DANILO DOS SANTOS DA FONTE

SMAP-SOFTWARE MÓVEL DE ANÁLISE POSTURAL

LAVRAS - MG

2014

DANILO DOS SANTOS DA FONTE

SMAP-SOFTWARE MÓVEL DE ANÁLISE POSTURAL

Monografia de Graduação apresentada ao Departamento de Ciência da Computação para obtenção do título de Bacharel em Sistemas de Informação

Orientador

Prof[a]. Dr[a]. Ana Paula Piovesan Melchiori

LAVRAS - MG

2014

DANILO DOS SANTOS DA FONTE

SMAP-SOFTWARE MÓVEL DE ANÁLISE POSTURAL

Monografia de Graduação apresentada ao Departamento de Ciência da Computação para obtenção do título de Bacharel em Sistemas de Informação

Prof[a]. Dr[a]. Ana Paula Piovesan Melchiori

Orientador

LAVRAS - MG

2014

SMAP – SOFTWARE MÓVEL DE ANÁLISE POSTURAL

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Sistemas de Informação para obtenção do título de Bacharel em Sistemas de Informação.

APROVADA em 11 de fevereiro de 2014.

Dr. André Pimenta Freire UFLA

Dr. Raphael Winckler Bettio UFLA

Drª. Ana Paula Piovesan Melchiori - UFLA

(Orientadora)

LAVRAS - MG
2014

Dedico esta monografia aos meus pais, que me me apoiaram e incentivaram ao longo da graduação.

AGRADECIMENTOS

Agradeço a minha orientadora Professora Ana Paula, que me ajudou ao longo do desenvolvimento deste trabalho.

Agradeço meus pais, que apesar das dificuldades sempre me apoiaram e me deram condições para chegar até aqui.

Agradeço a minha namorada e amiga Jossidele, que me incentivou e deu bronca nos momentos necessários.

RESUMO

O objetivo deste trabalho foi desenvolver um software para dispositivos móveis capaz de realizar análises de postura através de imagens, como aquelas realizadas através de um simetógrafo. Tendo como base uma ferramenta de análise postural online desenvolvida através de um projeto de iniciação científica da UFLA chamada SOAP.

Palavras-chave: android, dispositivo móvel, análise postural

ABSTRACT

The objective of this work was to develop a software for mobile devices able to perform postural analyses through images such as those carried out through a postural grid. Based on a postural analysis online tool developed through a research project of UFLA called SOAP .

Keywords: android, mobile devices, postural analysis

LISTA DE FIGURAS

Figura 1 Problemas da Má Postura Corporal 18
Figura 2 Vista Frontal Anterior - SAPO 22
Figura 3 Avaliação Postural - SOAP 24
Figura 4 Arquitetura do sistema operacional Android 27
Figura 5 Ciclo de Vida Activity - Android 31
Figura 6 Ciclo de Vida Service - Android 32
Figura 7 Execução Content Provider 33
Figura 8 Broadcast Receiver 33
Figura 9 Hierarquia View e View Group 34
Figura 10 Diagrama de Ferramentas Utilizadas 37
Figura 11 Exemplo - Coleta de Requisitos 39
Figura 12 Requisitos 43
Figura 13 Diagrama de Casos de Uso 44
Figura 14 Diagrama de Interface - Relatórios 45
Figura 15 Código de layout - XML 48
Figura 16 Android manifest 49
Figura 17 TAB inicial 54
Figura 18 Tela de Cadastro de Pessoas 55
Figura 19 Mensagem de Alerta - Cadastro completado com sucesso 56
Figura 20 Tela de Listagem de Pessoas 57
Figura 21 TAB de Avaliação 57
Figura 22 Sequência de Captura de Fotos 58
Figura 23 Avaliação postural 59
Figura 24 Opções Relatório 60
Figura 25 Relatório de Avaliação Postural 60
Figura 26 SOAP x SMAP 61
Figura 27 Diagrama de Interface - Cadastro de Pessoas 99
Figura 28 Diagrama de Interface - Avaliação de Pessoas 100
Figura 29 Cadastro de Pessoa - SMAP x SOAP 101
Figura 30 Nova Análise Postural - SMAP x SOAP 102
Figura 31 Análise Postural - SMAP x SOAP 103
Figura 32 Opções de Análise - SMAP x SOAP 104
Figura 33 Relatório - SMAP x SOAP 105
Figura 34 Ícone - cadastro de pessoa 106
Figura 35 Ícone - lista de pessoas 107
Figura 36 Opção - avaliar 107
Figura 37 Ícone - foto 107

