugal, calcula-se que cerca de 14% da população é diabética e destes, 1 em cada 4 desenvolve na sua vida a condição de *pé diabético*. [8]

As causas mais importantes que podem conduzir a um maior risco de desenvolver pé diabético e úlcera são a neuropatia, a macroangiopatia (por arteriosclerose) e/ou a microangiopatia. Na presença de um ou mais destes fatores, a existência de um desencadeador como um traumatismo, pode ser o que conduz à formação da úlcera [41].

Esquematicamente, apresentam-se na figura 2.2 os fatores etiológicos do *pé diabético* [42], [43].

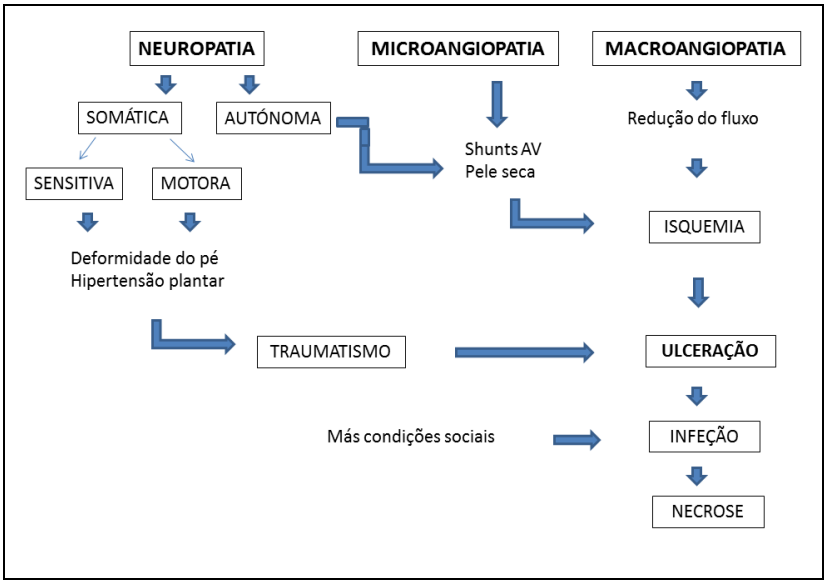

Figura 2.2 - Fatores etiológicos do *pé diabético* (adaptada).

A maioria das úlceras do pé é precedida por pequenos traumatismos, muitas vezes provocados pela pressão que o calçado exerce sobre o pé e não detetadas pelo doente, devido a alterações sensitivas decorrentes da neuropatia diabética e de graus variáveis de doença vascular periférica. Por isso, é tão importante a prevenção através de medidas como a informação proativa dos doentes desta problemática e, sobretudo, pelo diagnóstico muito precoce de pequenas áreas de má perfusão sanguínea e de lesões isquémicas tecidulares, mesmo antes do aparecimento de úlceras visíveis [1], [7].

Os fatores etiológicos primários da doença do pé diabético incluem neuropatia e/ou doença vascular periféricas. Embora ambos estejam presentes, a contribuição de cada um destes fatores é geralmente desigual para cada doente, encontrando-se, na prática clínica, lesões maioritariamente isquémicas, neuropáticas ou neuroisquémicas [44].

A polineuropatia periférica é uma complicação comum em doentes diabéticos, afetando os nervos periféricos das extremidades, nomeadamente do pé, estimando-se a sua presença nos membros inferiores em cerca de 66% destes doentes [38], [45]. A polineuropatia periférica resulta de alterações degenerativas dos axónios, mais fácil nos amielínicos e mais longos e, por isso, mais frequentes nas fibras do sistema nervoso autónomo, que não são mielinizadas e nos membros inferiores e pés [44]. No diabético, a lesão neurológica pode ser sensitiva ou motora (fibras somáticas) ou autonómica ou, na maioria das vezes, surgir em conjugação das três. A lesão dos nervos sensitivos implica a perda de sensibilidade tátil e de pressão no membro inferior, podendo haver lesão dos tecidos decorrentes de excesso de

tensão exercida e mantida sobre o pé. Da neuropatia motora podem resultar alteração da coordenação dos movimentos e atrofias musculares e subsequentes deformidades do pé (por exemplo pé equino ou cabeças metatársicas proeminentes), que também podem ser áreas de conflito de pressão e, consequentemente, de maior suscetibilidade de desenvolvimento de úlceras ou outras lesões [38].

A termorregulação da pele é coordenada e regulada pelo sistema nervoso autónomo, através da atividade vasomotora central e periférica. A degeneração das fibras nervosas simpáticas periféricas pode alterar os mecanismos de controlo neurogénicos que regulam os fluxos capilares e arteriovenosos, levando a um desvio do fluxo sanguíneo da circulação arteriolar para a venular, ultrapassando o leito capilar [46].

Assim, a neuropatia periférica autonómica é, por sua vez, um importante determinante na alteração dos fluxos sanguíneos no *pé diabético*, tendo-se estabelecido uma relação causal entre a disfunção da microcirculação nestes doentes e as complicações do *pé diabético*. A degeneração dos nervos simpáticos periféricos, na neuropatia avançada, poderá alterar os mecanismos de controlo neurogénicos que regulam os fluxos capilares e arteriovenosos, conduzindo a um aumento do *shunt* arteriovenoso no *pé diabético*. Assim, quando há neuropatia autonómica simpática, há maior fluxo sanguíneo arteriovenoso e desvio do fluxo sanguíneo dos capilares cutâneos, havendo alterações na microcirculação, podendo conduzir a lesões isquémicas [38], [46], [40].

Finalmente, a neuropatia autonómica é, de igual modo, responsável por alterações na função das glândulas sebáceas e sudoríparas, tornando o pé mais vulnerável a fissuras e infeções [38].

Assim, as úlceras ocorrem como consequência da interação entre fatores externos ambientais e alterações específicas dos membros inferiores de certos doentes diabéticos. Fatores pessoais decorrentes da doença como a microangiopatia e a neuropatia autonómica, em conjugação com alguma perda de sensibilidade tátil e dolorosa, altas pressões nos pés (muitas vezes por provocadas por calçado desadequado), deformidades locais, pequenos traumatismos e uma maior suscetibilidade às infeções, conduzem a uma maior vulnerabilidade e predisposição para o desenvolvimento de úlceras no pé, especialmente em locais sujeitos a uma maior tensão, como as cabeças metatársicas mais proeminentes. Por outro lado, as úlceras neuropáticas clássicas ocorrem mais frequentemente nas plantas dos pés [40].

As complicações decorrentes do *pé diabético* podem ser muito graves, com grave risco de evoluir para a morte ou para a amputação de segmentos variáveis das extremidades inferiores, sendo a causa principal de amputações nestes doentes [5], [37], [39], [40], [47]. Estima-se que possa ser responsável por cerca de 85% das amputações [5], [48].

Assim, é de uma enorme importância a identificação precoce e tratamento imediato das complicações do *pé diabético* e dos sinais que as antecedem, como a inflamação, a formação de calos, a formação da úlcera, para prevenir e diminuir a incidência destas consequências devastadoras. Em muitos casos, o desenvolvimento e a rápida deterioração das lesões ulcerosas podem ser evitados, ou substancialmente atrasados, com o tratamento adequado em fase precoce da doença [4], [37], [40].

A deteção precoce depende, deste modo, da avaliação frequente e minuciosa destes doentes, o que pode não ser muito fácil por razões várias. Por um lado, o auto-exame pode ser difícil devido às limitações da própria doença ou por motivos sociais e o exame frequente

por profissionais de saúde pode ser muito dispendioso e intrusivo na vida do doente, podendo aumentar o absentismo laboral.

A avaliação da neuropatia periférica é feita, mais frequentemente, através do método tradicional do monofilamento de *Semmes-Weinstein* (Figura 2.3 [49]) para avaliar se a perceção tátil e de pressão estão ou não preservadas, mas este é um método que não é completamente objetivo e não deteta lesões precoces que possam evoluir para úlcera do pé [38], [40].

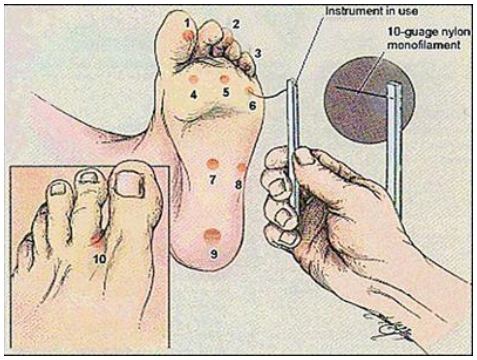

Figura 2.3 - Monofilamento de *Semmes-Weinstein*.

Para avaliar a presença de doença vascular periférica é útil a palpação dos pulsos das artérias pediosa e tibial posterior.

A análise térmica do pé pode ser útil, já que alterações da temperatura do pé são um sinal chave de inflamação e/ou de presença de neuropatia e microangiopatia periféricas [39].

Vários estudos [50], [51], [52] mostram uma associação entre o aumento da temperatura do pé e a presença de complicações do *pé diabético*. Esse aumento da temperatura pode estar presente mais do que uma semana antes do aparecimento da úlcera no pé. Neste estadio precoce da lesão, os doentes muitas vezes não sentem dor, devido à perda da sensibilidade, consequência da neuropatia, pelo que o aumento da temperatura do pé pode ser um útil e único sinal preditivo de ulceração e de inflamação subclínica [4], [38].

Por outro lado, as lesões isquémicas associam-se a temperaturas mais baixas, em virtude de ausência de fluxo sanguíneo [53].

Deste modo, as úlceras neuropáticas caracterizam-se por se localizarem em pontos de maior pressão, serem indolores, se associarem a pés quentes e pulsação periférica palpável. Por seu lado, as úlceras isquémicas aparecem maioritariamente nas extremidades, em pés frios, com dor e sem pulsação periférica palpável [54].

Assim, a monitorização da temperatura do pé na avaliação de rotina destes doentes, permite identificar precocemente complicações do *pé diabético* e alterar comportamentos, quer por parte dos profissionais de saúde, quer por parte dos doentes, o que contribui para diminuir a sua incidência. Deste modo, o mapeamento da temperatura do pé pode disponibilizar informações valiosas adicionais aos exames complementares de diagnóstico clássicos. No entanto, há que ter em conta que não existem valores de referência

standardizados de controlo da distribuição da temperatura do pé saudável, sendo a temperatura dos pés variável de pessoa para pessoa e dependente de fatores exógenos como a temperatura ambiente, ou endógenos como as condições térmicas e de metabolismo internos, idade, género, peso, entre outros [5], [4]. A variabilidade destes fatores pode ser ultrapassada pela análise comparativa da temperatura entre os dois pés, sendo a referência mais utilizada em ensaios clínicos a diferença de temperaturas em áreas sobreponíveis dos pés homo e contralaterais. Essa diferença de temperaturas não deve ultrapassar 1ºC, sendo que diferenças superiores a 2.2ºC são consideradas alteradas e patológicas [4], [47].

Em súmula, a *Diabetes Mellitus* é uma doença com uma enorme prevalência quer a nível mundial, quer em Portugal e a sua incidência tem vindo a aumentar de forma significativa. As complicações do *pé diabético* que se lhe associam conduzem a uma enorme morbilidade ou mesmo mortalidade nestes doentes. É, por isso, fundamental estar atento a alterações muito precoces no desenvolvimento destas complicações que permitam o diagnóstico e tratamento atempados. A monitorização da temperatura do pé pode ser uma ferramenta muito útil na identificação das primeiras lesões, mais subtis, invisíveis ao olho humano e difíceis de valorizar objetivamente pelos protocolos clínicos clássicos.

2.3 - Uso de imagens térmicas IR na avaliação do *pé diabético*

Nos últimos anos, vários trabalhos têm sido descritos na literatura em que se faz referência à utilização da termografia de IR, no sentido de realizar o mapeamento térmico dos pés de doentes diabéticos para rastreio precoce das complicações de *pé diabético* [5], [4], [37], [38], [39], [40], [46], [55], [56].

Idealmente o uso da termografia de IR para análise da temperatura pode ser uma ferramenta muito proveitosa para prevenir as úlceras no pé. Apesar disto, esta tecnologia não é ainda amplamente utilizada na prática clínica, nos protocolos de abordagem de diagnóstico destes doentes.

Lavery, et al. (2004 e 2007) e *Armstrong, et al.* (2007) em 3 estudos randomizados, controlados e cegos para os clínicos indicaram que a monitorização da temperatura do pé limita de forma significativa as taxas de reulceração na diabetes. Nestes ensaios foi usado um termómetro de IR (*Temp Touch Dermal Thermometer*®) para autoavaliação do pé [4], [47], [57].

Nos ensaios clínicos de *Lavery, et al.* (2007) e de *Armstrong, et al.* (2007) foram avaliados 173 e 225 doentes diabéticos, respetivamente, com risco de desenvolver *pé diabético* e com história de prévia úlcera no pé. As temperaturas de várias diferentes regiões de interesse do pé (*hallux*, primeira, terceira e quinta cabeça metatarsiana, região média da planta do pé e calcanhar) foram autoavaliadas diariamente e registadas num diário. No caso da diferença de temperatura de áreas sobreponíveis dos pés ser superior a 2.2ºC, foi recomendado ao doente diminuir a sua atividade e contactar a sua enfermeira de imediato.

Os doentes com diabetes e alto risco de ulceração foram divididos em 2 grupos: os com terapêutica padronizada habitual e os com terapêutica padronizada mais autoavaliação por termometria de IR. Em ambos os ensaios os grupos de doentes com terapêutica padronizada foram significativamente mais propensos a desenvolver úlceras no tempo de estudo (15-18 meses) do que os outros doentes com avaliação suplementar das temperaturas do pé. Os resultados dos estudos mostram mais de 60% de redução de formação de úlceras [4].

Van Netten, et al. (2013) [39] num estudo piloto com 15 doentes com *Diabetes Mellitus* tipo 1 ou tipo 2, usando uma câmara de IR (resolução de 1.2mm/*pixel*) obtiveram imagens térmicas da superfície plantar do pé destes doentes. Não encontraram diferenças superiores a 1.5ºC na temperatura média entre o pé homo e contralateral, nos doentes sem complicações de *pé diabético*. No grupo de doentes com complicações locais, as temperaturas médias dos pés homo e contralateral foram semelhantes, mas a temperatura nas regiões específicas de lesão foi > 2ºC em relação com a região correspondente do pé contralateral e com a média de todo o pé homolateral. No grupo de doentes com complicações difusas encontraram diferenças térmicas > 3ºC entre os pés homo e contralateral. Concluíram que com uma câmara de IR de alta resolução é possível distinguir, pelas imagens térmicas da planta do pé, entre doentes sem complicações e com complicações locais e difusas de *pé diabético*.

De igual modo, *Luciane Balbinot, et al.* (2013) [46] foram responsáveis por um estudo piloto que decorreu durante 4 meses, realizado com 10 doentes diabéticos recrutados num departamento de endocrinologia de um Hospital do Brasil e com um grupo de controlo com 10 indivíduos saudáveis. Através de uma câmara de IR (*IRISYS®, model IRI 4010, UK*) obtiveram imagens térmicas da planta do pé, pré e 10 minutos após teste de *stress* com o frio, em 2 dias diferentes, com 7 dias de intervalo. A média das temperaturas absolutas da região plantar do pé pré (p=0.033) e pós-*stress* do frio (p=0.019) diferiu entre os 2 dias, nos doentes não diabéticos, sendo que não foi diferente nos doentes com diabetes. O índice de reaquecimento após *stress* do frio apresentou boa repetibilidade entre os 2 dias, com uma semana de intervalo, nos 2 grupos. Concluem que, apesar de os doentes diabéticos não apresentarem diferenças nos valores de temperatura absoluta (temperaturas médias), medidas térmicas como a variação de temperatura e o índice de reaquecimento após stress do frio, são recomendadas para fins clínicos, apoiando a sugestão de autores prévios.

Liu, C, van Netten, et al. [37], publicaram, em 2015, um estudo cujo objetivo era desenvolver meios tecnológicos que permitissem a avaliação automática do pé para detetar complicações do *pé diabético*. Esta tecnologia, usando a termografia IR, poderia ser aplicada num sistema inteligente de telemedicina a ser desenvolvido nas casas dos doentes, centros de saúde e hospitais. Referiram que as diferenças de temperatura entre áreas correspondentes dos pés contralaterais parecem ser os parâmetros clínicos mais significativos a serem avaliados e valorizados. No entanto, esta análise de assimetria térmica é prejudicada por erros de segmentação do pé, em particular quando a temperatura do pé e a temperatura ambiente são semelhantes e, também, dadas as formas e tamanhos diferentes entre os pés, devido a deformações ou amputações *minor*. Para ultrapassar o primeiro problema usaram uma imagem RGB e outra térmica adquiridas simultaneamente. As regiões do pé detetadas pelas imagens RGB, foram rigorosamente registadas sobre as suas imagens térmicas. Isto resultou numa sensibilidade de 97.8 ± 1.1% e uma especificidade de 98.4 ± 0.5%, em mais de 76 diabéticos de alto risco, tendo como referência anotações manuais sobre estes doentes. Tentou-se ultrapassar o segundo problema através de registos não rígidos do contorno dos pés através de funções *B-spline*. Assim, áreas correspondentes nos dois pés puderam ser obtidas, e as diferenças de temperaturas entre os pés direito e esquerdo avaliadas,

independentemente das formas e tamanhos dos pés. No entanto, os autores reforçam que a limitação por análise assimétrica das complicações do *pé diabético* é que esta só é possível por comparação dos dois pés de cada doente. Isto significa que, quando um dos pés é amputado, qualquer complicação do outro pé não pode ser detetada por comparação. Acresce, ainda, que se os dois pés tiverem complicações semelhantes, estas podem não ser diagnosticadas. Outra limitação deste estudo é que não há um *gold standard* para avaliação da segmentação e registo do pé. Foram usadas referências registadas manualmente por um investigador, o que pode introduzir um viés ao estudo, dado que outro investigador poderia obter e registar diferentes observações.

Vardasca, et al. [53], num ensaio clínico publicado em 2015, fizeram a monitorização da distribuição da temperatura da planta do pé através da termografia médica. Foram avaliados 50 doentes diabéticos, 30 com lesões neuropáticas, 2 com lesões isquémicas e 18 com lesões neuroisquémicas. Foram definidas as regiões de interesse nas localizações, na planta do pé, mais frequentes de aparecimento deste tipo de úlceras ilustradas na figuar 2.4 [53].

Figura 2.4 - Definição das regiões de interesse da planta do pé em diabéticos.

As imagens térmicas foram obtidas em temperatura ambiente a 22 ± 1°C e com humidade relativa <50%, após os doentes estarem num período de equilíbrio térmico de 15 minutos. Foi usada uma câmara térmica FLIR A325sc, com um tamanho de matriz de sensores de 320x240, *Non-Equivalent Temperature Difference* (NETD) < 70mK a 30°C e uma rastreabilidade de medição de ± 2% da gama de leitura, montada num tripé, a 70cm dos pés dos doentes. Os resultados obtidos, para cada um dos três tipos de pé diabético e para cada região de interesse da planta do pé, mostraram que: os casos isquémicos apresentaram maior simetria térmica nas regiões A2, A3, A4, A6 e A8; os casos neuropáticos nas regiões A1 e A7 e os neuroisquémicos nas regiões A9 e A10.

Na figura 2.5 [53], apresenta-se um exemplo de imagem térmica de cada tipo de pé diabético.

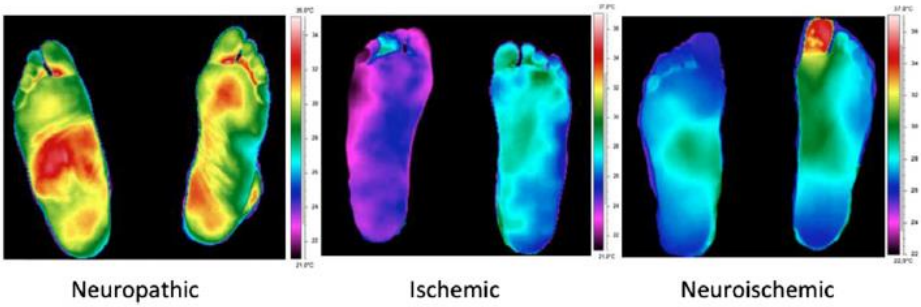

Figura 2.5 - Imagens térmicas dos 3 tipos de pé diabético.

Neste estudo, as informações obtidas pela termografia foram relacionadas com a fisiologia de cada doente, o que permitiu uma avaliação fisiológica em tempo real. O uso da termografia confirmou a influência de fatores como o tipo de diabetes, o nível de glicose sérica, a idade, o IMC na apresentação clínica e os autores afirmam que esta metodologia de diagnóstico pode ser valiosa e aplicada na prática clínica.

T Kananza, et al. (2016) [55] publicaram um estudo piloto transversal e observacional em que participaram 8 doentes recrutados na consulta de *pé diabético* (5) e de úlceras de pressão (3) de um Hospital Universitário de Tóquio, em janeiro de 2015. Foi feita a avaliação da imagem térmica dos seus pés com o recurso ao dispositivo FLIR ONE, 1ª geração, conectado a um *smartphone*. O intervalo de medidas do aparelho variava entre 0ºC e 100ºC, com um intervalo de erro indefinido e um NETD de 0.1ºC e um tamanho de matriz de sensores de 80x60. A termografia de IR usada por rotina era *Thermo Tracer TH700N (Nippon Avionics Co.,Ltd.,Tokio, Japan*), que é um dispositivo manual de termografia de topo. Este dispositivo foi usado para fornecer os dados de referência. O coeficiente de *Coehen´s Kappa*, com intervalos de 95% de confiança, foi usado para avaliar a validade de critério e confiabilidade inter e intra-avaliador de avaliação por imagem térmica. Foi feita a comparação entre as imagens térmicas obtidas com o FLIR ONE e com o *Thermo Tracer TH700N*. Nas 16 imagens térmicas (figura 2.6 [55]) obtidas de 8 doentes, o coeficiente de *Kappa* para cada valor foi o seguinte: para o intervalo predeterminado e para o intervalo de alcance automático, respetivamente, a validade de critério foi 1.0 (95% intervalo de confiança 1.00-1.00) e 1.0 (95% intervalo de confiança 1.00-1.00); a confiabilidade entre avaliadores foi 1.0 (95% intervalo de confiança 1.00-1.00) e 1.0 (95% intervalo de confiança 1.00-1.00); a confiabilidade intra-avaliadores foi 1.0 (95% intervalo de confiança 1.00-1.00) e 1.0 (95% intervalo de confiança 1.00-1.00).

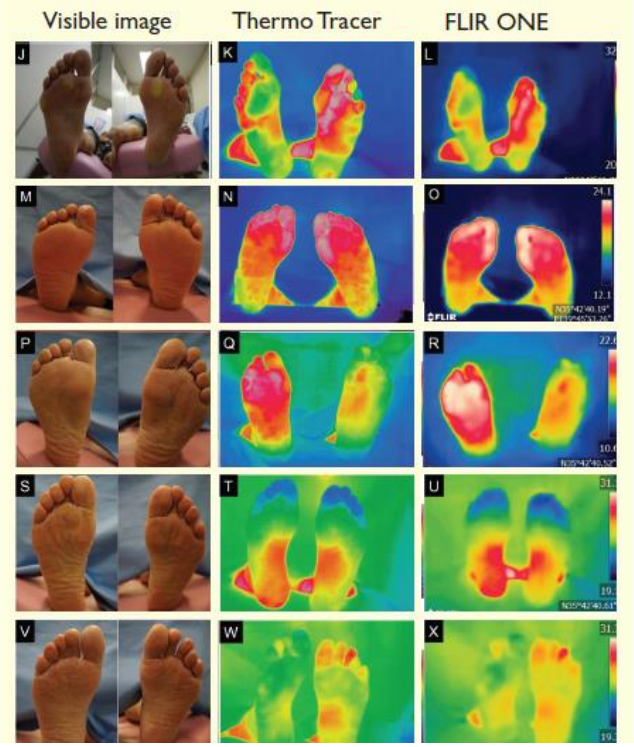

Figura 2.6 - Imagens térmicas e visíveis de pés de doentes diabéticos obtidas usando a Thermo Tracer TH700N e a FLIR ONE num intervalo predefinido.

Apesar da FLIR ONE poder não indicar a temperatura absoluta com uma acuidade adequada, neste contexto, a diferença de temperatura relativa foi suficiente para avaliar a inflamação. Concluíram, assim, que o estudo sugere que o FLIR ONE pode ser um dispositivo alternativo para avaliar a inflamação subclínica nas úlceras de pressão e no *pé diabético*, nos protocolos de avaliação clínica, superando os dispositivos clássicos de termografia pela portabilidade (apenas 110g) e pelo custo (cerca de $200, ou seja, aproximadamente 178.5€). A grande limitação deste estudo é tratar-se de uma amostra pequena, o que pode afetar a sua relevância clínica no que diz respeito à eficácia de diagnóstico deste dispositivo.

Apesar de vários trabalhos documentarem que o uso de imagens térmicas de IR pode ser promissor como meio de diagnóstico de *pé diabético* em fase muito precoce, são necessários mais estudos clínicos bem desenhados e documentados para validar esta técnica para este propósito. Importante seria também validar o uso de câmaras térmicas acopladas dispositivos móveis do cuidador e/ou do profissional de saúde, o que seria uma mais-valia adicional, quer em termos de custo, quer em termos de portabilidade e aplicação prática.

2.4 - Sistema operativo *iOS*

Este sistema foi desenvolvido em 2007 pela *Apple Inc* para dispositivos móveis, podendo apenas ser utilizado em aparelhos móveis da *Apple*® como *iPhones*, *iPads* e *iPods*. A sua versão atual é a 10.3.

A interface com o utilizador é concebida no sentido de se tornar muito simples e intuitiva, através do toque no ecrã, sendo pioneiros no "gesto de pinça" para aumentar e diminuir a imagem. Utilizam-se acelerómetros e giroscópios embutidos no dispositivo para,

por exemplo, manter a estabilidade das imagens e rodar a orientação destas, rodando o aparelho para a orientação desejada [58], [59], [60].

A linguagem utilizada para o desenvolvimento de aplicações neste sistema operativo é a *Objective-C*. Esta é um superconjunto da linguagem C que é orientada a objetos e fornece um tempo de execução dinâmico. Desta forma, herda a sintaxe, tipos primitivos e instruções de controlo de fluxo de C, adicionando sintaxe para definir classes e métodos. Existe ainda uma outra linguagem de programação para *iOS*, mais moderna e recentemente desenvolvida, chamada *Swift*. Esta baseia-se no melhor da linguagem C e *Objective-C*, sem as restrições de compatibilidade de C, adotando padrões de programação seguros e adicionando recursos modernos para tornar a programação mais fácil e flexível [61], [62], [63].

A *Apple*® disponibiliza um SDK próprio, lançado em 2008, com funções e bibliotecas por si produzidas, possibilitando, assim, o desenvolvimento de aplicações concebidas para uso específico e particular.

No entanto, existem requisitos mínimos para se conseguir desenvolver aplicações em *iOS*. Em primeiro lugar, é necessário utilizar um computador Mac com processador Intel, com o sistema operativo *Mac OS X10.7 – LION* (ou outra versão mais recente). É indispensável a criação de uma conta na *Apple* de *iOS Developer*. É preciso instalar o *Xcode*, uma ferramenta fundamental para o desenvolvimento de aplicações para os dispositivos *iOS*, que possui um simulador virtual e fazer o *download* do *iOS* SDK, cuja versão atual é a 8. Por último, é essencial estar familiarizado com uma linguagem de programação, como por exemplo a linguagem C, uma vez que *Objective-C* tem por base esta linguagem. É de notar que se o desenvolvedor escolher utilizar linguagem *Swift*, para desenvolver a sua aplicação, é na mesma importante ter conhecimento de uma linguagem de programação como a C.

É de salientar que a *Apple*® disponibiliza a possibilidade de desenvolver aplicações individualizadas, porém, para as testar em dispositivos móveis é necessário pagar uma subscrição e a aplicação tem de passar pela certificação da *Apple Store*, no caso de se desejar obter um certificado [62].

2.5 - Sistema operativo *Android*

A *Android Inc* foi criada em 2003 por *Andy Rubin*, *Rich Miner*, *Nick Sears* e *Chris White*, mas em 2005, a *Google* adquiriu-a, sendo que o sistema operativo *Android* é atualmente suportado e desenvolvido pela *Google*. A sua versão atual é a 7.1, existindo já uma versão 8.0.0 em fase de avaliação.

É baseado no núcleo do Linux e o seu foco de desenvolvimento são os *smartphones* e os *tablets*, que são constituídos por ecrãs táteis. A interface com o utilizador é, da mesma forma que para o sistema operativo *iOS*, gerada com o intuito de se tornar muito simples e intuitiva.

Tem a vantagem do desenvolvimento e teste, em simuladores ou em dispositivos móveis, de aplicações em *Android* ser gratuito e acessível a qualquer utilizador. Para isso, é

necessário fazer o *download* do *Android Studio*, acedendo, por exemplo, a https://developer.android.com/studio/index.html. Esta é a ferramenta de desenvolvimento mais usada, no entanto há outras. O *Android Studio* é o IDE (ambiente de desenvolvimento integrado) oficial do *Android*, foi criado pela *Google* e a sua versão atual é a 2.3.2. Este tem um ambiente de desenvolvimento muito simples e intuitivo e oferece ferramentas para edição, depuração, testes e criação de perfis de código e a linguagem utilizada é a *Java*. Este contém um SDK, com funções e bibliotecas desenvolvidos pela mesma organização que é crucial para a criação de aplicações [64], [65].

2.6 - FLIR

A FLIR® (*Foward Looking Infrared*) foi criada em 1978, nos Estados Unidos da América, para promover o desenvolvimento de sistemas de imagens de infravermelhos (térmicas) de alto desempenho e baixo custo para aplicações aéreas. As suas câmaras de infravermelhos permitem que o operador veja na escuridão ou em clima adverso.

Em 1990 associou-se ao grupo industrial da *Hughes Aircraft Co.*, de que resultou um avanço tecnológico nas suas câmaras térmicas, com qualidade de imagem superior e também a capacidade de detetar e medir mais pequenas diferenças de temperatura.

Em 2003, a FLIR® adquiriu a *Indigo Systems*, um dos principais desenvolvedores e fornecedores de uma ampla gama de produtos de imagens de infravermelhos, incluindo detetores infravermelhos *cooled* e *uncooled*.

Desde então, a FLIR® investiu em inúmeros mercados, tecnologias e produtos adjacentes para ampliar o seu conjunto de soluções de sensores e sua capacidade de atender a um conjunto mais amplo de clientes. Esses investimentos permitiram um crescimento significativo das receitas e dos volumes unitários vendidos, o que ajudou a reduzir o custo e, portanto, os preços dos produtos, o que aumentou muito o número de clientes. Possibilitou, também, uma maior sensibilização para as vantagens da tecnologia das câmaras térmicas.

A FLIR® Systems, Inc. projeta, desenvolve, fabrica, comercializa e distribui tecnologias que melhoram perceção que têm aplicações práticas diversas. Trazem soluções de deteção inovadoras para o quotidiano, através dos seus sistemas de imagens térmicas e visíveis, sistemas de localização, sistemas de avaliação e rastreio (nomeadamente na clínica) e sistemas avançados de deteção de ameaças.

Os seus produtos melhoram a forma como as pessoas interagem com o mundo, permitem aumentar a segurança pública e o bem-estar, bem como a eficiência energética, conduzindo a comunidades saudáveis e divertidas [66].

2.7 - Levantamento de aplicações *iOS* e *Android* já existentes no mercado

Como ponto de partida, foram pesquisadas as aplicações que utilizam a câmara FLIR ONE existentes no mercado quer para *iOS*, quer para *Android*. Foi realizado, para o efeito, um levantamento referindo características, organizado por plataforma, ano, tipo, desenvolvedor e compatibilidade.

Posteriormente, organizaram-se de forma única, esquemática e mais intuitiva todas as aplicações encontradas, definindo para cada aplicação as suas características próprias, que se registaram na secção dos Anexos (Anexo A).

Não existe nenhuma aplicação criada para uso clínico, com o objetivo de obter imagens térmicas que possam ser analisadas e integradas nos protocolos clínicos de avaliação dos doentes.

Na pesquisa foram encontradas algumas aplicações interessantes, mas que foram desenvolvidas com fins lúdicos ou artísticos e das quais poucas características poderiam ser utilizadas numa aplicação clínica. Um exemplo é a aplicação *GrafHEATi* [67]. Em vez das latas de graffiti, com uma câmara FLIR ONE e com um secador de cabelo, a aplicação permite que o utilizador crie um desenho, num fundo escolhido, possibilitando a sua edição posterior para aperfeiçoamento. Para além de guardar a imagem na galeria de imagens do dispositivo móvel, não possui nenhuma característica que possa ser aproveitada para a aplicação que se pretende desenvolver.

As aplicações relevantes que encontrou para *iOS* foram as seguintes: *FLIR ONE* [68], *FLIR Tools* [69], *FLIRExperience* [70], *Vernier Thermal Analysis for FLIR ONE* [71], *Thermal Camera Burst Mode* [72], *ThermoPro: Thermal Imaging Reporting App* [73] e *Thermal Preso* [74].

Para *Android*, as aplicações mais pertinentes foram as seguintes: *Remote Thermal Cam f. FLIR ONE* [75], *Thermal Camera For Flir One* [76], *Thermal Camera + For Flir One* [77], *FLIR ONE* [78], *FLIR Tools Mobile* [79], *HeatReview For FLIR ONE* [80] e *DiveFLIR* [81].

A aplicação que estudou em maior profundidade foi a *FLIR ONE*, dado que é a aplicação oficial dos fabricantes da câmara térmica a partir da qual o trabalho vai ser desenvolvido. Está disponível tanto para o sistema operativo *Android* como para o *iOS*. É gratuita e está disponível em vários idiomas. Transforma o ecrã do dispositivo numa câmara térmica, mas para isso é necessário conectar a câmara acessória, de segunda geração, sendo que é possível utilizar a câmara de primeira geração nos *iPhones 5* e *5s*. Tem como principais características a capacidade de ver no escuro ou através do fumo, a disponibilidade de imagens com diferenças mínimas de temperatura, a visualização de fontes de calor invisíveis ao olho humano e a comparação de temperaturas relativas. Permite captar imagens ou vídeos em direto, sendo possível carregá-los diretamente para a galeria de imagens ou até partilhá-los nas redes sociais. Após a captura de uma imagem, é possível vê-la diretamente no ecrã principal da aplicação, sem ter de se aceder à galeria do *smartphone* ou *tablet*. Oferece uma interface com o utilizador simples e intuitiva e permite alterar a paleta de cores falsas a usar. Possibilita, também, a alteração do valor da emissividade, sendo que só são permitidos quatro valores: Mate (0.95), que é o valor que recomenda, Semimate (0.80), Semibrilhante (0.60) e Brilhante (0.30). Captura imagens e vídeos em formato *Multi Spectral Dynamic Imaging (*MSX), propriedade da FLIR®, ou seja, uma imagem com as imagens visível e térmica combinadas, não permitindo captar imagens apenas térmicas ou apenas visíveis. Viabiliza a mudança da orientação da imagem e a unidade de temperatura. Por último, permite guardar a localização GPS onde a imagem/vídeo foi capturada e calibra a imagem automaticamente para garantir qualidade e precisão do valor de temperatura obtido.

A FLIR® tem ainda uma outra aplicação chamada *FLIR Tools*, que é gratuita e está disponível em vários idiomas e para os sistemas operativos *Android* e *iOS*. Esta aplicação apresenta uma imagem térmica em direto e capta fotografias e vídeos, permite importar imagens da câmara de IR, alteras as ferramentas de medição diretamente na imagem e realizar *zoom*. A câmara da FLIR® pode ser controlada por ligação sem fios (acesso remoto) e as imagens/vídeos capturados são automaticamente guardados na galeria de imagens. Dá informação sobre as temperaturas medidas, permitindo a sua análise e fornece as coordenadas GPS da imagem. Permite apagar imagens diretamente no *smartphone* ou *tablet* e ainda criar imagens em papel e relatórios que podem ser enviados por *email*. A impressão das imagens é compatível com qualquer impressora *AirPrint* e é possível alterar a paleta de cores falsas da imagem. A aplicação possibilita, também, a reprodução de comentários de voz e mostra informações da imagem, como por exemplo, comentários ou informações do ficheiro. Permite guardar imagens em serviços de partilha e tem registo e *login* obrigatórios. Viabiliza o controlo da emissividade, humidade relativa e temperatura refletida e atmosférica.

Para o sistema operativo *iOS*, existe a aplicação *FLIRExperience* que está disponível em vários idiomas, é gratuita e inclui visão de 360º. Explica como funcionam as imagens térmicas e dá acesso a ficheiros da FLIR® com características exclusivas.

A aplicação *Vernier Thermal Analysis for FLIR ONE* está disponível apenas para o sistema operativo *iOS*, é gratuita e em inglês. Permite marcar até quatro localizações ou regiões numa imagem térmica e exportar os dados para a aplicação *Graphical Analysis* para análise posterior. Pode determinar-se a temperatura média, máxima ou mínima numa região selecionada e fornece dados de temperatura durante um ensaio. As imagens e vídeos são guardados na galeria de imagens e é possível analisar as alterações de temperatura na pele, ilustrar convecção e mostrar calor devido à fricção. A aplicação compara a condução do calor em diferentes materiais e, também, a transparência dos materiais sob IR e luz visível.

Thermal Camera Burst Mode é outra aplicação existente para o sistema operativo *iOS* e está disponível em inglês, mas tem um custo de $1.99 (aproximadamente 1.83€). Pode ser usada com a câmara FLIR ONE de primeira geração no *iPhone 5* ou *5s* ou com a de segunda geração. Permite captar imagens térmicas e alterar a paleta de cores falsas.

Ainda para o *iOS*, existe a aplicação *ThermoPro: Thermal Imaging Reporting App*. Esta tem um custo de 79.99€ e está disponível apenas em inglês. Obtém imagens térmicas diretamente da câmara da FLIR ONE, mostra a temperatura mais alta e mais baixa na imagem em tempo real e cria relatórios instantâneos com imagens de IR, localização GPS, notas e assinaturas. Permite, ainda, editar a imagem no relatório e desenhar sobre ela, enviar por *email*, imprimir abrir ou guardar o relatório com um só toque e guarda automaticamente e de forma contínua as edições de imagens.

A última aplicação encontrada para *iOS*, com características relevantes para este projeto chama-se *Thermal Preso* e custa $69.00 (aproximadamente 63.48€). Está disponível em inglês e inclui instruções de definições e transmissões. Requer a câmara FLIR ONE, sendo que é compatível quer com a câmara de primeira geração, para *iPhone 5* e *5s*, quer com a de segunda geração. Permite captar imagens térmicas e transmitir vídeos de análise térmica para a *Apple TV*. Fornece dados de temperatura durante um ensaio e exporta para a galeria de imagens os vídeos e imagens capturados. Permite a opção de rotação vertical ou horizontal e, ainda, imagens em espelho.

A *Remote Thermal Cam f. FLIR ONE* é uma aplicação *Android*, gratuita, que envia um fluxo MJPEG da câmara térmica FLIR ONE para o computador, através de rede sem fios. Permite mostrar os valores de temperatura máxima, média e mínima e requer um servidor *SmartCam* para *Windows*. Foi criada com base na aplicação exemplo da FLIR ONE.

Para o sistema operativo *Android* existe a aplicação *Thermal Camera For Flir One*, que é gratuita. Requer a conexão do dispositivo FLIR ONE ou S60 (*smartphone* com câmara térmica integrada), permite regular manualmente a escala de cores falsas a utilizar e capta imagens em formato MSX. Tem maior precisão para os dados de calibração, permite o bloqueio da faixa de temperatura e a visualização desta no ecrã.

A aplicação anteriormente mencionada tem uma versão paga, com um custo de 3.29 € que se chama *Thermal Camera + For Flir One*. Para além de todas as características encontradas na versão gratuita, esta aplicação tem galeria de imagens térmicas, permite resolução *multi-frame*, a alteração do modo de nitidez nas configurações e esconder a cor real da imagem. Corrige o *offset* MSX e permite guardar imagens de cor real, MSX e Non-MSX.

A *HeatReview For FLIR ONE* está disponível para *Android*, é gratuita e permite encontrar a temperatura máxima, média e mínima de uma região selecionada. Guarda a imagem captada na galeria de imagens do *smartphone* ou *tablet* e permite a partilha da imagem em qualquer rede social. É, também, possível fazer comentários na imagem.

Por último, *DiveFLIR* é uma aplicação para o sistema operativo *Android*, gratuita, que permite obter a imagem térmica sobreposta na imagem visível e ver a temperatura média do centro em graus *Celsius* e *Fahrenheit*.

Concluiu-se que não existe, atualmente, nenhuma aplicação no mercado com todas as características desejadas para a avaliação do risco de *pé diabético*.

Assim, foi construída uma base de estudo, nomeadamente de interfaces com o utilizador, e foi aberto o caminho para o que poderá ainda ser explorado.

2.8 - Estudo do SDK

O SDK (*Software Development Kit*) da FLIR ONE consiste no pacote de *software*, criado pela FLIR, que é incluído no projeto. Este contém as bibliotecas e API que permitem a criação do código para manipulação da câmara FLIR ONE por um dispositivo móvel.

O objetivo do estudo do SDK foi procurar saber quais as funções que este tem disponíveis, nomeadamente, se existem funções que permitam tirar a imagem, definir intervalos de temperatura, definir emissividade, definir intervalos de focagem e em que formato guardar a imagem (JPG) – térmica, térmica e visível, térmica e visível (MSX), separando a imagem térmica da visível.

Para se poder utilizar o SDK e obter documentação a ele referente, é necessário aceder e criar uma conta em https://developer.flir.com/flir-one-software-development-kit/.

Foi feito um levantamento e registaram-se todos os dados referentes a este estudo. Posteriormente, os dados foram agrupados e apresentam-se no Anexo B.

As funções do SDK que foram sendo utilizadas no desenvolvimento deste projeto foram expostas e explicadas nos capítulos referentes à sua utilização.

2.9 - Levantamento das câmaras térmicas adaptáveis a dispositivos móveis

Foi feito um estudo de mercado para se avaliar quais as câmaras térmicas, adaptáveis a dispositivos móveis, disponíveis atualmente.

Na tabela seguinte apresentam-se as várias câmaras térmicas encontradas, bem como algumas das suas características.

Tabela 2.1 - Características das câmaras térmicas.

Câmara	Características
FLIR ONE 1ª geração [82]	• Capa para telemóvel • Escala de temperatura: 0°C a 100°C • *Lepton camera* • Tecnologia MSX • Tamanho da matriz de sensores: 80x60 • NETD (diferença de temperatura não equivalente) < 0.1°C • Bateria própria (4h) • Peso: 110g • Sistema operativo: *Android* e *iOS*
FLIR ONE 2ª geração [83]	• Escala de temperatura: -20°C a 120°C • Tamanho da matriz de sensores: 160x120 • Tecnologia MSX • NETD < 0.1°C • Bateria própria (1h) • Peso: 29g • Preço: $240.99 (222.653€) • Sistema operativo: *Android* e *iOS*
FLIR ONE 3ª geração [84]	• Escala de temperatura: -20°C a 120°C • Tamanho da matriz de sensores: 80x60 • NETD < 0.1°C • Tecnologia MSX • Bateria própria (1h) • Peso: 34.5g • Preço: $199 (177.239€) • Repetibilidade da medição: ± 3% • Sistema operativo: *Android* e *iOS*

FLIR ONE 3ª geração PRO [85]	• Escala de temperatura: -20°C a 120°C • Tamanho da matriz de sensores: 160x120 • NETD < 0.1°C • Tecnologia MSX • Bateria própria (1h) • Peso: 36.5g • Preço: $399 (355.368€) • Repetibilidade da medição: ± 3% • Sistema operativo: *Android* e *iOS*
Seek thermal CompactPro [86]	• Escala de temperatura: -40°C a 330°C • Tamanho da matriz de sensores: 320x240 • Preço: $499 (444.443€) • Sistema operativo: *Android* e *iOS*
Themal Expert Q1 [87]	• Lentes de 6.8mm (13mm opcional) • Escala de temperatura: -10°C a 150°C • Tamanho da matriz de sensores: 384x288 • NETD < 50mK • Peso <39g, com lentes • Preço: $499 (444.443€) • Sistema operativo: *Android* e *iOS*
Themal Expert Q1 Plus [88]	• Lentes de 13mm • Escala de temperatura: -10°C a 120°C • Tamanho da matriz de sensores: 384x288 • NETD < 50mK • Peso <42g, com lentes • Preço: $599 (533.5€) • Sistema operativo: *Android* e *iOS*
Themal Expert Q1 Pro [89]	• Lentes de 6.8mm (13mm opcional) • Escala de temperatura: -10°C a 250°C • Tamanho da matriz de sensores: 384x288 • NETD < 50mK • Peso<39g, com lentes • Preço: $699 (622.562€) • Sistema operativo: *Android* e *iOS*
Themal Expert V1 [89]	• Lentes de 19mm (25 e 35mm opcional)

	• Escala de temperatura: -10°C a 120°C • Tamanho da matriz de sensores: 640x480 • NETD < 50mK • Peso<90g, com lentes • Preço: $2990 (2663.03€) • Sistema operativo: *Android* e *iOS*
ThermApp [91]	• Lentes de 6.8mm (13mm, 19mm e 35mm opcional) • Escala de temperatura: 5°C a 90°C • Tamanho da matriz de sensores: 384x288 • NETD < 0.07°C • Repetibilidade da medição: ± 3% (@25°C) • Peso 138g, com lentes • Preço: $999 (889.756€) • Sistema operativo: *Android*

2.9.1 -Câmara FLIR ONE de 2ª Geração

As câmaras da FLIR ONE de 2ª geração têm sido utilizadas na prática clínica como complemento dos exames de diagnóstico tradicionais. Há relatos na literatura do seu uso clínico, nomeadamente por *Hardwicke et al*. (2016) [92] *e S. Suphachokauychai et al*. (2016) [93]. Realçam a sua portabilidade e mais baixo custo e a melhor resolução de imagem térmica que é interpolada a partir do tamanho original da matriz de sensores.