Figura 38 Ícone - avaliação postural 108
Figura 39 Opção - avaliar 108
Figura 40 Ícone - avaliações 108

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
GPS	Global Positioning System
JNI	Java Native Interface
JVM	Máquina Virtual Java
MVD	Máquina Virtual Dalvik
SAPO	Software de Avaliação Postural
SMAP	Software Móvel de Análise Postural
SOAP	Software Online de Análise Postural
UFLA	Universidade Federal de Lavras

SUMÁRIO

1 Introdução 14
2 Referencial Teórico 16
2.1 Postura Corporal 16
2.2 Desvio Postural 17
2.3 Problemas da Má Postura Corporal 17
2.4 Avaliação Postural 18
2.5 Ferramentas de Análise Postural 19
2.5.1 SAPO 19
2.5.1.1 Funcionalidades do SAPO 20
2.5.1.2 Protocolo de Avaliação SAPO 21
2.5.2 SOAP 22
2.5.2.1 Funcionalidades 22
2.5.2.2 Processo de Avaliação Postural através da ferramenta SOAP 23
2.6 Plataforma Android 24
2.7 Arquitetura do Sistema Operacional 25
2.8 Máquina Virtual *Dalvik*(MVD) 28
2.9 Sobre o Android 29
2.9.1 Conceitos Fundamentais de uma Aplicação Android 29
2.9.2 Activity 30
2.9.3 Service 31
2.9.4 Content Provider 32
2.9.5 Broadcast Receiver 33
2.9.6 Intent 34
2.9.7 Interface com o Usuário 34
2.9.7.1 View e ViewGroup 34
2.9.7.2 Layout 35
2.9.7.3 Widgets 35
2.9.8 Recursos 35
2.9.9 Armazenamento de Informações 36
2.10 Publicando uma Aplicação 36
3 Material e Métodos 37
3.1 Método 39
4 Desenvolvimento 42
4.1 Modelagem do Software 42
4.1.1 Requisitos Funcionais 42
4.1.2 Diagrama de Casos de Uso 43

4.1.3 Diagramas de Interface 44
4.2 Execução e Depuração do Aplicativo 45
4.3 Testes Funcionais 46
5 Implementação 47
5.1 Interface 47
5.2 Permissões 48
5.3 Captura da Foto 49
5.4 GPS 50
5.5 Persistência de Dados 50
5.6 Rotação de Linha 51
5.7 Manutenção da Sessão 51
5.8 Fragments 52
5.9 Adapter 52
6 Resultados 54
6.1 Cadastrar Pessoas 54
6.2 Análise Postural por Linhas 57
6.3 Relatório de Análise Postural 59
6.4 Comparativo com a ferramenta SOAP 61
7 CONCLUSÃO 63
7.1 Problemas Encontrados 63
7.2 Trabalhos Futuros 64
REFERÊNCIAS 65
APÊNDICE 68

1 Introdução

A crescente preocupação com a saúde e qualidade de vida no trabalho tem sido foco de diversos estudos(TEIXEIRA; KOTHE, 2012)(BATIZ; SANTOS; ANZARDO, 2009). Nesse contexto, alguns fatores, como boa postura corporal, condições de trabalho inadequadas e atividades repetitivas(BATIZ; SANTOS; ANZARDO, 2009), devem ser analisados e estudados visando melhores condições de trabalho, maior satisfação profissional ,pessoal e melhor qualidade de vida.

Dentre esses fatores a boa postura no trabalho das posições sentada e em pé, bem como a garantia de que os membros superiores e inferiores estejam em posições confortáveis(BATIZ; SANTOS; ANZARDO, 2009) deve ser assegurada.

Tornando-se necessário a realização da análise postural desses diversos profissionais, assim como orientações relacionadas à postura, a execução correta das atividades e a manutenção da aptidão física em níveis favoráveis com a colaboração de exercícios físicos e ações ergonômicas.

Essas ações além de serem um procedimento para prevenir os problemas da coluna vertebral, contribuem na postura corporal durante as funções diárias com economia de energia sem exceder o limite tolerável da região músculo articular(JR, 1995 apud TOSCANO; EGYPTO, 2001)

Uma dos métodos de análise postural é análise postural estática, utilizando um simetógrafo e um fio de prumo, no qual são identificados diversos pontos pré-definidos, que ajudam a identificar possíveis assimetrias, que podem ser indícios de um possível desvio da postura correta.

Entre as ferramentas encontradas, capazes de realizar uma análise postural, através de um processo parecido com o estático utilizando o sime-

tógrafo, podem ser destacadas as ferramentas SAPO e SOAP, está última desenvolvida através de um projeto de iniciação científica da UFLA.

Essas ferramentas assimilam-se devido ao fato de gerarem um relatório de análise postural através de imagens, embora a ferramenta SAPO tenha um processo de análise e uso mais complexo que a SOAP.

Ambas as ferramentas são acessíveis aos profissionais da área, entretanto existe a ausência de uma ferramenta que seja utilizada de forma mais prática e interativa no processo de análise postural, que foi a motivação para este trabalho.

Devido a ausência de uma ferramenta que seja utilizada de forma mais prática e interativa no processo de análise postural o presente trabalho objetiva desenvolver uma ferramenta que supra essa necessidade e possa integrar-se a ferramenta SOAP em trabalhos futuros.

Para isto foi escolhido desenvolver um aplicativo para dispositivos móveis da plataforma Android, similar a ferramenta SOAP, que possibilite a captura de imagens e a análise postural do indivíduo através destas, gerando um relatório postural.

2 Referencial Teórico

Neste capítulo tem-se a conceituação de postura corporal, desvio postural, possíveis problemas ocasionados pela má postural corporal e a análise postural, assim como sua importância. Também são apresentados duas ferramentas de análise postural, sendo uma delas, aquela no qual o software desenvolvido neste projeto foi baseado. Ao final o sistema operacional Android é abordado, com sua arquitetura, características e particularidades.

2.1 Postura Corporal

A postura corporal pode ser entendida como a posição que nosso corpo adota no espaço, visando o maior conforto possível. Para que este conforto possa existir também é necessário o equilíbrio do sistema neuro-musculoesquelético(KENDALL; MCCREARY; PROVANCE, 1986).

A postura corporal também pode ser definida como a posição que nosso corpo adota em determinado momento, sendo este um conjunto das diferentes articulações do corpo e a maneira como atuam entre si naquele momento (MAGEE, 2002).

Para buscar harmonia o corpo adota uma nova postura sempre que existe algum desconforto, devido a diversos fatores, como compressão articular e contração muscular contínua. Através disto podemos entender que a postura corporal mais adequada é aquela que melhor se ajusta ao nosso corpo de forma a favorecer nossas atividades diárias (SMITHL; WEISS; LEHMKUHL, 1997).

2.2 Desvio Postural

O alinhamento ideal como definido por (KENDALL; MCCREARY; PROVANCE, 1986) consiste naquele em que os músculos e articulações, junto a suas estruturas encontrem-se em equilíbrio, possibilitando o menor esforço possível do indivíduo, acarretando na maior eficiência possível(KENDALL; MCCREARY; PROVANCE, 1986).