Esta câmara possui SDK próprio e foi a eleita para ser utilizada neste estudo. Assim, é fundamental fazer um estudo das características desta câmara da FLIR ONE. As suas características encontram-se listadas em baixo.

2.9.1.1 - *Android*

- Intervalo de temperatura de -20ºC a 120ºC, mas temperatura operacional de 0ºC a 35ºC;
- Peso: 29g;
- Dimensões: 72x26x18 mm;
- Capacidade da bateria: 350 mA-h;
- Duração da bateria: ≈ 1h;
- Câmara Visível: VGA;
- NETD (diferença de temperatura não equivalente): consegue detetar diferenças de temperatura de 0.1ºC:
- Modo de carregamento: micro-USB com carregador de parede 1A (carrega o FLIR ONE apenas, não o dispositivo);
- Certificações: FCC, CE, RoHS, CAN ICES-3 (B)/NMB-3(B), UL;
- Compatibilidade: tem de ter os seguintes requisitos mínimos:
    - Versão *Android* 4.4.2 ou superior;

- GPS, localização, OTG USB, lanterna e microfone (para funcionalidade de panorama tem de ter também giroscópio e acelerómetro);
- Permissões de GPS, localização, microfone e lanterna [83].

Figura 2.7 - Câmara FLIR ONE de 2ª geração para *Android*.

2.9.1.2 - *iOS*

- Intervalo de temperatura de -20ºC a 120ºC, mas temperatura operacional de 0ºC a 35ºC;
- Peso: 29g;
- Dimensões: 72x26x18 mm;
- Capacidade da bateria: 350 mA-h;
- Duração da bateria: ≈ 1h;
- Câmara Visível: VGA;
- NETD (diferença de temperatura não equivalente): consegue detetar diferenças de temperatura de 0.1ºC;
- Modo de carregamento: conector *lightning* (*Apple*®) com carregador de parede 1A (carrega o FLIR ONE apenas, não o dispositivo);
- Certificações: FCC, CE, RoHS, CAN ICES-3 (B)/NMB-3(B), UL;
- Compatibilidade: dispositivos da Apple que contenham um conector *lightning* (*iPhone 7* / *iPhone 7 Plus*, *iPhone SE*, *iPhone 6* / *iPhone 6 Plus*, *iPhone 5* / *iPhone 5s*, *iPad Air* / *iPad Air 2*, *iPad Mini* / *iPad Mini 3*, *iPad* (4ª geração) [94].

Figura 2.8 - Câmara FLIR ONE 2ª geração para *iOS*.

De seguida, apresenta-se uma figura exemplificativa da câmara FLIR ONE de segunda geração, que é visualmente igual para *Android* e *iOS*, sendo a única diferença o Micro USB que faz a conexão do dispositivo com a câmara, descrevendo todas as características que lhe são inerentes [95].

Figura 2.9 - Câmara FLIR ONE de segunda geração.

Deve, no entanto, salientar-se que estas câmaras não satisfazem os requisitos mínimos para serem utilizadas em medicina, que são [96], [97]:

- Tamanho da matriz de sensores de 320x240;
- NETD < 50mK a 30 ºC;
- Rastreabilidade de medição de ± 2% da gama medida.

2.10 - *SQLite*

O ***SQLite*** é uma biblioteca, iniciada no ano 2000, desenvolvida em linguagem C, que implementa uma base de dados SQL. A sua versão atual é a 3.5.9, lançada em maio de 2017. Esta base de dados possui inúmeras características, das quais se destacam:

- Autonomia (*self-contained*) – no sentido de ter muito poucas dependências. Corre em qualquer sistema operativo e não usa bibliotecas ou interfaces externas (excluindo algumas chamadas da biblioteca C standard como *malloc()* ou *strlen()*). Toda a biblioteca *SQLite* está num único ficheiro de código fonte que não requer instalações especiais.
- Não necessita de servidor (*serveless*) – A maioria das bases de dados SQL são implementadas com recurso a um *webservice* remoto. Aplicações que desejam aceder a bases de dados têm de comunicar com um servidor utilizando algum tipo de comunicação (por exemplo, TCP/IP) para enviar os pedidos e receber resultados. No entanto, o *SQLite* não funciona desta forma. O processo que deseja aceder à base de dados lê e escreve diretamente nos ficheiros da base de dados no disco, não existindo nenhum processo intermediário do servidor.
- Dispensa configurações (*zero configurations*) – O *SQLite* não precisa de ser instalado para ser utilizado, não existindo nenhum procedimento de configuração. Para informar o sistema de que o *SQLite* está a ser executado, não é requerida qualquer ação. Os dispositivos *Android* e *iOS* possuem o *SQLite* embutido, estando esta biblioteca presente no SDK destes sistemas operativos. Assim, para criar e atualizar a base de dados só é necessário definir os comandos SQL respetivos.
- Transacional (*transactional*) – O *SQLite* implementa transações que são ACID, ou seja, atómicas (requerem que cada transação seja tudo ou nada), consistentes (asseguram que qualquer transação levará a base de dados de um estado válido para outro), isoladas (garantem que a execução simultânea de transações resulta num estado que seria obtido se estas fossem executadas sequencialmente) e duradouras (a propriedade de durabilidade garante que uma vez que a transação tenha sido comprometida, ela permanecerá assim), mesmo que a transação seja interrompida por uma falha no programa, no sistema operacional ou mesmo uma falha de energia.
- Leve (*small-size*) – é uma biblioteca muito compacta, cujo tamanho pode ser inferior a 500 kB, dependendo da plataforma de destino e das configurações de otimização do compilador (o código de 64 bits é maior e algumas otimizações do compilador, como o desdobramento do *loop*, podem fazer com que o código do objeto seja muito maior).

O *SQLite* é uma componente de *software* de um sistema de gestão de bases de dados que é usado para criar, ler, atualizar e apagar (CRUD) uma base de dados SQL.

É *open source* e o seu código está em domínio público, sendo a sua utilização, desta forma, gratuita. Este suporta tipos de dados TEXT (*String* em *Java*), INTEGER (*long* em *Java*) e REAL (*double* em *Java*). Os outros tipos devem ser convertidos num destes tipos antes de ser guardado na base de dados. No entanto, o *SQLite* não verifica se os dados inseridos em cada coluna são do tipo definido.

Por outro lado, implementa grande parte da linguagem SQL *standard*, existindo algumas funcionalidades que ainda não são implementadas, como, por exemplo, FULL JOIN.

O ficheiro local onde a base de dados fica armazenada pode ser visualizado num computador, usando o *SQLiteManager*.

Todas estas características contribuem para que o *SQLite* seja muito popular no desenvolvimento de aplicações, sendo a base de dados mais usada em todo o mundo [98].

2.11 - *Webservices*

Um *webservice* é um padrão usado para trocar informações entre aplicações ou sistemas de tipos heterogéneos (exemplo: *Java* e *PHP*). Quer isto dizer que aplicações escritas em várias linguagens de programação e em execução em várias plataformas podem usar *webservices* para trocar informações através da internet, usando, por exemplo, o protocolo HTTP.

Assim, um *webservice* é uma porção de código escrito numa linguagem de programação que pode ser invocada remotamente através do protocolo HTTP. Dado que os métodos *web* são expostos publicamente, todas as aplicações podem invocar o *webservice* e usar a funcionalidade dos métodos da *web*, tornando possível a exposição do método como um serviço através da rede.

Por outro lado, com a ajuda de um *webservice*, aplicações heterogéneas, ou seja, que utilizam tipos de linguagens diferentes, podem comunicar entre si. Desta forma, a interoperabilidade entre aplicações é uma grande vantagem da utilização de *webservices*.

Os *webservices* utilizam um protocolo padronizado para a comunicação, que deve ser adotado e seguido pelas aplicações que criam o serviço.

Hypertext Transfer Protocol (HTTP) é o principal e mais comum protocolo de comunicação utilizado. Permite a transferência de dados entre redes de computadores. Para que o protocolo HTTP consiga transferir os dados, é imperativo que os protocolos TCP/IP tornem possível a conexão entre clientes e servidores. Para isso, utilizam-se *sockets* TCP/IP[99].

Representational State Transfer (REST) é um estilo arquitetónico concebido por *Roy Fielding* para projetar aplicações em rede. É uma alternativa ao uso de mecanismos complexos como *Simple Object Access Protocol* (SOAP) para estabelecer uma comunicação entre um cliente e um servidor. O REST facilita a comunicação entre computadores remotos, usando o protocolo HTTP simples que suporta operações CRUD (*Create*, *Read*, *Update and Delete*, ou seja, criar, ler, atualizar e apagar) no servidor. O essencial desta arquitetura são os URLs (*Uniform Resource Locator*), que é a forma de identificar os endereços aos quais se pretende aceder, e os recursos. Usa 4 métodos HTTP para executar tarefas:

- GET – é um dos métodos HTTP mais simples. Tem como principal objetivo pedir ao servidor um recurso, que pode ser uma página HTML, um ficheiro de som, ou até uma imagem. Assim, o método GET serve para obter algo de um servidor. Neste método, os dados que são enviados estão anexados ao URL.
- POST – é usado para enviar dados que vão ser processados. Os dados estão incluídos no corpo do pedido. Assim, o método POST serve para enviar algo para um servidor.
- PUT – é um método usado para atualizar dados. Tal como no método POST, os dados estão incluídos no corpo do pedido.
- DELETE – é um método utilizado para apagar um recurso específico.

Criando-se um *webservice* com a arquitetura REST, a sua denominação passa a ser *RESTful* [100].

SOAP é um protocolo projetado para invocar aplicações remotas através de trocas de mensagens. É um padrão aceite para se utilizar com *webservices*. Com ele, garante-se interoperabilidade e intercomunicação entre diferentes sistemas.

O SOAP especifica exatamente como codificar um cabeçalho HTTP e um ficheiro XML para que um programa num computador possa chamar um programa noutro computador e transmitir informações. O SOAP também especifica como o programa chamado pode retornar uma resposta [101], [102].

Capítulo 3

Metodologia

Nesta secção é descrita a metodologia utilizada para a realização e desenvolvimento deste projeto, desde a solução proposta para alcançar a finalidade proposta, até à sua implementação e validação.

O trabalho foi dividido em 5 partes, nomeadamente a definição da solução proposta, o estudo das câmaras, nomeadamente da FLIR ONE (*Android versus iOS*), o estudo do formato JPG, o desenvolvimento da aplicação *Android* e os casos de uso.

3.1 - Solução proposta

Como mencionado no capítulo 2, atualmente, existem no mercado diversas aplicações que utilizam a termografia de IR e que permitem ao utilizador a captura de imagens térmicas para vários fins.

Segundo *P. Crosby* [103], e numa perspetiva de *Business-to-Business*, a "Qualidade é a conformidade com as especificações". Assim, garantir a qualidade é assegurar que o produto final entregue não tem defeitos e vai ao encontro das especificações exigidas.

Por outro lado, e, agora, numa perspetiva de *Business-to-Consumer*, a definição anterior de qualidade torna-se insuficiente. É uma condição indispensável o produto/serviço estar em conformidade com as especificações, mas por si só não define qualidade no conceito deste tipo de relação, do ponto de vista do cliente final. Desta forma, segundo *J. Juran*, a "Qualidade é a aptidão ao uso" [104]. Por outras palavras, para garantir a qualidade é fundamental que os produtos finais vão ao encontro das necessidades e preferências dos consumidores. Na verdade, não vale a pena criar produtos ou fornecer um serviço, perfeito e em conformidade com as especificações se essas mesmas especificações não correspondem ao que o consumidor necessita ou que realmente quer.

Neste caso em concreto, torna-se necessário criar um produto com qualidade segundo todas as perspetivas acima referidas. As aplicações existentes possuem qualidade, na medida em que todas elas funcionam e desempenham todos os requisitos previstos pelo seu desenvolvedor. No entanto, existe uma lacuna nas características necessárias para a

finalidade deste projeto de investigação: avaliar o risco do *pé diabético*, através do processamento de imagens térmicas.

Assim, a solução proposta consiste no desenvolvimento de uma aplicação, para os sistemas operativos *Android* e *iOS*, que permita obter imagens térmicas, imagens visíveis e imagens MSX, apenas mostrando na interface com o utilizador o tipo de imagem escolhido, que por omissão é térmico, utilizando a câmara FLIR ONE de segunda geração.

A aplicação deve ter uma interface acessível e intuitiva para os profissionais de saúde e outros prestadores de cuidados, de modo a que possam capturar as imagens e armazená-las.

Possibilitará, igualmente, escolher a emissividade que, por omissão, deverá ser 0.98 (emissividade da pele) e a escala de cores a utilizar (por omissão *Rainbow*). A gama de temperaturas deve poder ser definida e as temperaturas máxima e mínima devem constar no ecrã, podendo o utilizador escolher a unidade (*Celsius* ou *Fahrenheit*, sendo a primeira a estar por omissão), nas definições.

A imagem térmica capturada deve poder ser guardada diretamente no dispositivo móvel e o utilizador deve poder aceder-lhe sempre que dela necessitar. A integração dos dados clínicos, como a conjugação das imagens térmicas no conjunto de dados definidos *a priori* ou o seu envio para um *webservice* remoto são características a incluir.

Para isso, será criado um *webservice* RESTful, onde os dados serão armazenados para posterior análise.

A análise automática das imagens e a sua classificação fazem parte de outro projeto de dissertação sob a mesma orientação e a decorrer em paralelo.

Por último, a aplicação deverá ser capaz de receber e exibir um relatório da análise e classificação das imagens obtidas para profissionais de saúde e doentes.

Por outras palavras e concluindo, a aplicação envia as imagens térmicas capturadas e um conjunto de dados associado para o *webservice* remoto e recebe de lá um relatório da análise e interpretação desses dados.

A aplicação deve permitir obter imagens térmicas de qualquer região do corpo humano, mas, neste caso particular, o objetivo final é capturar imagens dos pés. Espera-se que esta seja vantajosa na avaliação e tratamento de doentes diabéticos com possibilidade de desenvolverem as complicações do *pé diabético*.

De forma a facilitar o posicionamento e distância entre o equipamento de captura e o doente, deve existir um *overlay*, onde os limites externos da planta do pé (máscara) estarão delineados. O doente pode estar sentado ou deitado em posição supina, com o pé fletido, definindo um ângulo de aproximadamente 90º com a perna.

3.1.1 -Arquitetura funcional do sistema

A arquitetura funcional de um sistema consiste num modelo de arquitetura que identifica as funções do sistema e o modo como estas interagem entre si, de modo a executarem o objetivo requerido. É útil para perceber os passos que o sistema tem de realizar.

A figura 3.1 [105], [106], [107], [108] ilustra, de forma esquemática, a arquitetura funcional do sistema a desenvolver.

Assim, a aplicação a implementar deve permitir a captura de imagens térmicas do pé, através de uma câmara térmica adaptada ao dispositivo móvel e o registo de alguns dados pessoais e fisiológicos do doente. A partir daqui, deve possibilitar o envio deste conjunto de dados para um *webservice* remoto. Para isso é necessária a existência de uma ligação de rede sem fios. No *webservice* remoto, os dados são arquivados e analisados numa base de dados.

A aplicação deve, ainda, estar habilitada a receber um relatório de análise com os resultados obtidos.

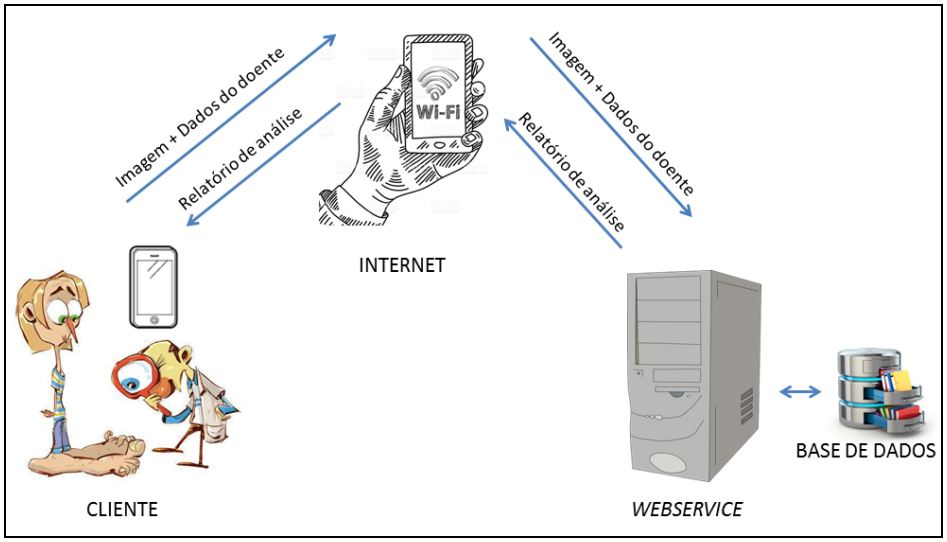

Figura 3.1- Arquitetura funcional do sistema (adaptada).

3.1.2 - Especificação do sistema

Para fazer a especificação do sistema é necessário e importante definir os seus objetivos, ou seja, a necessidade da sua existência, quem o vai utilizar e que competências é necessário que possua e qual a sua dimensão, por outras palavras, qual será o campo da sua utilização. É igualmente relevante conhecer os recursos tecnológicos que irão ser necessários, um orçamento que reflita a verba correspondente aos custos, tendo em conta não só o equipamento, mas também os custos humanos (desenvolvimento e supervisão) e, por último, o prazo que se estabelece para a conclusão do projeto.

Desta forma, a especificação do sistema é a que se apresenta, de seguida, de forma abreviada.

Objetivo: Garantir que a aplicação captura, guarda e envia imagens térmicas para um *webservice* remoto e recebe de lá relatórios de análise.

Utilizadores:

Tabela 3.1 - Utilizadores e respetivas competências.

Utilizadores	Competências básicas
Profissionais de saúde	Ter um mínimo de conhecimentos de sistemas operativos *Android* ou *iOS* e saber interpretar os resultados dos relatórios.
Outros cuidadores	Ter um mínimo de conhecimentos de sistemas operativos *Android* ou *iOS*.
Doentes	Ter um mínimo de conhecimentos de sistemas operativos *Android* ou *iOS*. NOTA: o doente não consegue capturar imagens dos próprios pés.
Familiares e amigos do doente	Ter um mínimo de conhecimentos de sistemas operativos *Android* ou *iOS*.

Dimensão: Usado num ACES (Agrupamento de Centros de Saúde).

Recursos:

- 1 *webservice* remoto;
- 1 dispositivo móvel com sistema operativo *Android* ou *iOS*;
- 1 câmara da FLIR ONE de segunda geração.

Orçamento:

- Recursos: *webservice* (±120€/ano) + dispositivo móvel (±180€ (*Android*) + ±350€ (*ipad mini*)) + câmara da FLIR ONE de segunda geração (300€) = 1250€.
- Horas de desenvolvedor: 18sem * 35h = 630h. 630h * ± 20€/h = 12600€.
- Custos de supervisão: 18h * 60€/h = 1080€.
- Orçamento global: 1250€ + 12600€ + 1080€ = 14930€.

Prazo: Implementado em 18 semanas.

3.1.3 - Levantamento dos requisitos

Um processo de engenharia de sistemas pode ser definido como um conjunto de atividades executadas em sequência, que, a partir de uma necessidade (*input*), dão origem a um sistema que responde a essa necessidade (*output*). Este conceito é crucial, já que permite enumerar as atividades que servem de guia para transformar as entradas em saídas e, assim, conseguir desenvolver o sistema.

A análise de requisitos corresponde à primeira fase de um processo de engenharia de sistemas. O seu principal objetivo é conhecer a necessidade que justifica o desenvolvimento do sistema. É nesta fase que se produz uma especificação das características de que o sistema deve ser dotado para satisfazer essa necessidade. Estas representam aquilo que se pretende do sistema a desenvolver. Por outras palavras, tem como objetivo apresentar, como o próprio nome indica, a análise e especificação de requisitos.

Para a realização de projetos de engenharia complexos, é fundamental fazer uma análise de requisitos. Esta tem como principais vantagens as seguintes:

- Permite fazer o levantamento de todos os requisitos, de forma objetiva e clara, que devem estar presentes no produto final, de acordo com as especificações do cliente;
- Permite verificar se o que está idealizado para o sistema corresponde ao que é desejado;
- Reduz a probabilidade de virem a ser identificadas, mais tarde, novas necessidades e restrições que comprometam as soluções;
- Proporciona uma base para estimar os custos e os prazos do projeto e para a verificação e validação do resultado do mesmo.

Um requisito consiste numa funcionalidade que o sistema deve oferecer ou uma característica que o sistema deve possuir. Deve ser realista, não ambíguo, executável e verificável. Existem 2 tipos de requisitos: os funcionais e os não funcionais.

Os requisitos funcionais são aqueles que têm de ser implementados para o sistema funcionar, ou seja, representam funções que o sistema deve desempenhar. É de notar que sem eles o sistema não funciona de forma global.

Por seu lado, os requisitos não funcionais são requisitos que otimizam o sistema e garantem uma melhor manutenção, flexibilidade e qualidade deste. Fazem parte das características mínimas de um *software* de qualidade. Não estão diretamente ligados às funcionalidades do sistema, mas colocam restrições ao sistema a ser desenvolvido. Estes incluem confiabilidade, segurança, adaptabilidade, portabilidade e performance, mas, no entanto, não estão limitados a isso [109].

Existem alguns requisitos mínimos para que as aplicações, que utilizam termografia de infravermelhos, possam ser utilizadas na prática clínica. Assim, o utilizador deve conseguir escolher qual o valor de emissividade que pretende utilizar, tendo à sua disposição o valor da emissividade da pele (0.98).

A temperatura é um parâmetro muito importante nesta avaliação, pelo que o utilizador deve conseguir especificar a escala de temperaturas que pretende utilizar.

A aplicação deve ainda capturar imagens térmicas com qualidade, permitir o registo de alguns dados do doente e possibilitar a calibração automática.

A aplicação pode ou não fazer a avaliação das imagens, mas deve permitir a receção de um relatório com os resultados da análise.

Por outro lado, a privacidade do doente deve ser uma prioridade. Desta forma, a aplicação tem de ter algumas medidas de segurança para que não ocorram fugas de informação. Um exemplo seria o registo e *login*.

Na tabela 3.2 são apresentados os requisitos funcionais e na tabela 3.3 os não funcionais, definidos para o sistema a desenvolver.

Na avaliação da prioridade e do risco de cada requisito, foram mais valorizados os requisitos que correspondem a funcionalidades pioneiras na aplicação a desenvolver.

Tabela 3.2 - Requisitos Funcionais.

ID	Requisito	Prioridade	Risco
F1	A aplicação deve verificar se a câmara FLIR ONE está ligada	Elevada	Elevado
F2	A aplicação deve verificar se há ligação à internet	Baixa	Médio
F3	A aplicação deve verificar se existe comunicação com o *webservice*	Alta	Médio
F4	O utilizador deve conseguir especificar a emissividade	Alta	Alto
F5	O utilizador deve conseguir especificar a gama de temperatura	Média	Médio
F6	A aplicação deve mostrar apenas o tipo de imagem escolhido pelo utilizador	Baixa	Médio
F7	O utilizador deve conseguir escolher a paleta de cores falsas	Média	Médio
F8	A aplicação deve capturar imagens	Elevada	Elevado
F9	A aplicação deve armazenar as imagens capturadas	Média	Baixo
F10	A aplicação deve permitir que o utilizador veja as imagens capturadas diretamente no dispositivo móvel	Baixa	Médio
F11	O utilizador deve conseguir escolher as unidades de temperatura	Baixa	Baixo
F12	A aplicação deve permitir o registo dos dados pessoais e fisiológicos do doente	Elevada	Elevado
F13	A aplicação deve enviar os dados a um *webservice* remoto	Alta	Alto
F14	A aplicação deve receber relatórios do *webservice* remoto	Alta	Alto
F15	A aplicação deve permitir controlar o posicionamento da imagem, através do *overlay*	Alta	Médio

Tabela 3.3 - Requisitos Não Funcionais.

ID	Requisito	Prioridade	Risco
NF1	Por omissão, a emissividade deve ser 0.98	Elevada	Elevado
NF2	Por omissão, a paleta de cores é a *rainbow*	Alta	Médio
NF3	Por omissão, o tipo de imagem a capturar é térmica	Elevada	Elevado
NF4	A aplicação deve ter uma interface com o utilizador simples e intuitiva	Elevada	Alto
NF5	A aplicação deve comunicar com o *webservice* remoto sem perda de informação	Alta	Elevado

NF6	A aplicação deve conseguir saber se o pé em estudo está na posição correta	Alta	Médio
NF7	O código da aplicação deve ser modular (se for necessário fazer alterações, estas devem ser as mínimas possíveis)	Média	Baixo
NF8	O código da aplicação deve ser bem documentado, com comentários, para que futuros utilizadores percebam o que foi desenvolvido	Baixa	Baixo

3.1.4 -Interface inicial com o utilizador

Um dos requisitos do projeto é desenvolver a aplicação com uma interface com o utilizador atrativa, intuitiva e simples de utilizar. Assim, é necessário conceber um esboço de uma possível interface, que poderá sofrer alterações depois de se iniciar o desenvolvimento da aplicação e de sondada a opinião de profissionais de saúde e eventuais utilizadores.

As figuras 3.2 a 3.7 ilustram a interface principal e as sequenciais, existindo uma repetição de imagens no sentido de tentar ilustrar a sua aplicabilidade prática. Assim, para cada passo a efetuar, faz-se uma demonstração, através da colocação de um círculo vermelho, sobre o comando a ser utilizado.

As figuras 3.2 a 3.7 referem-se ao *mockup* inicialmente idealizado.

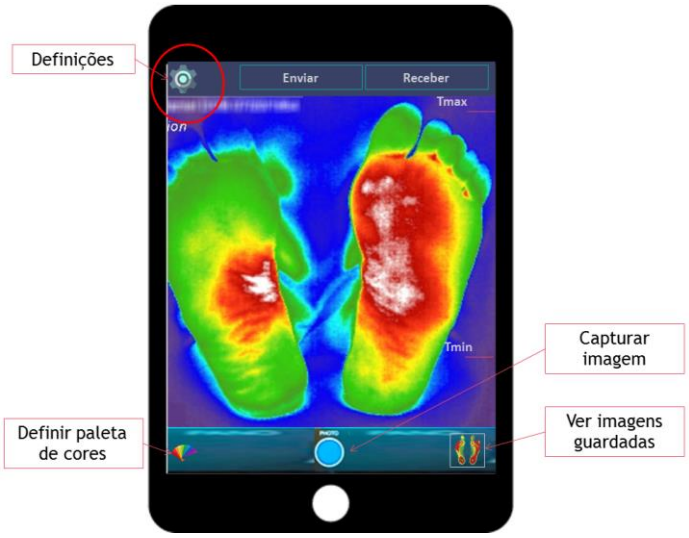

Figura 3.2 - Página principal.

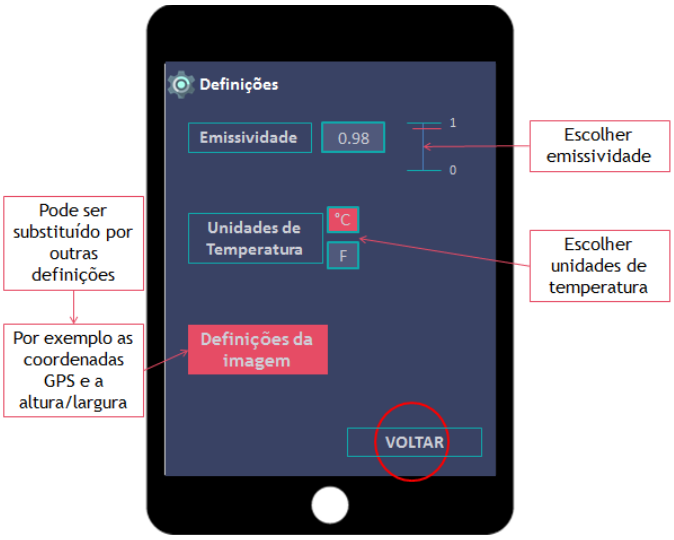

Figura 3.3 - Definições.

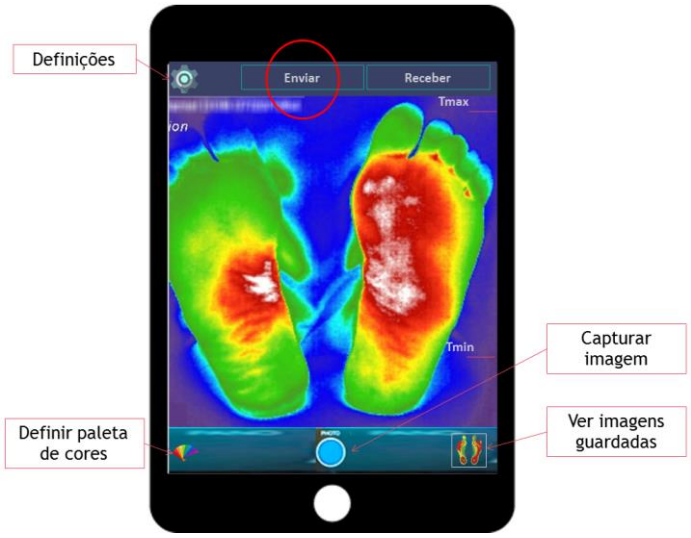

Figura 3.4 - Página principal.

Figura 3.5 - Enviar dados.

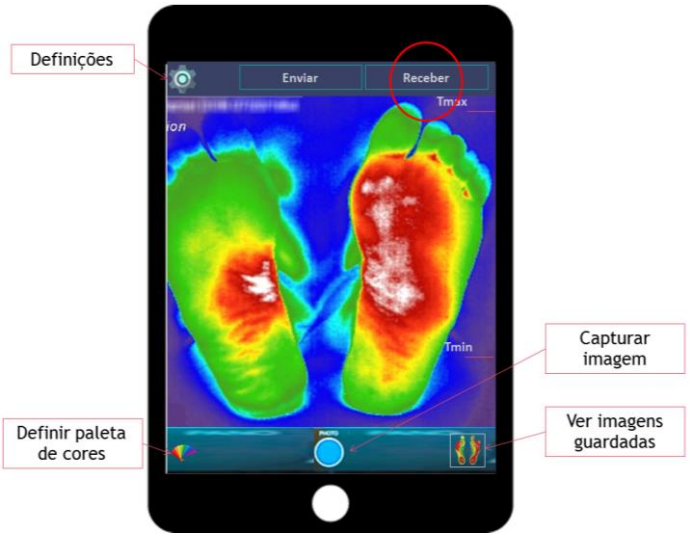

Figura 3.6 - Página principal.

Figura 3.7 - Receber relatório.

3.2 - Estudo de câmaras térmicas

A não existência de nenhum estudo comparativo entre as várias câmaras térmicas da FLIR®, adaptáveis dispositivos móveis, que cumpram os requisitos mínimos em aplicações clínicas, em relação às temperaturas obtidas por cada câmara, contra uma referência para verificação de qualidade, despertou a necessidade de realizar essa verificação. Assim, as câmaras FLIR E60 e FLIR ONE de segunda geração para os sistemas operativos *Android* e *iOS* foram estudadas para avaliar a sua performance e aferir qual o seu erro de medição de temperaturas.

Para isso, foram capturadas imagens térmicas a um corpo negro (referência de calibração) a diferentes temperaturas. As câmaras foram avaliadas num período de tempo de cerca de 60 minutos, visto que é o tempo de durabilidade da bateria das câmaras FLIR ONE.

Foram obtidas várias imagens térmicas, ao longo do tempo de estudo, pelas diferentes câmaras, de um corpo negro, inicialmente a temperatura constante (30ºC) e, posteriormente, a temperaturas variáveis (as da superfície do corpo humano: 20 a 38 ºC), com aumento ou diminuição da temperatura do corpo negro, para avaliar o comportamento das diferentes câmaras.

Os testes encontram-se descritos com maior detalhe, no capítulo seguinte, "Testes e Resultados", bem como os resultados obtidos.

3.3 - Estudo do formato JPG radiométrico

Para complementar o trabalho foi feito o estudo ao formato JPG radiométrico, propriedade da FLIR®. Isto é relevante para a necessidade da criação de aplicações específicas de análise destas imagens, de forma a não depender das aplicações fornecidas pelo fornecedor das câmaras, que não estão preparadas para as necessidades na prática clínica.

Uma imagem no formato JPG não é uma imagem sequencial, mas sim por blocos, isto é, por linhas. Todos os dados importantes da imagem térmica estão presentes no seu cabeçalho. Desta forma, é importante fazer um estudo deste formato para saber a posição de alguns dados que vão ser usados nesta aplicação clínica.

Considerou-se crucial conseguir separar o cabeçalho, a imagem térmica e a visível, assim como descobrir os valores de certos parâmetros, como a emissividade, temperatura máxima e mínima, constantes de *Planck* (R1, B, F, O e R2) e, também, o modelo da câmara da FLIR, bem como o seu número de série. Para tal, utilizou-se engenharia reversa.

Em primeiro lugar, usou-se o *ExifTool*, um programa de *software* para leitura, escrita e manipulação de dados de uma imagem, de modo a obter e guardar num ficheiro de texto os dados da imagem a ser analisada. Para isso, escreveu-se na linha de comandos o seguinte comando: **exiftool nome_da_imagem.jpg > nome_do_ficheiro.txt**.

De seguida, converteram-se os valores obtidos no *ExifTool* para hexadecimal. Alguns valores foram convertidos diretamente, como, por exemplo, a temperatura máxima e mínima, que já se encontrava em graus *Kelvin* (unidades do sistema internacional). Por outro lado, alguns valores tiveram de ser previamente transformados antes de serem convertidos para hexadecimal, como, por exemplo, a data e hora originais, que foi alterada para a Era *Unix* (definida como o número de segundos passados desde o *epoch* – 1 de janeiro de 1970 às 00:00:00 – não considerando segundos bissextos), com o auxílio de um conversor *online* (http://www.onlineconversion.com/unix_time.htm). Para além deste conversor, foram, ainda, utilizados outros dois conversores: um que converte valores decimais em hexadecimais (http://www.binaryhexconverter.com/decimal-to-hex-converter) e outro que converte valores *float* em hexadecimais (https://gregstoll.dyndns.org/~gregstoll/floattohex/).

Descobertos todos os valores a procurar, e através de um editor hexadecimal, neste caso, o *UltraEdit*, identificaram-se os endereços onde estão os valores a analisar. Por vezes, os valores hexadecimais a pesquisar estavam pela ordem inversa da ponderada.

Para garantir que os valores estão no mesmo endereço em todas as imagens, é necessário testar com mais do que uma imagem.

Desta forma, foram testadas 9 imagens e os seus endereços foram devidamente anotados, para posterior estudo. Os testes e resultados obtidos estão relatados no capítulo "Testes e Resultados".

Assim, depois de analisados os resultados e encontrada uma referência, que neste caso foi a palavra "FOL", facilmente se conseguiu aceder aos outros parâmetros, através do *offset* constante. Em relação aos parâmetros que tinham o mesmo endereço dentro do mesmo grupo de imagens, também são simples de alcançar. No caso das imagens da FLIR ONE, procura-se por "FOL2" e depois, como já se sabe o endereço, basta aceder ao endereço pretendido. Nas restantes imagens, pesquisa-se por "FOL1" e, de seguida, acede-se ao endereço desejado. Apresenta-se, seguidamente, um exemplo de código em C#, desenvolvido no *Visual Studio 2015*, referente ao que foi anteriormente mencionado.

```
List<int> res3 = searchforstr(bytes, "FOL2");
if (res3.Count == 1)
{
    ivalue = BitConverter.ToInt32(bytes, 592);
    listBox1.Items.Add("Temperature Min: " + ivalue.ToString());
    ivalue = BitConverter.ToInt32(bytes, 584);
    listBox1.Items.Add("Temperature Max: " + ivalue.ToString());
    ivalue = BitConverter.ToInt32(bytes, 438);
    listBox1.Items.Add("Focal Length: " + ivalue.ToString());
    ivalue = BitConverter.ToInt32(bytes, 430);
    listBox1.Items.Add("Subject Distance: " + ivalue.ToString());
}
```

Ultrapassada esta primeira fase, seguiu-se o estudo de como separar a imagem térmica e a visível. Esta etapa não foi executada nas imagens da câmara FLIR ONE.

Para se extrair a imagem térmica diretamente do *ExifTool* escreveu-se na linha de comandos o seguinte comando: **exiftool nome_da_imagem.jpg -b -rawthermalimage > nome_do_ficheiro.termica.jpg**.

Depois de analisada a imagem térmica obtida no *ExifTool*, usando, novamente, engenharia reversa, foi possível extrair a imagem térmica com um pequeno código em C#, no *Visual Studio 2015*.

Os resultados obtidos estão expressos no capítulo seguinte (4.2).

Para se extrair a imagem visível diretamente do *ExifTool* escreveu-se na linha de comandos o seguinte comando: **exiftool nome_da_imagem.jpg -b -embeddedimage > nome_do_ficheiro.visivel.png**.

Depois de estudada a imagem visível obtida no *ExifTool*, usando, mais uma vez, engenharia reversa, foi possível extrair a imagem visível desejada com um pequeno código em C#, no *Visual Studio 2015*.

No Capítulo "Testes e Resultados" (4.2) relatam-se os resultados obtidos.

3.4 - Desenvolvimento da aplicação *Android*

Nesta secção descreve-se o desenvolvimento, etapa a etapa, da aplicação *Androi*d que criou.

Para conceber esta aplicação utilizou-se o *Android Studio*, tendo sido *Java* a linguagem adotada. Assim, o primeiro passo neste desenvolvimento, foi fazer o *download* da última versão do *Android Studio* (2.3.2) e do SDK da FLIR ONE, que se retirou do website da FLIR.

Para uma melhor organização e explicação dos assuntos expostos, este segmento foi dividido em *Work Breakdown Structure*, cronologia do desenvolvimento, processo de inicialização e, por último, processo de desenvolvimento.

3.4.1 -Work Breakdown Structure

O projeto foi desenvolvido a partir da divisão em etapas que, agregadas, formavam o trabalho completo. Assim, e de uma forma esquemática, apresenta-se o *Work Breakdown Strutcture* (WBS) do denominado projeto, ou seja, a sua decomposição em pequenas tarefas mais detalhadas e específicas, como pode ser observado na figura 3.8.

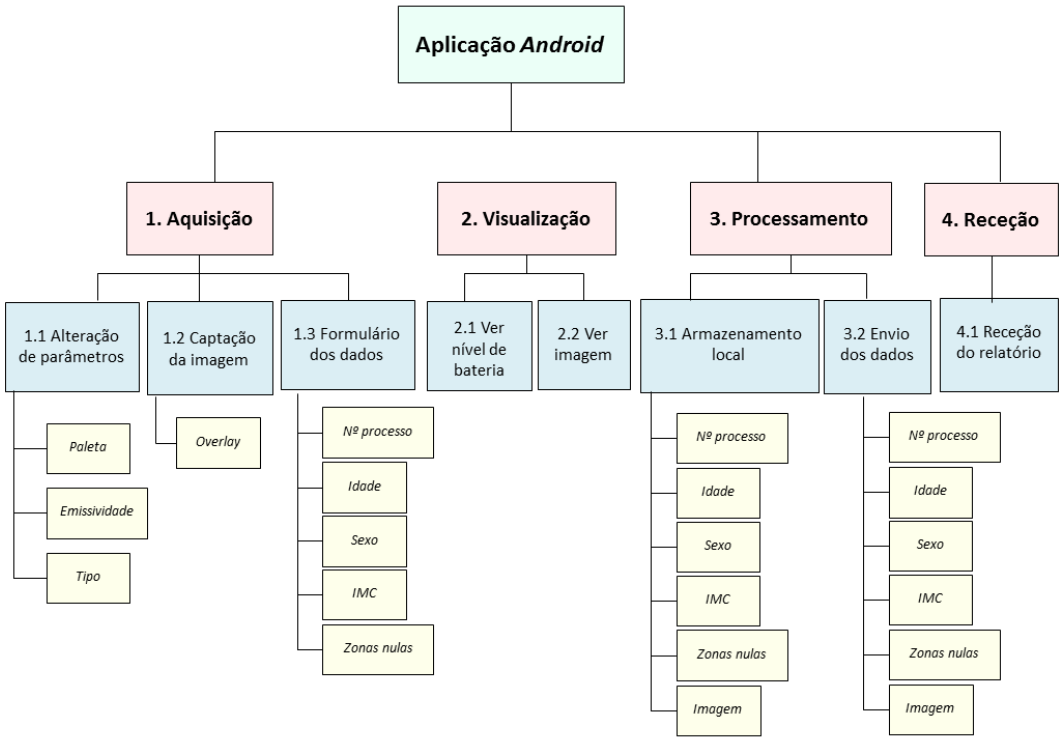

Figura 3.8 – WBS.

Como ilustrado na figura 3.8, no WBS, o sistema foi dividido em 4 tarefas principais (aquisição, visualização, processamento e receção), de onde derivaram outros níveis inferiores e mais detalhados de tarefas.

Assim, relativamente à aquisição, equacionaram-se 3 principais abordagens: a alteração dos parâmetros, a captura da imagem e o preenchimento do formulário com os dados clínicos e pessoais do doente. Os parâmetros a serem alterados são a paleta de cores falsas, o valor da emissividade e o tipo de imagem.

A visualização divide-se em ver o nível de bateria da câmara FLIR ONE e em abrir e ver uma imagem diretamente da galeria de imagens do *smartphone* ou *tablet*.

Em relação ao processamento, as tarefas dividem-se em envio dos dados para um *webservice* remoto e armazenamento local destes. Esses dados referentes a cada doente são o seu número de processo clínico, a sua idade, o seu sexo, o seu índice de massa corporal (IMC) e as zonas do seu pé eventualmente amputadas.

A última etapa é a receção de um relatório do *webservice*.

3.4.2 -Cronologia do desenvolvimento

Nesta secção descreve-se, com maior detalhe, como foi sendo desenvolvida a aplicação, ao longo do tempo.

Tudo começou com o desenvolvimento de uma aplicação básica, sem a utilização do SDK da FLIR ONE. Esta aplicação consistia em escrever uma mensagem e, depois de clicar no único botão existente, a mensagem aparecia numa segunda atividade. Este projeto foi muito importante para a ambientação ao *Android Studio*, uma vez que de uma forma simples se aprendeu a utilizar funções importantes e a comunicação entre duas atividades diferentes.

De seguida, adicionou-se o SDK da FLIR ONE ao projeto, não se alterando nada no código, e este funcionou de igual forma, sem qualquer problema.

O passo seguinte foi tentar usar algumas funções existentes no SDK da FLIR ONE. Criou-se, então, um novo projeto que transmitia uma imagem térmica, em tempo real, ao conectar a câmara FLIR ONE com o *tablet*. Nesta fase, ainda não era possível capturar imagens. Conquistada esta etapa, a tarefa seguinte foi a de alterar a emissividade, a paleta de cores falsas e o tipo de imagem. Nesta fase, ainda não era possível para o utilizador escolher os valores que desejava, mas forçaram-se valores para se testar a utilização das funções referentes à alteração dos parâmetros.

Ultrapassado este estadio, criou-se uma nova aplicação que capturava imagens térmicas, com emissividade a 0.98 e *Rainbow* como paleta de cores falsas, valores forçados por omissão. Esta é uma das tarefas mais cruciais do trabalho.

De seguida, criaram-se novas aplicações que, para além de capturarem imagens, já permitiam ao utilizador escolher qual o valor desejado para cada parâmetro. Foram, assim, obtidas mais 3 aplicações, uma que permitia alterar a paleta de cores falsas, outra que permitia alterar, também, o tipo de imagens a capturar e, por último, uma outra que também permitia alterar o valor da emissividade.

Superada esta tarefa, seguiu-se a de permitir ao utilizador abrir e visualizar uma imagem já capturada e guardada na galeria de imagens do *tablet*. Desenvolveu-se uma aplicação com este intuito.

Para ajudar o utilizador a posicionar os pés para capturar a imagem, acrescentou-se um *overlay*, que funciona como uma máscara ou silhueta dos pés, de forma a fazer corresponder o mais possível essa silhueta com o contorno real dos pés.

Para terminar a fase de aquisição, criou-se uma nova aplicação que acrescentou às funcionalidades já conseguidas e descritas, um formulário dos dados clínicos e pessoais do doente.

Foi, também, desenvolvida uma aplicação que permite ver o nível de bateria da câmara.

Concluído o desenvolvimento da tarefa de aquisição, criou-se uma nova aplicação que era capaz de armazenar os dados mencionados anteriormente num ficheiro local que contém uma base de dados SQL, no *tablet* (*SQLite*), como medida de tolerância a falhas, para, no caso de falta de conectividade ou falha da bateria do dispostivo, garantir que estes não se perdessem.

Com a garantia de que os dados não se perdiam, desenvolveu-se uma última aplicação capaz de comunicar com um *webservice* remoto (também criado). Desta forma, a aplicação envia os dados para o *webservice* e este armazena-os num ficheiro local (que contém a base de dados SQL), no servidor (*SQLite*). Foi, ainda, criada uma interface que permite visualizar os dados armazenados na base de dados.

3.4.3 -Processo de inicialização

O processo de inicialização foi realizado para cada aplicação que foi desenvolvida. Na criação de um novo projeto (***Start a new Android Studio project***) é necessário fazer a sua configuração. É nesta fase que se define qual o nome do projeto, a localização deste e o nome do pacote. A única coisa que aqui foi alterada foi o nome da aplicação, mantendo-se o restante, por omissão. Na etapa seguinte, é necessário escolher quais os dispositivos nos quais se pretende correr e usar a aplicação e qual o nível de API. Neste caso, o *target* escolhido foi "***Phone and Tablet***" e o nível de API não foi alterado nesta fase. De seguida, é preciso adicionar uma atividade e, para isso, selecionou-se "***Empty Activity***". Para concluir a fase de criação de um novo projeto é necessário configurar a atividade, dando-lhe um nome. O nome da atividade principal não foi alterado, chamando-se, então, "***Main Activity***".

Gerado o novo projeto, o *Android Studio* criou a estrutura padrão e abriu o ambiente de desenvolvimento. Antes de começar a desenvolver a aplicação, foi adicionado o pacote do SDK da FLIR ONE ao projeto. Assim, no *Android Studio*, é necessário "colocar o SDK em utilização". Para isso, criou-se um novo módulo (***File → New → New Module***) e, em seguida, escolheu-se "***import .JAR/.AAR package***" e carregou-se no botão "***Next***". Após isto, apareceu uma janela onde se teve de escolher o ficheiro que contém a biblioteca. O ficheiro a escolher chama-se ***flironesdk.aar***. Escolhido o ficheiro, carregou-se no botão "***Finish***".

Concluída esta etapa, tornou-se, então, imperativo adicionar o módulo criado como uma dependência (***File → Project Structure***) (***Modules → app → Dependencies***). Na janela das dependências, clicou-se no "**+**" e de seguida no "***module dependencies***" e adicionou-se o *flironesdk*. Depois de acrescentado, carregou-se em "***OK***" e o SDK foi adicionado ao projeto, podendo ser utilizado.

Foi, ainda, necessário alterar a API mínima (***File → Project Structure***) (***Modules → app → Flavors***). Nesta janela, alterou-se o valor do parâmetro referido para 21.

Por último, importou-se o SDK da FLIR ONE e implementaram-se todos os pacotes deste, como se mostra no códido escrito em Java seguinte:

```
import com.flir.flironesdk.*;
  public class MainActivity extends AppCompatActivity implements Device.Delegate,
FrameProcessor.Delegate, Device.StreamDelegate, Device.PowerUpdateDelegate {
```

3.4.4 -Processo de desenvolvimento

Passa-se, agora, à explicação detalhada de como o projeto foi desenvolvido.

Nesta fase do desenvolvimento foi atribuído um nome à aplicação – ***JT's Diabetic Foot Teller*** e foi desenhado um logótipo que se apresenta no capítulo "Testes e Resultados". Procurou-se que o nome fosse o mais explícito possível, assim como o logótipo.

É de salientar que sempre que se utiliza uma função do SDK, esta é mencionada e explicada.

Ao longo desta secção, far-se-á referência mais detalhadamente às subdivisões do WBS.

O ciclo de vida de uma atividade de uma aplicação *Android* é composto por 6 fases principais: ***onCreate()***, ***onStart()***, ***onResume()***, ***onPause()***, ***onStop()*** e ***onDestroy()***. Por

vezes, existe uma sétima fase, *onRestart()*, que, neste caso, não foi utilizada. A figura seguinte ilustra o funcionamento do ciclo de vida de uma atividade [110].

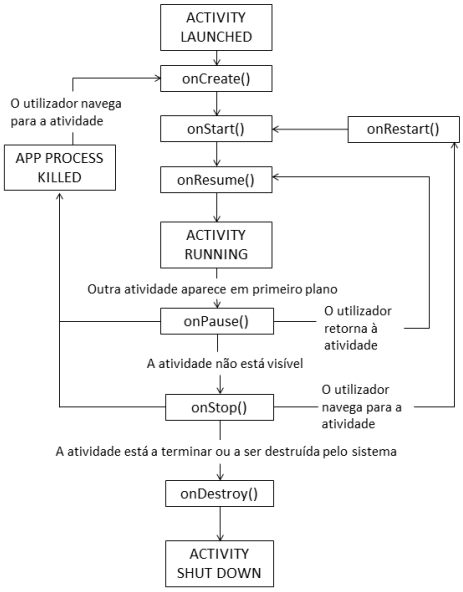

Figura 3.9 - Ciclo de vida de uma atividade.

Quando a atividade é criada pela primeira vez chama-se o *onCreate()*. É aqui que se faz toda a configuração estática, como, por exemplo, criar exibições. Ele recebe um *Bundle* (pacote) que contém o estado anterior da atividade. Na aplicação desenvolvida, são definidos os valores iniciais de algumas variáveis globais criadas.

Considerou-se necessário criar uma variável booleana chamada ***live*** que indica se estamos na página principal ou não. Desta forma, quando a variável está a *true*, quer dizer que estamos na página principal e se está a *false* é porque nos encontramos numa outra página. Então, quando o *onCreate()* é chamado, a variável toma o valor verdadeiro.

As variáveis booleanas ***cameraAtiva*** e ***discoveryAtivo*** foram, também, geradas. Inicialmente, elas encontram-se a *false*. As variáveis estão associadas a 4 funções: *CameraIniciar()*, *CameraParar()*, *DiscoveryIniciar()* e *DiscoveryParar()*. Estas funções foram desenvolvidas com o código que se encontra de seguida:

```
public void CameraIniciar()
   {
      if (!cameraAtiva)
      {
         if (flirDevice != null)
         {
            flirDevice.startFrameStream(this);
            cameraAtiva = true;
            findViewById(R.id.batteryLevelprogressBar).setVisibility(View.VISIBLE);
            findViewById(R.id.batteryLevelTextView).setVisibility(View.VISIBLE);
            findViewById(R.id.imageView_overlay).setVisibility(View.VISIBLE);
         }
      }}
```

```
public void CameraParar()
   {
      if (cameraAtiva)
      {
         if (flirDevice != null) {
            flirDevice.stopFrameStream();
            cameraAtiva = false;
         }
      }
   }
```

```
public void DiscoveryIniciar()
   {
      if (!discoveryAtivo)
      {
         Device.startDiscovery(this,this);
         discoveryAtivo = true;
      }
   }
```

```
public void DiscoveryParar()
   {
      if (discoveryAtivo) {
         Device.stopDiscovery();
         discoveryAtivo = false;
      }
   }
```

A *DiscoveryIniciar()* verifica se algum dispositivo da FLIR está conectado ao *smartphone* ou *tablet*. Por seu lado, a *DiscoveryParar()* averigua se o dispositivo foi desconectado.

Por outro lado, sempre que se deseja iniciar/parar a transmissão, em tempo real, de imagens térmicas, chamam-se as funções *CameraIniciar()* e *CameraParar()*, respetivamente.

As linhas de código que se encontram a verde são aquelas que contêm os métodos do SDK da FLIR ONE responsáveis por fazer o que foi mencionado. Assim, ***startDiscovery(Context, Device.Delegate)*** é um método da classe *com.flir.flironesdk.Device* que é chamado para se conectar a um dispositivo FLIR ONE. O método ***stopDiscovery()*** pertence à classe *com.flir.flironesdk.Device* e deve ser chamado para parar de ouvir os dispositivos conectados por USB, evitando que a aplicação atual capte eventos de conexão em segundo plano. O método ***startFrameStream(Device.StreamDelegate)*** da classe *com.flir.flironesdk.Device* deve ser chamado no delegado do dispositivo para iniciar o *frame streaming*. Por último, o método ***stopFrameStream()*** da classe *com.flir.flironesdk.Device* deve ser chamado para parar *frame streaming*.

Ainda no *onCreate()*, é gerado um novo *frameprocessor*. ***FrameProcessor(Context, FrameProcessor.Delegate, Set<RenderedImage.ImageType>)*** é um construtor da classe *com.flir.flironesdk.FrameProcessor* que cria um processador de *frames* que converte *frames* térmicas em imagens que podem ser exibidas.

Os menus *popup* que permitem que o utilizador altere alguns parâmetros da imagem são criados ainda nesta fase. Estes menus serão analisados na secção 3.4.4.1.

O *onCreate()* é sempre seguido pelo *onStart()*. Este é chamado imediatamente antes da atividade se tornar visível para o utilizador.

De seguida, apresenta-se o código desenvolvido no *onStart()*.

```
    @Override
       protected void onStart(){
          super.onStart();
          ProgressBar BatteryLevelprogrBar = (ProgressBar)
findViewById(R.id.batteryLevelprogressBar);
          BatteryLevelprogrBar.setProgress(100);
          TextView BatteryLevelText = (TextView)findViewById(R.id.batteryLevelTextView);
          BatteryLevelText.setText(String.valueOf(100 + "%"));
          thermalImageView = (ImageView) findViewById(R.id.imageView);
          findViewById(R.id.batteryLevelprogressBar).setVisibility(View.INVISIBLE);
          findViewById(R.id.batteryLevelTextView).setVisibility(View.INVISIBLE);
          findViewById(R.id.imageView_overlay).setVisibility(View.INVISIBLE);
          state=1;
          DiscoveryIniciar();
    }
```

Como se pode verificar, a função *DiscoveryIniciar()* é chamada e são tornados invisíveis os indicadores do nível da bateria da câmara FLIR ONE, bem como o *overlay* dos pés.

Se a atividade for para segundo plano, o *onStart()* é seguido pelo *onResume()*. Este é chamado antes de a atividade interagir com o utilizador. Aqui, a atividade está no topo do conjunto de atividades com a entrada do utilizador direcionada para ela.

De seguida, apresenta-se o código desenvolvido no *onResume()*.

```
@Override
    public void onResume(){
        super.onResume();
        if (flirDevice != null && live == true){
            CameraIniciar();
        }
    }
```

Como se pode observar, a função *CameraIniciar()* é chamada se existir algum dispositivo FLIR ONE conectado e se nos encontrarmos na página inicial.

O *onResume()* é sempre seguido pelo *onPause()*. Este é chamado quando o sistema está quase a retomar outra atividade. Pode ser usado para confirmar alterações não guardadas.

De seguida, apresenta-se o código desenvolvido para o *onPause()*.

```
@Override
    public void onPause(){
        super.onPause();
        if (flirDevice != null){
            CameraParar();
        }
    }
```

Como se pode ver, a função *CameraParar()* é chamada se não for encontrado nenhum dispositivo FLIR ONE conectado.

O *onStop()* é seguido do *onPause()* se a atividade ficar invisível ao utilizador. Assim, este é chamado quando a atividade não está mais visível para o utilizador. Isto pode acontecer se ela estiver a ser destruída ou se outra atividade for retomada.

De seguida, apresenta-se o código desenvolvido para o *onStop()*.

```
@Override
    public void onStop() {
        super.onStop();
        CameraParar();
}
```

Como se pode confirmar, a função *CameraParar()* é chamada, pelo que o *frame streaming* é interrompido.

Por último, é chamado o *onDestroy()* antes da atividade ser destruída. É, então, a última chamada que a atividade recebe. Pode ser chamado para finalizar a atividade ou porque o sistema está a destruir temporariamente essa instância da atividade para poupar espaço.

De seguida, apresenta-se o código desenvolvido para o *onDestroy()*.

```
@Override
   public void onDestroy() {
      super.onDestroy();
      flirDevice = null;
      DiscoveryParar();
   }
```

Como se pode observar, a função *DiscoveryParar()* é chamada, deixando, assim, de se ter dispositivos conectados.

Este é o ciclo de vida da atividade principal da aplicação *Android* desenvolvida.

3.4.4.1 - Alteração dos parâmetros (1.1 do WBS)

Nesta subsecção serão mencionados os parâmetros que podem ser alterados e como foi desenvolvido o código que permite essa alteração.

A **paleta de cores falsas** é um dos parâmetros que o utilizador pode alterar. Existem 9 paletas disponíveis: ***Arctic***, ***Coldest***, ***Contrast***, ***Gray***, ***Hottest***, ***Iron***, ***Lava***, ***Rainbow*** e ***Wheel***. A paleta recomendada para utilização clínica, por ajudar na discriminação visual subjetiva, é a *Rainbow* e, por essa razão, é a paleta que está presente, por omissão, na imagem quando a câmara é conectada ao dispositivo móvel, ou seja, esta é a paleta definida. Para isso, utiliza-se o método da classe *com.flir.flironesdk.FrameProcessor* ***setImagePalette(RenderedImage.Palette.Rainbow)*** do SDK da FLIR ONE.

De seguida, apresenta-se uma imagem que exemplifica cada paleta de cores falsas existente [95].

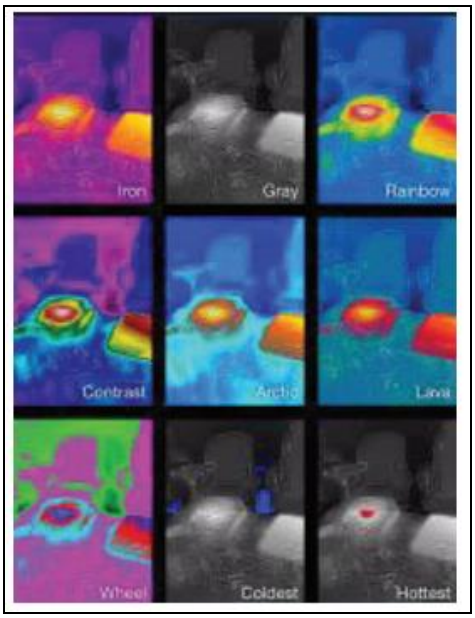

Figura 3.10 - Paletas de cores falsas disponíveis no SDK da FLIR ONE.

Para o utilizador escolher qual a paleta de cores falsas que deseja utilizar, optou-se por se desenvolver um menu *popup*. Quando o utilizador carrega no botão que tem o símbolo de uma paleta, o menu com todas as possibilidades de escolha aparece. Quando se seleciona uma das opções, esta fica de outra cor, a imagem fica automaticamente com a paleta escolhida e aparece uma mensagem de aviso (*Toast*) com o nome da paleta adotada. Foi criada a variável ***pal***, do tipo *RenderedImage.Palette*, onde é guardada a paleta escolhida. Seguidamente, apresenta-se o código criado para este efeito. A linha de código apresentada a verde é onde se define a paleta, depois de esta ser selecionada.

```
    buttonPal.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               PopupMenu popupMenu = new PopupMenu(MainActivity.this, buttonPal);
               popupMenu.getMenuInflater().inflate(R.menu.popup_menu_paleta,
popupMenu.getMenu());

               popupMenu.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener() {
                  @Override
                  public boolean onMenuItemClick(MenuItem item) {
                     if(item.getItemId()==R.id.rainbow){
                        pal = RenderedImage.Palette.Rainbow;
                     } else if(item.getItemId()==R.id.iron){
                        pal = RenderedImage.Palette.Iron;
                     }else if(item.getItemId()==R.id.arctic){
                        pal = RenderedImage.Palette.Arctic;
                     }else if(item.getItemId()==R.id.coldest){
                        pal = RenderedImage.Palette.Coldest;
                     }else if(item.getItemId()==R.id.contrast){
```

```
                    pal = RenderedImage.Palette.Contrast;
                }else if(item.getItemId()==R.id.gray){
                    pal = RenderedImage.Palette.Gray;
                }else if(item.getItemId()==R.id.hottest){
                    pal = RenderedImage.Palette.Hottest;
                }else if(item.getItemId()==R.id.lava){
                    pal = RenderedImage.Palette.Lava;
                }else if(item.getItemId()==R.id.wheel){
                    pal = RenderedImage.Palette.Wheel;
                }else{
                    pal = RenderedImage.Palette.Rainbow;
                }
                frameProcessor.setImagePalette(pal);
                Toast.makeText(MainActivity.this, "" + item.getTitle(),
Toast.LENGTH_SHORT).show();
                return true;
            }
        });
        popupMenu.show();
    }
});
```

Existe ainda um outro método da classe *com.flir.flironesdk.FrameProcessor*, chamado ***getImagePalette()***, que retorna o tipo de paleta de cores falsas que está a ser utilizado.

A **emissividade**, como já foi referido, é uma propriedade de um material que está relacionada com a sua capacidade em emitir energia por radiação, a partir da sua superfície.

O SDK da FLIR ONE tem 4 tipos de emissividade, a ***EMISSIVITY_GLOSSY***, a ***EMISSIVITY_SEMI_GLOSSY***, a ***EMISSIVITY_SEMI_MATTE*** e a ***EMISSIVITY_MATTE***. A emissividade brilhante (EMISSIVITY_GLOSSY) é boa para a leitura de temperatura de materiais como o aço. A emissividade semibrilhante (EMISSIVITY_SEMI_GLOSSY) é boa para a leitura de temperaturas de materiais semirreflexivos. Por seu lado, a emissividade matte é a emissividade padrão, segundo a FLIR, e é útil para a leitura de temperaturas da pele e madeiras. Os valores destas constantes apresentam-se na tabela 3.4.

Tabela 3.4 - Tipos de emissividade disponíveis no SDK da FLIR ONE.

Tipo de emissividade	Valor
EMISSIVITY_GLOSSY	0.30000001192092896f
EMISSIVITY_SEMI_GLOSSY	0.6000000238418579f
EMISSIVITY_SEMI_MATTE	0.800000011920929f
EMISSIVITY_MATTE	0.949999988079071f

A emissividade da pele é 0.98 e, por isso, este é o valor definido, por omissão. Como se verifica na tabela 3.4, não existe nenhum tipo de emissividade predefinido no SDK da FLIR ONE cujo valor seja 0.98. A função do SDK da FLIR ONE que permite alterar este parâmetro é a ***setEmissivity(float)***. Por seu lado, a função ***getEmissivity()*** retorna o valor de emissividade que está a ser utilizado. Ambas são métodos da classe *com.flir.flironesdk.FrameProcessor*.

Para o utilizador conseguir escolher o valor de emissividade que deseja definir, optou-se por se chamar uma nova atividade denominada *ChangeSettings* que contém uma *seekbar* capaz de selecionar o valor pretendido, uma vez que a gama de valores que este parâmetro pode tomar varia entre 0 e 1.

Quando o utilizador retorna o valor por si selecionado, aparece uma mensagem de aviso (*Toast*) com o valor de emissividade adotado.

O código seguinte é o que foi desenvolvido para se proceder à alteração do parâmetro emissividade e está dividido em três partes: a parte referente à chamada da segunda atividade, a que diz respeito ao envio do valor escolhido (na segunda atividade) e a definição do valor de emissividade escolhido (na primeira atividade). Estas secções de código encontram-se apresentadas na ordem referida. A linha de código apresentada a verde, na última secção do código, é onde se define o valor da emissividade, depois deste ser selecionado.

```
@Override
   public boolean onOptionsItemSelected(MenuItem item) {
      live = false;
      Intent c_settings_intent = new Intent(MainActivity.this, ChangeSettings.class);
      startActivityForResult(c_settings_intent,9);
      return super.onOptionsItemSelected(item);
   }
public void passSettings(View view) {
      Intent data = new Intent();
      data.putExtra("emi",Math.round(emi*100));
      setResult(RESULT_OK,data);
      finish();
   }
```

```
public void onActivityResult(final int requestCode, int resultCode, Intent data){
         if(requestCode == 9){
         if(resultCode!=RESULT_OK){
            live = true;
         } else{
            float value = data.getIntExtra("emi",0);
            value = value/100;
            frameProcessor.setEmissivity(value);
            Toast.makeText(MainActivity.this, "Emissividade alterada para: " +
frameProcessor.getEmissivity() + " com sucesso!" , Toast.LENGTH_SHORT).show();
            live = true; }}}
```

O **tipo de imagens** é o último dos parâmetros que o utilizador pode alterar a ser analisado. Os formatos de saída de *frames* suportados pelo SDK da FLIR ONE, ou seja, os tipos de imagem possíveis, são os que estão presentes na tabela 3.5:

Tabela 3.5 - Tipos de imagem disponíveis no SDK da FLIR ONE.

Tipos de imagem
BlendedMSXRGBA8888Image
MSX (térmica + visual) Dados de imagem RGBA
ThermalLinearFlux14BitImage
Dados de imagem lineares de 14 bits, preenchidos para 16 bits por *pixel*
ThermalRadiometricKelvinImage
Dados de temperatura do *centikelvin* radiométrico (cK)
ThermalRGBA8888Image
Dados de imagem RGBA térmica
VisualJPEGImage
Dados de imagem do Visual JPEG
VisualYCbCr888Image
Dados de imagem do Visual YCbCr
VisibleAlignedRGBA8888Image
Dados de imagem Visual RGBA

Optou-se por se utilizar apenas três tipos de imagens: ***BlendedMSXRGBA8888Image***, ***ThermalRGBA8888Image*** e ***VisibleAlignedRGBA8888Image***. Desta forma, o utilizador pode escolher capturar uma imagem térmica, MSX ou visível.

Como o objetivo principal é capturar imagens térmicas, o tipo de imagem definido, por omissão, é *ThermalRGBA8888Image*. Para o utilizador escolher que tipo de imagem deseja usar, decidiu-se criar, novamente, um menu *popup*. Foi criada a variável ***tipo***, do tipo *RenderedImage.ImageType*, onde é guardado o tipo de imagem escolhido.

Quando o utilizador seleciona um tipo, esta opção muda de cor e aparece uma mensagem de aviso (*Toast*) com o tipo de imagem adotado.

O código seguinte ilustra o que foi mencionado e, desta forma, como se altera este parâmetro.

```
    buttonVis.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               final PopupMenu popupMenu = new PopupMenu(MainActivity.this, buttonVis);
               popupMenu.getMenuInflater().inflate(R.menu.popup_menu_vis,
popupMenu.getMenu());
               popupMenu.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener() {
                  @Override
                  public boolean onMenuItemClick(MenuItem item) {
                     findViewById(R.id.button).setVisibility(View.VISIBLE);
                     if(item.getItemId()==R.id.one){
                        tipo = RenderedImage.ImageType.ThermalRGBA8888Image;
                     } else if(item.getItemId()==R.id.two){
                        tipo = RenderedImage.ImageType.VisibleAlignedRGBA8888Image;
                        findViewById(R.id.button).setVisibility(View.INVISIBLE);
                     }else if(item.getItemId()==R.id.three){
                        tipo = RenderedImage.ImageType.BlendedMSXRGBA8888Image;
                     }else{
                        tipo = RenderedImage.ImageType.ThermalRGBA8888Image;
                     }
                     frameProcessor.setImageTypes(EnumSet.of(tipo));
          Toast.makeText(MainActivity.this, "" + item.getTitle(),
Toast.LENGTH_SHORT).show();
                     return true;
                  }
               });
               popupMenu.show();
            }
         });
```

A linha de código apresentada a verde é onde se define o tipo de imagem, depois deste ser selecionado. O método usado é ***setImageTypes(Set<RenderedImage.ImageType>)*** que pertence à classe *com.flir.flironesdk.FrameProcessor*.

É de salientar que quando o tipo de imagem escolhido é visível, o botão que permite escolher a paleta de cores falsas que se deseja utilizar fica invisível. Não faz sentido alterar a paleta de cores falsas de uma imagem visível e, por isso, este desaparece para não confundir o utilizador.

3.4.4.2 - Captação da imagem (1.2 do WBS)

Para se poder capturar imagens térmicas é necessário conectar a câmara FLIR ONE ao *smartphone* ou *tablet*. Existem dois métodos da interface com.flir.flironesdk.Device.Delegate que são muito importantes nesta fase. São eles o ***onDeviceConnected(Device)***, que é chamado quando o dispositivo da FLIR tiver concluído a conexão e estiver pronto para a transmissão, e o ***onDeviceDisconnected(Device)***, que é chamado quando o dispositivo foi desconectado por algum motivo.

Já com a câmara FLIR ONE conectada ao *tablet*, o primeiro passo para se poder captar imagens é fazer a transmissão de imagens térmicas, em tempo real, no ecrã. Para isso, escreveu-se o seguinte código:

```
    @Override
        public void onFrameProcessed(final RenderedImage renderedImage) {
            final Bitmap imageBitmap = Bitmap.createBitmap(renderedImage.width(),
renderedImage.height(), Bitmap.Config.ARGB_8888);
            imageBitmap.copyPixelsFromBuffer(ByteBuffer.wrap(renderedImage.pixelData()));
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    thermalImageView.setImageBitmap(imageBitmap);

                }
            });
```

É de salientar que o *thermalImageView* do código acima corresponde à *ImageView* criada para mostrar o que a câmara capta em tempo real.

Para capturar a imagem, declarou-se uma variável global booleana chamada ***imageCaptureRequested*** que inicialmente tem o valor falso.

Quando o utilizador carrega no botão definido para capturar imagens, essa variável passa a verdadeiro e o seguinte código é executado:

```
            if (this.imageCaptureRequested) {
                imageCaptureRequested = false;
                final Context context = this;
                new Thread(new Runnable() {
                    public void run() {
                        String path = Environment.getExternalStoragePublicDirectory
(Environment.DIRECTORY_PICTURES).toString();
                        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd-HH-mm-ssZ",
Locale.getDefault());
                        String formatedDate = sdf.format(new Date());
                        String fileName = "JTDFT-" + formatedDate + ".jpg";
                        try{
                            lastSavedPath = path+ "/" + fileName;
                            renderedImage.getFrame().save(new File(lastSavedPath), pal, tipo);
                            runOnUiThread(new Runnable() {
                                @Override
```

```
                        public void run() {
                            Toast.makeText(MainActivity.this, "Guardada " + lastSavedPath,
Toast.LENGTH_SHORT).show();
                        }
                    });

                    MediaScannerConnection.scanFile(context,
                            new String[]{path + "/" + fileName}, null,
                            new MediaScannerConnection.OnScanCompletedListener() {
                                @Override
                                public void onScanCompleted(final String path,final Uri uri) {
                                    runOnUiThread(new Runnable() {
                                        @Override
                                        public void run() {
                                            Toast.makeText(MainActivity.this, "\"Guardada \" + path +
\"\\n uri=\" + uri" + lastSavedPath, Toast.LENGTH_SHORT).show();
                                        }
                                    });
                                }

                            });
                }catch (Exception e){
                    e.printStackTrace();
                }
    }
```

Analisando o código, vemos que a variável booleana ***imageCaptureRequested*** volta a ter o valor falso. Aqui definimos qual o nome e formato de como a imagem vai ser guardada e qual o seu destino, ou seja, em que pasta esta pode ser encontrada.

As imagens são guardadas consoante o seu tipo e a paleta de cores falsas escolhidas (*pal*, *tipo*), o que já foi explicado anteriormente, quando se referiu a alteração de parâmetros.

Foi colocada uma mensagem de aviso (*Toast*) com o caminho para se chegar à imagem. Desta forma, o utilizador sabe onde esta está armazenada.

A linha de código que se encontra a verde representa as funções do SDK da FLIR ONE que permitem retornar e guardar a imagem capturada. Assim, ***getFrame()*** é um método da classe *com.flir.flironesdk.RenderedImage* que retorna o objeto original do qual a imagem renderizada foi criada e ***save(File, RenderedImage.Palette, RenderedImage.ImageType)*** é um método da classe *com.flir.flironesdk.Frame* que guarda um ficheiro JPG térmico.

Para ajudar o utilizador a perceber que captou uma imagem, para além da mensagem de aviso, foi ainda adicionada uma animação que faz desaparecer e reaparecer a imagem, numa orientação vertical. De seguida, apresenta-se o código desenvolvido para criar a animação referida.

```
    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {

thermalImageView.animate().setDuration(500).scaleY(0).withEndAction((new Runnable() {
                                public void run() {
                                    thermalImageView.animate().setDuration(500).scaleY(1);
                                }
                            }));
                        }
                    });
                }
            }).start();
```

3.4.4.3 - Formulário dos dados (1.3 do WBS)

Para além de capturar a imagem, a aplicação tem um formulário que deve ser preenchido com os dados clínicos e pessoais do doente: número do processo, idade, índice de massa corporal (IMC), sexo e também as zonas dos pés que foram amputadas (zonas nulas).

Quando o utilizador escolhe a imagem que pretende enviar ou armazenar para posterior análise, clica num botão que tem o símbolo de uma carta e é remetido para uma nova atividade que contém o formulário a preencher.

Um *RadioButton* só permite escolher uma das opções, ao passo que nas *CheckBoxe*s podemos escolher mais do que uma hipótese. Assim, o formulário é composto maioritariamente por *EditTexts*, ou seja, por campos de preenchimento, mas o sexo é escolhido através de um *RadioButton* e as zonas nulas através de *CheckBoxes*.

Os dados referentes às zonas nulas são armazenados num *array* que contém 24 elementos (os primeiros 12 referentes ao pé direito e os restantes 12 relativos ao pé esquerdo). Se uma zona do pé não existe, é assinalada como nula, através de um clique, e o seu elemento correspondente aparece como *true*. Pelo contrário, quando não é assinalada uma zona nula, o seu elemento correspondente aparece como *false*.

As regiões de interesse correspondentes às zonas a sinalizar, estão ilustradas na figura 2.4.

3.4.4.4 -- Ver nível de bateria (2.1 do WBS)

A câmara FLIR ONE tem uma bateria interna, que dura cerca de 1 hora. Decidiu-se criar um indicador do nível da bateria, para o utilizador se manter informado. Para isso utilizou-se uma barra de progresso com a percentagem respetiva numericamente escrita.

De seguida encontra-se o código desenvolvido com este intuito.

```
    @Override
        public void onBatteryPercentageReceived(final byte percentage) {
            final ProgressBar BatteryLevelprogrBar = (ProgressBar)
findViewById(R.id.batteryLevelprogressBar);
            final TextView BatteryLevelText = (TextView)
findViewById(R.id.batteryLevelTextView);
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    BatteryLevelprogrBar.setProgress((int)percentage);
                    BatteryLevelText.setText(String.valueOf(((int)percentage) + "%"));
                }
            });
    }
```

O método usado é o ***onBatteryPercentageReceived(byte)*** que pertence à interface com.flir.flironesdk.Device.PowerUpdateDelegate. Este é chamado sempre que a percentagem de carga da bateria se altera.

Se a bateria estiver a menos de 15%, aparecerá uma mensagem de alerta.

Existe, ainda, um método que indica quando ocorreu alguma alteração ao estado da carga da bateria. O método ***onBatteryChargingStateReceived(Device.BatteryChargingState)*** da interface *com.flir.flironesdk.Device.PowerUpdateDelegate* é chamado sempre que o estado de carga da bateria se altera.

A tabela 3.6 mostra quais os estados de carga da bateria existentes.

Tabela 3.6 - Estados de carga da bateria disponíveis no SDK da FLIR ONE.

Estados de carga da bateria
BAD
Indica que um estado de carregamento da bateria RBPDevice válido não estava disponível
CHARGING_SMART_PHONE_FAULT_HEAT
Indica que existe uma falha no carregamento, mas que o *iPhone* está a ser carregado
CHARGING_SMART_PHONE_ONLY
Indica que o dispositivo está no modo *phone-charging-only*
FAULT
Indica que uma falha inesperada ocorreu durante o carregamento (*bad battery*, etc.)
FAULT_BAD_CHARGER
Indica que ocorreu uma falha de carga devido à baixa corrente da fonte de carga
FAULT_HEAT
Indica que ocorreu uma falha de calor de carregamento
MANAGED_CHARGING
Indica que a bateria RBPDevice está a ser carregada a partir de uma fonte de alimentação externa
MANAGED_CHARGING_ONLY
Indica que o dispositivo está no modo *charge-only*

NO_CHARGING
Indica que a bateria RBPDevice não está a ser carregada porque o RBPDevice não está conectado a uma fonte de alimentação externa

De seguida encontra-se o código desenvolvido neste método.

```
    @Override
       public void onBatteryChargingStateReceived(final Device.BatteryChargingState
batteryChargingState) {
          runOnUiThread(new Runnable() {
             @Override
             public void run() {
                TextView chargingIndicator =
(TextView)findViewById(R.id.batteryLevelTextView);
                ProgressBar chargingIndicator2 =
(ProgressBar)findViewById(R.id.batteryLevelprogressBar);
                if (originalChargingIndicatorColor == null){
                   originalChargingIndicatorColor = chargingIndicator.getTextColors();
                }
                switch (batteryChargingState) {
                   case FAULT:
                   case FAULT_HEAT:
                      chargingIndicator.setTextColor(Color.MAGENTA);
               chargingIndicator2.setProgressTintList(ColorStateList.valueOf(Color.MAGENTA));
                      break;
                   case FAULT_BAD_CHARGER:
                      chargingIndicator.setTextColor(Color.DKGRAY);
    chargingIndicator2.setProgressTintList(ColorStateList.valueOf(Color.DKGRAY));
                   case MANAGED_CHARGING:
                      chargingIndicator.setTextColor(Color.GREEN);

chargingIndicator2.setProgressTintList(ColorStateList.valueOf(Color.GREEN));
                      break;
                   case NO_CHARGING:
                      chargingIndicator.setTextColor(originalChargingIndicatorColor);
                      chargingIndicator2.setProgressTintList(originalChargingIndicatorColor);
                      break;
                   default:
                      break;
                }}});}
```

Como se pode verificar, os estados da tabela que foram utilizados são: ***FAULT***, ***FAULT_HEAT***, ***FAULT_BAD_CHARGER***, ***MANAGED_CHARGING***, ***NO_CHARGING***. Foi definido um código de cores que identifica o estado de carregamento da bateria.

A tabela 3.7 mostra o estado correspondente a cada cor.

Tabela 3.7 - Código de cores adotado para o indicador do estado da bateria.

Cor	Estado
Cinzento-claro	Normal
Verde	A carregar
Rosa	Falha devido ao calor ao carregar
Cinzento-escuro	Não está a carregar porque ocorreu uma falha devido à baixa corrente da fonte de energia
Vermelho	A bateria é igual ou inferior a 15%

3.4.4.5 - Ver imagem (2.2 do WBS)

A aplicação permite que se abram e vejam imagens diretamente da galeria do dispositivo. Desta forma, é mais simples escolher a imagem que se deseja guardar e analisar.

O código que se encontra de seguida é aquele que foi realizado com o intuito de abrir uma imagem na aplicação.

```
    public void openImg(View view){
          CameraParar();
          live = false;
          Uri fold = Uri.parse (Environment.getExternalStoragePublicDirectory
(Environment.DIRECTORY_PICTURES).toString());
          Intent i = new Intent(Intent.ACTION_GET_CONTENT);
          i.setDataAndType(fold,"image/*");
          startActivityForResult(i,18);
       }

    public void onActivityResult(final int requestCode, int resultCode, Intent data){
          if(requestCode == 18) {
             if (resultCode != RESULT_OK) {
                live = true;
             } else {
                Uri img = data.getData();
                final String str = img.toString();
                ((ImageView) findViewById(R.id.imageView2)).setImageURI(img);
                findViewById(R.id.imageView2).setVisibility(View.VISIBLE);
          }
    }
```

Depois de vista, a imagem pode ser guardada numa base de dados (o que vai ser analisado noutra fase), pode abrir-se uma outra imagem, carregando-se, novamente, no rolo da câmara ou pode voltar-se à página principal para se capturar uma nova imagem.

Para se fechar a imagem e voltar à página principal, deve colocar-se a variável ***live*** a *true* e chamar a função ***CamaraIniciar()***.

3.4.4.6 - Armazenamento local (3.1 do WBS)

No desenvolvimento de uma aplicação é crucial ter em consideração o armazenamento de dados, ou seja, como é que os dados vão ser guardados. Sem isso, os dados só estão disponíveis durante o tempo de execução do programa, pelo que quando este termina todos os dados são perdidos.

Depois de capturar e escolher uma imagem e de preencher os dados do formulário, optou-se, então, por permitir que estes possam ser armazenados numa base de dados local, no dispositivo.

Por todas as vantagens mencionadas na secção 2.10 e, principalmente, porque a biblioteca geral do *Android* já suportava o *SQLite*, este foi o componente de *software* eleito para guardar os dados localmente, no dispositivo.

A atividade do formulário tem um botão que permite adicionar os dados à base de dados e outro que permite visualizar os dados que já lá estão armazenados.

Foi necessário criar uma nova atividade que é a responsável por criar e atualizar a base de dados. Esta estende a classe ***SQLiteOpenHelper*** responsável por fornecer os métodos ***getReadableDatabase()*** e ***getWriteableDatabase()***, que permitem aceder à base de dados *SQLite*, quer para leitura, quer para escrita.

A tabela criada tem 7 colunas. A primeira é um ID que é incrementado sempre que se acrescenta uma nova linha, as 5 seguintes são referentes aos dados clínicos e pessoais do doente (número do processo, idade, IMC, sexo e zonas nulas) e a última coluna é onde se coloca o nome e o caminho da imagem.

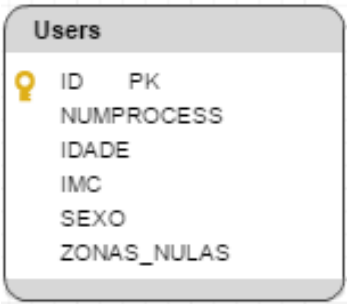

Figura 3.11 - Diagrama entidade relação da base de dados.

Para criar a base de dados, o código desenvolvido foi o seguinte:

```
    @Override
       public void onCreate(SQLiteDatabase db) {
          String createTable = "CREATE TABLE " + TABLE_NAME + " (ID INTEGER PRIMARY KEY
AUTOINCREMENT, " +
                " NUMPROCESS TEXT, IDADE TEXT, IMC TEXT, SEXO TEXT, ZONAS TEXT, URI
TEXT)";
          db.execSQL(createTable);
       }
```

Para adicionar os dados à base de dados, o código desenvolvido foi o seguinte:

```
    public boolean addData(String numProcess, String idade, String imc, String sexo, String
zona, String uri) {
          SQLiteDatabase db = this.getWritableDatabase();
          ContentValues contentValues = new ContentValues();
          contentValues.put(COL2, numProcess);
          contentValues.put(COL3, idade);
          contentValues.put(COL4, imc);
          contentValues.put(COL5, sexo);
          contentValues.put(COL6, zona);
          contentValues.put(COL7, uri);
          long result = db.insert(TABLE_NAME, null, contentValues);
          //if date as inserted incorrectly it will return -1
          if (result == -1) {
             return false;
          } else {
             return true;
         }
    }
```

Se os dados não forem inseridos na base de dados porque ocorreu algum erro, aparecerá uma mensagem de aviso (*Toast*). Por seu lado, se os dados forem inseridos com sucesso, aparecerá uma mensagem de aviso (*Toast*) a indicar o mesmo.

Foi gerada uma nova atividade que permite a visualização dos dados presentes na base de dados. Pode, também, passar-se o ficheiro que contém a tabela criada para um computador, por USB, e abrir a tabela num ecrã maior, com a ajuda do *SQLiteManager*.

O código desenvolvido para mostrar a base de dados foi o que se encontra de seguida.

```
    @Override
       protected void onCreate(@Nullable Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.viewcontents);
          myDB = new DatabaseHelper(this);
          userList = new ArrayList<>();
          Cursor data = myDB.getListContents();
          int numRows = data.getCount();
          if(numRows == 0){
             Toast.makeText(ViewListContents.this,"The Database is empty
:(.",Toast.LENGTH_LONG).show();
          }else{
             int i=0;
             while(data.moveToNext()){
                user = new User(data.getString(1),data.getString(2),data.getString(3),
data.getString(4), data.getString(5), data.getString(6));
                userList.add(i,user);
                System.out.println(data.getString(1)+" "+data.getString(2)+"
"+data.getString(3)+" "+data.getString(4)+" "+data.getString(5)+" "+data.getString(6));
                System.out.println(userList.get(i).getNumProcess());
                i++;
             }
    }
```

Como se pode analisar, na tabela que o utilizador visualizará não se expõe a coluna do ID, mostrando-se todas as restantes.

3.4.4.7 - Envio dos dados (3.2 do WBS)

Em alternativa a guardar os dados e a imagem numa base de dados local, no dispositivo, estes podem ser enviados para um *webservice* e serem lá armazenados para posterior análise.

Optou-se por se desenvolver um *webservice RESTful*, uma vez que este é menos pesado e mais flexível e simples de usar. Embora o SOAP apresente um nível de segurança elevado, o *RESTful* utiliza ao máximo o protocolo HTTP, evitando a construção de protocolos adicionais. Permite formatos de dados muito diferentes, nomeadamente JSON que é melhor para transferência de dados e mais rápido, enquanto que o SOAP apenas permite XML.

A linguagem escolhida para o desenvolver foi *python*, usando a biblioteca *tornado*, e utiliza-se HTTP e JSON para passar a informação.

Assim, foi necessário fazer o *download* e instalação do *python3*, acedendo a https://www.python.org/downloads/. Depois de instalado, foi preciso instalar a biblioteca *tornado*, no terminal do *Windows*.

O *webservice* responde a dois tipos de pedidos, *submit* e *report*. O código desenvolvido para responder a cada um destes pedidos encontra-se exposto, de seguida.

```
class ReportHandler(tornado.web.RequestHandler):
    def get(self, basicauth_user, basicauth_pass):
        if not validateAuth(str(basicauth_user), str(basicauth_pass)):
            return

        global db_conn
        c = db_conn.cursor()
        c.execute('SELECT * FROM users_data')
        output = io.open("report.html", "r", encoding="utf-8").read()
        output_table = ['<table>']
        for line in c.fetchall():
            output_table.append('<tr>')
            for cell in line[1:]:
                output_table.append('<td>')
                output_table.append(cell)
                output_table.append('</td>')
            output_table.append('</tr>')
        output_table.append('</table>')
        self.write(output.format(report=".join(output_table)))

class SubmitHandler(tornado.web.RequestHandler):
    def post(self):
        if db_conn == None:
            self.write('')
            return

        c = db_conn.cursor()
        data = tornado.escape.json_decode(self.request.body)
        cols = []
        values = []
        for col in data:
            cols.append(col)
            values.append('\"+data[col]+'\")
        sql = "INSERT INTO users_data (%s) VALUES(%s)" % (','.join(cols), ','.join(values))
        c.execute(sql)
        db_conn.commit()

        self.write(json.dumps({'result' : 'ok'}, indent=4, separators=(',', ': ')))
```

Foi ainda criada uma função de autenticação, cujo nome de utilizador é **“*user*”** e a *password* é **“*joana*”**.

Por outro lado, como já referido, é essencial ter em consideração como é que os dados vão ser guardados. Optou-se, novamente, pela utilização do *SQLite*. Assim, os dados são guardados num ficheiro local no servidor, podendo ser, posteriormente analisados.

Para se conseguir visualizar os dados armazenados, foi, ainda, criada uma interface básica que os mostra.

Do lado do *Android*, foi utilizada a biblioteca *volley*. *Volley* é uma biblioteca HTTP que está disponível no *GitHub*. Para usar esta biblioteca é necessário adicionar ***compile 'com.android.volley:volley:1.0.0'*** às dependências do *build.gradle* do projeto *Android*.