(KENDALL; MCCREARY; PROVANCE, 1986) também propõem que a postura ideal é aquela, no qual o corpo esteja alinhado verticalmente no plano sagital. Para isto é necessário que um fio de prumo estendido do chão até a altura da cabeça passe simetricamente "pelo maléolo lateral e pelo lóbulo da orelha, ambos do mesmo lado"(FERREIRA, 2008).

Segundo (SANTOS *et al.*, 2009) um desvio postural consiste em uma posição adotada pelo corpo de forma que este apresente curvaturas anormais da coluna, de forma que os ossos dos membros inferiores estejam fora do alinhamento ideal para a sustentação do corpo.

Um desvio postural também pode ser entendido como qualquer posição que aumente o estresse sobre as articulações (MAGEE, 2002).

2.3 Problemas da Má Postura Corporal

Entre os principais problemas ocasionados pela má postura corporal podemos destacar(UNIMEDRIO, 2013):

- **Lordose**: Consiste no aumento anormal da curva lombar.
- **Cifose**: Consiste no aumento anormal da concavidade posterior da coluna vertebral.

- **Escoliose**: Consiste em uma curvatura lateral da coluna vertebral.

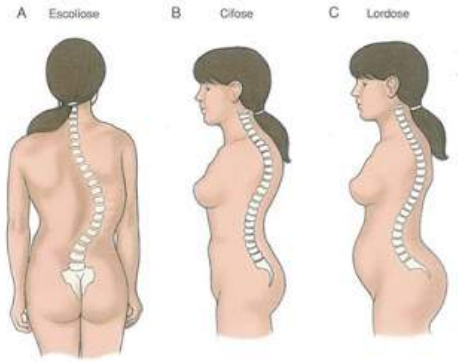


Figura 1 Problemas da Má Postura Corporal
Fonte: <http://marcoshanry.blogspot.com.br/2011/09/coluna-vertebral-e-os-movimentos.html>

2.4 Avaliação Postural

A avaliação postural é necessária para mensurar possíveis desequilíbrios do padrão dado como uma postura correta (KENDALL; MCCREARY; PROVANCE, 1986), possibilitando a adoção de técnicas capazes de realizar melhorias da postura do indivíduo , diminuindo problemas ocasionados por uma má postura corporal.(BRACCIALLI; VILARTA, 2000)

Essa avaliação pode ser realizada de diversas maneiras, sendo a utilizada neste trabalho a avaliação postural estática através de imagens, semelhante àquela realizada com o simetrógrafo, facilitando a percepção e a visualização de assimetrias com base em critérios pré-definidos.

Esses critérios devem estar relacionados ao processo de adaptação do corpo, onde os segmentos avaliados devem estar associados à linha de

gravidade, de forma que segmentos incompatíveis sejam considerados como um desiquilíbrio (KENDALL; MCCREARY; PROVANCE, 1986).

2.5 Ferramentas de Análise Postural

Conforme (NORIEGA, 2012) o uso da tecnologia tem possibilitado a criação de diversas ferramentas capazes de realizar análises posturais de maneira relativamente simples e intuitiva. Entre as diversas ferramentas existentes no mercado podemos destacar a ferramenta SAPO e a ferramenta SOAP, na qual este trabalho foi baseado.