De seguida, apresenta-se o código desenvolvido para enviar os dados para o servidor.

```
// TODO: read data
            JSONObject object = new JSONObject();
            try {
               object.put("NUMPROCESS", nProcess);
               object.put("IDADE", idade);
               object.put("IMC", imc);
               object.put("SEXO", sexo);
               object.put("ZONAS", z);
               object.put("URI", img);
            } catch (JSONException e) {
               e.printStackTrace();
            }
```

```
            Log.d("STATE", object.toString());

            RequestQueue queue = Volley.newRequestQueue(getApplicationContext());
            // TODO: address for the localhost in the emulator
            String url = "http://10.0.2.2:8888/submit";
            JsonObjectRequest request = new
JsonObjectRequest(Request.Method.POST,url,
                  object,new Response.Listener<JSONObject>() {
               @Override
               public void onResponse(JSONObject response) {
                  try {
                     Log.d("STATE", response.getString("result"));
                  } catch (JSONException e) {
                     e.printStackTrace();
                  }
               }
            },new Response.ErrorListener() {
```

```
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d("STATE", error.getMessage());
            }
        });

        queue.add(request);
```

Como se pode verificar, é utilizado o método POST.

Salienta-se a necessidade de alterar a linha de código que se encontra a verde se se pretender (como é o caso) utilizar num dispositivo real e não no emulador. Para isso, é necessário saber o IP da zona onde se pretende utilizar a aplicação.

Para usar o servidor, é necessário abrir o terminal do *Windows* e escrever o seguinte comando: **C:\Users\joanasdk\Documents\Servidor>py servidor.py**. Se a pasta que contém o servidor mudar de localização, é necessário alterar o comando. Depois disto, o servidor está apto a receber pedidos.

Para se visualizar os dados que estão no servidor, basta aceder a **localhost:8888/** e no URL acrescentar o pedido, ou seja, **localhost:8888/report**.

É de notar que no ficheiro local, que contém a base de dados, presente no disco, só é possível ver os dados inseridos por esse dispositivo móvel. Por seu lado, no ficheiro local que está presente no servidor, é possível observar os dados enviados por mais do que um dispositivo.

3.4.4.8 - Receção do relatório (4.1 do WBS)

Uma vez que os dados não serão analisados no âmbito desta dissertação, esta tarefa fica sem efeito. No entanto, para a realizar, seria necessário utilizar o método GET.

3.5 - Casos de Uso

Cada caso de uso corresponde a uma operação que o utilizador pode executar num sistema, neste caso, na aplicação. Na figura seguinte, apresenta-se este conceito de forma esquemática.

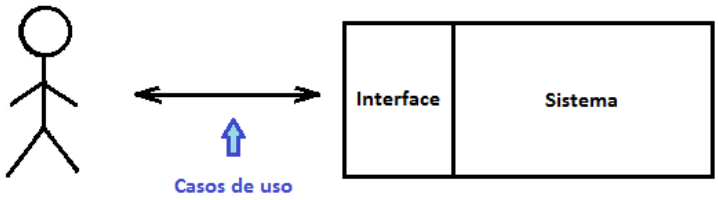

Figura 3.12 - Casos de uso.

De seguida, apresentam-se os casos de uso da aplicação desenvolvida. Recorda-se que na secção " Especificação do sistema", já se enumeraram quais os possíveis utilizadores desta aplicação.

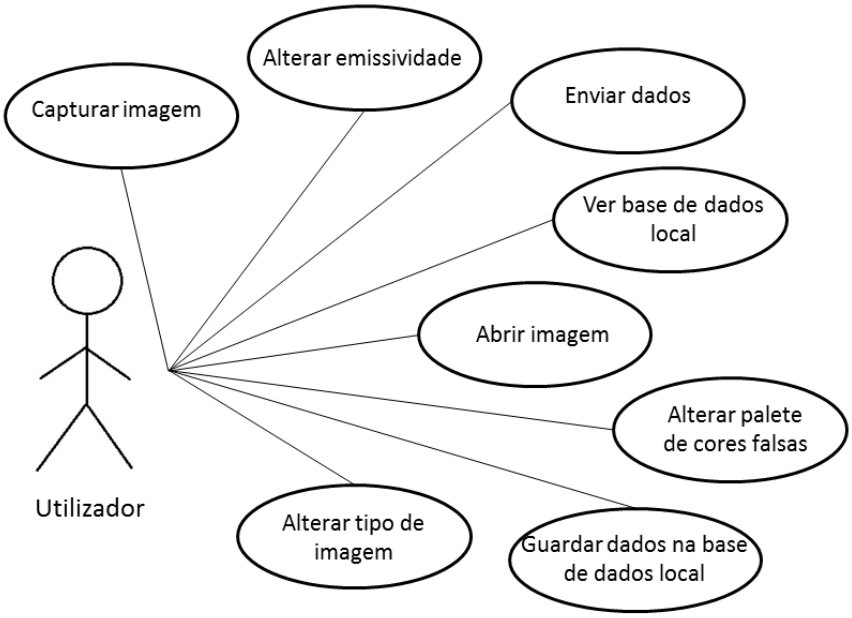

Figura 3.13 - Casos de uso da aplicação *Android*.

Os casos de uso devem ser detalhados e devem descrever a interação pretendida com o utilizador. Por isso e de modo a que o utilizador saiba usar a aplicação para executar cada operação, a tabela 3.8 contém uma breve descrição de cada caso de uso mencionado.

Tabela 3.8 - Descrição dos casos de uso da aplicação *Android*.

Caso de uso	Descrição
Capturar imagem	Para capturar uma imagem, o utilizador deve conectar a câmara FLIR ONE ao dispositivo móvel. Se quiser alterar algum parâmetro, deve fazê-lo antes de capturar a imagem. Para capturar a imagem, basta apontar para o sítio pretendido e carregar no botão central.
Alterar emissividade	Para alterar a emissividade, o utilizador deve ir ao menu das definições e escolher o valor que deseja. De seguida, deve carregar no botão "OK".
Alterar tipo de imagem	Para alterar o tipo de imagem, o utilizador deve carregar no botão que abre um menu

	com as possibilidades de escolha. Depois disso, basta selecionar a opção desejada.
Alterar paleta de cores falsas	Para alterar a paleta de cores falsas, o utilizador deve carregar no botão que abre um menu com as possibilidades de escolha. Depois disso, basta selecionar a opção desejada.
Abrir imagem	Para abrir e ver uma imagem já capturada no ecrã do dispositivo móvel, o utilizador deve carregar no botão do lado direito. Será, então, reencaminhado para a pasta que contém as imagens captadas e deve escolher qual a imagem que pretende abrir. Para concluir esta etapa, basta clicar na imagem selecionada. NOTA: não é necessário ter a câmara FLIR ONE conectada ao dispositivo móvel.
Guardar dados na base de dados local	Para que os dados não sejam perdidos, o utilizador pode guardá-los numa base de dados local. Para isso, depois de escolhida a imagem, aparecerá um botão que permite armazenar os dados localmente. Clicando no botão, aparece um formulário que contém os dados que devem ser preenchidos. Depois de completados todos os campos obrigatórios, o utilizador deve carregar no botão "Adicionar". Para além dos dados, o nome da imagem, bem como o caminho para a encontrar, são, também, guardados na base de dados. NOTA: não é necessário ter a câmara FLIR ONE conectada ao dispositivo móvel.
Ver base de dados local	Para ver os dados já armazenados, o utilizador deve fazer os mesmos passos que no caso de uso "Guardar dados na base de dados local". No entanto, em vez de preencher o formulário e adicionar os dados, deve carregar no botão "Ver". NOTA: não é necessário ter a câmara FLIR ONE conectada ao dispositivo móvel.
Enviar dados	Para enviar os dados para o servidor, o utilizador tem de seguir os mesmo passos explicados no caso de uso "Guardar dados na base de dados local". No entanto, não deve carregar no botão "Adicionar", mas sim no botão "Enviar". NOTA: não é necessário ter a

	câmara FLIR ONE conectada ao dispositivo móvel.

Seguidamente, exibe-se um fluxograma, ou seja, uma representação gráfica, que representa o percurso que o utilizador deve/pode seguir.

É de notar que os dados a serem enviados para uma base de dados, local ou remota, são representados da mesma forma neste fluxograma.

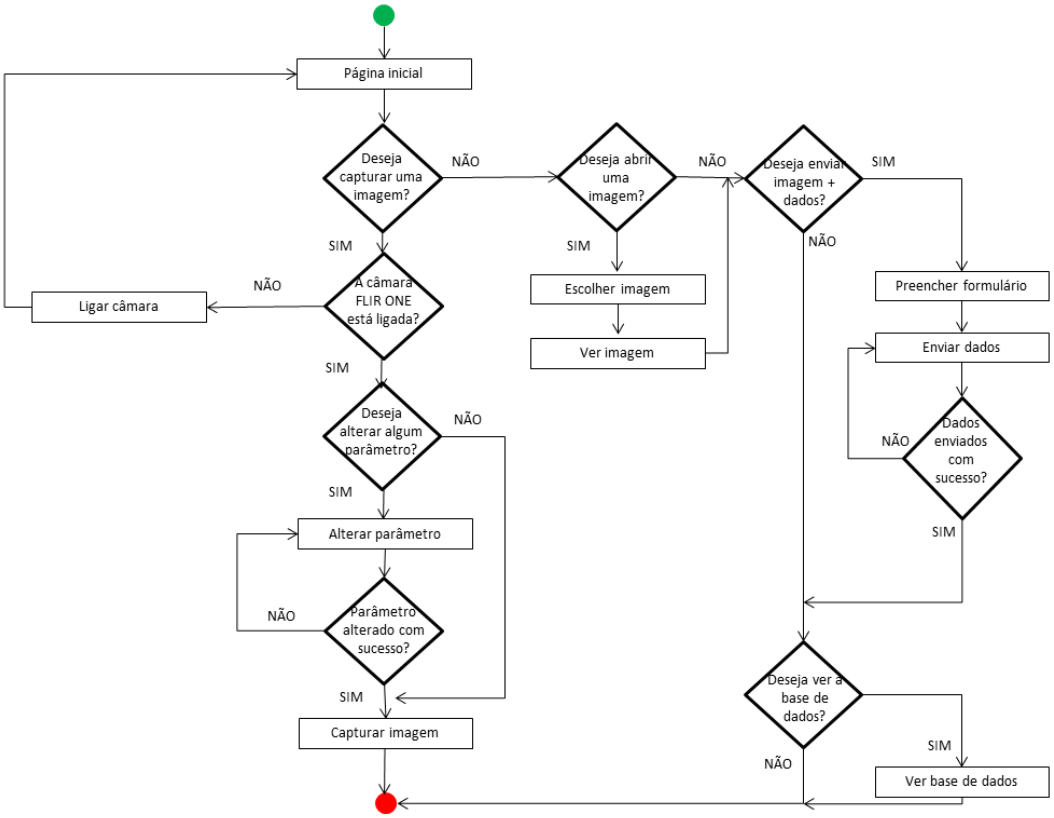

Figura 3.14 - Fluxograma.

Em sumário, foi proposta uma solução com características e funcionalidades mais adequadas à prática clínica, fazendo-se um levantamento dos requisitos, funcionais e não funcionais, necessários. Foi feita a arquitetura do sistema e a especificação do mesmo.

De seguida, fez o estudo de câmaras térmicas e do formato JPG radiométrico.

A partir daqui, iniciou-se o desenvolvimento da aplicação. Para isso dividiu-se o trabalho em pequenas tarefas, nomeadamente a aquisição, a visualização e o processamento, com

mais detalhe na alteração de parâmetros, na captação da imagem, no preenchimento do formulário dos dados e no seu armazenamento local e envio para um *webservice* remoto. Foi, também, trabalhada a possibilidade da visualização da imagem e do nível de bateria da câmara, no ecrã do dispositivo móvel.

Capítulo 4

Testes e Resultados

Neste capítulo apresentam-se os testes efetuados, nomeadamente o estudo de câmaras térmicas da FLIR e do formato JPG radiométrico, bem como a interface final com o utilizador. Faz-se, ainda, o teste da aplicação *Android* desenvolvida, apresentando-se os resultados obtidos e quais as aplicações que resultaram deste projeto.

4.1 - Estudo de câmaras térmicas

Como já referido no capítulo anterior, as câmaras FLIR E60 e FLIR ONE de segunda geração para os sistemas operativos *Android* e *iOS* foram objeto de estudo.

Foram captadas imagens térmicas do corpo negro a 30°C, pelas várias câmaras, de 5 em 5 minutos, para avaliar o *drift* de ligação da câmara (conhecer o seu tempo de estabilização) obtendo-se os resultados que se apresentam na tabela 4.1 e na figura 4.1.

Tabela 4.1 - Leituras de temperatura da fonte de calibração a 30°C.

Time (m)	FLIR E60	FLIR ONE *iOS*	FLIR ONE *Android*
1	30.9	30.9	32.2
5	30.8	30.8	32.3
10	30.7	32	32.5
15	30.7	31.9	32.1
20	30.4	31.2	31.4
25	30.5	31.6	32.6
30	30.2	31.4	31.3
35	30.5	31.3	31.9
40	30.4	31.4	31.9
45	30.3	31.1	32.2
50	30.5	30.8	31.7
55	30.4	31.3	31.4
60	30.3	31.4	31.6

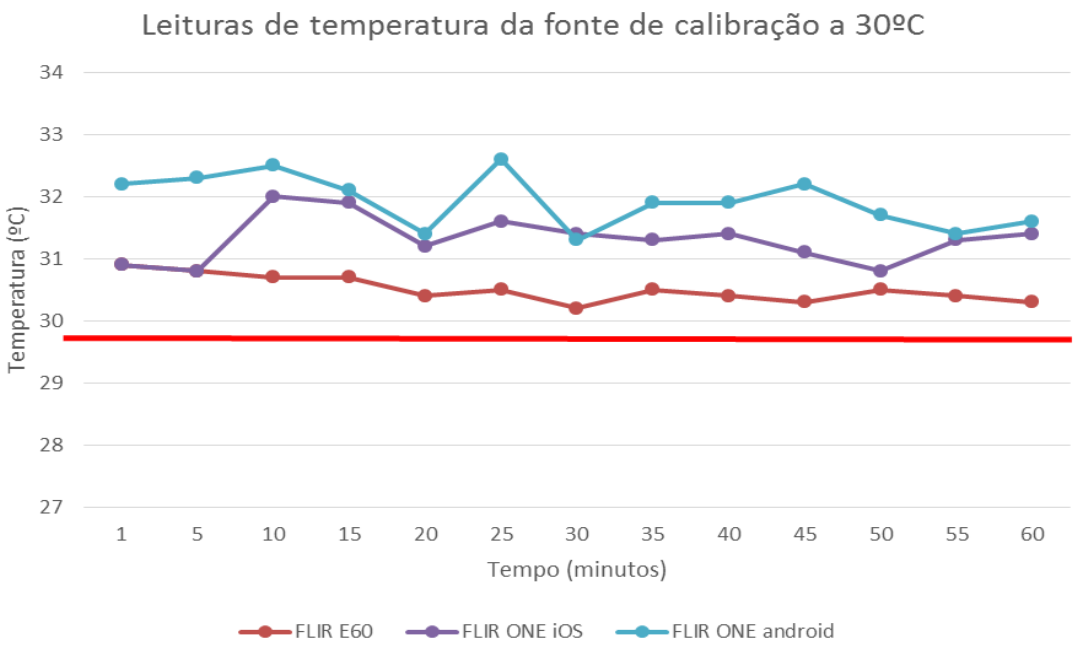

Figura 4.1 - Leituras de temperatura da fonte de calibração a 30°C.

Expressa a vermelho, na figura 4.1, está a linha que representa a temperatura de calibração de 30°C. Pode verificar-se que todos os equipamentos fazem uma sobrestimação do valor real da temperatura.

De seguida, fez-se a determinação da diferença entre a temperatura do corpo negro e a temperatura obtida em cada câmara, para determinar, desta forma, o erro de medição. Os resultados obtidos apresentam-se na tabela 4.2.

Tabela 4.2 - Erros de medição obtidos.

Time (m)	FLIR E60	FLIR ONE *iOS*	FLIR ONE *Android*
1	0.9	0.9	2.2
5	0.8	0.8	2.3
10	0.7	2	2.5
15	0.7	1.9	2.1
20	0.4	1.2	1.4
25	0.5	1.6	2.6
30	0.2	1.4	1.3
35	0.5	1.3	1.9
40	0.4	1.4	1.9
45	0.3	1.1	2.2
50	0.5	0.8	1.7
55	0.4	1.3	1.4
60	0.3	1.4	1.6

Com base nos dados expostos na tabela 4.2, calculou-se qual a média (μ) e desvio padrão (σ) do erro obtido em cada câmara. Os resultados encontram-se na seguinte tabela:

Tabela 4.3 - Média e desvio padrão do erro de medição obtido.

	FLIR E60	FLIR ONE *iOS*	FLIR ONE *Android*
μ	0.51	1.32	1.93
σ	0.21	0.37	0.43

Numa outra análise, foi aumentada a temperatura do corpo negro em 1°C, dos 20°C iniciais para os 29°C e, de seguida, a partir de uma temperatura do corpo negro de 38°C, foi diminuído 1°C até atingir os 30°C. As temperaturas obtidas por cada câmara térmica foram as que se expõem na tabela 4.4.

Tabela 4.4 - Temperaturas obtidas por cada câmara térmica.

Temperatura	FLIR E60	FLIR ONE *iOS*	FLIR ONE *Android*
20	20.2	21.3	22.1
21	21.1	22.4	22.3
22	22.2	23.1	23.1
23	23.1	24	24.1
24	24.2	25.2	24.5
25	25.1	25.3	25.9
26	26.5	26.2	27.2
27	27.5	27.4	27.6
28	28.5	28.3	28.6
29	29.5	30.2	30.3
30	30.3	31.3	31.7
31	31.4	32.9	33.5
32	32.4	34.1	34.8
33	33.4	35	35.7
34	34.5	35.9	36.3
35	35.4	36.7	37.9
36	36.5	37.8	38.8
37	37.5	38.8	39.2
38	38.5	40.2	39.6

De seguida, avaliou-se a diferença entre a temperatura do corpo negro e a obtida em cada câmara, para determinar, desta forma, o erro de medição. Os resultados obtidos apresentam-se na tabela 4.5 e na figura 4.2.

Tabela 4.5 - Erros de medição obtidos.

Temperatura	FLIR E60	FLIR ONE *iOS*	FLIR ONE *Android*
20	0.2	1.3	2.1
21	0.1	1.4	1.3
22	0.2	1.1	1.1
23	0.1	1	1.1
24	0.2	1.2	0.5
25	0.1	0.3	0.9
26	0.5	0.2	1.2
27	0.5	0.4	0.6
28	0.5	0.3	0.6
29	0.5	1.2	1.3
30	0.3	1.3	1.7
31	0.4	1.9	2.5
32	0.4	2.1	2.8
33	0.4	2	2.7
34	0.5	1.9	2.3
35	0.4	1.7	2.9
36	0.5	1.8	2.8
37	0.5	1.8	2.2
38	0.5	2.2	1.6

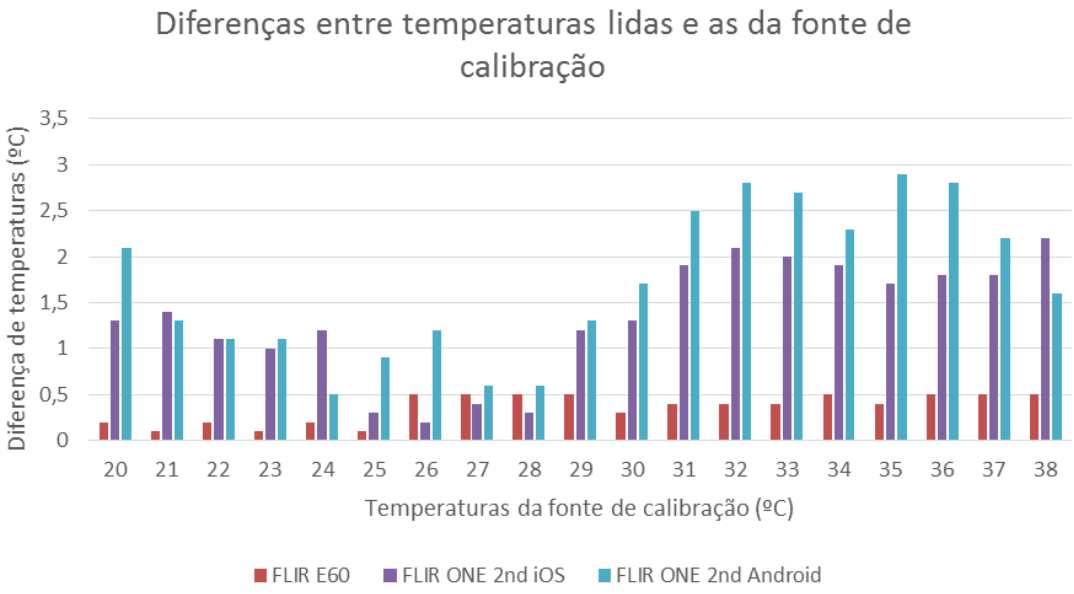

Figura 4.2 - Diferenças entre temperaturas lidas e as da fonte de calibração.

A partir dos dados mostrados na tabela 4.5 e na figura 4.2, determinou-se qual a média (μ), desvio padrão (σ) e os valores máximo e mínimo do erro obtido para cada câmara. Os resultados encontram-se na seguinte tabela:

Tabela 4.6 - Média, desvio padrão e valor máximo e mínimo do erro obtido.

	FLIR E60	FLIR ONE *iOS*	FLIR ONE *Android*
μ	0.36	1.32	1.69
σ	0.16	0.65	0.82
Máx	0.5	2.2	2.9
Mín	0.1	0.2	0.5

Para determinar a intensidade da associação linear entre a temperatura medida por cada câmara térmica e a fonte de calibração, foi calculado o coeficiente de correlação linear de *Pearson*. Nas figuras seguintes, ilustram-se os valores dos coeficientes de correlação obtidos, entre os valores de temperatura determinados pelas várias câmaras e os valores de temperatura definidos pela fonte de calibração.

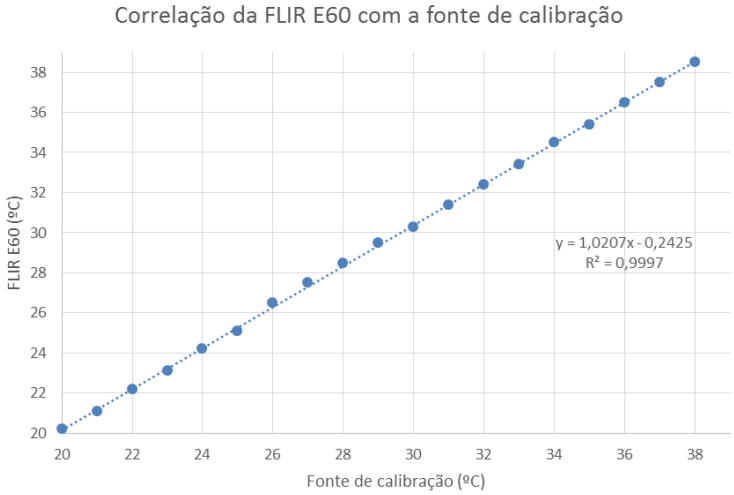

Figura 4.3 - Correlação da FLIR E60 com a fonte de calibração.

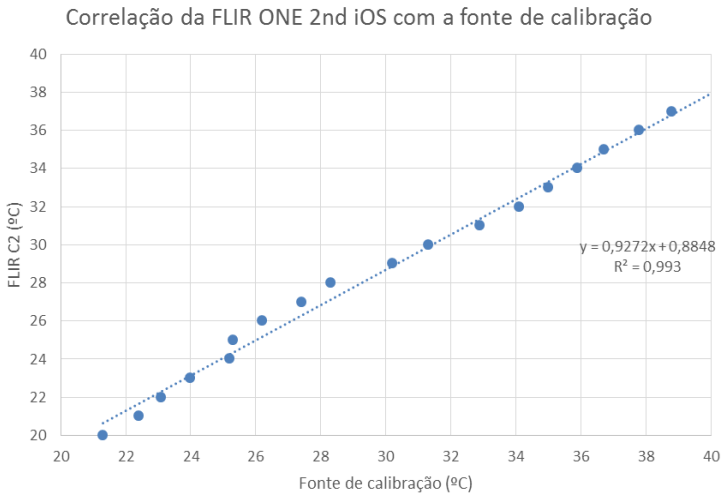

Figura 4.4 - Correlação da FLIR ONE 2nd iOS com a fonte de calibração.

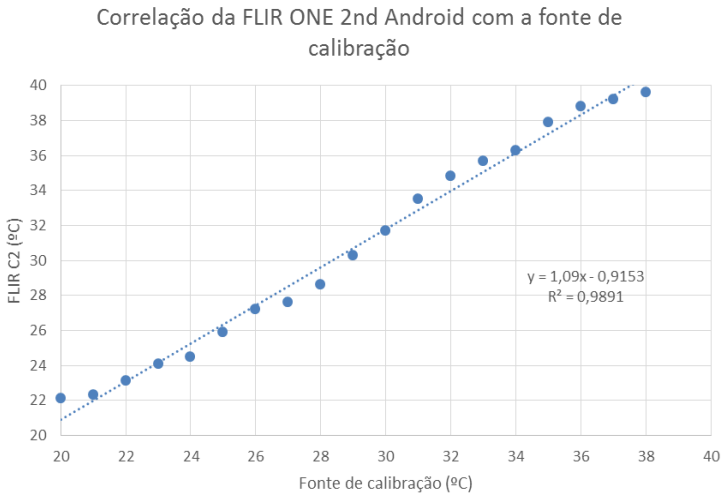

Figura 4.5 - Correlação da FLIR ONE 2nd *Android* com a fonte de calibração.

Por outro lado, foi também avaliado o coeficiente de correlação de *Pearson* entre os valores de temperatura obtidos pela câmara pela câmara FLIR ONE para *iOS* e *Android*. Foram obtidos o valor e o gráfico que se ilustram na figura 4.6.

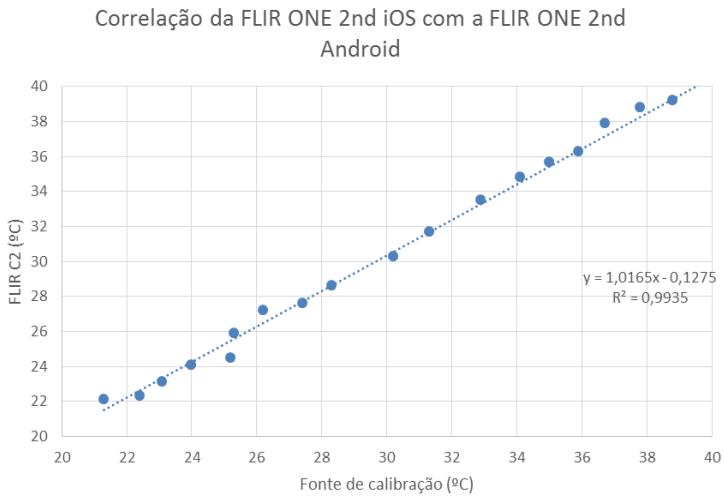

Figura 4.6 - Correlação da FLIR ONE 2nd iOS com a FLIR ONE 2nd Android.

No capítulo seguinte, estes resultados são analisados e discutidos.

4.2 - Estudo do formato JPG radiométrico

Na sequência do que já foi mencionado no capítulo "Metodologia", foi feita a análise do formato JPG radiométrico, propriedade da FLIR® com o intuito de desvincular a imagem térmica da visível e determinar os valores de emissividade, temperatura máxima e mínima, constantes de *Planck* (R1, B, F, O e R2) e o modelo da câmara da FLIR®, por exemplo.

De seguida, com um objetivo meramente exemplificativo, encontra-se uma tabela para uma das imagens estudadas, precedida pela sua respetiva imagem. Na tabela, encontra-se o nome do parâmetro cujo endereço está a ser pesquisado, o valor obtido no *ExifTool*, o valor a procurar (diretamente, em decimal, ou em *float*), o valor a procurar em hexadecimal, o valor em hexadecimal encontrado, assim como o endereço em hexadecimal e decimal encontrado.

Tabela 4.7 - Legenda a ser utilizada nos endereços encontrados para a imagem FLIR1528.

	Diretamente
	Unix + Decimal
	Decimal
	Float

Figura 4.7 - Exemplo de imagem térmica - FLIR1528.

Tabela 4.8 - Endereços encontrados para a imagem FLIR1528.

Nome	*Exiftool*	Valor a Procurar	Valor Hexadecimal	Valor Encontrado	Endereço Hexadecimal	Endereço Decimal
Subject Distance	1 m	100	00 00 00 64	64 00 00 00	00 00 01 C8 - 00 00 01 CB	456 - 459
Image Temp.Max	305	305	00 00 01 31	31 01 00 00	00 00 02 62 - 00 00 02 65	610 - 613
Image Temp.Min	294	294	00 00 01 26	26 01 00 00	00 00 02 6A - 00 00 02 6D	618 - 621
Focal Length	18.0 mm	180	00 00 00 B4	B4 00 00 00	00 00 01 D0 - 00 00 01 D3	464 - 467
Emissivity	0.98	0.98	3F 7A E1 48	48 E1 7A 3F	00 00 11 88 - 00 00 11 8B	4488 - 4491
Object Distance	1.00 m	1.00 m	3F 80 00 00	00 00 80 3F	00 00 11 8C - 00 00 11 8F	4492 - 4495
Reflected Apparent Temp.	20.0 C	20 + 273.15 K	43 92 93 33	33 93 92 43	00 00 11 90 - 00 00 11 93	4496 - 4499
Atmospheric Temperature	25.0 C	25 + 273.15 K	43 95 13 33	33 13 95 43	00 00 11 94 - 00 00 11 97	4500 - 4503
Relative Humidity	50.0 %	0.5	3F 00 00 00	00 00 00 3F	00 00 11 A4 - 00 00 11 A7	4516 - 4519
Planck R1	16358.414	16358.414	46 7F 99 A8	A8 99 7F 46	00 00 11 C0 - 00 00 11 C3	4544 - 4547
Planck B	1424.4	1424.4	44 B2 0C CD	CD 0C B2 44	00 00 11 C4 - 00 00 11 C7	4548 - 4551
Planck F	1	1	3F 80 00 00	00 00 80 3F	00 00 11 C8 - 00 00 11 CB	4552 - 4555
Camera Model	FLIR E60	FLIR E60	46 4C 49 52 20 45 36 30	46 4C 49 52 20 45 36 30	00 00 12 3C - 00 00 12 43	4668 - 4675
Camera Serial Number	64510820	64510820	36 34 35 31 30 38 32 20	36 34 35 31 30 38 32 20	00 00 12 6C - 00 00 12 73	4716 - 4723
Lens Model	FOL18	FOL18	46 4F 4C 31 38	46 4F 4C 31 38	00 00 12 D8 - 00 00 12 DC	4824 - 4828
Planck O	-5815	-5815	FF FF E9 49	49 E9 FF FF	00 00 14 70 - 00 00 14 73	5232 - 5235
Planck R2	0.011263178	0.011263178	3C 38 89 31	31 89 38 3C	00 00 14 74 - 00 00 14 77	5236 - 5239
Raw Value Median	19182	19182	00 00 4A EE	EE 4A 00 00	00 00 14 A0 - 00 00 14 A3	5280 - 5283
Raw Value Range	3246	3246	00 00 0C AE	AE 0C 00 00	00 00 14 A4 - 00 00 14 A7	5284 - 5287
Date/Time Original	2015:05:25 12:33:38.325+00:00	1432557218 + 325	55 63 16 A2. 01 45	A2 16 63 55 45 01	00 00 14 EC - 00 00 14 F1	5356 - 5361
Embedded Image Width	2048	2048	00 00 08 00	00 00 08 00	00 01 79 3B - 00 01 79 3E	96571 - 96574
Embedded Image Height	1536	1536	00 00 06 00	00 00 06 00	00 01 79 47 - 00 01 79 4A	96583 - 96586

Para uma melhor comparação dos endereços dos vários parâmetros de todas as imagens analisadas anteriormente, foi criada uma nova tabela que contém, apenas, os seus endereços decimais e que se apresenta, de seguida.

Tabela 4.9 - Endereços decimais das imagens analisadas.

	FLIR 1528	FLIR 0608	FLIR 7728	FLIR 7646	FLIR 7628	FLIR 7642		flir 20170201T190042	flir 20170201T193756	flir 20170201T193816
	FFF	FFF	FFF	FFF	FFF	FFF		JFIF	JFIF	JFIF
Nome	Endereço Decimal	Endereço Decimal	Endereço Decimal	Endereço Decimal	Endereço Decimal	Endereço Decimal		Endereço Decimal	Endereço Decimal	Endereço Decimal
Subject Distance	456 – 459	456 - 459	456 - 459	456 - 459	456 - 459	456 - 459		430 - 433	430 - 433	430 - 433
Image Temperature Max	610 – 613	610 - 613	610 - 613	610 - 613	610 - 613	610 - 613		584 - 587	584 - 587	584 - 587
Image Temperature Min	618 – 621	618 - 621	618 - 621	618 - 621	618 - 621	618 - 621		592 - 595	592 - 595	592 - 595
Focal Length	464 – 467	464 - 467	464 - 467	464 - 467	464 - 467	464 - 467		438 - 441	438 - 441	438 - 441
Emissivity	4488 – 4491	4638 - 4641	5044 - 5047	4566 - 4569	4840 - 4843	4608 - 4611		51708 - 51711	33432 - 33435	34858 - 34862
Object Distance	4492 – 4495	4642 - 4645	5048 - 5051	4570 - 4573	4844 - 4847	4612 - 4615		51712 - 51715	33436 - 33439	34862 - 34865
Reflected Apparent Temperature	4496 – 4499	4646 - 4649	5052 - 5055	4574 - 4577	4848 - 4851	4616 - 4619		51716 - 51719	33440 - 33443	34866 - 34869
Atmospheric Temperature	4500 – 4503	4650 - 4653	5056 - 5059	4578 - 4581	4852 - 4855	4620 - 4623		51720 - 51723	33444 - 33447	34870 - 34873
Relative Humidity	4516 – 4519	4666 - 4669	5072 - 5075	4594 - 4597	4868 - 4871	4636 - 4639		51736 - 51739	33460 - 33463	34886 - 34889
Planck R1	4544 – 4547	4694 - 4697	5100 - 5103	4622 - 4625	4896 - 4899	4664 - 4667		51764 - 51767	33488 - 33491	34914 - 34917
Planck B	4548 – 4551	4698 - 4701	5104 - 5107	4626 - 4629	4900 - 4903	4668 - 4671		51768 - 51771	33492 - 33495	34918 - 34921
Planck F	4552 – 4555	4702 - 4705	5108 - 5111	4630 - 4633	4904 - 4907	4672 - 4675		51772 - 51775	33496 - 33499	34922 - 34925
Camera Model	4668 – 4675	4818 - 4825	5224 - 5231	4746 - 4753	5020 - 5027	4788 - 4795		51888	33612	35038
Camera Serial Number	4716 – 4723	4866 - 4873	5272 - 5279	4794 - 4801	5068 - 5075	4836 - 4843		51936	33660	35086
Lens Model	4824 – 4828	4974 - 4978	5380 - 5384	4902 - 4906	5176 - 5180	4944 - 4948		52044 - 52047	33768 - 33771	35194 - 35197
Planck O	5232 – 5235	5382 - 5385	5788 - 5791	5310 - 5313	5584 - 5587	5352 - 5355		52452 - 52455	34176 - 34179	35602 - 35605
Planck R2	5236 – 5239	5386 - 5389	5792 - 5795	5314 - 5317	5588 - 5591	5356 - 5359		52456 - 52459	34180 - 34183	35606 - 35609
Raw Value Median	5280 – 5283	5430 - 5433	5836 - 5839	5358 - 5361	5632 - 5635	5400 - 5403		52500 - 52503	34224 - 34227	35650 - 35653
Raw Value Range	5284 – 5287	5434 - 5437	5840 - 5843	5362 - 5365	5636 - 5639	5404- 5407		52504 - 52507	34228 - 34231	35654 - 35657
Date/Time Original	5356 – 5361	5506 - 5511	5912 - 5917	5434 - 5438	5708- 5713	5476 - 5480		52576 - 52580	34300 - 34305	35726 - 35730
Embedded Image Width	96571 - 96574	101777 - 101780	96571 - 96574	96061 - 96064	101959 - 101962	97275 - 97277		5636 - 5639	4596 - 4599	4322 - 4325
Embedded Image Height	96583 - 96586	101789 - 101792	96583 - 96586	96073 - 96076	101971 - 101974	97287 - 97290		5648 - 5651	4608 - 4611	4334 - 4337

Os nomes dos parâmetros que se encontram a verde, na tabela acima, são aqueles que apresentam o mesmo endereço para todas as imagens analisadas, para cada grupo. Ou seja, os dois grupos têm endereços diferentes, mas dentro do mesmo grupo os endereços são iguais.

Para todos os outros parâmetros, foi realizado um novo estudo para se tentar encontrar alguma relação entre os endereços destes.

Começou-se por se subtrair os endereços dos parâmetros para se averiguar se a diferença entre estes era constante em todas as imagens. Os resultados obtidos encontram-se na tabela abaixo.

Tabela 4.10 - Diferença entre os endereços.

Nome	FLIR 1528	FLIR 0608	FLIR 7728	FLIR 7646	FLIR 7628	FLIR 7642		flir 20170201T19 0042	flir 20170201T19 3756	flir 20170201T19 3816
Emissivity - O. Distance	4	4	4	4	4	4		4	4	4
O. Distance - R.A.Temp	4	4	4	4	4	4		4	4	4
R.A.Temp - A. Temp	4	4	4	4	4	4		4	4	4
A. Temp - R. Humidity	16	16	16	16	16	16		16	16	16
R. Humidity - Planck R1	28	28	28	28	28	28		28	28	28
Planck R1 - Planck B	4	4	4	4	4	4		4	4	4
Planck B - Planck F	4	4	4	4	4	4		4	4	4
Planck F - Camera Model	116	116	116	116	116	116		116	116	116
Camera Model - Camera Serial N	48	48	48	48	48	48		48	48	48
Camera Serial N - Lens Model	108	108	108	108	108	108		108	108	108
Lens Model - Planck O	408	408	408	408	408	408		408	408	408
Planck O - Planck R2	4	4	4	4	4	4		4	4	4
Planck R2 - Raw Value Median	44	44	44	44	44	44		44	44	44
R. Value Median - R. Value Range	4	4	4	4	4	4		4	4	4
R. Value Range - Date/Time O.	72	72	72	72	72	72		72	72	72
Date/Time O. - E. Image Width	91215	96271	90659	90627	96251	91799		46940	29704	31404
E. Image Width - E. Image Height	12	12	12	12	12	12		12	12	12

Posteriormente, foi feito um estudo para saber como separar a imagem térmica e a visível.

De seguida, mostram-se as imagens obtidas. Na figura 4.8, a imagem térmica obtida em tons cinzento (*gray*), relativa à imagem exemplo em cima referida e examinada, juntamente com a sua imagem "de origem" e na figura 4.9, a imagem visível obtida, relativa à imagem exemplo em cima referida e examinada, juntamente com as suas imagens "de origem" e térmica, em tons cinzento (*gray*).

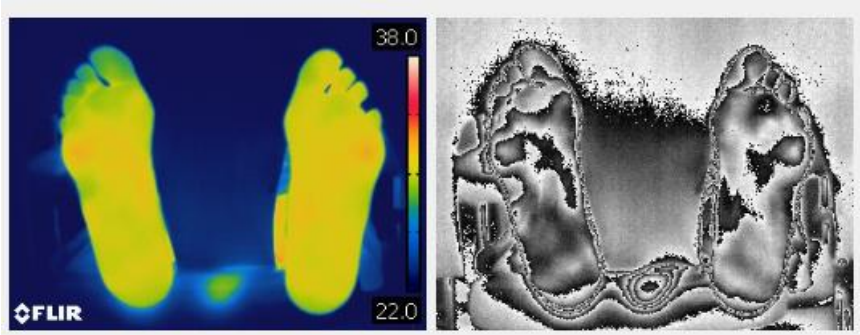

Figura 4.8 - Imagem térmica obtida.

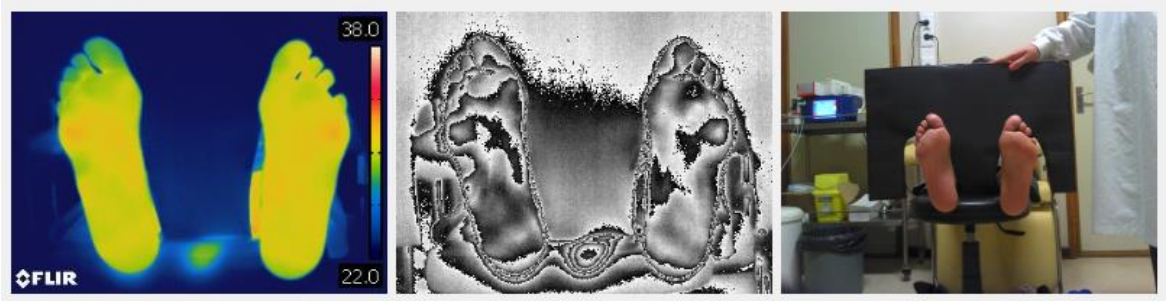

Figura 4.9 - Imagem visível obtida.

4.3 - Interface final com o utilizador

De seguida apresenta-se a interface final com o utilizador. Esta foi sofrendo alterações sucessivas, desde a versão inicial, de modo a tornar-se mais atrativa, intuitiva e simples para o utilizador.

Na figura 4.10 apresenta-se o logótipo da aplicação.

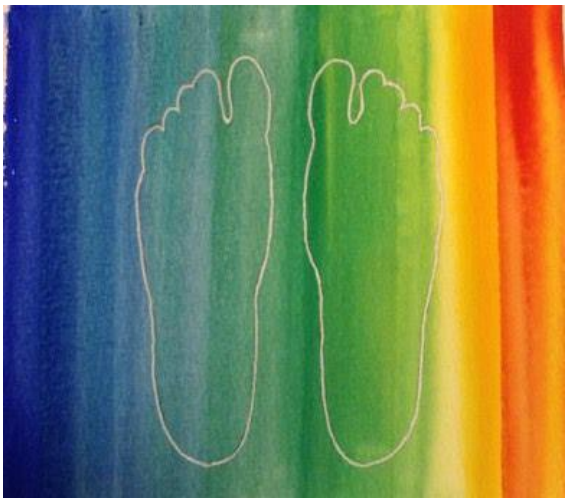

Figura 4.10 - Logótipo da aplicação Android.

Apresenta-se agora uma captura do ecrã do dispositivo móvel onde a aplicação final foi testada, estando esta em funcionamento. Nesta sequência pode haver repetição de imagens para mostrar melhor o modo como funciona. Em cada etapa, faz-se uma demonstração, através da colocação de um círculo vermelho, sobre o comando a ser utilizado.

Figura 4.11 - Página inicial.

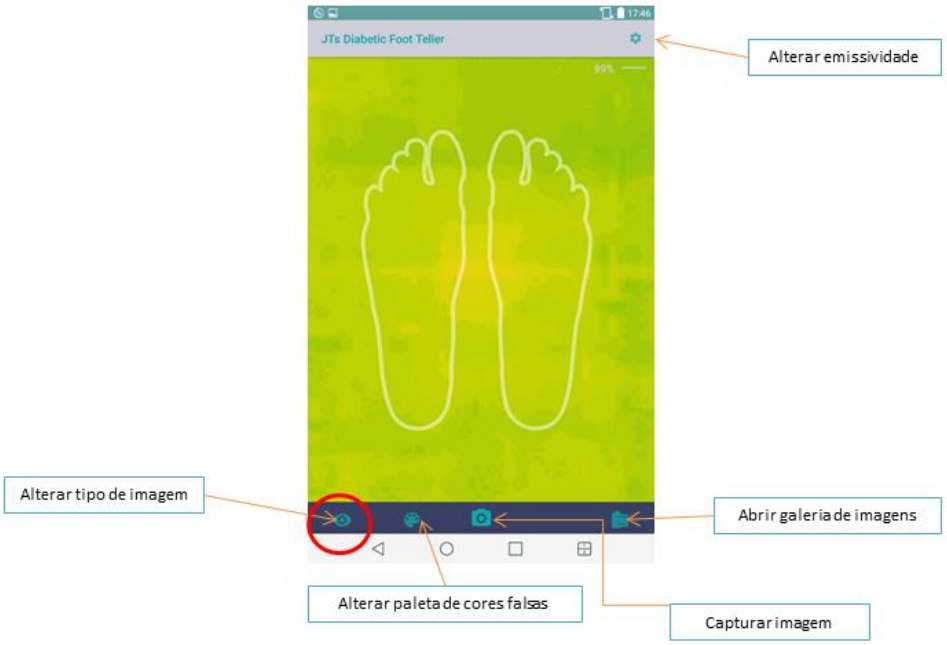

Figura 4.12- Página principal.

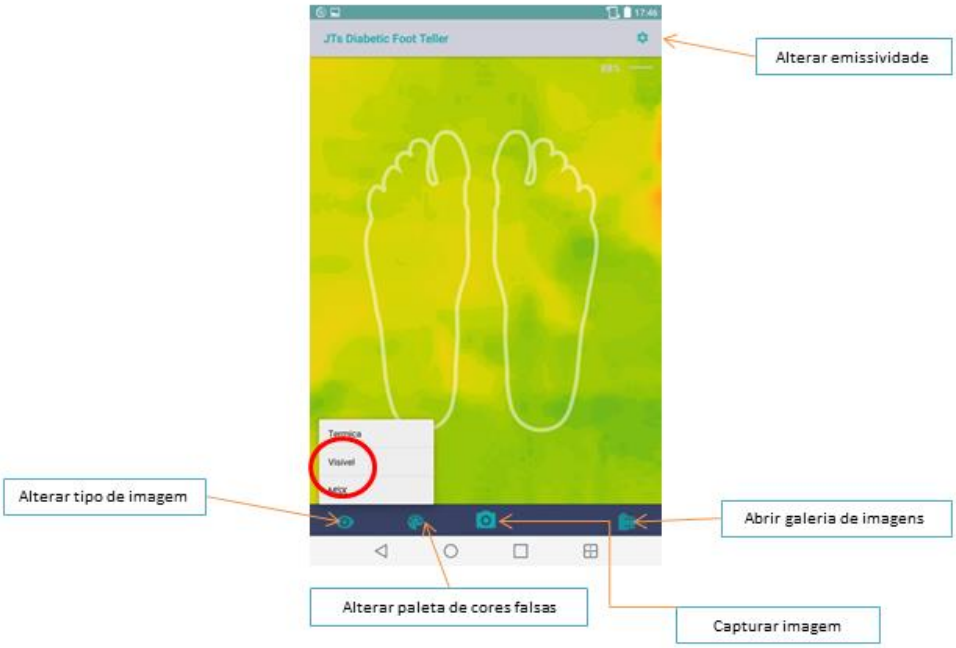

Figura 4.13 - Alterar tipo de imagem.

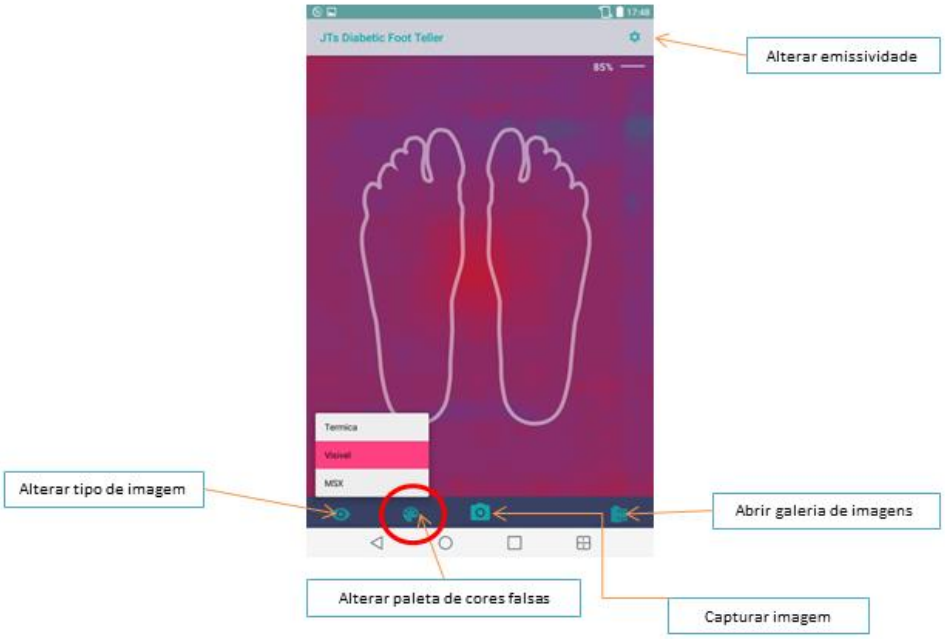

Figura 4.14 - Tipo de imagem selecionado.

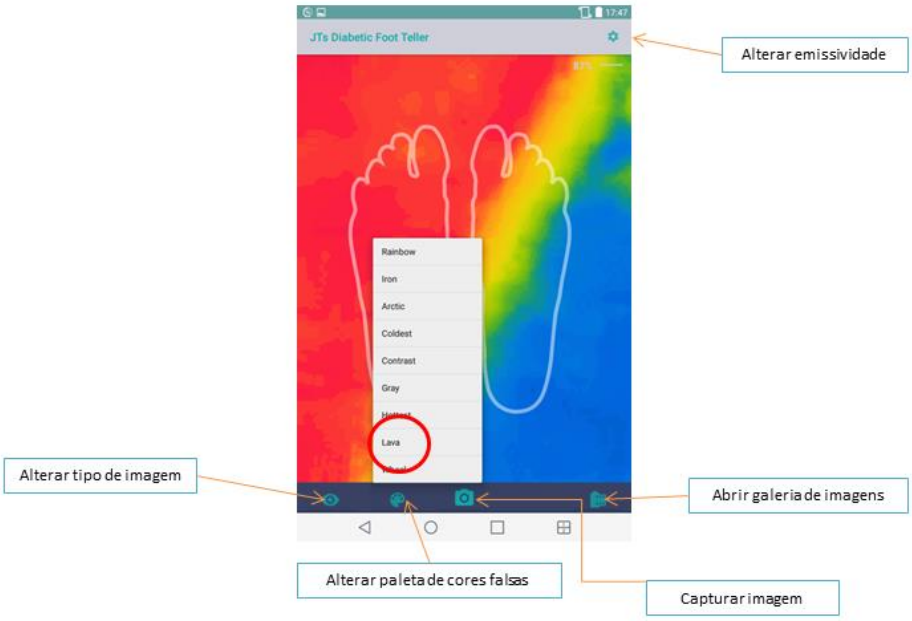

Figura 4.15 - Alterar paleta de cores falsas.

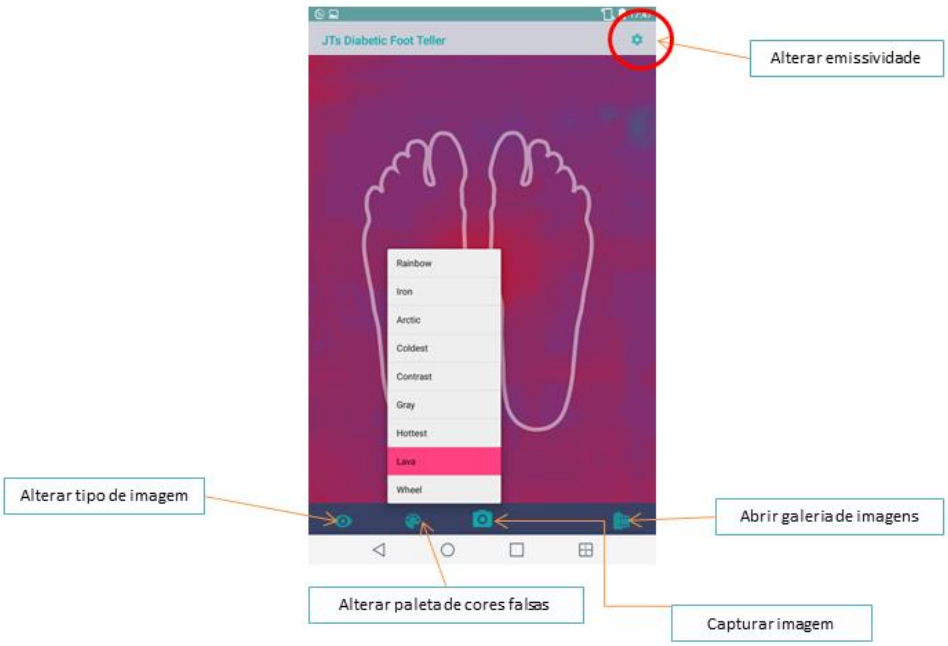

Figura 4.16 - Paleta de cores falsas selecionada.

Figura 4.17 - Alterar emissividade.

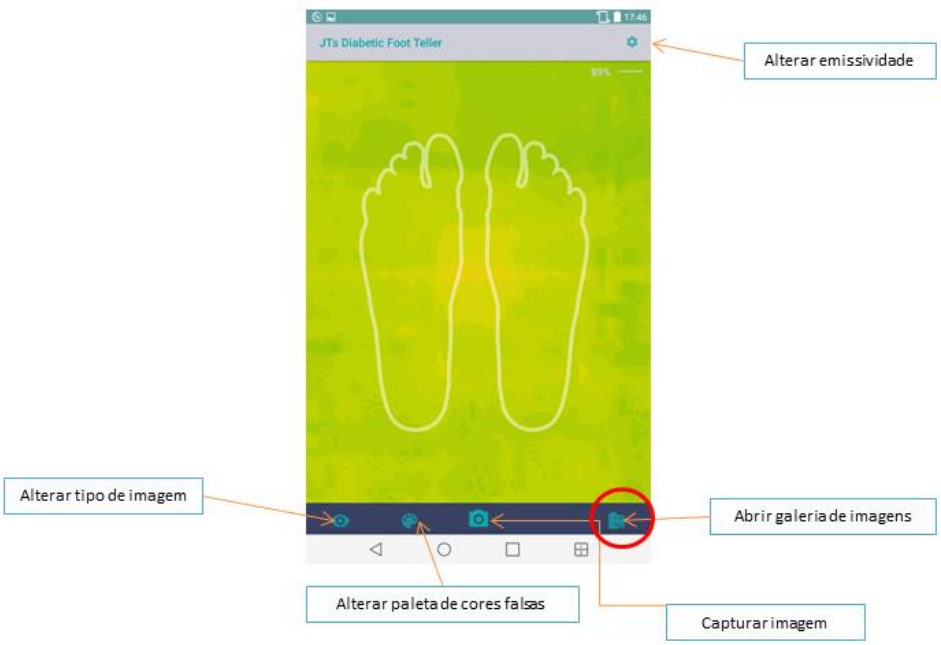

Figura 4.18 - Página principal.

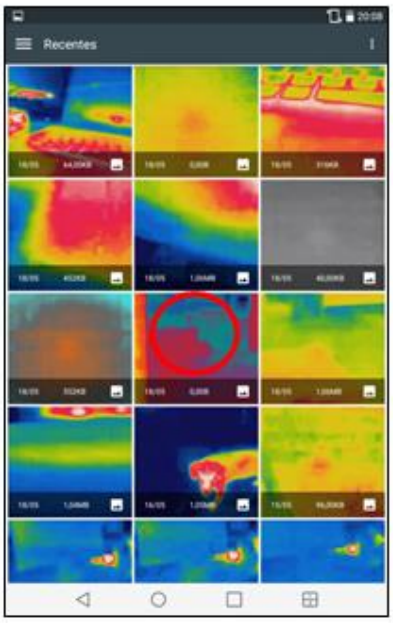

Figura 4.19 - Escolher imagem a abrir.

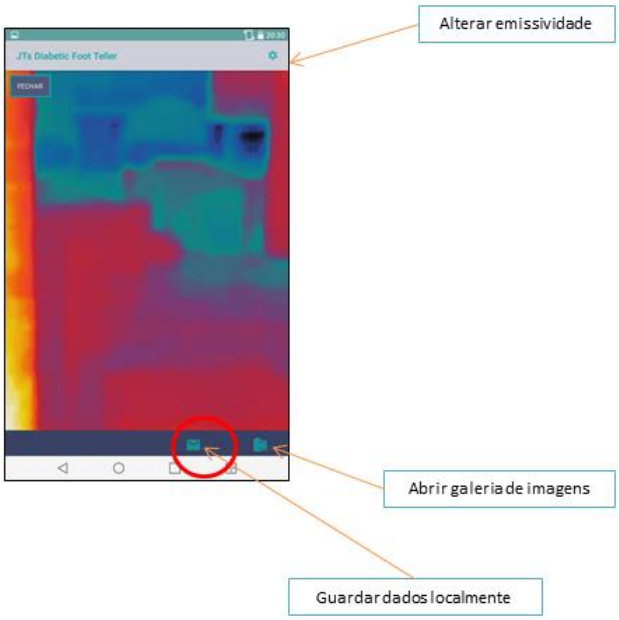

Figura 4.20 - Visualização da imagem.

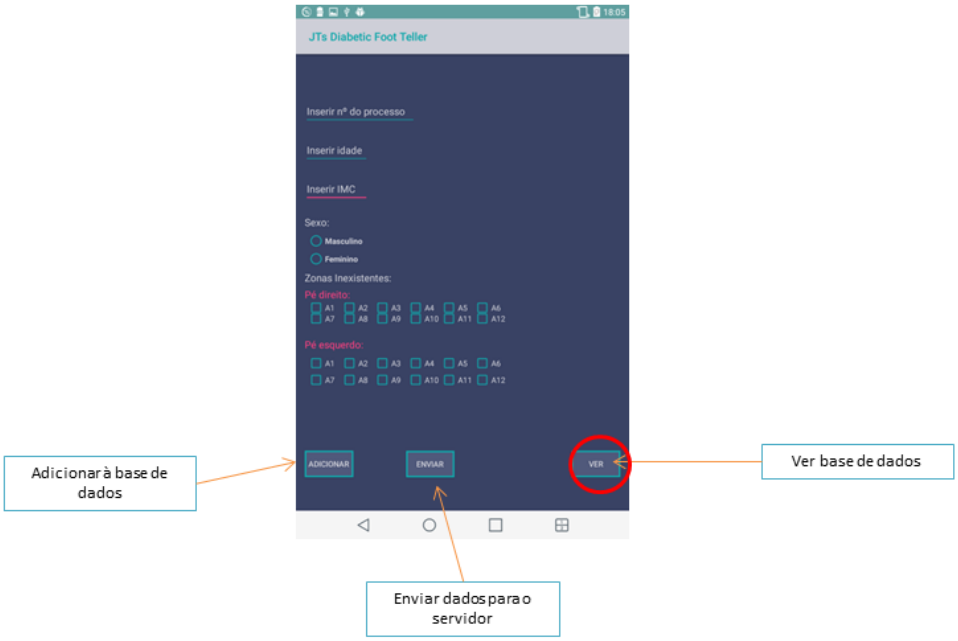

Figura 4.21 - Formulário dos dados.

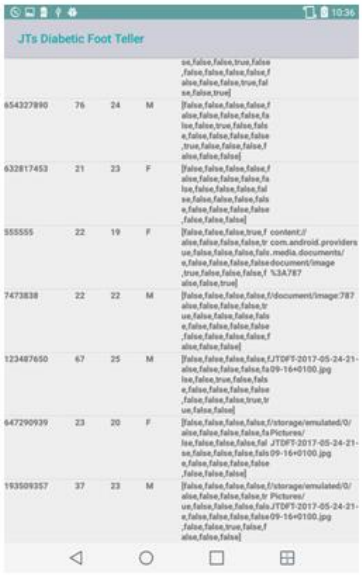

Figura 4.22 - Visualização da base de dados local.

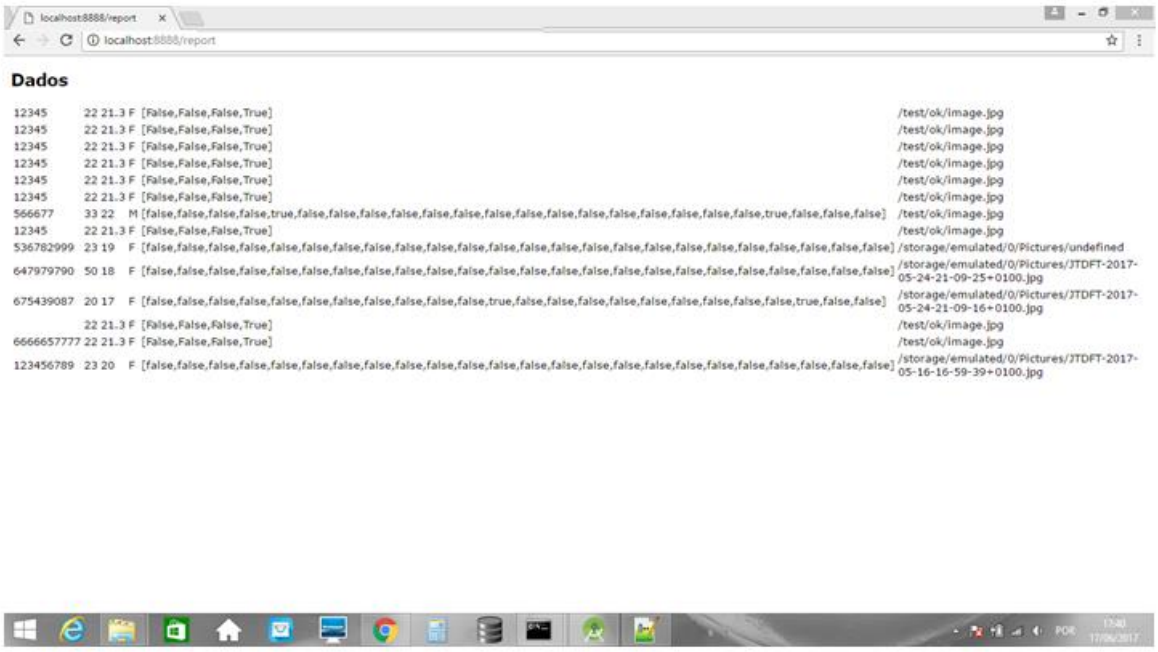

Figura 4.23 - Visualização da base de dados do *webservice*.

4.4 - Teste da aplicação *Android* desenvolvida

À medida que a aplicação foi sendo criada, cada tarefa foi submetida a testes sucessivos. Para isso, a aplicação foi descarregada e simulada num *tablet*, LG-V400 e versão do *Android* 5.0.2. No final, a aplicação foi testada no seu conjunto, já num ambiente parecido com a realidade para que foi criada.

Idealmente os testes deveriam ter sido realizados em ambiente hospitalar, de consulta externa de pé diabético, em doentes diabéticos, numa situação real. Isto não se concretizou, tendo sido os testes efetuados num grupo de 3 voluntários, aparentemente saudáveis, de acordo com o protocolo internacional de captura de imagens de Glamorgan [111].

Os voluntários foram colocados todos numa posição *standard*, sentados com as pernas estendidas e apoiadas num banco e os pés fletidos num ângulo de ≈90° com as pernas. Foi adicionada uma barreira retangular opaca, preta, com dois orifícios para a saída dos pés, junto dos tornozelos, para assegurar o anonimato e para servir de moldura para a imagem térmica, bloqueando as interferências das imagens circundantes. Depois de se descalçarem, os pés dos voluntários sem tocar no chão ficaram à temperatura ambiente durante cerca de 10 minutos, para estabilização da sua temperatura.

As fotografias representadas nas figuras 4.24, 4.25 e 4.26 foram captadas durante a realização de um desses testes, onde é possível ver a posição do voluntário, a barreira protetora e a aplicação em ação.

Figura 4.24 - Teste real 1.

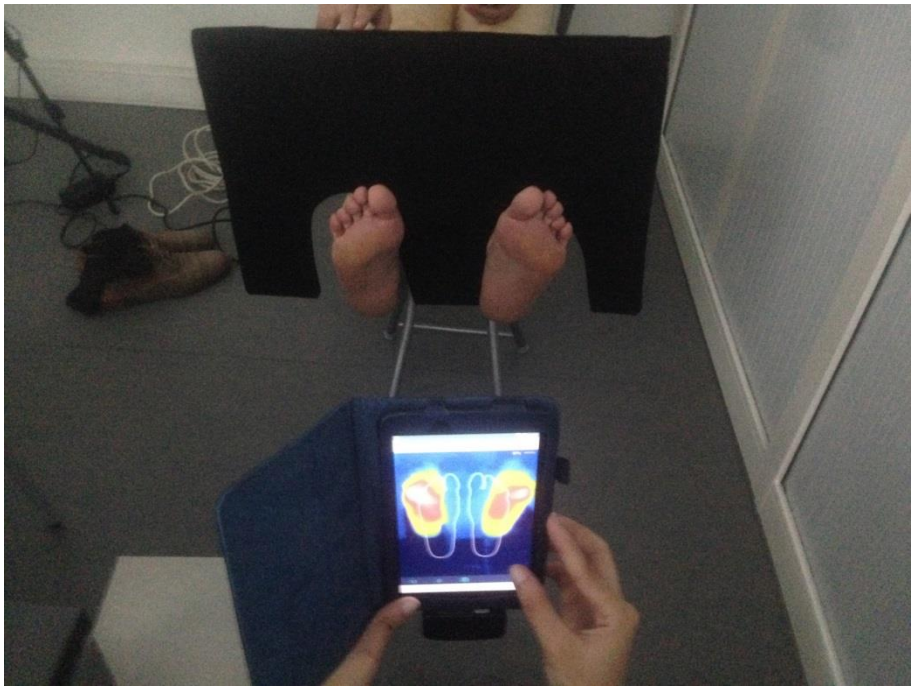

Figura 4.25 - Teste real 2.

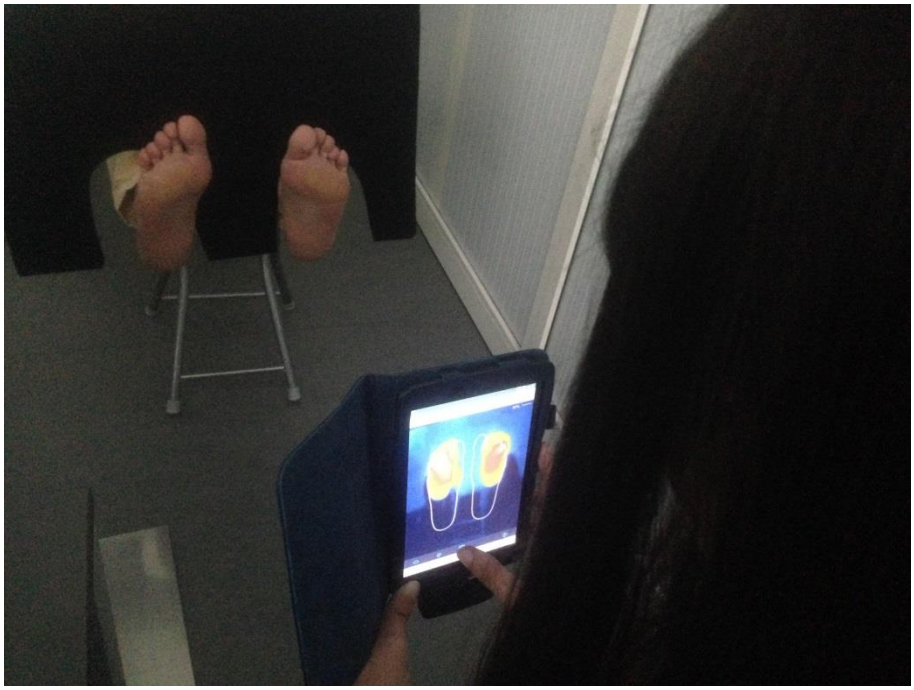

Figura 4.26 - Teste real 3.

As imagens obtidas pela aplicação, durante os testes, são as que se apresentam de seguida.

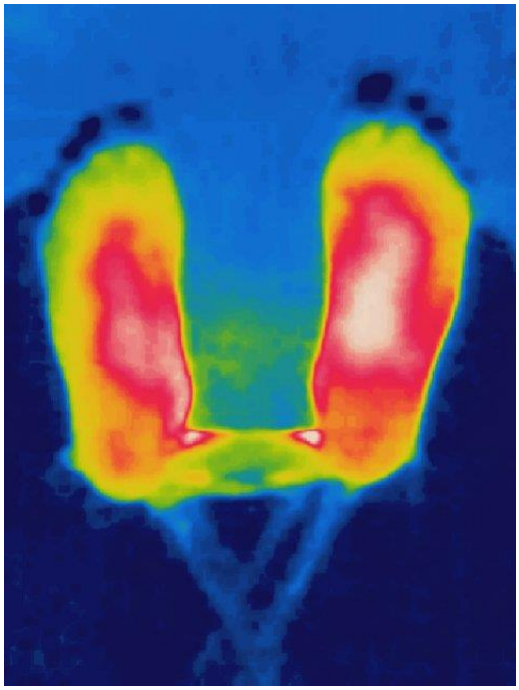

Figura 4.27 – Imagem obtida do voluntário número 1.

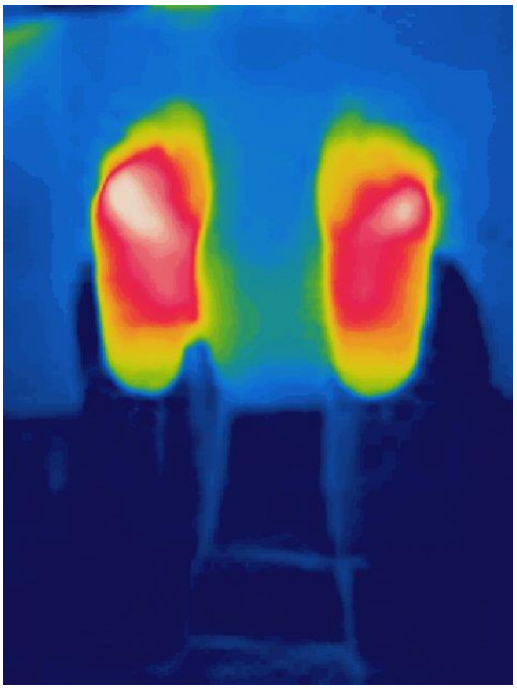

Figura 4.28 - Imagem obtida do voluntário número 2.

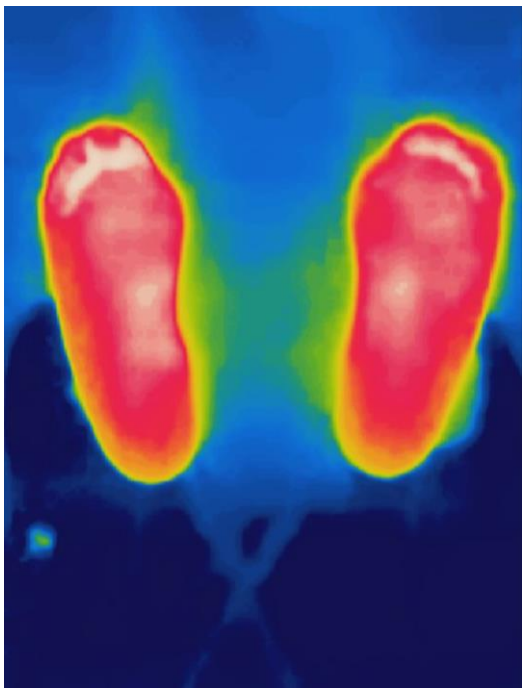

Figura 4.29 - Imagem obtida do voluntário número 3.

Também foi testado, numa outra fase, o armazenamento dos dados na base de dados local. As figuras seguintes mostram o formulário a ser preenchido e a visualização, no *tablet* e no computador, dos dados adicionados.

É de salientar que os valores que foram introduzidos são fictícios, criados só para os testes.

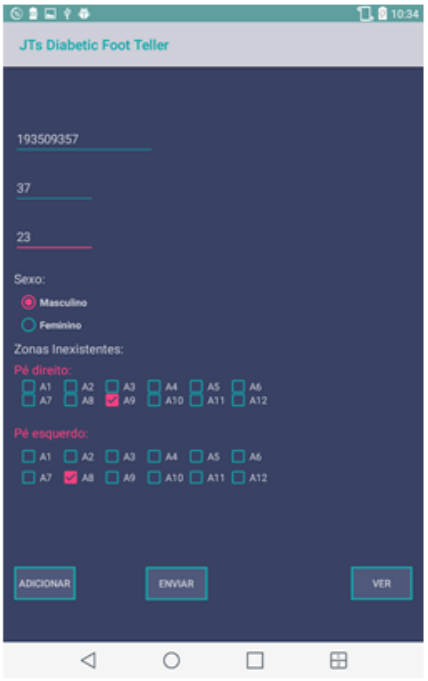

Figura 4.30 - Dados inseridos no formulário.

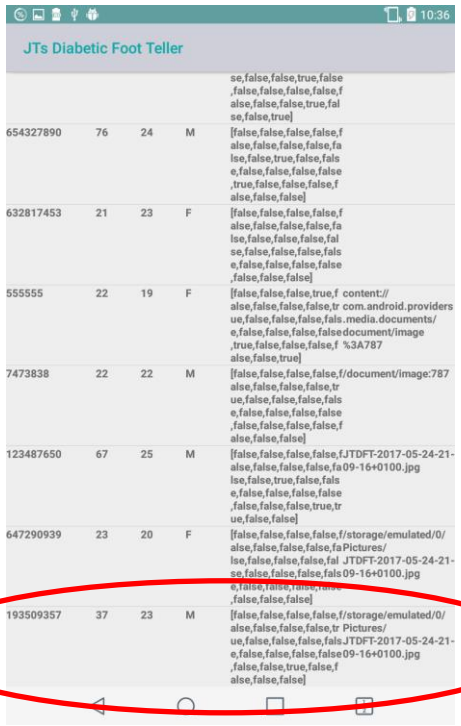

Figura 4.31 - Dados inseridos na base de dados local.

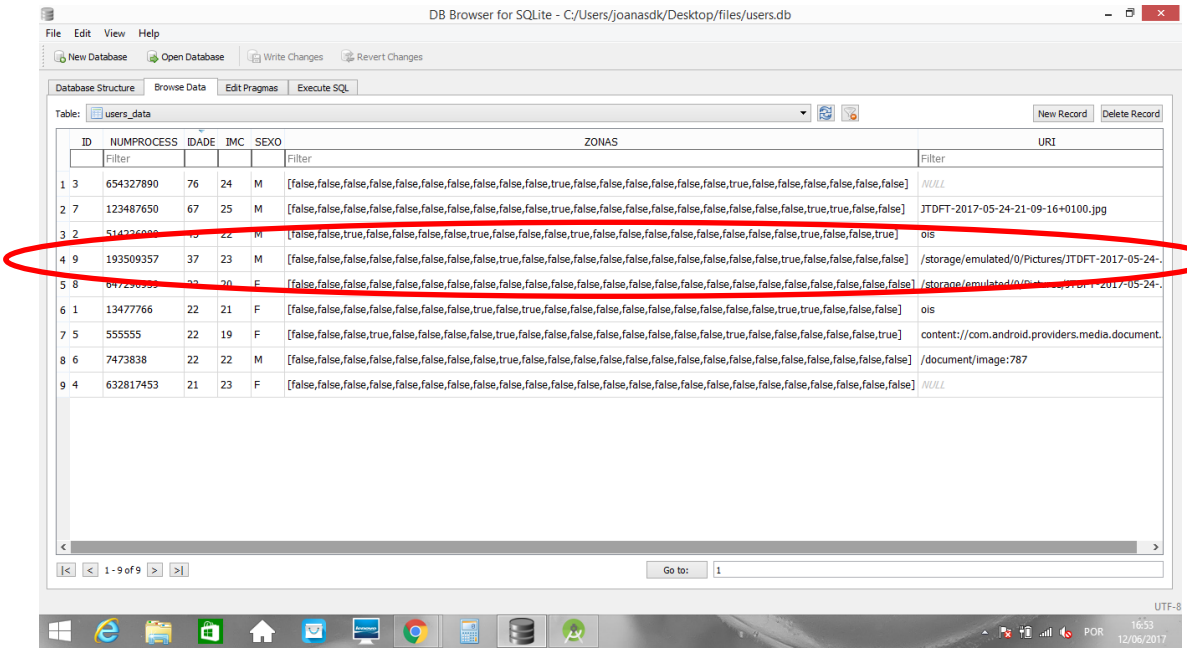

Figura 4.32 - Dados inseridos na base de dados local vistos com o *SQLiteManager*.

Por último, foi testado o envio dos dados para o servidor. As figuras seguintes mostram o formulário a ser preenchido e a visualização, no computador, dos dados enviados.

É, novamente, de salientar que os valores que foram introduzidos são fictícios, criados só para os testes.

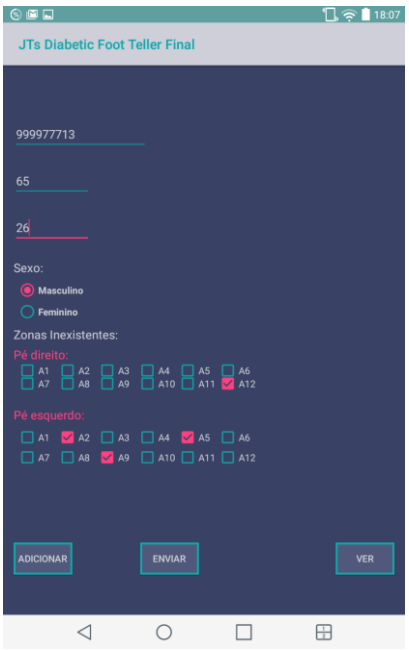

Figura 4.33 - Dados inseridos no formulário.

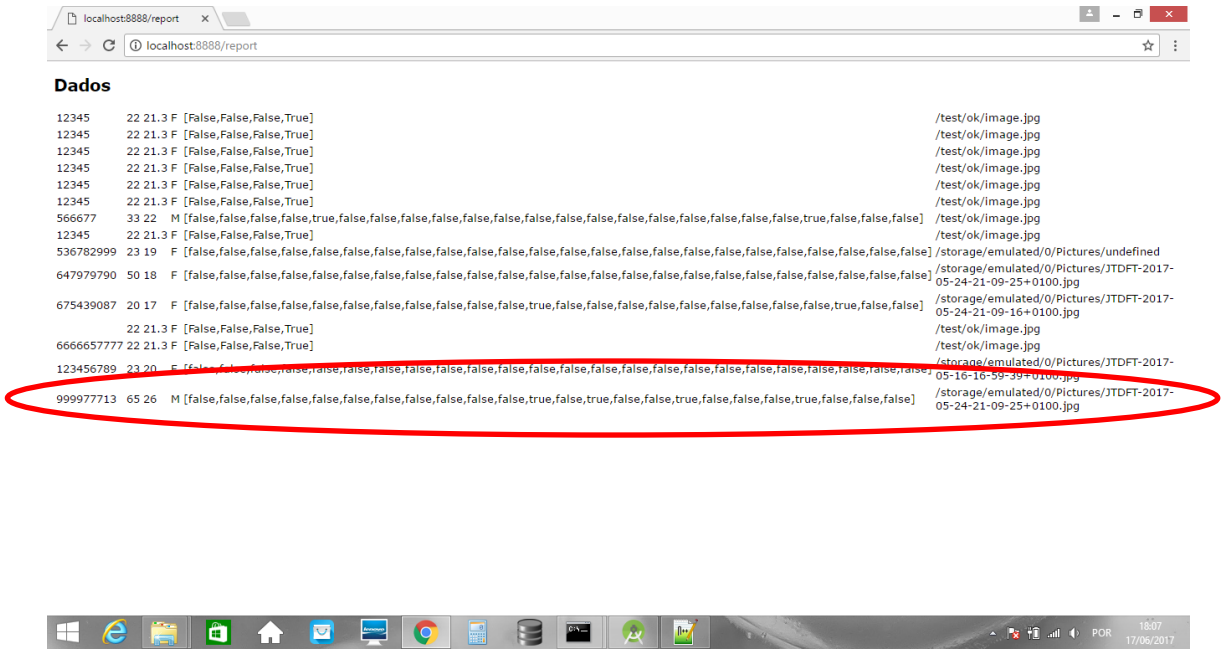

Figura 4.34 - Dados existentes no servidor.

4.5 - Aplicações finais

Deste trabalho resultaram 5 aplicações. Uma mais completa com o *overlay* dos pés e capaz de armazenar os dados do doente na base de dados local e de os enviar para um *webservice*. Outra, também com o *overlay*, mas só com a capacidade de guardar os dados do doente na base de dados local. Outra, ainda com *overlay*, mas só com a capacidade de enviar os dados para o *webservice*. Por último, uma aplicação que não tem a funcionalidade de arquivar os dados do doente na base de dados local nem de os enviar para o servidor, mas que apresenta o *overlay* dos pés e uma aplicação igual à anteriormente apresentada, mas sem o *overlay* dos pés.

Capítulo 5

Discussão

Neste capítulo apresenta-se a análise dos resultados obtidos e se estes estiveram de acordo com os objetivos desta tese.

Será feita referência a imagens e tabelas já expostas no capítulo anterior "Testes e Resultados".

Como descrito no capítulo anterior, as câmaras FLIR E60 e FLIR ONE de segunda geração para os sistemas operativos *Android* e *iOS* foram estudadas para aferir qual o seu erro de medição de temperaturas.

Obtiveram-se imagens térmicas do corpo negro a 30ºC, de 5 em 5 minutos, por cada uma das câmaras estudadas. Para determinar o erro de medição, posteriormente, calculou-se a diferença entre a temperatura do corpo negro e a temperatura obtida para cada câmara. A partir dos resultados obtidos, calculou-se qual a média (μ) e desvio padrão (σ) do erro obtido em cada câmara.

Da análise dos dados registados nas tabelas 4.1, 4.2 e 4.3, é possível inferir que as câmaras adaptáveis a dispositivos móveis, a FLIR ONE *iOS* e a FLIR ONE *Android*, são as que apresentam um maior erro médio na aferição da temperatura, sendo que a câmara FLIR ONE Android apresenta um erro médio de quase 2°C.

Por outro lado, em relação ao desvio padrão, os valores encontrados são mais próximos, mas, mesmo assim, as câmaras FLIR ONE têm um pior desempenho do que a FLIR E60, embora o seu valor não atinja os 0.5°C.

Também se verificou que as câmaras FLIR ONE só apresentaram estabilidade na medição 25 minutos após terem sido ligadas, o que limita o seu tempo de utilização (a bateria dura cerca de 1h). Mesmo assim, a versão *iOS* mostrou melhor desempenho do que a sua congénere *Android*.

Num estudo posterior, avaliou-se o comportamento das câmaras térmicas face a um aumento ou diminuição gradual de 1°C da temperatura do corpo negro, a partir de temperaturas basais de 20°C e 38°C, respetivamente. As temperaturas obtidas por cada câmara foram as registadas na tabela 4.4. De seguida, avaliou-se a diferença entre a temperatura do corpo negro e a obtida em cada câmara, para determinar, desta forma, o erro de medição (tabela 4.5 e figura 4.2). A partir destes resultados, determinou-se qual a

média (μ), desvio padrão (σ) e os valores máximo e mínimo do erro obtido para cada câmara (tabela 4.6).

A partir da tabela 4.6, é possível depreender que as câmaras FLIR ONE *iOS* e FLIR ONE *Android*, são as que apresentam um maior erro médio na aferição da temperatura, principalmente a câmara FLIR ONE *Android*.

Também para o desvio padrão, as câmaras FLIR ONE são as que apresentam os valores absolutos mais elevados.

Por outro lado, relativamente ao valor máximo do erro encontrado, a câmara FLIR ONE *Android* apresenta um valor elevado, de 2.9°C, logo seguida pela sua versão para *iOS* que ostenta um valor máximo de 2.2°C.

O valor mínimo do erro obtido tem uma distribuição mais uniforme pelas 3 câmaras, mas a FLIR ONE *Android* continua a apresentar um dos maiores valores, de 0.5°C.

Por último, foi calculado o coeficiente linear de *Pearson*, para determinar a intensidade da associação linear entre a temperatura medida por cada câmara térmica e a fonte de calibração (figuras 4.3, 4.4, 4.5).

Como se pode verificar, os valores do coeficiente de correlação linear de *Pearson*, para qualquer dos casos estudados, foi superior a 0.9 e quase igual a 1, o que significa que a correlação entre os valores de temperatura conseguidos pelas várias câmaras e os valores de temperatura da fonte de calibração foi muito forte, quase perfeita.

Calculou-se, também, o coeficiente de correlação de *Pearson* entre os valores de temperatura obtidos pelas câmaras FLIR ONE para *iOS* e *Android* (figuras 4.6).

Obteve-se um valor de coeficiente de correlação linear de *Pearson* superior a 0.9 e quase iguais a 1, o que significa que a correlação entre os valores de temperatura conseguidos pelas câmaras FLIR ONE, foi, também, muito forte, quase perfeita.

Nesta aplicação clínica em particular, são valorizadas diferenças de temperatura entre pés contralaterais muito pequenas (≈ 2°C), pelo que é fundamental uma avaliação da temperatura muito rigorosa. As câmaras FLIR ONE, para *Android* (sobretudo esta) e *iOS*, apresentam um erro muito elevado, por vezes superiores a 2°C, principalmente para temperaturas acima dos 30°C, que são as que mais se aproximam da temperatura superficial da pele. Assim, apesar da vantagem da sua maior portabilidade e menor peso, perdem em relação às câmaras mais comumente utilizadas, nomeadamente a FLIR E60, na precisão e rigor de avaliação de temperatura, podendo apresentar erros consideráveis e induzir a falhas na caracterização clínica do pé diabético. Apesar de tudo, a câmara FLIR ONE para *iOS*, de entre as câmaras adaptáveis a dispositivos móveis, é a que parece mais aconselhável para este tipo de utilização no rastreio do *pé diabético*.

Para apoiar estes dados, seria interessante comprovar através de mais testes semelhantes com um número apreciável de câmaras do mesmo tipo, focando principalmente nas câmaras FLIR ONE.

No capítulo "Testes e Resultados" descrevem-se os testes e respetivos resultados referentes ao estudo do formato JPG radiométrico.

Relativamente à etapa de localizar os vários parâmeros no cabeçalho da imagem, como é possível verificar na tabela 4.9, a maior parte dos endereços não se encontram todos na mesma posição, o que não era esperado. É de salientar que existem dois grupos distintos de imagens a serem analisados, que facilmente se distinguem distinguidos pelo nome da imagem: os que começam por FLIR® pertencem a um grupo e os que se iniciam por

flir20120201T19 pertencem ao outro grupo. Estas últimas são imagens da câmara FLIR ONE, enquanto as outras foram captadas pela câmara FLIR E60.

Assim, dado que se verificou que os endereços não se encontravam todos na mesma posição, decidiu-se fazer um outro teste, como já foi referido anteriormente. Como é possível observar na tabela 4.10, tirando a diferença de parâmetros presentes na célula que se encontra a vermelho, os parâmetros, apesar de não estarem todos no mesmo endereço, encontram-se todos à mesma distância.

Com estes testes conseguiu-se descobrir como chegar aos parâmetros da imagem que se pretendem obter, para todas as imagens captadas pela câmara FLIR E60.

Por outro lado, nos últimos testes realizados (figuras 4.8 e 4.9), comprovou-se ser possível separar a imagem térmica da visível.

No entanto, verificou-se que a imagem térmica se encontra comprimida, tendo os dados radiométricos um tamanho de sensivelmente 93kB, quando seria expectável um tamanho de acordo com o tamanho da imagem de 152 kB. Futuramente, deve ser estudado o algoritmo de compressão usado para se poderem descomprimir estas imagens e se realizarem aplicações de análise específicas para uso clínico.

Ao longo do desenvolvimento da aplicação, e após os testes de validação realizados já referidos, verificou-se que os requisitos foram sendo gradualmente cumpridos. Assim, os testes demonstraram que a aplicação é funcional, desde a transmissão, em tempo real, de imagens térmicas para o ecrã, à capacidade de utilização das funções referentes à alteração dos parâmetros, tais como a emissividade, a paleta de cores falsas e o tipo de imagem. Através dos testes, verificou-se que esta possibilita, ainda, a captura e armazenamento de uma imagem, assim como permite a sua visualização diretamente da galeria de fotografias. Foi, ainda, testada com sucesso a capacidade de guardar os dados clínicos e pessoais do doente na base de dados local e do envio desses mesmos dados para o servidor, também com êxito.

Nas figuras 4.24, 4.25 e 4.26 demonstra-se que a posição do doente escolhida é favorável à obtenção das imagens e simples de conseguir. Por outro lado, o *overlay* criado na aplicação revelou ser muito útil no posicionamento e fácil de ajustar aos pés a serem estudados. Estas imagens permitem, também, comprovar que a aplicação desenvolvida funciona na prática.

As imagens térmicas captadas aos voluntários, através da aplicação criada, são claras e nítidas, podendo ser avaliadas e discutidas em termos de temperatura da pele dos pés e, portanto, podem ter aplicação clínica no futuro. Para isso foi muito vantajoso o uso da barreira que serviu de moldura de destaque para os pés, atenuando potenciais interferências de diferentes temperaturas de outras zonas da imagem e o uso do *overlay*, que permitiu que os pés ficassem bem centrados e posicionados.