2.5.1 SAPO

A ferramenta SAPO foi desenvolvida através de um projeto científico pela FAPESP, sendo disponibilizada gratuitamente através do endereço: <http://puig.pro.br/sapo/>. Entre seus principais objetivos estâo (SAPO, 2005a)(SAPO, 2014):

- Desenvolvimento de software livre para avaliação postural
- Desenvolvimento de estudos metrológicos sobre avaliação postural computadorizada
- Criação de tutoriais científicos sobre avaliação postural e o software
- Criação de banco de dados com resultados de avaliações feitas pelos centros colaboradores

Seu desenvolvimento foi feito através da linguagem de programação java, podendo ser utilizado em qualquer sistema operacional que possua

uma JVM. Sendo uma aplicação *Open Source*, seu código fonte pode ser encontrado no seguinte endereço: <https://code.google.com/p/sapo-desktop/source/browse/>

2.5.1.1 Funcionalidades do SAPO

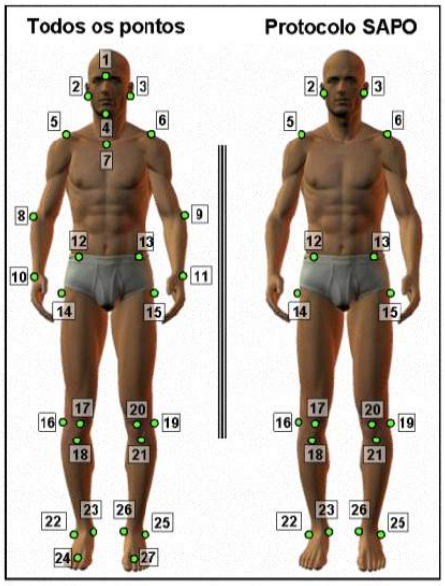
Segundo (NORIEGA, 2012) a ferramenta SAPO possui as seguintes funcionalidades principais:

- **Carregar imagens**: funcionalidade que permite que uma ou mais imagens possam ser carregadas para a análise.
- **Processamento de Imagem**: funcionalidade que permite o tratamento da imagem para melhor avaliação.
- **Calibrar Imagem**: funcionalidade que estabelece uma relação matemática entre o espaço real e as dimensões da imagem.
- **Medição livre de distância, pontos, ângulos e áreas na imagem**: funcionalidade que permite a livre medição dessas informações.
- **Digitalização de Pontos na imagem**: funcionalidade no qual os pontos necessários para a avaliação são plotados sobre a imagem.
- **Cálculo de medidas a partir dos pontos**: funcionalidade que permite o cálculo de medidas a partir dos pontos plotados na imagem.
- **Vizualização dos pontos e das medidas calculadas**: funcionalidade que apresenta de forma organizada os pontos plotados sobre a imagem, bem como os resultados gerados a partir destes.

- **Geração de Relátorios**: apresentação das informações obtidas através da análise realizada pelo software.
- **Documentação**: documentação sobre as funcionalidades existentes no software.

2.5.1.2 Protocolo de Avaliação SAPO

Conforme (SAPO, 2005b) a ferramenta SAPO possui um protocolo de avaliação postural padrão, chamado de Protocolo SAPO de Marcação de Pontos. O Protocolo foi sugerido pela equipe de desenvolvimento da ferramenta com base na relevância clínica, base científica, viabilidade metodológica e aplicabilidade(SAPO, 2005b). Ele possui como base para sua avaliação a vista de uma pessoa por quatro perspectivas diferentes, sendo elas: frontal anterior, frontal posterior, lateral direita e lateral esquerda no qual os pontos pré definidos pelo protocolo são plotados.



1. Glabela
2. Trago direito
3. Trago esquerdo
4. Mento
5. Acrômio direito
6. Acrômio esquerdo
7. Manúbrio do esterno
8. Epicôndilo lateral direito
9. Epicôndilo lateral esquerdo
10. Ponto médio entre o processo estilóide do rádio e a cabeça da ulna direita
11. Ponto médio entre o processo estilóide do rádio e a cabeça da ulna esquerda
12. Espinha ilíaca ântero-superior direita
13. Espinha ilíaca ântero-superior esquerda
14. Trocânter maior do fêmur direito
15. Trocânter maior do fêmur esquerdo
16. Linha articular do joelho direito
17. Ponto medial da patela direita
18. Tuberosidade da tíbia direita
19. Linha articular do joelho esquerdo
20. Ponto medial da patela esquerda
21. Tuberosidade da tíbia esquerda
22. Maléolo lateral direito
23. Maléolo medial direito
24. Ponto entre a cabeça do 2° e 3° metatarso direito
25. Maléolo lateral esquerdo
26. Maléolo medial esquerdo
27. Ponto entre a cabeça do 2° e 3° metatarso esquerdo