As figuras 4.30, 4.31 e 4.32 demonstram que a aplicação funcionou no que diz respeito ao armazenamento na base de dados local, dos dados fictícios referentes aos voluntários e ao nome e caminho da imagem já capturada escolhida aleatoriamente.

Na figura 4.30 é possível observar um utilizador a inserir os dados no formulário. Já na figura 4.31 pode ver-se que os dados foram inseridos com sucesso na base de dados, uma vez que os dados que foram inseridos são aqueles que aparecem na linha circundada a vermelho. O texto da última coluna, que corresponde ao nome e caminho para aceder à imagem, é comprovadamente correto.

O ficheiro criado no dispositivo *Android*, que contém a base de dados, foi transferido, por cabo USB, para um computador. Com o apoio do *SQLliteManager* foi possível visualizar a mesma base de dados, num ecrã maior e, por isso, de mais fácil visualização (figura 4.32). A linha oval vermelha, mais uma vez, demonstra essa passagem de dados com sucesso.

As figuras 4.33 e 4.34 mostram que a aplicação funcionou no que diz respeito ao envio dos dados para o servidor. Os dados eram, mais uma vez, fictícios e a imagem já capturada foi escolhida aleatoriamente.

Na figura 4.33 é possível ver um utilizador a inserir os dados no formulário. Já na figura 4.34 pode ver-se que os dados foram enviados com sucesso para o servidor, através da interface criada, uma vez que os dados que foram inseridos são aqueles que aparecem na linha circundada a vermelho. O texto da última coluna, que corresponde ao nome e caminho para aceder à imagem é comprovadamente correto.

É de salientar que a imagem não é enviada para o servidor nem é armazenada. Em vez disso, envia-se o nome e o caminho da imagem que corresponde aos dados inseridos. No futuro, isto pode ser realizado com facilidade. Para isso, bastará alterar a base de dados para uma que suporte campos binários como são exemplo a *MySQL* e a *PostgreSQL*.

No desenvolvimento da aplicação houve um parâmetro que não foi trabalhado, mas que merece ser mencionado pela sua importância. Na verdade, a possibilidade de alterar a escala da temperatura, permitindo que o utilizador introduza valores, de temperaturas máximas e mínimas, fixando-os, pode ser muito valiosa. Como se sabe, a análise subjetiva de pequenas diferenças de temperatura é crucial na avaliação dos pés destes doentes e, assim, poderiam ser mais objetivas quer a análise quer a comparação das imagens térmicas. Esta tarefa não foi levada a cabo porque a documentação do SDK da FLIR ONE não é muito abrangente, não se conseguindo retirar grande informação sobre este parâmetro. Numa primeira análise não existe uma função ou um método óbvios que permitam a alteração da temperatura de forma imediata, como existe, por exemplo, para modificar a emissividade (*setEmissivity(float)*).

Por outro lado, a partir de determinado momento do desenrolar do projeto, verificou-se que, no tempo disponível, não era exequível uma avaliação automática das imagens térmicas, através da conjugação com o trabalho desenvolvido noutra dissertação de mestrado sob a mesma orientação. Como essa tarefa não chegou a ser executada, não se tornou possível a receção de um relatório da análise acerca do risco de desenvolver o *pé diabético*. No entanto, futuramente, isso pode ser conseguido utilizando bibliotecas que permitam lidar com ficheiros XML e PDF.

Para desenvolver uma aplicação com os mesmos objetivos e requisitos em *iOS* era necessário obter uma licença de desenvolvedor por parte da empresa *Apple*®. Esse é um processo dispendioso e com um prazo de aceitação muito moroso, pelo que já não foi possível obtê-la no tempo de execução deste projeto de mestrado.

Capítulo 6

Conclusão

Em suma, o objetivo principal de criar uma aplicação capaz de capturar imagens térmicas e de enviar dados para um *webservice* foi atingido, sendo que dos 23 requisitos (funcionais e não funcionais) definidos, apenas 3 não foram atingidos: não é possível especificar a gama de temperaturas a utilizar, nem quais as unidades desta (*Celsius* ou *Fahrenheit*), dado que não existe uma função específica do SDK da FLIR ONE para este efeito, e não se recebe um relatório com o resultado da análise das imagens do pé, uma vez que esta análise não é efetuada.

Mesmo assim, pode afirmar-se que a grande maioria dos objetivos foram atingidos com êxito.

A execução deste projeto permitiu a aquisição e aprofundamento de conhecimentos na área da informática, nomeadamente no que diz respeito à programação em *Java* e no ambiente do *Android Studio*, na utilização de bases de dados *SQLite*, bem como no desenvolvimento de *webservices* e na linguagem *python*. Permitiu, ainda, documentar e conhecer o FLIR ONE SDK para criar aplicações que usem este tipo de câmaras térmicas acopladas a dispositivos móveis. No futuro, isto poderá ser muito útil para a conceção de outras aplicações destas imagens noutras áreas.

Este projeto foi um primeiro passo no desenvolvimento de um método objetivo, rápido e indolor a ser utilizado na prática clínica, nomeadamento no rastreio do *pé diabético*.

6.1 - Trabalho futuro

No futuro, seria interessante replicar a aplicação *Android* desenvolvida para o sistema operativo *iOS*.

Por outro lado, sugere-se a junção desta dissertação de mestrado com outra que analisa e classifica os dados enviados, sob a mesma orientação, por forma a que a aplicação permita a receção automática do relatório que contém a análise desses dados. Nesse relatório, deveria ser permitida a inserção de comentários por parte do utilizador.

Como já se referiu, a temperatura é o parâmetro mais importante nesta avaliação. Deste modo, é crucial a implementação de uma função que permita ao utilizador definir e fixar

qual a gama de temperaturas que pretende utilizar. A escala de temperaturas deveria aparecer no ecrã no momento de captura da imagem.

Para se ter um histórico dos dados enviados e resultados recebidos de cada doente, devia ser implementada uma autenticação na aplicação, ou seja, devia ser obrigatório fazer o registo e *login*. Assim era possível ver a evolução de cada doente.

Seria, também, muito importante testar a aplicação em ambiente real, na prática clínica, seja em consultas de *pé diabético*, seja em contexto de cuidados de saúde primários, nos ACES, ou mesmo em ambulatório, no domicílio dos doentes.

É, também, muito importante efetuar um novo estudo de câmaras térmicas da FLIR para comprovar os resultados aqui obtidos, podendo ser reavaliadas e analisadas outras opções no futuro.

Por último, para o funcionamento da aplicação não depender da existência de internet, poderia fazer-se uma rede *ad hoc* entre um computador (que tenha o servidor) e o dispositivo móvel.

Referências

[1] Powers, Alvin, "Diabetes Mellitus," in *Harrison's Principles of Internal Medicine*, 19th ed., Mc Graw Hill, 2015, pp. 2399–2407.

[2] Longmore M, Wilkinson I B, Baldwin A, Wallin E, "Diabetes Mellitus," in *Oxford Handbook of Clinical Medicine*, 9th Edition., Oxford University Press, 2014, pp. 198–205.

[3] "IDF Diabetes Atlas," 2015. [Online]. Available: http://www.diabetesatlas.org/resources/2015-atlas.html. [Accessed: 07-May-2017].

[4] K. Roback, "An overview of temperature monitoring devices for early detection of diabetic foot disorders," *Expert Rev. Med. Devices*, vol. 7, no. 5, pp. 711–718, Sep. 2010.

[5] P. Sousa, V. Felizardo, D. Oliveira, R. Couto, and N. M. Garcia, "A review of thermal methods and technologies for diabetic foot assessment," *Expert Rev. Med. Devices*, vol. 12, no. 4, pp. 439–448, Jul. 2015.

[6] Sociedade Portuguesa de Diabetologia, "Diabetes - Factos e Números - O Ano de 2015," Dezembro 2016.

[7] Powers, Alvin, "Diabetes Mellitus: Complications," in *Harrison's Principles of Internal Medicine*, 19th ed., Mc Graw Hill, 2015, pp. 2422–2430.

[8] "FEUP - Proposta de Dissertação / Projecto." [Online]. Available: https://sigarra.up.pt/feup/pt/estagios_empresas.ver_dados_proposta?p_id=206712&p_processo=21673&p_inst_codigo=&pv_perfil=ALU&p_aluno_id=117561. [Accessed: 28-Nov-2016].

[9] Vinik A, Maser R, Mitchell B, Freeman R, "Diabetic autonomic neuropathy," *Diabetes Care 265*, pp. 1553–1579, May 2003.

[10] A. J. Boulton, "Lowering the risk of neuropathy, foot ulcers and amputations," *Diabet. Med. J. Br. Diabet. Assoc.*, vol. 15 Suppl 4, pp. S57-59, 1998.

[11] J. Apelqvist, D. Bergqvist, M. Eneroth, and J. Larsson, "[The diabetic foot. Optimal prevention and treatment can halve the risk of amputation]," *Lakartidningen*, vol. 96, no. 1–2, pp. 37–41, Jan. 1999.

[12] Meola C, Carlomagno G, "Recent advances in the use of infrared thermography," *Meas. Sci. Technol. 159*, p. R27, 2004.

[13] Tan J H, et al, "Infrared thermography on ocular surface temperature: a review," *Infrared Phys. Technol. 524*, pp. 97–108, 2009.

[14] Nagase T, Sanada H, Takehara K, Oe M, Iisaka S, Ohashi Y, Oba M, Kadowaki T, Nakagami G, "Variations of plantar thermographic patterns in normal controls and non-ulcer diabetic patients: novel classification using angiosome concept," *J. Plast. Reconstr. Aesthetic Surg. 647*, pp. 860–866, 2011.

[15] "Home - PubMed - NCBI." [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/. [Accessed: 28-Nov-2016].

[16] "Scopus - Welcome to Scopus." [Online]. Available: https://www.scopus.com/. [Accessed: 28-Nov-2016].

[17] "Web of Science [v.5.23.1] - Principal Coleção do Web of SciencePágina inicial." [Online]. Available: http://apps.webofknowledge.com/WOS_GeneralSearch_input.do?product=WOS&search_

mode=GeneralSearch&SID=W2OKnfCUEW7784ke3Ec&preferencesSaved=. [Accessed: 28-Nov-2016].
[18] "Temperature – The Physics Hypertextbook." [Online]. Available: http://physics.info/temperature/. [Accessed: 29-Nov-2016].
[19] R. Vardasca and R. Simoes, "Current Issues in Medical Thermography," in *Topics in Medical Image Processing and Computational Vision*, J. M. R. S. Tavares and R. M. N. Jorge, Eds. Springer Netherlands, 2013, pp. 223–237.
[20] Tipler, Paul and Mosca, Gene, *Física para Cientistas e Engenheiros*, 6ª., vol. 1. gen/LTC, 2012.
[21] "Infrared Waves: Definition, Uses & Examples - Video & Lesson Transcript | Study.com." [Online]. Available: http://study.com/academy/lesson/infrared-waves-definition-uses-examples.html. [Accessed: 28-Nov-2016].
[22] Thomas R, *The Thermography Monitoring Handbook*. Coxmoor Publishing Company, 1999.
[23] "fig-1-cmi.gif (550×289)." [Online]. Available: http://www.edmundoptics.eu/contentassets/639fec6d719643a8a3400e1205e6fa84/fig-1-cmi.gif. [Accessed: 28-Nov-2016].
[24] "Infrared Waves." [Online]. Available: http://science.hq.nasa.gov/kids/imagers/ems/infrared.html. [Accessed: 28-Nov-2016].
[25] "What Is Infrared?" [Online]. Available: http://www.livescience.com/50260-infrared-radiation.html. [Accessed: 28-Nov-2016].
[26] Jones, B, "A reappraisal of the use of infrared thermal image analysis in medicine," *Med. Imaging IEEE Trans. 176*, pp. 1019–1027, 1998.
[27] Brioschi M, "Anatomia e Fisiologia Termográfica.," in *Manual de Termografia Médica*, 2012.
[28] Jones, B.F. and Plassmann, P, "Digital IR Thermal Imaging of Human Skin," *IEEE Eng. Med. Biol.*, pp. 41–48, Dec. 2002.
[29] Incropera F, *Fundamentals of heat and mass transfer*. John Wiley & Sons, 2011.
[30] Love T, "Thermography as an indicator of blood perfusion," *Ann. N. Y. Acad. Sci.*, no. 335.1, pp. 429–437, 1980.
[31] E. F. J. Ring and K. Ammer, "Infrared thermal imaging in medicine," *Physiol. Meas.*, vol. 33, no. 3, pp. R33-46, Mar. 2012.
[32] R. Vardasca, P. Plassmann, J. Gabriel, and E. F. J. Ring, "Towards a Medical Imaging Standard Capture and Analysis Software," in *ResearchGate*, 2014.
[33] Ring E, "Foot Technology, Part 1 of 2: Thermal Imaging Today and Its Relevance to Diabetes," *J. Diabetes Sci. Technol.*, no. 4.4, p. 857, 2010.
[34] Vardasca, Ricardo and Gabriel, J, "A proposal of a standard rainbow false color scale for TMI."
[35] American Diabetes Association (ADA), "Diagnosis and Classification of Diabetes Mellitus," *Diabetes Care*, no. 35(1), p. S11, 2012.
[36] A. J. M. Boulton, L. Vileikyte, G. Ragnarson-Tennvall, and J. Apelqvist, "The global burden of diabetic foot disease," *Lancet Lond. Engl.*, vol. 366, no. 9498, pp. 1719–1724, Nov. 2005.
[37] C. Liu, J. J. van Netten, J. G. van Baal, S. A. Bus, and F. van der Heijden, "Automatic detection of diabetic foot complications with infrared thermography by asymmetric analysis," *J. Biomed. Opt.*, vol. 20, no. 2, p. 26003, Feb. 2015.
[38] D. Hernandez-Contreras, H. Peregrina-Barreto, J. Rangel-Magdaleno, and J. Gonzalez-Bernal, "Narrative review: Diabetic foot and infrared thermography," *Infrared Phys. Technol.*, vol. 78, pp. 105–117, Sep. 2016.
[39] J. J. van Netten, J. G. van Baal, C. Liu, F. van der Heijden, and S. A. Bus, "Infrared thermal imaging for automated detection of diabetic foot complications," *J. Diabetes Sci. Technol.*, vol. 7, no. 5, pp. 1122–1129, Sep. 2013.
[40] M. Bharara, J. E. Cobb, and D. J. Claremont, "Thermography and thermometry in the assessment of diabetic neuropathic foot: a case for furthering the role of thermal techniques," *Int. J. Low. Extrem. Wounds*, vol. 5, no. 4, pp. 250–260, Dec. 2006.
[41] Sanches J, *Pé Diabético: fisiopatologia, manifestações e principais formas de diagnóstico e rastreio*. 2008.
[42] Serra L, Serra M B, Carvalho R, Martins J, "Diagrama etiológico no pé neuropático," in *Pé Diabético - Manual para a prevenção da catástrofe*, 2º Edição., 2008, p. 33.

[43] Gomez D A M, Garriga G, Mompo I B, Sanchez F L, Barberan J, Rodriguez J A G, Gobernado M, Mensa J, "Documento de consenso sobre el tratamiento antimicrobiano de las infecciones en el pie del diabe tico," *Rev. Esp. Quimioter.*, vol. 20, no. 1, pp. 77–92.
[44] Serra L, Serra M B, Carvalho R, Martins J, *Pé Diabético - Manual para a Prevenção da Catástrofe*, 2º Edição. LIDEL, 2008.
[45] P. C. Leung, "Diabetic foot ulcers--a comprehensive review," *Surg. J. R. Coll. Surg. Edinb. Irel.*, vol. 5, no. 4, pp. 219–231, Aug. 2007.
[46] L. F. Balbinot, C. C. Robinson, M. Achaval, M. A. Zaro, and M. L. Brioschi, "Repeatability of infrared plantar thermography in diabetes patients: a pilot study," *J. Diabetes Sci. Technol.*, vol. 7, no. 5, pp. 1130–1137, Sep. 2013.
[47] D. G. Armstrong, K. Holtz-Neiderer, C. Wendel, M. J. Mohler, H. R. Kimbriel, and L. A. Lavery, "Skin temperature monitoring reduces the risk for diabetic foot ulceration in high-risk patients," *Am. J. Med.*, vol. 120, no. 12, pp. 1042–1046, Dec. 2007.
[48] D. R. Whiting, L. Guariguata, C. Weil, and J. Shaw, "IDF diabetes atlas: global estimates of the prevalence of diabetes for 2011 and 2030," *Diabetes Res. Clin. Pract.*, vol. 94, no. 3, pp. 311–321, Dec. 2011.
[49] "Monofilamento de Semmes-Weinstein." [Online]. Available: http://semiologiamedica.blogspot.pt/2009/10/teste-do-monofilamento-de-simmes.html. [Accessed: 05-Mar-2017].
[50] R. M. Stess *et al.*, "Use of liquid crystal thermography in the evaluation of the diabetic foot," *Diabetes Care*, vol. 9, no. 3, pp. 267–272, Jun. 1986.
[51] E. J. Boyko, J. H. Ahroni, V. Stensel, R. C. Forsberg, D. R. Davignon, and D. G. Smith, "A prospective study of risk factors for diabetic foot ulcer. The Seattle Diabetic Foot Study," *Diabetes Care*, vol. 22, no. 7, pp. 1036–1042, Jul. 1999.
[52] P.-C. Sun, H.-D. Lin, S.-H. E. Jao, Y.-C. Ku, R.-C. Chan, and C.-K. Cheng, "Relationship of skin temperature to sympathetic dysfunction in diabetic at-risk feet," *Diabetes Res. Clin. Pract.*, vol. 73, no. 1, pp. 41–46, Jul. 2006.
[53] Vardasca R, Marques A, Carvalho R, Gabriel J, "Thermal imaging of the foot in different forms of diabetic disease," in *Infrared Imaging A casebook in clinical medicine*, IOP Publishing, 2015, pp. 27-1-3.
[54] Donnelly R, et al, "ABC of arterial and venous disease: vascular complications of diabetes," *BMJ*, no. 320.7241, p. 1062, 2000.
[55] T. Kanazawa *et al.*, "Use of smartphone attached mobile thermography assessing subclinical inflammation: a pilot study," *J. Wound Care*, vol. 25, no. 4, pp. 177–180, 182, Apr. 2016.
[56] F.-F. Lee, F. Chen, and J. Liu, "Infrared thermal imaging system on a mobile phone," *Sensors*, vol. 15, no. 5, pp. 10166–10179, Apr. 2015.
[57] L. A. Lavery *et al.*, "Home monitoring of foot skin temperatures to prevent ulceration," *Diabetes Care*, vol. 27, no. 11, pp. 2642–2647, Nov. 2004.
[58] "iOS - o que é e para que serve." [Online]. Available: http://originsuk.com/o-que-ios-e-para-que-serve-. [Accessed: 17-May-2017].
[59] "iOS|Informática|TechTudo." [Online]. Available: http://www.techtudo.com.br/tudo-sobre/ios.html. [Accessed: 17-May-2017].
[60] "Apple utiliza 2 acelerómetros." [Online]. Available: http://www.tudocelular.com/apple/noticias/n42979/apple-usa-dois-acelerometros-no-iphone-6.html. [Accessed: 17-May-2017].
[61] "About - Objective C." [Online]. Available: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html. [Accessed: 08-May-2017].
[62] "Queres ser um iOS Developer?" [Online]. Available: https://pplware.sapo.pt/apple/queres-ser-um-ios-developer/. [Accessed: 08-May-2017].

[63] “The Swift Programming Language (Swift 4): About Swift.” [Online]. Available: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/index.html#//apple_ref/doc/uid/TP40014097-CH3-ID0. [Accessed: 08-May-2017].
[64] “A história do Android.” [Online]. Available: https://www.oficinadanet.com.br/post/13939-a-historia-do-android. [Accessed: 09-May-2017].
[65] “What is Android? What is an Android Phone? Beginners Guide to Android.” [Online]. Available: https://recombu.com/mobile/article/what-is-android-and-what-is-an-android-phone_M12615.html#. [Accessed: 09-May-2017].
[66] “FLIR Company History.” [Online]. Available: http://www.flir.com/about/display/?id=55679. [Accessed: 15-Jul-2017].
[67] “GrafHEATi.” [Online]. Available: http://grafheati-supporting-flir-one.appstor.io/. [Accessed: 28-Nov-2016].
[68] “FLIR ONE na App Store.” [Online]. Available: https://itunes.apple.com/pt/app/flir-one/id875842742?mt=8. [Accessed: 28-Nov-2016].
[69] “FLIR Tools na App Store.” [Online]. Available: https://itunes.apple.com/pt/app/flir-tools/id511247887?mt=8. [Accessed: 28-Nov-2016].
[70] “FLIRExperience na App Store.” [Online]. Available: https://itunes.apple.com/pt/app/flirexperience/id695092829?mt=8. [Accessed: 28-Nov-2016].
[71] “App Shopper: Vernier Thermal Analysis for FLIR ONE (Education).” [Online]. Available: http://appshopper.com/education/vernier-thermal-analysis-for-flir-one. [Accessed: 28-Nov-2016].
[72] “Thermal Camera Burst Mode on the App Store.” [Online]. Available: https://itunes.apple.com/us/app/thermal-camera-burst-mode/id995940862?mt=8. [Accessed: 28-Nov-2016].
[73] “ThermoPro: Thermal Imaging Reporting App on the App Store.” [Online]. Available: https://itunes.apple.com/us/app/thermopro-thermal-imaging/id1024216614?mt=8. [Accessed: 28-Nov-2016].
[74] “Thermal Preso on the App Store.” [Online]. Available: https://itunes.apple.com/th/app/thermal-preso/id975042259?mt=8&ign-mpt=uo%3D2. [Accessed: 28-Nov-2016].
[75] “Remote thermal cam f. FLIR ONE – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=de.killig.remotethermalcam. [Accessed: 28-Nov-2016].
[76] “Thermal Camera For Flir One – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=georg.com.flironetest_01. [Accessed: 28-Nov-2016].
[77] “Thermal Camera+ for Flir One – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=georg.com.thermal_camera_plus. [Accessed: 28-Nov-2016].
[78] “FLIR ONE – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.flir.flirone. [Accessed: 28-Nov-2016].
[79] “FLIR Tools Mobile – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.flir.viewer. [Accessed: 28-Nov-2016].
[80] “HeatReview For FLIR ONE – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.lancelotlam.heatreview. [Accessed: 28-Nov-2016].
[81] “DiveFLIR – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.divegames.diveflir. [Accessed: 28-Nov-2016].
[82] “FLIR ONE 1st generation.” [Online]. Available: http://www.flir.com/flirone/content/?id=69323. [Accessed: 28-Nov-2016].
[83] “FLIR ONE 2nd generation.” [Online]. Available: www.flir.com/flirone/android/. [Accessed: 28-Nov-2016].

[84] “FLIR ONE 3rd generation.” [Online]. Available: http://www.flir.com/flirone/ios-android/. [Accessed: 28-Nov-2016].
[85] “FLIR ONE 3rd generation PRO.” [Online]. Available: http://www.flir.com/flirone/pro/. [Accessed: 28-Nov-2016].
[86] “Seek Thermal CompactPro.” [Online]. Available: http://www.thermal.com/products/compactpro/. [Accessed: 28-Nov-2016].
[87] “TE Q1.” [Online]. Available: http://www.i3-thermalexpert.com/products/t-e-q1/. [Accessed: 28-Nov-2016].
[88] “TE Q1 Plus.” [Online]. Available: http://www.i3-thermalexpert.com/products/t-e-narrow/. [Accessed: 28-Nov-2016].
[89] “TE Q1 Pro.” [Online]. Available: http://www.i3-thermalexpert.com/products/t-e-q1-pro/. [Accessed: 28-Nov-2016].
[90] “TE V1.” [Online]. Available: http://www.i3-thermalexpert.com/products/t-e-v1/. [Accessed: 28-Nov-2016].
[91] “ThermApp.” [Online]. Available: https://therm-app.com/therm-app/. [Accessed: 28-Nov-2016].
[92] J. T. Hardwicke, O. Osmani, and J. M. Skillman, “Detection of Perforators Using Smartphone Thermal Imaging,” *Plast. Reconstr. Surg.*, vol. 137, no. 1, pp. 39–41, Jan. 2016.
[93] S. Suphachokauychai, K. Kiranantawat, and K. Sananpanich, “Detection of Perforators Using Smartphone Thermal Imaging,” *Plast. Reconstr. Surg. Glob. Open*, vol. 4, no. 5, p. e722, May 2016.
[94] “FLIR ONE 2nd generation iOS.” [Online]. Available: http://www.flir.com/flirone/ios/. [Accessed: 28-Nov-2016].
[95] “User Guide for Android.” [Online]. Available: http://www.flir.com/uploadedFiles/FLIR_ONE/Resources/FLIR-ONE-User-Guide-Android.pdf. [Accessed: 02-Feb-2017].
[96] Howel K J, Smith R E, “Guidelines for specifying and testing a thermal camera for medical applications,” *Thermol. Int.*, vol. 19(1), pp. 5–12, 2009.
[97] IEC 80601 2-59, “Medical electrical equipment - part 2 - 59: particular requirements for the basic safety and essential performance of screening thermography for human febrile temperature screening,” 1.0., 2008.
[98] “About SQLite.” [Online]. Available: http://www.sqlite.org/about.html. [Accessed: 05-Apr-2017].
[99] “What is HTTP? Definition from WhatIs.com.” [Online]. Available: http://searchwindevelopment.techtarget.com/definition/HTTP. [Accessed: 17-May-2017].
[100] “Developing a REST Webservice using C# - A walkthrough - Code Project.” [Online]. Available: https://www.codeproject.com/Articles/112470/Developing-a-REST-Web-Service-using-C-A-walkthroug. [Accessed: 17-May-2017].
[101] “Understanding SOAP and REST Basics and Differences.” [Online]. Available: https://blog.smartbear.com/apis/understanding-soap-and-rest-basics/. [Accessed: 17-May-2017].
[102] “What is SOAP? Definition from WhatIs.com.” [Online]. Available: http://searchmicroservices.techtarget.com/definition/SOAP-Simple-Object-Access-Protocol. [Accessed: 17-May-2017].
[103] “Philip Crosby: Zero Defects Thinker.” [Online]. Available: https://mbsportal.bl.uk/taster/subjareas/busmanhist/mgmtthinkers/crosby.aspx. [Accessed: 02-Jun-2017].

[104]“Joseph M Juran: Quality Management Thinker.” [Online]. Available: https://mbsportal.bl.uk/taster/subjareas/busmanhist/mgmtthinkers/juran.aspx. [Accessed: 02-Jun-2017].
[105]“img cliente.” [Online]. Available: http://2.bp.blogspot.com/-aAnYk5JsMmg/UdHXTULRiYI/AAAAAAAAFMs/kVbVbcMhNlA/s590/podologia+n%C3%A3o_lixar.jpg + https://contrastefashion.files.wordpress.com/2014/05/iphone-tn.gif?w=240. [Accessed: 15-Jul-2017].
[106]“img internet.” [Online]. Available: http://www.istockphoto.com/pt/vetorial/sinal-de-wi-fi-no-ecr%C3%A3-de-dispositivo-m%C3%B3vel-desenho-gm500260634-80642417?esource=SEO_GIS_CDN_Redirect. [Accessed: 15-Jul-2017].
[107]“img servidor.” [Online]. Available: https://pixabay.com/p-30050/?no_redirect. [Accessed: 15-Jul-2017].
[108]“img base de dados.” [Online]. Available: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSYW8dJBObVYFRCNu9FjJ7E10LcZnTYp25RhuOrg7VlRJeeojFaEw. [Accessed: 15-Jul-2017].
[109]Department of Defense, Systems Management College, “SYSTEMS ENGINEERING FUNDAMENTALS.” Defense Acquisition University Press, Jan-2001.
[110]“The Activity Lifecycle | Android Developers.” [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle.html. [Accessed: 08-Mar-2017].
[111]Ammer K, “The Glamorgan Protocol for recording and evaluation of thermal images of the human body,” *Thermol. Int.*, vol. 18(4), pp. 125–144, 2008.

Anexo A – Aplicações existentes no mercado

Nome	FLIR ONE	FLIR Tools	FLIRExperience	ThermoPro: Thermal Imaging Reporting App	Thermal Camera Burst Mode	Vernier Thermal Analysis for FLIR ONE	Thermal Preso	Remote Thermal Cam f. FLIR ONE	Thermal Camera For Flir One	Thermal Camera + For Flir One	HeatReview For Flir ONE	FLIR Tools Mobile	FLIR ONE	DiveFlir
Características														
Gratuita	x	x	x			x		x	x		x	x	x	x
Paga				79.99€	1,99$		69.03$			3.29€				
Precisa do acessório da FLIR One	x						x		x	x				
Transforma o ecrã numa câmara térmica	x													
Capacidade de ver no escuro	x												x	
Fornece uma imagem com diferenças mínimas de temperatura	x													
Observa fontes de calor invisíveis	x												x	
Compara temperaturas relativas	x												x	
Permite ver através do fumo	x												x	
Interface simples e intuitiva	x												x	
Apresenta uma imagem térmica em direto	x	x												
Capta fotografias e vídeos	x	x												
Permite carregar imagens diretamente para o rolo da câmara	x	x		x							x	x	x	
Partilha rápida e fácil das imagens nas redes sociais	x										x		x	
Inclui a sua própria finte de bateria	x												x	
Disponível em vários idiomas	x	x	x											
A camara da FLIR pode ser colocada num local remoto e ser controlada via wireless		x										x		
Permite importar imagens da câmara de infra vermelhos		x												
Faz zoom		x										x		
Permite mexer em ferramentas de medição diretamente na imagem		x												
Apaga imagens no rolo da câmara		x										x		
Fornece coordenadas GPS da imagem	x	x											x	
Diz a temperatura medida, permitindo a sua análise		x												
Permite criar imagens em papel e relatórios que podem ser enviados por email		x												
Impressão das imagens compatível com qualquer impressora AirPrint		x												
Permite guardar as imagens em serviços de partilha (ex. Dropbox)	x	x										x	x	
Mostra informações da imagem	x	x											x	
Permite reprodução de comentários de voz		x										x		
Possível mudar a palete de cores da imagem e ajustar níveis e medidas		x										x		
Permite alterar definições gerais na aplicação	x	x											x	
Tem registo e login obrigatórios		x												
Permite controlar emissividade, humidade relativa e temperatura refletida e atmosférica		x												
Vídeos: MPEG; Imagens:JPEG		x												
Permite a ligação a aparelhos de avaliação da FLIR e retirar informações de DM93 e CM83 e receber ficheiros DM93		x												
Inclui visão 360º			x											
Explica como funcionam as imagens térmicas			x											
Dá acesso a ficheiros da FLIR com características exclusivas			x											
Trta imagens térmicas da FLIR One				x										
Mostra a temperatura mais alta e mais baixa em tempo real				x										
Cria relatórios instantâneos com imagem de infra vermelhos, localização GPS, notas e assinatura				x										
Permite ecitar imagem no relatório e desenhar sobre ela				x										
Permite enviar por email, imprimir, abrir ou guardar o relatório só com um toque				x										
Guarda automaticamente e de forma contínua as edições de imagens				x										
Cria uma completa hierarquia do projeto e guarda como base de dados				x										
Permite avaliar medidas dessa hierarquia do projeto e criar relatórios				x										
Disponível em Inglês				x	x	x	x							
Permite só com um toque obter todas ou algumas das seguintes funções: Hottest, Artic, Coldest, Gray, Iron, Contrast, Rainbow e Normal					x									
Permite marcar até 4 localizações ou regiões numa imagem térmica						x								
Numa região selecicnada pode ser determinada a temperatura média, máxima ou mínima						x		x			x			
Fornece dados de temperatura durante um ensaio						x	x							
Permite exportar os dados para a aplicação Graphical Analysis para análise posterior						x								
Vídeos de imagens térmicas podem também ser exportados para o rolo da câmara						x	x							
Permite analisar as alterações de temperatura na pele						x								
Ilustra convecção						x								
Mostra calor devido à fricção						x								
Compara a condução de calor em diferentes materiais						x								
Compara a transparência dos materiais sob IR vs luz visível						x								
Instruções de definições e transmissões incluídas							x							
Permite transmitir o vídeo da análise térmica para a Apple TV							x							
Permite opção de rotação vertical ou horizontal e imagens em espelho							x							
Envia um fluxo MJPEG da cam. Térmica FLIR ONE ao PC via wifi								x						
Baseado no exemplo da FLIR ONE SDK	x							x					x	
Requer um servidor SmartCam para Windows								x						
Permite regular manualmente a gama/escala de cores	x								x	x			x	
Modo SDK MSX originais	x								x	x			x	
Tem maior precisão para os dados de calibração	x								x	x			x	
Permite bloqueio da faixa de temperatura									x	x				
Permite ver a temperatura no ecrã	x								x	x			x	
Tem galeria de imagens térmicas	x									x			x	
Permite ajusatar a faixa de temperatura										x				
Permite super resolução multi frame										x				
Permite alterar modo de nitidez (nas configurações)										x				
Correção do offset MSX										x				
Permte esconder a cor real da imagem										x				
Permite guardar : imagens MSX (Flir radiometric JPG), imagens Non-MSX (Algoritmo novo), imagens cor real										x				
Permite fazer comentários na imagem											x			
Não é compatível com a transmissão ao vivo de Cat S60 smartphone (para isso usar MyFlir App)												x		
Permite analisar, gerir e distribuir imagens IR												x		
Permite streaming de vídeos e de fotos												x		
MSX	x											x	x	
Tem mensagem de texto ecitável												x		
Tem uma tabela de emissividade												x		
Permite ler medições de temperatura												x		
Permite fazer log dos dados de medição e exportá-los como um arquivo* .csv												x		
Apresenta as coorderadas GPS de uma imagem no Google Maps												x		
Permite criar e enviar relatórios												x		
Mostra informações da imagem como comentários de texto e detalhes do ficheiro												x		
Tem capacidade para ver o que o olho nu não consegue	x											x	x	
Fornece uma imagem visual das diferenças de temperatura	x											x	x	
Tem uma variedade de funções que facilitam a colheita e partilha de magens térmicas atraentes e úties	x												x	
Permite obter a imagem térmica sobreposta (em cima) da imagem visual com shader de deteção de borda														x
Permite ver a temperatura média do centro em graus Celsius e Fahrenheit														x

Figura A. 1 - Aplicações existentes no mercado.

Anexo B – Estudo do SDK da FLIR ONE

ANDROID

Hierarquia

Hierarquia de pacote: com.flir.flironesdk

Hierarquia de classe:

- java.lang.Object
  - com.flir.flironesdk.Device (implements java.io.Closeable)
  - com.flir.flironesdk.Frame
  - com.flir.flironesdk.FrameProcessor
  - com.flir.flironesdk.RenderedImage

Hierarquia de interface:

- com.flir.flironesdk.Device.Delegate
- com.flir.flironesdk.Device.PowerUpdateDelegate
- com.flir.flironesdk.Device.StreamDelegate
- com.flir.flironesdk.FrameProcessor.Delegate

Hierarquia de enum:

- java.lang.Object
  - java.lang.Enum<E>(implementsjava.lang.Comparable<T>,java.io.Serializable)
    - com.flir.flironesdk.RenderedImage.Palette
    - com.flir.flironesdk.RenderedImage.ImageType
    - com.flir.flironesdk.Device.TuningState
    - com.flir.flironesdk.Device.BatteryChargingState

Pacote com.flir.flironesdk

- Sumário da interface

Interface	Descrição
Device.Delegate	Define os métodos de retorno de chamada para as atualizações do estado do dispositivo
Device.PowerUpdateDelegate	Métodos de retorno de chamadas de energia específicos
Device.StreamDelegate	Define o método de retorno da chamada para manipular eventos de captura de fluxo (*stream capture events*)
FrameProcessor.Delegate	Define o retorno de chamada para resultados de processamento de *frames*

- Sumário da classe

Classe	Descrição
Device	Representa um dispositivo FLIR ONE conectado, incluindo os métodos para interagir e capturar imagens e vídeos

EmbeddedDevice	Representa uma câmara da FLIR embutida
FlirUsbDevice	Suporte a dispositivos USB, FLIR ONE
Frame	Representa uma *raw frame* recebida da câmara que pode ser renderizada (*rendered*) numa imagem colorida, MSX ou visível com uma FrameProcessor
FrameProcessor	Converte de Frame para RenderedImage Processing. Inclui colorização, fusão MSX e termografia de temperatura precisa
LoadedFrame	Representa uma *frame* que foi carregada de um ficheiro
RenderedImage	Representa uma imagem renderizada (*rendered*) criada a partir de uma *frame*, a partir do dispositivo
SimulatedDevice	Um dispositivo simulado permite testar o código sem um dispositivo físico

- Sumário da enum

Enum	Descrição
Device.BatteryChargingState	Se estiver presente, representa o estado da bateria
Device.TuningState	Representa o estado de ajuste de calibração de campo plano (*flat-field calibration tuning*)
RenderedImage.ImageType	Contém todos os formatos de saída de *frames* suportados
RenderedImage.Palette	A paleta da *frame* colorida será nula para tipos de *frames* não aplicáveis

Constantes

com.flir.flironesdk.FrameProcessor		
Modificador e tipo	**Campo (*Constant Field*)**	**Valor**
public static final float	EMISSIVITY_GLOSSY	0.30000001192092896f
public static final float	EMISSIVITY_MATTE	0.949999988079071f
public static final float	EMISSIVITY_SEMI_GLOSSY	0.6000000238418579f
public static final float	EMISSIVITY_SEMI_MATTE	0.800000011920929f

Descrição Detalhada

- **Enum Device.BatteryChargingState**
  - java.lang.Object
    - java.lang.Enum<Device.BatteryChargingState>
      - com.flir.flironesdk.Device.BatteryChargingState

Todas as interfaces implementadas:

- java.io.Serializable, java.lang.Comparable<Device.BatteryChargingState>

Classe de encerramento:

- Device

---

public static enum Device.BatteryChargingState
extends java.lang.Enum<Device.BatteryChargingState>

Se presente, representa o estado da bateria.

- **Sumário do método**

Modificador e tipo	Método e descrição
static Device.BatteryChargingState	get(int value)
static Device.BatteryChargingState	valueOf(java.lang.String name) Retorna a constante enum desse tipo com o nome especificado
static Device.BatteryChargingState[]	values() Retorna uma matriz que contém as constantes desse tipo de enum, na ordem por que são declaradas

- Métodos herdados da classe java.lang.Enum

compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

- Métodos herdados da classe java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

- **Detalhes da constante enum**

NO_CHARGING
public static final Device.BatteryChargingState NO_CHARGING
Indica que a bateria RBPDevice não está a ser carregada porque o RBPDevice não está conectado a uma fonte de alimentação externa
MANAGED_CHARGING
public static final Device.BatteryChargingState MANAGED_CHARGING
Indica que a bateria RBPDevice está a ser carregada a partir de uma fonte de alimentação externa
FAULT
public static final Device.BatteryChargingState FAULT
Indica que ocorreu uma falha de carga inesperada (*bad battery*, etc.)
FAULT_HEAT
public static final Device.BatteryChargingState FAULT_HEAT

Indica que ocorreu uma falha de carregamento, devido ao calor
FAULT_BAD_CHARGER
public static final Device.BatteryChargingState FAULT_BAD_CHARGER
Indica que ocorreu uma falha de carga devido à baixa corrente da fonte de carga
CHARGING_SMART_PHONE_FAULT_HEAT
public static final Device.BatteryChargingState CHARGING_SMART_PHONE_FAULT_HEAT
Indica que existe uma falha ao carregar, mas que o iPhone está a ser carregado
MANAGED_CHARGING_ONLY
public static final Device.BatteryChargingState MANAGED_CHARGING_ONLY
Indica que o dispositivo está no modo *charge-only*
CHARGING_SMART_PHONE_ONLY
public static final Device.BatteryChargingState CHARGING_SMART_PHONE_ONLY
Indica que o dispositivo está no modo *phone-charging-only*
BAD
public static final Device.BatteryChargingState BAD
Indica que um estado de carga da bateria RBPDevice, que é válido, não estava disponível

- **Detalhes do método**

values
public static Device.BatteryChargingState[] values()
Retorna uma matriz que contém as constantes desse tipo de enum, na ordem por que são declaradas. Este método pode ser usado para iterar sobre as constantes da seguinte forma:
for (Device.BatteryChargingState c : Device.BatteryChargingState.values()) System.out.println(c);
Retorna:
Uma matriz que contém as constantes desse tipo de enum, na ordem por que são declaradas
valueOf
public static Device.BatteryChargingState valueOf(java.lang.String name)
Retorna a constante enum desse tipo com o nome especificado. A sequência de caracteres deve coincidir exatamente com um identificador usado para declarar uma constante de enumeração nesse tipo.
Parâmetros:
name - O nome do enum constante a ser retornado
Retorna:
O enum constante com o nome especificado
Lança:
• java.lang.IllegalArgumentException - Se esse tipo de enum não tiver nenhuma constante com o nome especificado

• java.lang.NullPointerException - se o argumento for nulo
get
public static Device.BatteryChargingState get(int value)

- **Interface Device.Delegate**

Classe de encerramento:

- Device

---

public static interface Device.Delegate

Define métodos de retorno de chamada para atualizações de estado do dispositivo.

- **Detalhes do método**

onTuningStateChanged
void onTuningStateChanged(Device.TuningState newTuningState)
Chamado quando o estado de sintonia (*tuning*) do dispositivo muda, o que indicará que: • o dispositivo está em processo de sintonia (*tuning*), • concluiu a sintonia (*tuning*) ou • pode requerer sintonia (*tuning*) para alta precisão térmica.
Parâmetros:
newTuningState - O novo estado de sintonia (*tuning*) do dispositivo
onAutomaticTuningChanged
void onAutomaticTuningChanged(boolean deviceWillTuneAutomatically)
Chamado quando o dispositivo confirma a ativação ou desativação da função de ajuste (*tuning*) automático.
Parâmetros:
deviceWillTuneAutomatically - A configuração recém-aplicada para ajuste (*tuning*) automático. *true* se o dispositivo sintonizar automaticamente.
onDeviceConnected
void onDeviceConnected(Device device)
Chamado quando o dispositivo tiver concluído a conexão e estiver pronto para transmissão em fluxo contínuo (*streaming*)
Parâmetros:
device - O objeto que representa o dispositivo conectado. Este objeto deve ser usado para executar operações de dispositivo, como iniciar *streaming* de *frames* e controlar o ajuste (*tuning*)
onDeviceDisconnected
void onDeviceDisconnected(Device device)
Chamado quando o dispositivo foi desconectado por alguma razão
Parâmetros:
device - o dispositivo que foi desconectado

- **Class Device**
  - java.lang.Object
    - com.flir.flironesdk.Device

Todas as interfaces implementadas:

- java.io.Closeable, java.lang.AutoCloseable

---

public class Device
extends java.lang.Object
implements java.io.Closeable

Representa um dispositivo FLIR ONE conectado, incluindo os métodos para interação e captura de imagens e vídeos.

- **Sumário da classe**

Modificador e tipo	Classe e descrição
static class	Device.BatteryChargingState Se presente, representa o estado da bateria
static interface	Device.Delegate Define métodos de retorno de chamada para atualizações de estado do dispositivo
static interface	Device.PowerUpdateDelegate Métodos de retorno de chamada específicos de energia
static interface	Device.StreamDelegate Define o método de retorno de chamada para manipular eventos de captura de fluxo (*stream capture events*)
static class	Device.TuningState Representa o estado do ajuste de calibração de campo plano (*flat-field calibration tuning*)

- Métodos herdados da classe java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- **Detalhes do construtor**

Construtor e tipo
Device
public Device(Device.Delegate delegate)
Constrói um dispositivo simulado que pode transmitir um *frame loop* fornecido num ficheiro zip com connectSimulatedDevice

- **Detalhes do método**

startFrameStream
public void startFrameStream(Device.StreamDelegate delegate)
Chamar depois de onDeviceConnected ser chamado no delegado do dispositivo para iniciar o *streaming* da *frame*. Este método não terá efeito se for chamado antes de o dispositivo estar pronto
stopFrameStream
public void stopFrameStream()
Chamar para parar de fazer *streaming* de *frames*. Este método não terá efeito quando chamado se as *frames* não estão em *streaming*
startDiscovery
public static void startDiscovery(Context context, Device.Delegate delegate) throws java.lang.IllegalStateException
Chamar para se conectar a um dispositivo FLIR ONE: normalmente deve-se chamar o método onResume da sua atividade, passando o contexto da app e o delegado do dispositivo. Isto irá iniciar um recetor de difusão que ouve os dispositivos conectados. Ele chamará o método deviceDiscovered do delegado quando algum for criado dessa maneira
Parâmetros:
• **context -** O contexto da aplicação, usado para registar recetor de broadcast
• **delegate -** O delegado para dispositivos descobertos a serem criados com
Lança:
java.lang.IllegalStateException - Quando chamado várias vezes sucessivas sem chamar stopDiscovery
stopDiscovery
public static void stopDiscovery()
Chamar durante o onPause para parar de ouvir os dispositivos conectados por USB. Isso impede que a aplicação atual capture eventos de conexão em segundo plano, quando outras aplicações podem precisar de os receber
connectSimulatedDevice
public void connectSimulatedDevice(java.io.InputStream sampleFramesZipData) throws java.io.IOException
Parâmetros:
sampleFramesZipData - Deve ser um InputStream para um ficheiro zip de pares de *raw frames* térmicos e visíveis, com nomes de ficheiro do formato lep-00001 para térmica e vis-00001 para o visível. A aplicação de exemplo SDK tem um recurso *raw* em

R.raw.sampleframes
Lança:
java.io.IOException
setPowerUpdateDelegate
public void setPowerUpdateDelegate(Device.PowerUpdateDelegate powerUpdateDelegate)
Se o dispositivo tiver uma bateria, as atualizações do estado de energia serão enviadas para o delegado de atualização de energia
Parâmetros:
powerUpdateDelegate - O objeto para receber atualizações de energia
performTuning
public void performTuning()
Chamar se a afinação (*tuning*) for necessária ou solicitada para alta precisão térmica
setAutomaticTuning
public void setAutomaticTuning(boolean shouldAutomaticallyTune)
Desativar ou ativar o recurso de ajuste (*tuning*) automático do dispositivo. O dispositivo tem como padrão a sintonia (*tuning*) automática.
Parâmetros:
shouldAutomaticallyTune - Definir como *false* para evitar que a função de ajuste (*tuning*) automático seja executada. Definir como *true* para reativar
Close
public void close()
Chamar quando o dispositivo USB é desconectado, não é mais necessário ou se a aplicação está em pausa
Especificado por:
• close in interface java.io.Closeable
• close in interface java.lang.AutoCloseable

- **Interface Device.PowerUpdateDelegate**

Classe de encerramento:

- Device

---

public static interface Device.PowerUpdateDelegate

Métodos de retorno de chamada específicos de energia.

- **Detalhes do método**

onBatteryChargingStateReceived
void onBatteryChargingStateReceived(Device.BatteryChargingState batteryChargingState)
Chamado sempre que o estado de carga da bateria muda
Parâmetros:
batteryChargingState
onBatteryPercentageReceived
void onBatteryPercentageReceived(byte percentage)
Chamado sempre que a percentagem de carga da bateria se altera
Parâmetros:
percentage - entre 0 e 100, inclusive

- **Interface Device.StreamDelegate**

Classe de encerramento:

- Device

---

public static interface Device.StreamDelegate

Define o método de retorno de chamada para manipular eventos de captura de fluxo (*stream capture events*).

- **Detalhes do método**

onFrameReceived
void onFrameReceived(Frame frame)
Chamado quando uma *frame* foi recebida durante o *streaming*
Parâmetros:
Frame

- **Enum Device.TuningState**
  - java.lang.Object
    - java.lang.Enum<Device.TuningState>
      - com.flir.flironesdk.Device.TuningState

Todas as interfaces implementadas:

- java.io.Serializable, java.lang.Comparable<Device.TuningState>

Classe de encerramento:

- Device

---

public static enum Device.TuningState
extends java.lang.Enum<Device.TuningState>

Representa o estado do ajuste de calibração de campo plano (*flat-field calibration tuning*).

- Métodos herdados da classe java.lang.Enum

compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

- Métodos herdados da classe java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

- **Detalhes da constante enum**

TuningRequired
public static final Device.TuningState TuningRequired
O dispositivo FLIR ONE necessita de afinação (*tuning*). Quando está neste estado, a afinação (*tuning*) é necessária para que as *frames* tenham dados de temperatura radiométrica
InProgress
public static final Device.TuningState InProgress
O dispositivo FLIR ONE está a ser ajustado, mas ainda não terminou. A UI (interface) de sintonia deve ser atualizada, permitindo que o utilizador saiba que o processo de sintonia está em andamento.
Tuned
public static final Device.TuningState Tuned
O dispositivo FLIR ONE foi ajustado (*tuned*).
TuningSuggested
public static final Device.TuningState TuningSuggested
A câmara está a solicitar, mas não necessita de ajuste (*tuning*). Se a aplicação requer dados de temperatura de alta qualidade, deve chamar-se performTuning.
ApproximatelyTuned
public static final Device.TuningState ApproximatelyTuned
A câmara iniciou-se recentemente e os valores de temperatura medidos podem ser ligeiramente imprecisos.
Unknown
public static final Device.TuningState Unknown
Um estado de sintonia (*tuning*) inválido. Não se deve receber este estado do dispositivo FLIR ONE.

- **Detalhes do método**

Values
public static Device.TuningState[] values()
Retorna uma matriz contendo as constantes desse tipo de enum, na ordem por que são declaradas. Este método pode ser usado para iterar sobre as constantes da seguinte maneira:
for (Device.TuningState c : Device.TuningState.values()) System.out.println(c);
Retorna:
Uma matriz que contém as constantes desse tipo de enumeração, na ordem por que são declaradas
valueOf
public static Device.TuningState valueOf(java.lang.String name)
Retorna a constante enum desse tipo com o nome especificado. A sequência de caracteres deve coincidir exatamente com um identificador usado para declarar uma constante de enumeração nesse tipo.
Parâmetros:
name - O nome do enum constante a ser retornado
Retorna:
O enum constante com o nome especificado
Lança:
• java.lang.IllegalArgumentException - Se esse tipo de enum não tiver nenhuma constante com o nome especificado
• java.lang.NullPointerException - se o argumento for nulo

- **Class Frame**
  - java.lang.Object
    - com.flir.flironesdk.Frame

public class Frame
extends java.lang.Object

Representa uma *raw frame* recebida da câmara que pode ser renderizada (*rendered*) numa imagem colorida, MSX ou apenas visível com uma instância do FrameProcessor.

- **Sumário do método**

Modificador e tipo	Método e descrição
static Frame	load(java.io.InputStream inputStream) Carrega uma *frame* guardada de um fluxo de entrada (*input stream*)
void	save(java.io.OutputStream outputStream, RenderedImage.Palette previewPalette) Guarda uma *frame* num fluxo de saída (*output stream*)

void	save(java.lang.String path, RenderedImage.Palette previewPalette) Guarda um ficheiro JPEG térmico, que possui uma visualização (*preview*) visual renderizada (*rendered*) e dados térmicos incorporados.

- Métodos herdados da classe java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- **Detalhes do método**

Load
public static Frame load(java.io.InputStream inputStream) throws java.io.IOException
Carrega uma *frame* guardada de um fluxo de entrada (*input stream*). AVISO: o formato destes dados será alterado antes da versão final
Parâmetros:
inputStream
Retorna:
instantiated Frame object
Lança:
java.io.IOException

- **Interface FrameProcessor.Delegate**

Classe de encerramento:

- FrameProcessor

public static interface FrameProcessor.Delegate

Define o retorno de chamada para resultados de processamento de *frames*.

- **Detalhes do método**

onFrameProcessed
void onFrameProcessed(RenderedImage renderedImage)
Chamado quando uma *frame* foi recebida e processada. Observação: se N formatos foram solicitados, este método será chamado N vezes por *frame* passada para a instância FrameProcessor. Se o objetivo é gravar um vídeo a partir de uma sequência de *frames*, usar

android.media.MediaCodec e chamar mediaCodec.queueInputBuffer a partir deste método com o *frame's* pixelData *byte array* colocar na entrada ByteBuffer do mediaCodec
Parâmetros:
renderedImage - A imagem após o processamento foi aplicada

- **Class FrameProcessor**
  - java.lang.Object
    - com.flir.flironesdk.FrameProcessor

public final class FrameProcessor
extends java.lang.Object

Esta classe converte de Frame para RenderedImage. O processamento inclui colorização, MSX fusão e termografia de precisão de temperatura.

- **Sumário da classe**

Modificador e tipo	Classe e descrição
static interface	FrameProcessor.Delegate Define o retorno de chamada para resultados de processamento de *frames*

  - Métodos herdados da classe java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- **Detalhes do campo**

EMISSIVITY_GLOSSY
public static final float EMISSIVITY_GLOSSY
A emissividade brilhante é para a leitura de temperaturas de materiais como aço
EMISSIVITY_SEMI_GLOSSY
public static final float EMISSIVITY_SEMI_GLOSSY
Semi-glossy é para a leitura de temperaturas de materiais semirreflexivos
EMISSIVITY_SEMI_MATTE
public static final float EMISSIVITY_SEMI_MATTE
EMISSIVITY_MATTE
public static final float EMISSIVITY_MATTE
Matte é a emissividade padrão e é útil para a leitura de temperaturas de pele, madeira e a maioria dos outros itens domésticos

- **Detalhes do construtor**

Construtor e tipo
FrameProcessor
public FrameProcessor(Context context, FrameProcessor.Delegate delegate, java.util.Set<RenderedImage.ImageType> types) throws java.lang.IllegalArgumentException
Cria um processador de *frames* para converter *frames* térmicas em imagens que podem ser exibidas
Parâmetros:
• **context** - O contexto da aplicação, necessário para armazenar em cache os dados de calibração
• **delegate** - Uma instância de sua implementação de delegado que receberá *frames* processadas
• **types** - Um conjunto de tipos de *frames* renderizadas (*rendered*) para serem entregues ao delegado ao processar uma *frame*. Embora não seja necessário, recomenda-se usar um ficheiro java.util.EnumSet
Lança:
java.lang.IllegalArgumentException - Se for dada uma combinação inválida de tipos de *frames*

- **Detalhes do método**

setImageTypes
public void setImageTypes(java.util.Set<RenderedImage.ImageType> types)
Para receber imagens renderizadas (*rendered*), o tipo de *frame* deve primeiro ser definido para permitir que o FrameProcessor saiba quais os formatos de imagem a enviar ao delegado. Passar uma lista de valores Enum ImageType em qualquer combinação com a exceção de ThermalRGBA8888Image e BlendedMSXRGBA8888Image, que pode não ser definida ao mesmo tempo. Quando uma *frame* é processada, cada tipo desta será enviado para delegate.onFrameReceived
Parâmetros:
types - Conjunto de FrameTypes a ser processado. Embora não seja necessário, recomenda-se o uso de um ficheiro java.util.EnumSet
Lança:
java.lang.IllegalArgumentException - Se uma combinação inválida de tipos de *frames* for fornecida
getImageTypes
public java.util.Set<RenderedImage.ImageType> getImageTypes()

Obtém um conjunto somente de leitura dos tipos de *frames* atualmente selecionados. Não tentar modificar este conjunto. Pode-se copiar o conjunto, fazer modificações para essa cópia e, em seguida, passá-lo para setImageTypes
setImagePalette
public void setImagePalette(RenderedImage.Palette palette)
Define a paleta a ser usada para imagens coloridas
Parâmetros:
palette
getImagePalette
public RenderedImage.Palette getImagePalette()
Obtém a paleta atualmente selecionada para imagens coloridas
setEmissivity
public void setEmissivity(float emissivity)
Define o fator de emissividade a ser usado ao processar a *frame* (avançado)
Parâmetros:
emissivity
getEmissivity
public float getEmissivity()
Obtém o fator de emissividade que está a ser usado para processar *frames*.
setMSXDistance
public void setMSXDistance(float distance)
Definir o parâmetro de distância do ponto de foco MSX
Parâmetros:
distance - em metros
getMSXDistance
public double getMSXDistance()
Obter o parâmetro de distância MSX atual
Retorna:
MSX atual especificando a distância MSX em metros
processFrame
public void processFrame(Frame frame)
Processa a *frame*, resultando numa *frame* renderizada (*rendered*) que é enviada para o método onFrameProcessed do delegado para cada tipo de *frame* solicitada por setImageTypes ou passada para o construtor.
Parâmetros:
frame - A *frame* recebida do dispositivo ou carregada a partir do armazenamento.

- **Class RenderedImage**
  - java.lang.Object
    - com.flir.flironesdk.RenderedImage

public class RenderedImage
extends java.lang.Object

Representa uma imagem renderizada (*rendered*) criada a partir de uma *frame* a partir do dispositivo.

- **Sumário da classe**

Modificador e tipo	Classe e descrição
static class	RenderedImage.ImageType Contém todos os formatos de saída de *frames* suportados
static class	RenderedImage.Palette A paleta de uma *frame* colorida será nula para tipos de *frames* não aplicáveis

- Métodos herdados da classe java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- **Detalhes do método**

getFrame
public Frame getFrame()
Retorna o objeto Frame original de que esta imagem renderizada (*rendered*) foi criada
Retorna:
O objeto raw Frame
imageType
public RenderedImage.ImageType imageType()
Obtém o tipo de *frame*
Retorna:
O formato da imagem
palette
public RenderedImage.Palette palette()
Obtém a paleta usada para colorir a imagem térmica para os tipos de *frame* aplicáveis
Retorna:
Paleta usada para a imagem colorida ou nula para uma imagem não colorida
emissivity
public float emissivity()
Emissividade refere-se ao grau de reflexividade de um material é
Retorna:
O ajuste de emissividade afeta somente as leituras de temperatura
width

public int width()
height
public int height()
pixelData
public byte[] pixelData()
Obtém os dados de *raw pixel* como uma matriz de bytes. Verificar o tipo de *frame* para o formato dos dados.
Retorna:
Dados *raw pixel*

- **Enum RenderedImage.ImageType**
  - java.lang.Object
    - java.lang.Enum<RenderedImage.ImageType>
      - com.flir.flironesdk.RenderedImage.ImageType

Todas as interfaces implementadas:

- java.io.Serializable, java.lang.Comparable<RenderedImage.ImageType>

Classe de encerramento:

- RenderedImage

---

public static enum RenderedImage.ImageType
extends java.lang.Enum<RenderedImage.ImageType>

Contém todos os formatos de saída de *frames* suportados.

- Métodos herdados da classe java.lang.Enum

compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

- Métodos herdados da classe java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

- **Detalhes da constante enum**

ThermalLinearFlux14BitImage
public static final RenderedImage.ImageType ThermalLinearFlux14BitImage
Dados de imagem lineares de 14 bits, preenchidos para 16 bits por pixel
ThermalRGBA8888Image
public static final RenderedImage.ImageType ThermalRGBA8888Image
Dados de imagem RGBA térmica externa
BlendedMSXRGBA8888Image
public static final RenderedImage.ImageType BlendedMSXRGBA8888Image
MSX (térmica + visual) Dados de imagem RGBA
VisualJPEGImage

public static final RenderedImage.ImageType VisualJPEGImage
Dados de imagem Visual JPEG
VisualYCbCr888Image
public static final RenderedImage.ImageType VisualYCbCr888Image
Dados de imagem do Visual YCbCr
ThermalRadiometricKelvinImage
public static final RenderedImage.ImageType ThermalRadiometricKelvinImage
Dados de temperatura do centikelvin radiométrico (cK)

- **Detalhes do método**

values
public static RenderedImage.ImageType[] values()
Retorna uma matriz que contém as constantes desse tipo de enum, na ordem por que são declaradas. Este método pode ser usado para iterar sobre as constantes da seguinte maneira:
for (RenderedImage.ImageType c : RenderedImage.ImageType.values()) System.out.println(c);
Retorna:
Uma matriz que contém as constantes desse tipo de enum, na ordem por que são declaradas
valueOf
public static RenderedImage.ImageType valueOf(java.lang.String name)
Retorna a constante enum desse tipo com o nome especificado. A sequência de caracteres deve coincidir exatamente com um identificador usado para declarar uma constante de enumeração nesse tipo. (Não são permitidos caracteres de espaço em branco estranhos.)
Parâmetros:
name - O nome do enum constante a ser retornado
Retorna:
O enum constante com o nome especificado
Lança:
• java.lang.IllegalArgumentException - Se esse tipo de enum não tiver nenhuma constante com o nome especificado
• java.lang.NullPointerException - se o argumento for nulo
isColorized
public boolean isColorized()

- **Enum RenderedImage.Palette**
  - java.lang.Object
    - java.lang.Enum<RenderedImage.Palette>
      - com.flir.flironesdk.RenderedImage.Palette

Todas as interfaces implementadas:

- java.io.Serializable, java.lang.Comparable<RenderedImage.Palette>

Classe de encerramento:

- RenderedImage

---

public static enum RenderedImage.Palette
extends java.lang.Enum<RenderedImage.Palette>

A paleta de uma *frame* colorida será nula para tipos de *frame* não aplicáveis.

- Métodos herdados da classe java.lang.Enum

compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

- Métodos herdados da classe java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

- **Detalhes da constante enum**

Iron
public static final RenderedImage.Palette Iron
Gray
public static final RenderedImage.Palette Gray
Rainbow
public static final RenderedImage.Palette Rainbow
Contrast
public static final RenderedImage.Palette Contrast
Arctic
public static final RenderedImage.Palette Arctic
Lava
public static final RenderedImage.Palette Lava
Wheel
public static final RenderedImage.Palette Wheel
Coldest
public static final RenderedImage.Palette Coldest
Hottest
public static final RenderedImage.Palette Hottest

- **Detalhes do método**

values
public static RenderedImage.Palette[] values()
Retorna uma matriz que contém as constantes desse tipo de enum, na ordem por que são declaradas. Este método pode ser usado para iterar sobre as constantes da seguinte maneira:
for (RenderedImage.Palette c : RenderedImage.Palette.values())

System.out.println(c);
Retorna:
Uma matriz que contém as constantes desse tipo de enum, na ordem por que são declaradas
valueOf
public static RenderedImage.Palette valueOf(java.lang.String name)
Retorna a constante enum desse tipo com o nome especificado. A sequência de caracteres deve coincidir exatamente com um identificador usado para declarar uma constante de enumeração nesse tipo. (Não são permitidos caracteres de espaço em branco estranhos.)
Parâmetros:
name - O nome do enum constante a ser retornado
Retorna:
O enum constante com o nome especificado
Lança:
• java.lang.IllegalArgumentException - Se esse tipo de enum não tiver nenhuma constante com o nome especificado
• java.lang.NullPointerException - se o argumento for nulo

iOS

Hierarquia

Hierarquia de classe:

- NSObject
  - FLIROneSDK
  - FLIROneSDKDelegateManager
    - FLIROneSDKImageEditor
    - FLIROneSDKStreamManager
  - FLIROneSDKImageMetadata
  - FLIROneSDKLibraryManager
  - FLIROneSDKPalette
- UIActivity
  - FLIROneSDKShareActivity
- UICollectionViewController
  - FLIROneSDKLibraryViewController
- UIImage
  - FLIROneSDKUIImage
- UIViewController
  - FLIROneSDKTuningViewController

Referências do protocolo:

- FLIROneSDKImageEditorDelegate
- FLIROneSDKImagePropertyController
- FLIROneSDKImageReceiverDelegate
- FLIROneSDKStreamManagerDelegate
- FLIROneSDKVideoRendererDelegate

Referências de constante:

- FLIROneSDKBatteryChargingState
- FLIROneSDKCaptureStatus
- FLIROneSDKImageIOStatus
- FLIROneSDKImageOptions
- FLIROneSDKTuningState

Classes

- **FLIROneSDK Class Reference**

Herda:

- NSObject

Declarado em:

- FLIROneSDK.h

- **Descrição**

Informa o desenvolvedor da versão do SDK;

Mantém as definições do utilizador nas unidades internacionais (UI) predefinidas - °C e °F.

- **Tarefas**

Aceder à SDK partilhada	sharedInstance
Aceder à versão SDK	version
	userInterfaceUsesCelsius (propriedade)

- **Propriedades**

userInterfaceUsesCelsius
O valor da temperatura por defeito é baseado na localização do telefone - °C ou °F
@property (nonatomic) BOOL userInterfaceUsesCelsius

- **Métodos da classe**

sharedInstance
Acessa, exclusivamente, e retorna uma referência ao *singleton* global (inicializa o *singleton* se este ainda não tiver sido inicializado)
+ (instancetype)sharedInstance
Retorna:
Uma referência ao *singleton* global
Detalhes:
Acessa, exclusivamente, e retorna uma referência ao *singleton* global (inicializa o *singleton* se este ainda não tiver sido inicializado)
Mantém um registo dos objetos registados como delegados do FLIROneSDK e é responsável por adicionar e remover delegados
O registo de um FLIROneSDKDelegate é necessário para receber todas e quaisquer informações do dispositivo FLIR ONE, incluindo o estado da conexão e as *frames* a ser recebidas

- **Métodos *Instance***

version
Retorna a versão atual do SDK (só alterar se se utilizar uma versão SDK diferente)
(NSString *)version

- **FLIROneSDKDelegateManager Class Reference**

Herda:
- NSObject

Declarado em:
- FLIROneSDKDelegateManager.h

- **Descrição**

É uma classe base, subdividida em ImageEditor and StreamManager.

- **FLIROneSDKImageEditor Class Reference**

Herda:
- FLIROneSDKDelegateManager: NSObject

Conforme:

- FLIROne SDKImagePropertyController

Declarado em:

- FLIROneSDKImageEditor.h

- **Descrição**

Usado para carregar, manipular, exibir e guardar imagens obtidas pelo FLIROneSDKStreamManager;

Usando o método - FLIROneSDKStreamManager capturePhoto - as imagens podem ser guardadas na biblioteca;

Esta classe só pode ser usada para editar fotos capturadas pelo FLIROneSDKStreamManager.

O workflow desta classe é o seguinte:

1. loadImageWithFilepath
2. Executar quaisquer métodos de edição como, por exemplo, setPalette:
3. finishEditingAndSaveCurrentImageToFilepath:withPreviewImage:

Os recursos implementados nesta classe estão incluídos no recurso de edição do UI (interface) pré-construído FLIROneSDKLibraryViewController incluído no SDK.

É de salientar que esta classe é *thread-safe*. Isto é, pode ser chamada em qualquer método de qualquer *thread* sem problema.

Avsio: Não tentar criar outra instância desta classe. Usar sempre o método da classe sharedInstance para recuperar o *singleton* global.

- **Tarefas**

Aceder à FLIROneSDKImageEditor partilhada	sharedInstance
Adicionar e remover delegados	addDelegate
	removeDelegate
Carregar e guardar uma imagem	loadImageWithFilepath
	refreshCurrentImageWithCurrentImageOptions
	finishEditingAndSaveCurrentImageToFilepath:withPreviewImage

- **Métodos da classe**

sharedInstance
Acessa, exclusivamente, e retorna uma referência ao *singleton* global (inicializa o *singleton* se este ainda não tiver sido inicializado)
+ (instancetype)sharedInstance
Retorna:
Uma referência ao *singleton* global

Detalhes:

O FLIROneSDKImageEditor é acedido exclusivamente através de uma referência à instância *singleton*. Todos os métodos da instância serão chamados neste objeto *singleton*

- **Métodos *Instance***

addDelegate

Registo de um novo delegado (sem efeito se o delegado já tiver sido adicionado anteriormente)

- (void)addDelegate:(id<FLIROneSDKImageEditorDelegate,FLIROneSDKImageReceiverDelegate>)delegate

Parâmetros:

delegate - O objeto para registar eventos de retorno de chamada FLIR ONE

Detalhes:

Como um exemplo para registar o controlador de exibição atual como um FLIROneSDKDelegate:

[FLIROneSDK sharedInstance] addDelegate:self];

finishEditingAndSaveCurrentImageToFilepath:withPreviewImage

Guarda no disco a imagem que está a ser editada.

Cada chamada para este método resultará em que o retorno de chamada FLIROneSDKEditorImageDidFinishSaving seja acionado para todos os delegados registados.

Além disso, seguindo este método, nenhuma edição adicional poderá ser executada até que uma nova imagem seja carregada usando loadImageWithFilepath:

(void)finishEditingAndSaveCurrentImageToFilepath:(NSURL *)filepath withPreviewImage:(UIImage *)previewImage

Parâmetros:

filepath - O caminho do ficheiro onde a imagem deve ser guardada. Para guardar no local, especificar o caminho do ficheiro usado para carregar a imagem originalmente

previewImage - A imagem de visualização (*preview*) para usar com a imagem de *dipslay* do JPEG e ficheiro de miniatura (*thumbnail file*)

loadImageWithFilepath

Carrega uma imagem com o nome especificado do ficheiro da biblioteca para o editor.

Todas as chamadas para este método resultam no retorno da chamada FLIROneSDKEditorImageDidFinishLoading a ser disparado para todos os delegados registados

- (void)loadImageWithFilepath:(NSURL *)filepath

Parâmetros:

filepath - A localização do caminho do ficheiro de onde a imagem vai ser carregada

NOTA: Se só for necessário recuperar uma miniatura (*thumbnail*) para um ficheiro, use o método thumbnailForFilename do FLIROneSDKLibraryManager

refreshCurrentImageWithCurrentImageOptions

Atualizar a imagem carregada usando a propriedade imageOptions do editor de imagens

- (void)refreshCurrentImageWithCurrentImageOptions
Detalhes:
Isto ativará *callbacks* para todos os dados associados.
removeDelegate
Anular a subscrição de um objeto previamente registado com addDelegate para receber eventos do dipositivo FLIR ONE (sem efeito se o delegado não foi adicionado)
- (void)removeDelegate:(id<FLIROneSDKImageEditorDelegate,FLIROneSDKImageReceiverDelegate>)delegate
Parâmetros:
delegate - O objeto para cancelar a inscrição dos eventos de retorno de chamada do dispositivo FLIR ONE

- **FLIROneSDKImageMetadata Class Reference**

Herda:
- NSObject

Declarado em:
- FLIROneSDKImageMetadata.h

- **Descrição**

Descreve o estado do editor de fluxo (*state of the stream*) ou imagem, incluindo paleta atual e emissividade;

- **Tarefas**

Paleta atual	palette (propriedade)
Emissividade	emissivity (propriedade)

- **Propriedades**

emissivity
Emissividade da imagem/*frame* atual
@property (nonatomic, readonly) CGFloat emissivity
palette
Paleta associada à imagem/*frame* atual
@property (strong, nonatomic, readonly) FLIROneSDKPalette *palette

- **FLIROneSDKLibraryManager Class Reference**

Herda:
- NSObject

Declarado em:
- FLIROneSDKLibraryManager.h

- **Descrição**

Gere imagens e videos que foram guardados no disco através das classes FLIROneSDKStreamManager e FLIROneSDKImageEditor.

É de salientar que esta classe só pode gerir fotos e vídeos guardados no disco via FLIROneSDKStreamManager e FLIROneSDKImageEditor.

AVISO: Deve usar-se esta classe para remover ficheiros em vez de o fazer manualmente.

É, ainda, de notar que se pode optar por guardar as imagens em qualquer local no sistema de ficheiros, mas quaisquer imagens guardadas no *media directory* poderão ser usadas pela biblioteca UI.

AVISO: Não tentar criar outra instância desta classe. Usar sempre o método de classe sharedInstance para recuperar o singleton global.

- **Tarefas**

Aceder à instância FLIROneSDKLibraryManager partilhada	+ sharedInstance
Outros métodos	- thumbnailForFilepath
	- previewImageForVideoWithFilepath
	- libraryFilepathForCurrentTimestampWithExtension
	- deleteLibraryFileWithFilepath
	- copyImageToLibraryWithFilepath:withThumbnailFilepath
	- fileIsVideo

- **Propriedades**

filepath
Uma matriz de caminhos de ficheiro (*array of filepaths*) para todas as imagens e vídeos na biblioteca. Cada item é um NSURL que representa a localização da imagem ou vídeo
@property (readonly, nonatomic, strong) NSArray *filepaths
mediapath
A representação *string* da pasta do dispositivo onde a *library media* está armazenada
@property (readonly, nonatomic, strong) NSURL *mediaPath
Detalhes:
Qualquer *media* guardada nesta pasta deve aparecer na *library UI* (interface com o utilizador). Para adicionar *media* a esta pasta, deve ser usado o método

finishEditingAndSaveCurrentImageToFilepath: withPreviewImage: do FLIROneSDKImageEditor.

- **Métodos da classe**

sharedInstance

Acessa, exclusivamente, e retorna uma referência ao *singleton* global (inicializa o *singleton* se este ainda não tiver sido inicializado)

+ (instancetype)sharedInstance

Retorna:

Uma referência ao *singleton* global

Detalhes:

O FLIROneSDKLibraryManager é acedido, exclusivamente, através de uma referência ao *singleton*. Todos os métodos da instância serão chamados neste objeto *singleton*.

- **Métodos *Instance***

copyImageToLibraryWithFilepath:withThumbnailFilepath

Copia uma imagem ou vídeo para a biblioteca

- (BOOL)copyImageToLibraryWithFilepath:(NSURL *)filepath withThumbnailFilepath:(NSURL *)thumbnailFilepath

Parâmetros:

filepath - O caminho do ficheiro (*filepath)* do ficheiro para copiar para o diretório da biblioteca

thumbnailFilepath - O caminho de miniatura (*thumbnail filepath*) para a miniatura (*thumbnail*) da imagem

Retorna:

YES se o ficheiro foi copiado com sucesso, caso contrário NO

Detalhes:

AVISO: Se existir um ficheiro com o mesmo nome de ficheiro na pasta da biblioteca, a operação de cópia vai falhar

deleteLibraryFileWithFilepath

Apaga uma imagem ou video da biblioteca

- (BOOL)deleteLibraryFileWithFilepath:(NSURL *)filepath

Parâmetros:

filepath - O caminho do ficheiro (*filepath*) do ficheiro a ser eliminado

Retorna:

YES se o ficheiro foi encontrado e eliminado, caso contrário NO

fileIsVideo

Caminho do ficheiro (*filepath*) do ficheiro a verificar

- (BOOL)fileIsVideo:(NSURL *)filepath

Parâmetros:

filepath - Caminho do ficheiro (*filepath*) do ficheiro a verificar

Retorna:

YES se o ficheiro com o nome especificado existir e for um vídeo, caso contrário NO

libraryFilepathForCurrentTimestampWithExtension

Retorna um novo caminho de ficheiro (*filepath*) na pasta *media* (*media folder*) com uma determinada extensão com base na data/hora atual
- (NSURL *)libraryFilepathForCurrentTimestampWithExtension:(NSString *)extension
Parâmetros:
extension – O tipo de extensão a usar do ficheiro. Exemplo: .png
Retorna:
Um caminho de ficheiro (*filepath*) para um ficheiro na pasta de *media* (*media folder*) com a extensão fornecida
previewImageForVideoWithFilepath
Recupera uma imagem de *preview* para um vídeo gravado anteriormente
- (UIImage *)previewImageForVideoWithFilepath:(NSURL *)filepath
Parâmetros:
filepath - O caminho de ficheiro (*filepath*) de vídeo para recuperar a *preview*
Retorna:
Uma representação UIImage da imagem de visualização (*preview*) do vídeo, ou nulo se o vídeo não existe
thumbnailForFilepath
Recupera um *thumbnail* de um vídeo ou de uma foto anteriormente capturados
- (UIImage *)thumbnailForFilepath:(NSURL *)filepath
Parâmetros:
filepath - O caminho do ficheiro (*filepath*) para recuperar a miniatura (*thumbnail*)
Retorna:
Uma representação UIImage do ficheiro *thumbnail*, ou nulo se o ficheiro não existe

- **FLIROneSDKLibraryViewController Class Reference**

Herda:

- UICollectionViewController

Conforme:

- UIAlertViewDelegate
- UICollectionViewDataSource
- UICollectionViewDelegate

Declarado em:

- FLIROneSDKLibraryViewController.h

- **Descrição**

Permite controlar a visualização de imagens e vídeos na biblioteca, possibilitando a sua edição, eliminação e visualização.

Funcionalidade é implementada usando as classes FLIROneSDKLibraryManager, FLIROneSDK, FLIROneSDKShareActivity.

É possível usar uma função criada pelo desenvolvedor para mostrar a biblioteca aos utilizadores.

AVISO: Todas as interações com essa classe devem ocorrer na *thread* principal.

- **Tarefas**

+ presentLibraryFromViewController:
shareButton (propriedade)
deleteButton (propriedade)
backButton (propriedade)
enableMultipleSelectionModeButton (propriedade)
exitMultipleSelectionModeButton (propriedade)
bottomBar (propriedade)
toolbarHeight (propriedade)
libraryEmptyView (propriedade)

- **Propriedades**

backButton
Botão usado para voltar atrás (*backwards*)
@property (nonatomic) UIBarButtonItem *backButton
bottomBar
Barra de ferramentas na parte inferior da exibição que contém os botões de partilhar, eliminar e cancelar
@property (nonatomic) UIToolbar *bottomBar
deleteButton
Botão usado para eliminar um ou mais itens da biblioteca
@property (nonatomic) UIBarButtonItem *deleteButton
enableMultipleSelectionModeButton
Botão usado para mudar o modo *multiple item selection*
@property (nonatomic) UIBarButtonItem
exitMultipleSelectionModeButton
Botão usado para passar do modo de *multiple item selection* para o modo de *single item display*
@property (nonatomic) UIBarButtonItem *exitMultipleSelectionModeButton
libraryEmptyView
Vista (*view*) que contém a informação “não há fotos nem vídeos”
@property (nonatomic) UIView *libraryEmptyView
shareButton
Botão usado para apresentar um controlador de visualização FLIROneSDKShareActivity
@property (nonatomic) UIBarButtonItem *shareButton
toolbarHeight
Altura da barra de baixo (*bottomBar*)

@property (nonatomic) float toolbarHeight

- **Métodos da classe**

presentLibraryFromViewController:
Apresenta um controlador de visualização modal da biblioteca (*library view controller modally*). A vista (*view*) vai ser ignorada quando o utilizador carrega no botão voltar (*back*)
+ (void)presentLibraryFromViewController:(UIViewController *)viewController
Parâmetros:
viewController - O controlador pai (*parent view controller*) a usar para apresentar o FLIROneSDKLibraryManager

- **FLIROneSDKPalette Class Reference**

Herda:

- NSObject

Declarado em:

- FLIROneSDKPalette.h

- **Descrição**

Usado para especificar o esquema de cores utilizado para renderizar (*rendering*) o componente térmico de imagens MSX ou imagens térmicas.

Obtém a lista de paletas a partir da SDK.

As paletas acedidas a partir desta classe são usadas para definir a paleta atual para o FLIROneSDKStreamManager antes da captura de imagem ou para editar a paleta de uma imagem após a sua captura através do método FLIROneSDKImageEditor setPalette. Por exemplo:

```
//set the current palette to the first palette in the list of palettes
[FLIROneSDKStreamManager sharedInstance].palette = [[FLIROneSDKPalette palettes] firstObject];
```

AVISO: Não tentar criar novas instâncias desta classe. Em vez disso, selecionar uma instância usando a propriedade palettes.

- **Tarefas**

Aceder objetos da paleta	+ palettes
Propridadades da paleta	name (propriedade)

- **Propriedades**

name
Pode ser o nome que aparece no ecrã ou pode ser o identificador único das paletas. Pode-se usar este valor para recuperar uma paleta específica, por exemplo através do método palletes class
@property (readonly, strong, nonatomic) NSString *name

- **Métodos da classe**

palletes
Um dicionário que mapeia nomes de sequências de caracteres para paletas correspondentes
+ (NSDictionary *)palettes

- **FLIROneSDKShareActivity Class Reference**

Herda:
- UIActivity

Declarado em:
- FLIROneSDKShareActivity.h

- **Descrição**

Permite ao utilizador partilhar vídeos e fotos, com serviços de partilha padrão do *iOS*, além da possibilidade de fazer o *upload* diretamente para o youtube, no caso de um ficheiro de vídeo.

Todas estas funcionalidades podem ser utilizadas usando a classe FLIROneSDKLibraryManager, mas o desenvolvedor pode implementar o seu próprio FLIROneSDKShareActivity personalizado.

AVISO: Todas as interações com esta classe devem ocorrer na *thread* principal

- **Tarefas**

+ presentActivityViewControllerWithFilepaths:withViewController:
+setClientID:andClientKey:
+clientID
+clientKey

- **Métodos da classe**

clientID
Usado com a interface de *upload* do youtube. Retorna o ID do cliente ou nada se nenhum cliente tiver sido definido
+ (NSString *)clientID
clientKey
Usado com a interface de *upload* do youtube. Retorna a chave do cliente ou nada se nenhum cliente tiver sido definido
+ (NSString *)clientKey
presentActivityViewControllerWithFilepaths:withViewController:
Apresenta o controlador de visualização de atividade que permite partilhar os ficheiros especificados na biblioteca
+ (void)presentActivityViewControllerWithFilepaths:(NSArray *)filepaths withViewController:(UIViewController *)viewController
Parâmetros:
filepath - Uma matriz de instâncias de caminhos de ficheiros NSURL (*filepath NSURL instances*) que correspondem a ficheiros na biblioteca da FLIR ONE
viewController - O controlador de visualização (*view* controller) com o qual se apresenta o FLIROneSDKShareActivity

- **FLIROneSDKStreamManager Class Reference**

Herda:

- FLIROneSDKDelegateManager : NSObject

Conforme:

- FLIROneSDKImagePropertyController

Declarado em:

- FLIROneSDKStreamManager.h

- **Descrição**

Usado para controlar a captura de dados de um fluxo de imagem ao vivo. FLIROneSDKDelegate é disparado em resposta aos métodos usados nesta classe.

O sharedInstance desta classe é responsável por definir as propriedades de entrada, como a paleta de cores atual ou a emissividade e para capturar imagens ou vídeos.

Todos os métodos devem ser chamados no *singleton* retornado pelo método FLIROneSDKStreamManager sharedInstance.

É de salientar que esta classe é *thread-safe*, ou seja, pode-se chamar qualquer método de qualquer *thread* sem problema.

AVISO: Não tentar criar outra instância desta classe. Usar sempre o método de classe sharedInstance para recuperar o *singleton* global.

- **Tarefas**

Aceder à instância FLIROneSDKStreamManager sharedInstance	+ sharedInstance
	– addDelegate:
	– removeDelegate:
Capturar uma imagem ou um vídeo	– startRecordingVideoWithFilepath:withVideoRendererDelegate:
	– stopRecordingVideo
	– capturePhotoWithFilepath:
	– performTuning
	– setAutomaticTuning:
	– isDongle
	msxDistanceEnabled (propriedade)
	msxDistance (propriedade)

- **Propriedades**

msxDistance
CGFloat entre 0 e 1, especificando a distância MSX, sendo 0 o mais próximo e 1 o mais afastado
@property (readwrite, nonatomic) CGFloat msxDistance
msxDistanceEnabled
Sinalizador *(flag*) que indica se a distância MSX, que controla o alinhamento dos elementos visuais e térmicos de uma imagem MSX, está ou não habilitada
@property (readwrite, nonatomic) BOOL msxDistanceEnabled

- **Métodos da classe**

sharedInstance
Acessa, exclusivamente, e retorna uma referência ao *singleton* global (inicializa o *singleton* se este ainda não tiver sido inicializado)
+ (instancetype)sharedInstance
Retorna:
Uma referência ao *singleton* global
Detalhes:
O FLIROneSDKStreamManager é acedido, exclusivamente, através de uma referência ao *singleton*. Todos os métodos de instância serão chamados neste objeto *singleton*. Este objeto *singleton* FLIROneSDK mantém um registo dos objetos registados como delegados do FLIROneSDK. Este objeto é responsável por adicionar e remover delegados. O registo de um FLIROneSDKDelegate é necessário para receber todas e quaisquer informações do dispositivo

FLIR ONE, incluindo o estado da conexão e as *frames* de receção
AVISO: O Stream Manager sharedInstance só deve ser acedido depois de um FLIROneSDKDelegate ter sido adicionado

- **Métodos *Instance***

addDelegate
Registo de um novo delegado (sem efeito se o delegado já tiver sido adicionado anteriormente)
- (void)addDelegate:(id<FLIROneSDKStreamManagerDelegate,FLIROneSDKImageReceiverDelegate>)delegate
Parâmetros:
delegate - O objeto para registar eventos de retorno de chamada da FLIR ONE
Detalhes:
Como um exemplo para registar o controlador de exibição atual como um FLIROneSDKDelegate: [[FLIROneSDK sharedInstance] addDelegate:self];
capturePhotoWithFilepath
Instrui o SDK a guardar a *frame* atual na biblioteca. Todas as chamadas para este método resultarão no seu retorno. É acionado para todos os delegados registados
- (void)capturePhotoWithFilepath:(NSURL *)filepath
Parâmetros:
filepath - O caminho do ficheiro onde a imagem capturada deve ser guardada
Detalhes:
NOTA: Se o *filepath* for um ficheiro na pasta de *media* do FLIROneSDKLibraryManager, a imagem será adicionada à biblioteca e aparecerá nos elementos da interface da biblioteca
AVISO: Deve-se aguardar o FLIROneSDKDidFinishCapturingPhoto: withFilepath: callback para disparar antes de chamar este método novamente, ou a solicitação falhará imediatamente.
AVISO: Não tentar capturar fotos enquanto o *sled* está num estado de sintonia (*tuning*). Aguardar até que a sintonia tenha sido concluída antes de capturar a *media* para a biblioteca
NOTA: Este método irá guardar uma *thumbnail* para o local fornecido, acrescentando ".thumb" para o caminho do ficheiro (*filepath*)
isDongle
Determina se o dispositivo conectado é um *dongle* ou *sled*
- (BOOL)isDongle
performTuning
Se a câmara tiver obturador automático, comanda a câmara para ajustar a imagem térmica usando o obturador automático
- (void)performTuning

removeDelegate
Anula a subscrição de um objeto previamente registado com addDelegate. Sem efeito se o delegado não tiver sido registado anteriormente
- (void)removeDelegate:(id<FLIROneSDKStreamManagerDelegate,FLIROneSDKImageReceiverDelegate>)delegate
Parâmetros:
delegate - O objeto para cancelar a inscrição dos eventos de retorno de chamada do dispositivo FLIR ONE
setAutomaticTuning
Para câmaras com obturador automático, esta será sintonizada automaticamente quando necessário
- (void)setAutomaticTuning:(BOOL)shouldTuneAutomatically
Parâmetros:
shouldTuneAutomatically – O estado desejado de sintonização automática (*automatic tuning*)
startRecordingVideoWithFilepath:withVideoRendererDelegate:
Começa a gravar o vídeo. Todas as chamadas resultam em retorno da chamada para FLIROneSDKDidStartRecordingVideo a ser disparado para todos os delegados registados
- (void)startRecordingVideoWithFilepath:(NSURL *)filepath withVideoRendererDelegate:(id<FLIROneSDKVideoRendererDelegate>)videoRendererDelegate
Parâmetros:
filepath - O caminho do ficheiro onde o vídeo deve ser guardado
videoRendererDelegate - Um delegado de vídeogravador opcional para gravação no formato de imagem à escolha, ou para fazer modificações em cada *frame*
Detalhes:
NOTA: Se o caminho do ficheiro (*filepath*) for um ficheiro na pasta de *media* do FLIROneSDKLibraryManager, o vídeo será adicionado à biblioteca e ficará visível na interface do utilizador da biblioteca
NOTA: Chamar este método se a gravação já começou não afetará a gravação atual, mas ainda resultará no disparo de chamada de delegado
NOTA: Este método irá guardar uma *thumbnail* para o local fornecido, acrescentando ".thumb" para o caminho do ficheiro (*filepath*)
NOTA: Este método irá guardar uma visualização em tamanho real (*full-size preview*) do vídeo para o local fornecido, acrescentando ".preview" ao caminho do ficheiro (*filepath*)
stopRecordingVideo
Termina uma gravação. Cada chamada a este método resulta no retorno da chamada FLIROneSDKDidStopRecording que é acionado para todos os delegados registados. Se estivesse a ocorrer uma gravação na altura, é expectável que FLIROneSDKDidFinishWritingVideo seja acionado posteriormente
- (void)stopRecordingVideo
Detalhes:
NOTA: Chamar este método após a gravação já ter parado não afetará a gravação atual, mas ainda resultará no disparo de retorno de chamada de delegado

- **FLIROneSDKTuningViewController Class Reference**

Herda:
- UIViewController

Conforme:
- FLIROneSDKImageReceiverDelegate
- FLIROneSDKStreamManagerDelegate

Declarado em:
- FLIROneSDKTuningViewController.h

- **Descrição**

Um controlador de visualização que pode ser usado para instruir o utilizador a puxar o obturador para ajustar o dispositivo FLIR ONE.

Não é obrigatório usar esta classe, o desenvolvedor pode criar o seu próprio FLIROneSDKDelegate para atualizar a interface do utilizador.

Para utilizar com êxito este controlador de exibição de ajuste (*tuning view controller*), é necessário alcançar os seguintes pontos:

- Ainda será necessário implementar uma lógica própria para lidar com a conexão e desconexão do dispositivo FLIR ONE. Esta vista de sintonização (*tuning view*) não pede ao utilizador para ligar o *sled* e irá despedir-se (*dismiss itself*) se o dispositivo desligar.
- É, ainda, necessário implementar uma lógica própria para saber como responder aos eventos de ajuste (*tuning events*) do método [FLIROneSDKDelegate FLIROneSDKTuningStateDidChange:]
- É necessário rastrear se este controlador de exibição de ajuste (*tuning view controller*) já foi apresentado. Se se tentar apresentar o FLIROneSDKTuningViewController, quando ele já é apresentado, ele retornará NO para indicar que ocorreu um erro.