Figura 2 Vista Frontal Anterior - SAPO
Fonte: (NORIEGA, 2012)

2.5.2 SOAP

A Ferramenta de análise postural SOAP foi desenvolvida através de um projeto de iniciação científica da UFLA. Ela pode ser encontrada e utilizada através do seguinte endereço:<http://www.anamelchiori.com.br/soap/>

2.5.2.1 Funcionalidades

Entre as funcionalidades do SOAP destacam-se:

- **Manutenção de pessoas**: funcionalidade que permite ao sistema manter uma base de dados contendo informações relevantes de diversas pessoas a serem avaliadas.
- **Carregar imagens**: funcionalidade que permite carregar uma imagem para ser avaliada.
- **Manutenção de imagens**: funcionalidade que permite ao sistema manter um banco de dados contendo imagens de diversas pessoas a serem avaliadas.
- **Plotagem de linhas**: funcionalidade que permite inserir/remover e movimentar linhas que identificam pontos anatômicos previamente definidos.
- **Rotação de linhas**: funcionalidade que permite rotacionar as linhas pré definidas, buscando identificar um possível desvio da postura padrão.
- **Geração de relatório**: funcionalidade que permite gerar um relatório de avaliação postural a partir da avaliação através das linhas.

2.5.2.2 Processo de Avaliação Postural através da ferramenta SOAP

A avaliação postural da ferramenta SOAP é realizada utilizando o mesmo processo de avaliação realizado através do simetógrafo. Esse processo consiste na identificação de pontos anatômicos pré-definidos, onde são traçadas linhas horizontais e verticais através desses pontos, podendo assim

realizar uma comparação entre os pontos de ambos os lados identificando possíveis desvios (AVALIACAO, 2014).

Figura 3 Avaliação Postural - SOAP

2.6 Plataforma Android

O Android é um sistema operacional para dispositivos móveis *open-source* baseado no Linux. Criado e mantido pelo *Google* com a colaboração da *Open Handseat Alliance*, que tem como finalidade acelerar a inovação em dispositivos móveis e oferecer aos consumidores a melhor, mais rica e também mais barata experiência com aparelhos móveis (GARGENTA, 2011).

Ele teve sua primeira versão disponibilizada no ano de 2009, sendo ela a versão 1.5, chamada de *cupcake*. Todas as versões subsequentes do

Android também receberem nomes de doces ou sobremesas. Atualmente o Android está na sua versão 4.3, chamada de *kit kat*.

Sendo uma plataforma completa para dispositivos móveis o Android é um SO de código aberto que possui máquina virtual, APIS, e diversos aplicativos nativos, como navegados, discador, agenda entre outros(OHA, 2013a)(OHA, 2013b).

Além das diversas aplicações nativas do Android, outras podem ser desenvolvidas e fornecidas através de terceiros(ARIMA, 2009). Essas aplicações podem facilmente substituir e expandir os recursos nativos disponibilizados pelo Android(MEIER, 2009).

Para disponibilizar as diversas aplicações existentes para o Android, este conta com uma loja virtual que pode ser acessada através de qualquer dispositivo Android, a *Android Market*.

Esta loja virtual possibilita ao desenvolvedor disponibilizar sua aplicação para o usuário final, em um modelo semelhante à AppStore,loja virtual de aplicativos para o iPhone, smartphone da marca Apple. (ARIMA, 2009).