Quando o controlador de exibição de ajuste (*tuning view controller*) é apresentado, ele torna-se o controlador de exibição ativo e inscreve-se como um FLIROneSDKDelegate que assina eventos de sintonia (*tuning events*) que ocorrem no método [FLIROneSDKDelegate FLIROneSDKTuningStateDidChange:]. Ele tratará estes eventos e descartar-se-á quando o processo de ajuste (*tuning*) for concluído.
Para aplicações personalizadas, é provável que se queira implementar uma UI própria de ajuste (*tuning*) para ter a aparência desejada para a aplicação.

AVISO: Todas as interações com esta classe devem ocorrer na *thread* principal.

- **Tarefas**

+ presentWithViewController:withTuningState:

- **Propriedades**

presentWithViewController:withTuningState:
Apresenta um controlador modal de visualização que irá guiar o utilizador no processo de sintonização (*tuning*)
+ (BOOL)presentWithViewController:(UIViewController *)viewController withTuningState:(FLIROneSDKTuningState)tuningState
Parâmetros:
viewController - O controlador de exibição pai a usar para apresentar o FLIROneSDKTuningViewController
tuningState – O estado de sintonia (*tuning state*) para fazer o *display* inicialmente
Retorna:
Valor booleano que representa o sucesso ou falha da ação de apresentação. YES significa que o controlador de visualização foi apresentado com êxito.
Detalhes:
Irá registar-se como um FLIROneSDKDelegate e ignora-se automaticamente quando o estado de sintonização (*tuning*) é alterado para FLIROneSDKTuningStateTunedWithOpenedShutter ou o dispositivo FLIR One se desliga
AVISO: Deve-se chamar este método dentro do FLIROneSDKTuningStateDidChange callback, quando o estado de ajuste (*tuning state*) muda para FLIROneSDKTuningStateTuningRequired, FLIROneSDKTuningStateInProgress ou FLIROneSDKTuningStateTunedWithClosedShutter
AVISO: Não chamar este método a não ser que o *sled* esteja conectado ao dispositivo

- **FLIROneSDKUIImage Class Reference**

Herda:

- UIImage

Declarado em:

- FLIROneSDKUIImage.h

- **Descrição**

É uma classe UIImage que permite a inicialização usando dados retornados pela chamada do FLIROneSDKStreamManagerDelegate ou do FLIROneSDKImageEditorDelegate.

- **Tarefas**

data (propriedade)
+ imageWithFormat:andData:andSize
- initWithFormat:andData:andSize

- **Propriedades**

data
Dados originais utilizados para gerar a UIImage
@property (strong, nonatomic, readonly) NSData *data

- **Métodos da classe**

imageWithFormat:andData:andSize:
Criar uma imagem a partir de dados formatados, usando o tamanho se necessário
+ (instancetype)imageWithFormat:(FLIROneSDKImageOptions)format andData:(NSData *)data andSize:(CGSize)size
Parâmetros:
format – O formato da imagem que corresponde aos dados
data – Os dados da imagem usados para criar a imagem
size – O tamanho da imagem. Para o formato JPEG, este parâmetro é ignorado, sendo CGSizeZero sugerido
Retorna:
O FLIROneSDKUIImage, ou nulo se não puder ser criado

- **Métodos *Instance***

initWithFormat:andData:andSize:
Inicializar uma imagem a partir de dados formatados, usando o tamanho se necessário
- (instancetype)initWithFormat:(FLIROneSDKImageOptions)format andData:(NSData *)data andSize:(CGSize)size
Parâmetros:
format – O formato da imagem que corresponde aos dados
data – Os dados da imagem usados para criar a imagem
size – O tamanho da imagem, se necessário.

Constantes

- **FLIROneSDKBatteryChargingState Constants Reference**

Declarado em:

- FLIROneSDKTypes.h

- **Descrição**

Enumera todos os estados possíveis do carregamento da bateria da câmara térmica.

- **Definição**

```
typedef NS_ENUM(NSInteger, FLIROneSDKBatteryChargingState ) {
        FLIROneSDKBatteryChargingStateDischarging,
        FLIROneSDKBatteryChargingStateCharging,
        FLIROneSDKBatteryChargingStateError,
        FLIROneSDKBatteryChargingStateInvalid, };
```

- **Constantes**

FLIROneSDKBatteryChargingStateDischarging	A bateria está a descarregar
FLIROneSDKBatteryChargingStateCharging	A bateria está a carregar a partir de uma fonte externa
FLIROneSDKBatteryChargingStateError	O estado de carregamento da bateria não pode ser lido devido a falhas ou algum tipo de erro
FLIROneSDKBatteryChargingStateInvalid	Estado de carregamento da bateria inválido

- **FLIROneSDKCaptureStatus Constants Reference**

Declarado em:

- FLIROneSDKTypes.h

- **Descrição**

O resultado de tentar capturar uma imagem ou vídeo.

- **Definição**

```
typedef NS_ENUM(NSUInteger, FLIROneSDKCaptureStatus ) {
        FLIROneSDKCaptureStatusSucceeded,
        FLIROneSDKCaptureStatusFailedWithUnknownError,
};
```

- **Constantes**

FLIROneSDKCaptureStatusSucceeded	A captura foi bem sucedida
FLIROneSDKCaptureStatusFailedWithUnknownError	A captura falhou por uma razão desconhecida

- **FLIROneSDKImageIOStatus Constants Reference**

Declarado em:

- FLIROneSDKTypes.h

- **Descrição**

O resultado de tentar uma operação IO com a FLIROneSDKImage.

- **Definição**

```
typedef NS_ENUM(NSUInteger, FLIROneSDKImageIOStatus ) {
        FLIROneSDKImageIOStatusFailedWithUnknownError,
        FLIROneSDKImageIOStatusFailedWithNoImageLoaded,
        FLIROneSDKImageIOStatusFailedWithInvalidURL,
        FLIROneSDKImageIOStatusFailedWithFileNotFound,
        FLIROneSDKImageIOStatusFailedWithFileFormatNotSupported,
        FLIROneSDKImageIOStatusSucceeded, };
```

- **Constantes**

FLIROneSDKImageIOStatusFailedWithUnknownError	A operação IO falhou por uma razão desconhecida
FLIROneSDKImageIOStatusFailedWithNoImageLoaded	O editor de imagem não carregou um ficheiro para gravar
FLIROneSDKImageIOStatusFailedWithInvalidURL	O URL fornecido estava ausente ou era inválido
FLIROneSDKImageIOStatusFailedWithFileNotFound	O ficheiro era de um tipo de ficheiros não suportado
FLIROneSDKImageIOStatusSucceeded	A operação IO foi completada com sucesso

- **FLIROneSDKImageOptions Constants Reference**

Declarado em:

- FLIROneSDKTypes.h

- **Descrição**

As opções de imagem permitem que o utilizador especifique quais os dados que são usados/retornados ao transmitir ou carregar imagens do disco.

- **Definição**

```
typedef NS_ENUM(uint64_t, FLIROneSDKImageOptions ) {
        FLIROneSDKImageOptionsThermalLinearFlux14BitImage = 0 x1,
        FLIROneSDKImageOptionsThermalRGBA8888Image = 0 x2,
        FLIROneSDKImageOptionsBlendedMSXRGBA8888Image = 0 x4,
        FLIROneSDKImageOptionsVisualJPEGImage = 0 x8,
        FLIROneSDKImageOptionsVisualYCbCr888Image = 0 x10,
FLIROneSDKImageOptionsThermalRadiometricKelvinImage = 0 x20, };
```

- **Constantes**

FLIROneSDKImageOptionsThermalLinearFlux14BitImage	Linear 14 bit image data
FLIROneSDKImageOptionsThermalRGBA8888Image	Thermal RGBA image data
FLIROneSDKImageOptionsBlendedMSXRGBA8888Image	MSX (thermal + visual) RGBA image data
FLIROneSDKImageOptionsVisualJPEGImage	Visual JPEG image data
FLIROneSDKImageOptionsVisualYCbCr888Image	Visual YCbCr image data
FLIROneSDKImageOptionsThermalRadiometricKelvinImage	Radiometric kelvin temperature data

- **FLIROneSDKTuningState Constants Reference**

Declarado em:

- FLIROneSDKTypes.h

- **Descrição**

O estado de sintonização (*tuning*) do dispositivo FLIR ONE.

O estado é atualizado quando o *sled* determina que os seus pontos de temperatura se estão a tornar imprecisos ou se o utilizador iniciar a sua afinação puxando a alavanca do obturador para baixo.

- **Definição**

```
typedef NS_ENUM(NSUInteger, FLIROneSDKTuningState ) {
        FLIROneSDKTuningStateTuningRequired,
        FLIROneSDKTuningStateInProgress,
        FLIROneSDKTuningStateTunedWithClosedShutter,
        FLIROneSDKTuningStateTunedWithOpenedShutter,
        FLIROneSDKTuningStateTuningSuggested,
        FLIROneSDKTuningStateApproximatelyTunedWithOpenedShutter,
        FLIROneSDKTuningStateUnknown,
};
```

- **Constantes**

FLIROneSDKTuningStateTuningRequired	Requer sintonização. Neste estado a afinação é necessária para que as *frames* tenham dados de temperatura radiométrica (se a aplicação requer dados de temperatura, o utilizador é forçado a ajustar o dispositivo)
FLIROneSDKTuningStateInProgress	O dispositivo FLIR ONE está a ser ajustado, mas ainda não está totalmente ajustado. A UI de ajuste deve ser atualizada pedindo ao utilizador para continuar o identificador do obturador
FLIROneSDKTuningStateTunedWithClosedShutter	O obturador foi libertado para a posição aberta (*open*) e a interface de sintonia (*tuning*) deve ser descartada
FLIROneSDKTuningStateTuningSuggested	A câmara pede, mas não exige que o obturador vá para a posição "fechada" para reajustar o dispositivo FLIR ONE. Se a aplicação requer dados de temperatura de alta qualidade deve-se forçar o utilizador a ajustar o dispositivo
FLIROneSDKTuningStateApproximatelyTunedWithOpenedShutter	A câmara iniciou recentemente e os valores de temperatura medidos podem ser ligeiramente imprecisos
FLIROneSDKTuningStateUnknown	Um estado de sintonia inválido. Não receber este estado do dispositivo FLIR ONE

Protocolos

- **FLIROneSDKImageEditorDelegate Protocol Reference**

Conforme:
- FLIROneSDKImageReceiverDelegate
- NSObject

Declarado em:
- FLIROneSDKImageEditorDelegate.h

- **Descrição**

O editor de imagens usa este protocolo para informar os delegados do sucesso ou falha no carregamento e na fase de guardar as imagens.

- **Tarefas**

FLIROneSDKImageEditor Events	- FLIROneSDKEditorImageDidFinishLoading
	- FLIROneSDKEditorImageDidFinishSaving:withFilepath

- **Métodos *Instance***

FLIROneSDKEditorImageDidFinishLoading:
Acionado depois de uma imagem ser carregada no editor através de uma solicitação FLIROneSDKImageEditor
- (void)FLIROneSDKEditorImageDidFinishLoading:(FLIROneSDKImageIOStatus)loadedStatus
Parâmetros:
loadedStatus - Se a imagem foi ou não carregada com êxito. Se a imagem foi carregada com êxito, o retorno de chamada FLIROneSDKEditorImageDidChange é disparado imediatamente depois, com a imagem inicial
FLIROneSDKEditorImageDidFinishSaving:withFilepath:
Acionado após solicitação de FLIROneSDKImageEditor finishEditingAndSaveCurrentImageToFilepath:withPreviewImage:. Este evento vai ser sempre acionado depois de chamar finishEditingAndSaveCurrentImageToFilepath:withPreviewImage:.
- (void)FLIROneSDKEditorImageDidFinishSaving:(FLIROneSDKImageIOStatus)savedStatus withFilepath:(NSURL *)filepath
Parâmetros:
savedStatus – Se a imagem foi ou não guardada com sucesso
filepath – O caminho do ficheiro (*filepath*) da imagem guardada

- **FLIROneSDKImagePropertyController Protocol Reference**

Conforme:
- NSObject

Declarado em:

- FLIROneSDKImagePropertyController.h

- **Descrição**

Neste protocolo o Image Editor e o StreamManager estão em conformidade.

Fornece uma interface uniforme para definir a paleta, a emissividade e o formato dos dados desejados.

- **Tarefas**

Propriedades da imagem	palette (propriedade, método requerido)
	emissivity (propriedade, método requerido)
	imageOptions (propriedade, método requerido)

- **Propriedades**

emissivity
A emissividade atual utilizada para a componente térmica do fluxo (*stream*). O seu valor pode ser lido a partir de qualquer *thread* com segurança
@property (readwrite, nonatomic) CGFloat emissivity
Detalhes:
Para atualizar a emissividade atual com que as *frames* recebidas serão coloridas, deve-se definir o valor desta propriedade como um *double*, opcionalmente usando um dos quatro valores predefinidos. Por exemplo: [[FLIROneSDKStreamManager sharedInstance] setEmissivity:FLIROneSDKEmissivityGlossy];
NOTA: Pode-se alterar esta propriedade enquanto se grava um vídeo
imageOptions
As opções de imagem ditam que formatos são entregues a partir do *sled* durante o *streaming* ou edição de imagem
@property (readwrite, nonatomic) FLIROneSDKImageOptions imageOptions
Detalhes:
Especificar as opções de imagem no Stream Manager afeta os retornos de chamada que serão recebidos na próxima *frame*.
palette
O FLIROneSDKPalette atual é usado para colorir o componente térmico do fluxo. Pode-se ler e escrever o valor desta propriedade de qualquer *thread* com segurança
@property (readwrite, nonatomic) FLIROneSDKPalette *palette
Detalhes:
Para atualizar a paleta atual com que as *frames* recebidas serão coloridas deve definir-se o

valor desta propriedade para um FLIROneSDKPalette. Isso deve ser feito com o nome de uma paleta da lista de paleta. Por exemplo:

```
FLIROneSDKPalette *palette = [[FLIROneSDKPalette palettes] objectAtIndex:self.paletteCount];
[[FLIROneSDKStreamManager sharedInstance] setCurrentPalette:palette];
```

NOTA: Pode-se alterar esta propriedade enquanto se grava um vídeo

- **FLIROneSDKImageReceiverDelegate Protocol Reference**

Conforme:

- NSObject

Declarado em:

- FLIROneSDKImageReceiverDelegate.h

- **Descrição**

Permite que o utilizador receba *frames* em direto nos formatos de imagem que escolha, podendo também carregar imagens recebidas geradas a partir de um ficheiro no mesmo formato.

Ao receber dados num dado formato, ele pode ser convertido num UIImage utilizando o FLIROneSDKUIImage class.

- **Tarefas**

FLIROneSDKDelegateManager:didReceiveFrameWithOptions:metadata:
FLIROneSDKDelegateManager:didReceiveBlendedMSXRGBA8888Image:imageSize:
FLIROneSDKDelegateManager:didReceiveThermal14BitLinearFluxImage:imageSize:
FLIROneSDKDelegateManager:didReceiveThermalRGBA8888Image:imageSize:
FLIROneSDKDelegateManager:didReceiveVisualJPEGImage:
FLIROneSDKDelegateManager:didReceiveVisualYCbCr888Image:imageSize:
FLIROneSDKDelegateManager:didReceiveRadiometricData:imageSize:

- **Métodos *Instance***

FLIROneSDKDelegateManager:didReceiveBlendedMSXRGBA8888Image:imageSize:
Chamado quando o utilizador solicitou o formato de imagem FSX (térmica+visual)
- (void)FLIROneSDKDelegateManager:(FLIROneSDKDelegateManager *)delegateManager didReceiveBlendedMSXRGBA8888Image:(NSData *)msxImage imageSize:(CGSize)size
Parâmetros:
deleteManager - O gestor de delegados, FLIROneSDKStreamManager ou FLIROneSDKImageEditor
msxImage - Os dados formatados no formato FLIROneSDKThermalRGBA8888
size – O tamanho dos dados, usado para renderizar (*rendering*)
AVISO: O parâmetro de tamanho é garantido para ser estático por sessão, mas alterar dispositivos da FLIR One pode alterar o parâmetro de tamanho

FLIROneSDKDelegateManager:didReceiveFrameWithOptions:metadata:
Chamado sempre que uma *frame* é entregue ao delegado pelo gestor de fluxo (*stream manager*) ou sempre que uma imagem é carregada pelo editor de imagens
- (void)FLIROneSDKDelegateManager:(FLIROneSDKDelegateManager *)delegateManager didReceiveFrameWithOptions:(FLIROneSDKImageOptions)options metadata:(FLIROneSDKImageMetadata *)metadata
Parâmetros:
delegateManager – O gestor de delegados, FLIROneSDKStreamManager ou FLIROneSDKImageEditor
options – Uma máscara de bits (*bitmask*) de opções relacionadas com os dados que estão sendo retornados
metadata - Metadados referentes à *frame* particular que está a ser entregue, como, por exemplo, a paleta usada, a emissividade, etc
FLIROneSDKDelegateManager:didReceiveRadiometricData:imageSize:
Retornado quando o utilizador solicitou dados radiométricos, seja por imagem carregada, seja por *frame* entregue pelo gestor de fluxo (*stream manager*)
- (void)FLIROneSDKDelegateManager:(FLIROneSDKDelegateManager *)delegateManager didReceiveRadiometricData:(NSData *)radiometricData imageSize:(CGSize)size
Parâmetros:
delegateManager – O gestor de delegados, FLIROneSDKStreamManager ou FLIROneSDKImageEditor
radiometricData - Uma matriz de dados radiométricos onde cada valor é igual à temperatura em kelvin num local na *frame*
size – O tamanho da matriz de dados
Detalhes:
AVISO: O parâmetro de tamanho é garantido para ser estático por sessão, mas alterar dispositivos da FLIR Um pode alterar o parâmetro de tamanho
FLIROneSDKDelegateManager:didReceiveThermal14BitLinearFluxImage:imageSize:
Chamado quando o utilizador solicitou os dados de fluxo linear de 14 bits
- (void)FLIROneSDKDelegateManager:(FLIROneSDKDelegateManager *)delegateManager didReceiveThermal14BitLinearFluxImage:(NSData *)linearFluxImage imageSize:(CGSize)size
Parâmetros:
delegateManager – O gestor de delegados, FLIROneSDKStreamManager ou FLIROneSDKImageEditor
linearFluxImage – Os dados da imagem
size – O tamanho dos dados, usado para renderizar (*rendering*)
AVISO: O parâmetro de tamanho é garantido para ser estático por sessão, mas alterar dispositivos da FLIR Um pode alterar o parâmetro de tamanho
FLIROneSDKDelegateManager:didReceiveThermalRGBA8888Image:imageSize:
Chamado quando o utilizador solicitou o formato de imagem térmica pura

- (void)FLIROneSDKDelegateManager:(FLIROneSDKDelegateManager *)delegateManager didReceiveThermalRGBA8888Image:(NSData *)thermalImage imageSize:(CGSize)size
Parâmetros:
delegateManager – O gestor de delegados, FLIROneSDKStreamManager ou FLIROneSDKImageEditor
thermalImage – Os dados da imagem
size – O tamanho dos dados, usado para renderizar (*rendering*)
AVISO: O parâmetro de tamanho é garantido para ser estático por sessão, mas alterar dispositivos da FLIR Um pode alterar o parâmetro de tamanho
FLIROneSDKDelegateManager:didReceiveVisualJPEGImage:
Chamado quando o utilizador solicitou a imagem visual formatada com JPEG
- (void)FLIROneSDKDelegateManager:(FLIROneSDKDelegateManager *)delegateManager didReceiveVisualJPEGImage:(NSData *)visualJPEGImage
Parâmetros:
delegateManager – O gestor de delegados, FLIROneSDKStreamManager ou FLIROneSDKImageEditor
visualJPEGImage – Os dados da imagem
FLIROneSDKDelegateManager:didReceiveVisualYCbCr888Image:imageSize:
Chamado quando o utilizador solicitou a imagem visual formatada com YCbCr
- (void)FLIROneSDKDelegateManager:(FLIROneSDKDelegateManager *)delegateManager didReceiveVisualYCbCr888Image:(NSData *)visualYCbCr888Image imageSize:(CGSize)size
Parâmetros:
delegateManager – O gestor de delegados, FLIROneSDKStreamManager ou FLIROneSDKImageEditor
visualYCbCr888Image – Os dados da imagem
size – O tamanho dos dados, usado para renderizar (*rendering*)
AVISO: O parâmetro de tamanho é garantido para ser estático por sessão, mas alterar dispositivos FLIR ONE pode alterar o parâmetro de tamanho

- **FLIROneSDKStreamManagerDelegate Protocol Reference**

Conforme:

- FLIROneSDKImageReceiverDelegate
- NSObject

Declarado em:

- FLIROneSDKStreamManagerDelegate.h

- **Descrição**

Fornece aos utilizadores a capacidade de detetar eventos de conexão, informações de ajuste e estado da bateria, bem como capturar videos e imagens térmicas em disco.

- **Tarefas**

Responder à conexão e desconexâo do dispositivo FLIR ONE	- FLIROneSDKDidConnect
	- FLIROneSDKDidDisconnect
Responder a alterações na bateria ou estado de carga	- FLIROneSDKBatteryChargingStateDidChange:
	- FLIROneSDKBatteryPercentageDidChange:
Responder a alterações no estado de sintonia	- FLIROneSDKTuningStateDidChange:
	- FLIROneSDKAutomaticTuningDidChange:
Responder a FLIROneSDKStreamManager, capturando fotos ou vídeos	- FLIROneSDKDidFinishCapturingPhoto:withFilepath:
	- FLIROneSDKDidStartRecordingVideo:
	- FLIROneSDKDidStopRecordingVideo:
	- FLIROneSDKDidFinishWritingVideo:withFilepath:

- **Métodos *Instance***

FLIROneSDKAutomaticTuningDidChange:
Acionado quando o recurso do ajuste automático é ativado ou desativado, para dispositivos com obturadores de ajuste automático
- (void)FLIROneSDKAutomaticTuningDidChange:(NSNumber *)deviceWillTuneAutomatically
Parâmetros:
deviceWillTuneAutomatically – É 0 se o ajuste automático (*automatic tuning*) estiver desligado
FLIROneSDKBatteryChargingStateDidChange:
Disparado quando o estado de carregamento da bateria do FLIR ONE muda (este método ainda não está implementado e nunca será chamado)
- (void)FLIROneSDKBatteryChargingStateDidChange:(FLIROneSDKBatteryChargingState)state
Parâmetros:
state – O novo estado de carregamento da bateria
Detalhes:
AVISO: É importante atualizar sempre a interface do utilizador (UI) na *thread* principal
FLIROneSDKBatteryPercentageDidChange:
Disparado quando a percentagem de carga da bateria do FLIR ONE muda (este método ainda não está implementado e nunca será chamado)
- (void)FLIROneSDKBatteryPercentageDidChange:(NSNumber *)percentage
Parâmetros:
percentage – O novo estado de percentagem da bateria, como um inteiro entre 0 e 100, inclusive
Detalhes:

AVISO: É importante atualizar sempre a interface do utilizador (UI) na *thread* principal

FLIROneSDKDidConnect:

Ativado logo após o dispositivo da FlirOne ter estabelecido a comunicação. Este é o primeiro evento recebido quando um dispositivo é conectado

É importante a conexão ao dispositivo FLIR ONE para receber *frames* ou informações do *sled*. Se a aplicação não estiver a receber este evento, assegure-se de que as dependências do projeto estão configuradas corretamente e que o *sled* está a funcionar

- (void)FLIROneSDKDidConnect

Detalhes:

Quando este evento é acionado, a aplicação deve rastrear que o estado da aplicação foi conectado e deve atualizar a interface do utilizador. Antes que este evento seja disparado, a interface do utilizador deve pedir ao utilizador para conectar o dispositivo.

AVISO: É importante atualizar sempre a interface do utilizador (UI) na *thread* principal

FLIROneSDKDidDisconnect

Disparado logo após o dispositivo FLIR ONE ter perdido a comunicação. Só é possível se a conectividade for previamente estabelecida

- (void)FLIROneSDKDidDisconnect

Detalhes

Quando este evento é acionado, a aplicação deve rastrear que o estado da aplicação foi desconectado e deve atualizar a interface do utilizador, pedindo ao utilizador para conectar o dispositivo.

AVISO: É importante atualizar sempre a interface do utilizador (UI) na *thread* principal

FLIROneSDKDidFinishCapturingPhoto:withFilepath:

Disparado quando uma solicitação de captura de imagem é enviada para o *singleton* FLIROneSDKStreamManager. Este evento será sempre acionado mesmo quando a captura falhar

- (void)FLIROneSDKDidFinishCapturingPhoto:(FLIROneSDKCaptureStatus)captureStatus withFilepath:(NSURL *)filepath

Parâmetros:

captureStatus – O resultado da captura de uma imagem. Indica se a captura foi ou não bem sucedida

filepath – Se a captura foi bem sucedida, será um NSURL a representar a localização da imagem capturada; caso contrário, será nulo. Usar esta sequência de caracteres com a instância de *singleton* FLIROneSDKLibraryManager para recuperar uma *thumbnail* e o caminho do ficheiro (*filepath*) absoluto do vídeo

FLIROneSDKDidFinishWritingVideo:withFilepath:

Acionado quando um vídeo capturado é gravado no disco. Este evento só irá disparar se FLIROneSDKDidStopRecordingVideo for acionado e a captura de vídeo estiver anteriormente ativa

- (void)FLIROneSDKDidFinishWritingVideo:(FLIROneSDKCaptureStatus)captureWriteStatus withFilepath:(NSURL *)filepath

Parâmetros:

captureWriteStatus – O resultado de uma tentativa de escrita do vídeo no disco

filepath – Se um ficheiro foi escrito com sucesso no disco, será um NSURL a representar a localização do vídeo guardado; caso contrário, será nulo. Usar esta sequência de caracteres

com a instância de *singleton* FLIROneSDKLibraryManager para recuperar uma *thumbnail* e o caminho do ficheiro (*filepath*) absoluto do vídeo

Detalhes:

AVISO: Se FLIROneSDKDidStartRecordingVideo foi acionado com um resultado com erro, este método não será accionado

FLIROneSDKDidStartRecordingVideo:

Acionado após uma solicitação FLIROneSDKStreamManager. Este evento será sempre accionado, mesmo que a captura de vídeo não possa ser iniciada

- (void)FLIROneSDKDidStartRecordingVideo:(FLIROneSDKCaptureStatus)captureStartStatus

Parâmetros:

captureStartStatus – O resultado de tentar iniciar uma captura de vídeo. Indica se a captura foi iniciada ou não com sucesso

FLIROneSDKDidStopRecordingVideo:

Disparado após uma solicitação FLIROneSDKStreamManager stopRecordingVideo ou se uma gravação é interrompida prematuramente devido a um erro

Este evento será sempre acionado depois de chamar stopRecordingVideo, mesmo se a captura do vídeo não tiver sido iniciada

- (void)FLIROneSDKDidStopRecordingVideo:(FLIROneSDKCaptureStatus)captureStopStatus

Parâmetros:

captureStopStatus – A razão pela qual uma captura foi parada, ou um erro a indicar que a captura não foi ativada

Detalhes:

AVISO: Se FLIROneSDKDidStartRecordingVideo foi acionado com um resultado com erro, este método não será acionado.

FLIROneSDKTuningStateDidChange:

Disparado quando o estado de sintonia do FLIROne muda

- (void)FLIROneSDKTuningStateDidChange:(FLIROneSDKTuningState)newTuningState

Parâmetros:

newTuningState – O novo estado de ajuste (*tuning state*) do dispositivo da FLIR One

Detalhes:

AVISO: É importante atualizar sempre a interface do utilizador (UI) na *thread* principal

- **FLIROneSDKVideoRendererDelegate Protocol Reference**

Conforme:

- NSObject

Declarado em:

- FLIROneSDKVideoRendererDelegate.h

- **Descrição**

Pode ser usado para permitir que os utilizadores usem *frames* do vídeo à sua maneira, usando o formato de imagem à sua escolha em vez do padrão MSX.

Por defeito, o gerador de fluxo (*stream manager*) gravará o vídeo na paleta atual como vídeo MSX, independentemente dos formatos transmitidos.

- **Tarefas**

imageForFrameAtTimestamp: (método requerido)

- **Métodos *Instance***

imageForFrameAtTimestamp:
Quando especificado, permite ao utilizador dispor de um UIImage personalizado para ser usado no vídeo de gravação num dado *timestamp*
- (UIImage *)imageForFrameAtTimestamp:(CMTime)timestamp
Parâmetros:
timestamp - *O timestamp* em que este UIImage será usado no vídeo
Retorna:
Um UIImage a ser gravado no vídeo no dado *timestamp*

Anexo C – Manual do utilizador

Introdução

Este manual contém algumas dicas e informações importantes de como utilizar a aplicação *JT's Diabetic Foot Teller* para o sistema operativo *Android*.

Em primeiro lugar, é importante conhecer a câmara FLIR ONE, uma vez que esta é um acessório crucial para o bom funcionamento da aplicação.

Figura C. 1 - Câmara FLIR ONE de segunda geração.

A figura C.1 foi retirada do manual de utilizador da câmara FLIR ONE de segunda geração da FLIR®.

A câmara tem uma bateria interna com uma duração de aproximadamente 1 hora. Como podemos observar na figura C.1, a câmara FLIR ONE tem um LED que indica se a bateria da câmara se encontra ou não a carregar, ou seja, se neste piscar uma luz verde, significa que esta se encontra a carregar e quando este emite uma luz verde contínua, indica que a bateria está carregada. Para fazer o carregamento da bateria da câmara, é necessário ligar um lado do cabo USB na entrada USB identificada na imagem como "*Micro USB power*" e o outro numa fonte de energia.

Por outro lado, para ligar a câmara é necessário carregar no botão *On/Off*. Inicialmente este emite uma luz alaranjada, ficando, depois, com uma luz verde intermitente.

Todas as outras funcionalidades são inferidas de forma intuitiva pela leitura da imagem.

Como utilizar a aplicação?

Depois de descarregar a aplicação, deve procurar no seu dispositivo móvel a aplicação chamada *JT's Diabetic Foot Teller* e clicar no seu ícone. Surgirá, de seguida, no seu ecrã o correspondente à figura C.2, indicando ao utilizador que deve ligar a câmara FLIR ONE. Nesta fase, ainda não é possível capturar imagens, uma vez que a câmara ainda não está conectada, mas a aplicação já permite a visualização de imagens da galeria e o envio dos dados para uma base de dados local e para um *webservice* remoto, o que será explicado mais à frente.

Figura C. 2 - Página inicial da aplicação.

Para utilizar a câmara, basta conectá-la na entrada USB do dispositivo móvel e ligá-la, carregando no botão *On/Off*, como ilustrado na figura seguinte adaptada do manual de utilizador fornecido pela FLIR®.

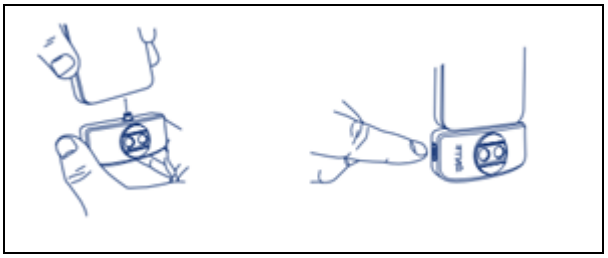

Figura C. 3 - Instruções de como ligar a câmara.

A figura C.3 foi retirada do manual de utilizador da câmara FLIR ONE de segunda geração da FLIR®.

Quando a câmara está conectada e funcional, aparece no ecrã o que se representa na figura C.4. Por omissão, a emissividade é de 0.98 (emissividade da pele), o tipo de imagem é térmica e a paleta de cores falsas é *rainbow*. Estes são os valores de utilização aconselhados.

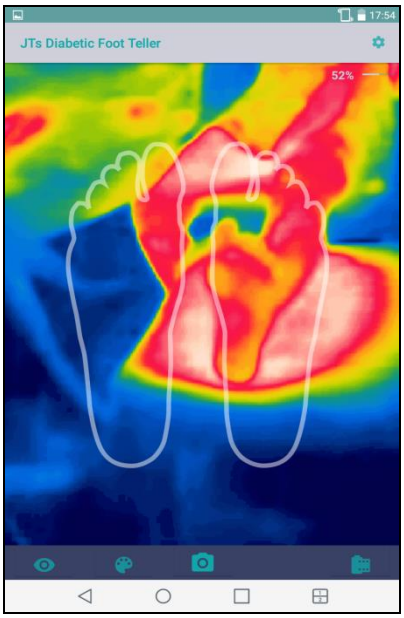

Figura C. 4 - Página principal da aplicação.

Neste momento, é visível no ecrã um *overlay* de uns pés, que funciona como um indicador de posição dos mesmos. Assim, o utilizador deve tentar posicionar os pés naquele sítio e naquela posição, para um melhor diagnóstico futuro.

Pode ver-se, no canto superior direito, um indicador da percentagem da bateria da câmara. As cores que este apresenta representam diferentes estados da carga da bateria. A tabela seguinte mostra o estado correspondente a cada cor.

Tabela C. 1 - Estado da bateria correspondente a cada cor.

Cor	Estado
Cinzento-claro	Normal
Verde	A carregar
Rosa	Falha devido ao calor ao carregar
Cinzento-escuro	Não está a carregar porque ocorreu uma falha devido à baixa corrente da fonte de energia
Vermelho	A bateria é igual ou inferior a 15%

Por cima do indicador do estado da bateria, ainda no canto superior direito, existe um ícone das definições . Este abre uma nova atividade que permite alterar a emissividade. A figura C.5 ilustra o que foi agora referido.

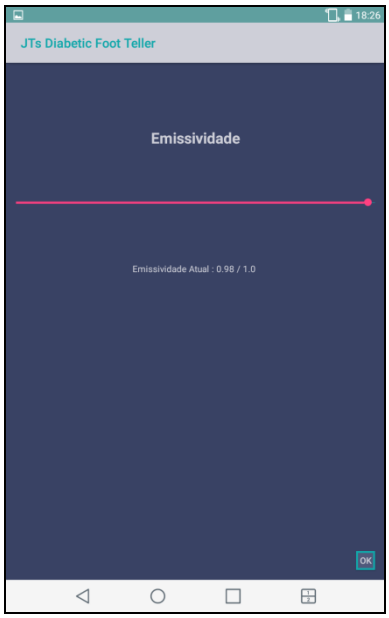

Figura C. 5 - Alterar emissividade.

Depois de alterada a emissividade, aparecerá uma mensagem, no fundo do ecrã, a avisar que a emissividade foi alterada com sucesso, bem como qual o valor para o qual esta foi alterada.

É de salientar que, sempre que esta atividade surge no ecrã, a emissividade que aparece como atual é 0.98, mesmo depois de esta já ter sido alterada. Isto acontece porque é o valor definido por defeito e quando esta atividade se fecha, ao clicar no botão OK, esta perde os valores definidos e envia-os para a atividade pai, ilustrada na figura C.4.

Regressando à figura C.4, no canto inferior esquerdo existe um ícone que tem a forma de um olho  . Ao clicar nele, é possível alterar o tipo de imagem a capturar. Existem três tipos de imagens possíveis: imagens **térmicas**, imagens **visíveis** ou imagens **MSX**, que contêm a imagem térmica sobreposta à imagem visível. Depois de alterado o tipo de imagem, aparecerá uma mensagem, no fundo do ecrã, com o tipo de imagem escolhido.

De seguida, apresenta-se uma imagem de cada tipo, bem como o menu *popup* que aparece ao carregar no botão mencionado.

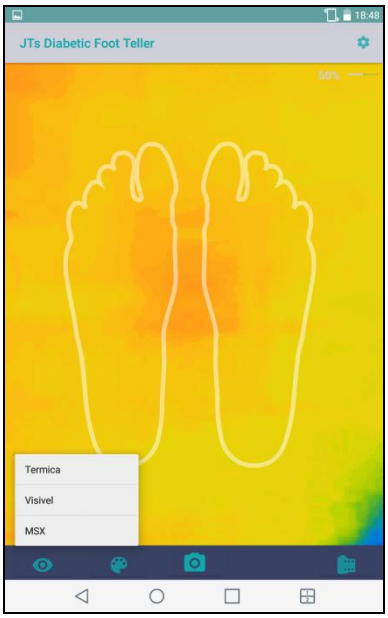

Figura C. 6 - Menu popup para alterar o tipo de imagem.

Figura C. 7 - Tipos de imagens.

NOTA: Se o tipo de imagem escolhido for visível, o ícone que corresponde à alteração da paleta de cores falsas, , fica invisível, uma vez que não se pode alterar a paleta das imagens visíveis.

Ao lado do ícone falado atrás, existe um outro, ainda no canto inferior esquerdo que tem a forma de uma paleta . Ao clicar nele, é possível alterar a paleta de cores falsas da imagem a capturar. Existem 9 diferentes paletas possíveis: **rainbow**, **iron**, **arctic**, **coldest**, **contrast**, **gray**, **hottest**, **lava** e **wheel**. Depois de alterada a paleta de cores falsas aparecerá uma mensagem, no fundo do ecrã, com o nome da paleta escolhida.

De seguida, apresenta-se uma imagem exemplo de cada paleta, retirada do manual de utilizador fornecido pela FLIR, bem como o menu *popup* que aparece ao carregar no botão mencionado.

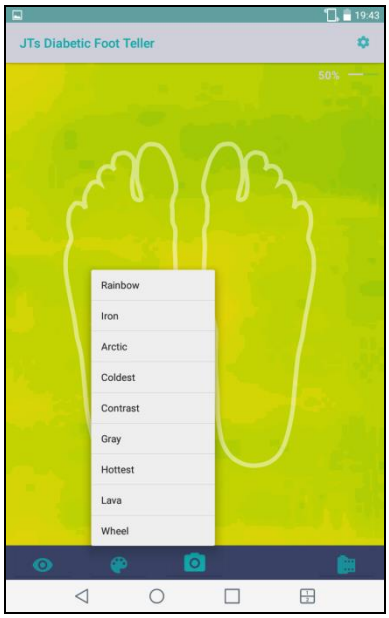

Figura C. 8 - Menu *popup* para alterar a paleta de cores falsas.

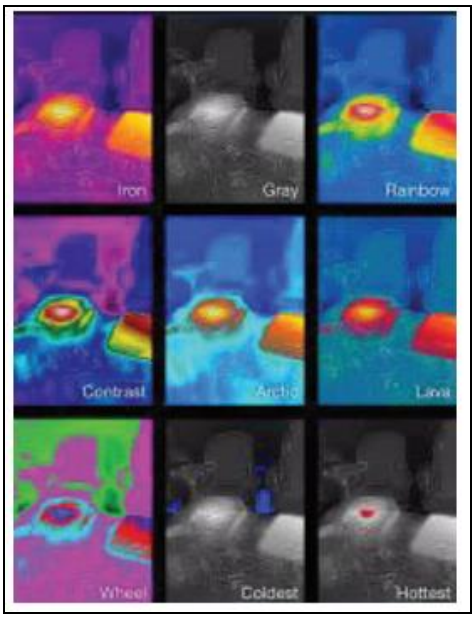

Figura C. 9 - Paletas de cores falsas disponíveis.

Como se pode observar na figura C.4, na parte inferior do ecrã, na zona central, existe um ícone de uma máquina fotográfica . Ao clicar nele, é possível capturar imagens. Após a sua captura, aparece uma mensagem, no fundo do ecrã, a dizer o nome e onde é que a imagem foi guardada.

Ainda na figura C.4, pode ver-se um ícone de um rolo fotográfico , no canto inferior direito. Ao clicar nele, o utilizador é redirecionado para a galeria do *smartphone* ou *tablet*. A figura C.10 mostra o que foi agora referido.

É de salientar que, por vezes, o utilizador pode ter de procurar a imagem noutro diretório que não os recentes, porque este nem sempre é atualizado.

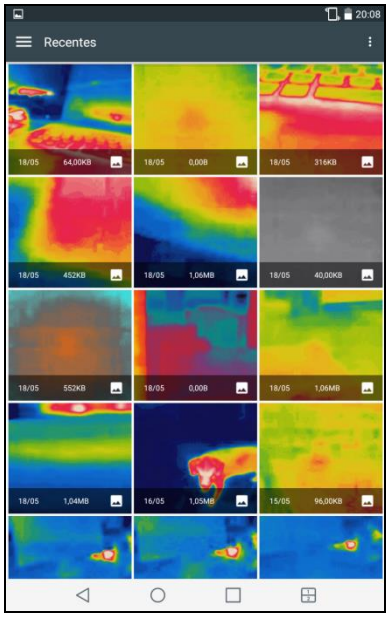

Figura C. 10 - Galeria de imagens.

Nesta fase, o utilizador deve escolher a imagem que pretende visualizar, clicando nela. Após isso, a imagem aparece no ecrã inteiro, como ilustrado na figura C.11.

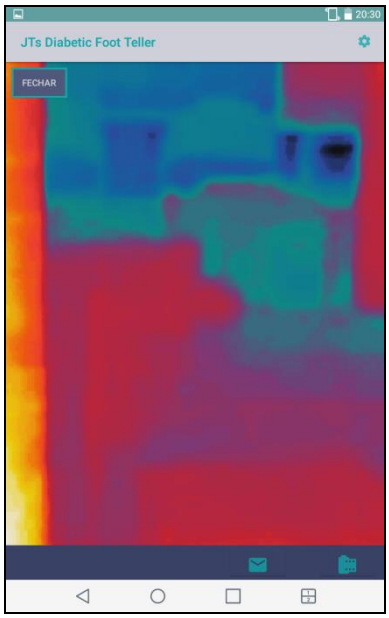

Figura C. 11 - Visualização da imagem escolhida.

Para voltar ao ilustrado na figura C.4, ou seja, para voltar a ver em tempo real o que se pretende capturar e talvez até capturar uma nova imagem, basta carregar no botão FECHAR.

Na figura C.11 é possível ver, de novo, no canto inferior direito o ícone do rolo fotográfico . Quer isto dizer que é possível abrir uma nova imagem.

Do lado esquerdo do botão atrás referido, no fundo do ecrã, encontra-se um ícone de uma carta . Depois de escolhida qual a imagem que se deseja enviar para o *webservice* ou armazenar na base de dados local, deve clicar-se no botão da carta. Este redirecionará o utilizador para um formulário que deve ser preenchido com os dados pessoais do doente. A figura C.12 mostra o que foi agora mencionado.

Figura C. 12 - Formulário dos dados.

Como se pode ver na figura C.12, existem 3 botões no fundo do ecrã. O botão que está no canto inferior esquerdo diz "Adicionar". Quando se carrega neste botão, os dados são armazenados localmente no dispositivo móvel. Se os dados forem adicionados à base de dados com sucesso, aparecerá uma mensagem de aviso no fundo do ecrã.

Os campos a preencher são autoexplicativos. No entanto dois deles merecem atenção especial. O IMC (índice de massa corporal) é calculado da seguinte forma: peso(kg)/altura^2(m).

No campo das zonas inexistentes, o utilizador deve clicar naquelas que não existem. A figura seguinte expõe as regiões de interesse na superfície plantar do pé do diabético. As zonas a preencher (ou não) estão aqui representadas.

Figura C. 13 - Regiões de interesse.

Estes dados podem depois ser vistos, carregando no botão "Ver", presente no canto inferior direito.

Para além dos dados preenchidos, é, ainda, enviado o nome e o caminho da imagem.

A figura seguinte exemplica como é apresentada a base de dados.

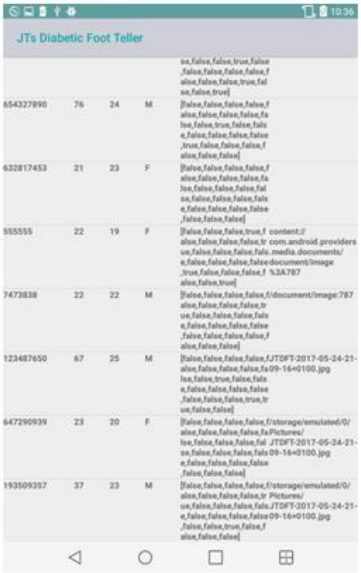

Figura C. 14 - Base de dados.

Para além de armazenar os dados, é, também, possível enviá-los para um servidor. Lá eles serão guardados e, numa outra fase, analisados.

Como presente na figura C.12, existe ainda um outro botão que diz “Enviar”. Ao carregar nele, os dados preenchidos no formulário são enviados para um *webservice* remoto.