A plataforma Android possui também outras características interessantes, como a fácil integração com serviços oferecidos pelo *Google* e diversas ferramentas disponíveis gratuitamente para os seus desenvolvedores (MEIER, 2009).

2.7 Arquitetura do Sistema Operacional

O sistema operacional do Android foi baseado no kernel 2.6 do Linux, e é responsável por gerenciar a memória, os processos, *threads*, e a segurança dos arquivos e pastas, além de redes e *drivers* (LECHETA, 2009).

Através do gerenciamento de processos, diversas aplicações podem ser executadas simultaneamente (*mult-thread*), no qual o kernel fica responsável por gerenciar toda a mémoria,permitindo que cada aplicação atue como um processo diferenciado, com recursos próprios disponibilizados por este.

Com este gerenciamento o kernel se encarrega de liberar recursos, reiniciar processos e recuperar um processo existente de falhas e erros, mantendo a estabilidade do sistema(LECHETA, 2009).

Sua segurança é baseada na segurança do Linux, onde cada aplicação executada possui acesso a seus próprios recursos através de uma *thread* dedicada, no qual um usuário diferente é criado para cada aplicação instalada. Permitindo que cada aplicação tenham acesso apenas aos seus próprios recursos(LECHETA, 2009).

O sistema operacional é dividido em quatro camadas (*Linux Kernel, Libraries, Application Framework, Applications*), conforme mostrado na Figura 4. Sendo que uma camada superior não utiliza os recursos da camada inferior (GOOGLE, 2013c).

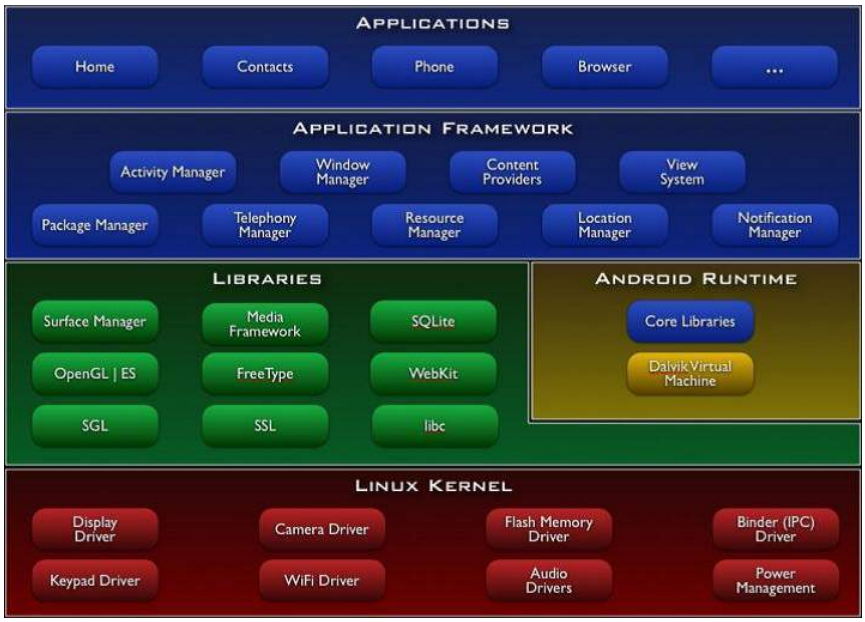


Figura 4 Arquitetura do sistema operacional Android
Fonte: (GOOGLE, 2013c)

A camada mais próxima do hardware é a Linux Kernel.Tornando-se a base do sistema operacional encarregando-se da segurança, gerenciamento de memória e processos, empilhamento de pacotes de rede e comunicação com *drivers*. Essa camada é a abstração entre o dispositivo físico e a camada de software (GOOGLE, 2013c).

A camada Libraries contém um conjunto de bibliotecas que auxiliam na execução das aplicações. Essas bibliotecas foram escritas nas linguagens C/C++ e customizadas para dispositivos móveis. Esta camada também possui a máquina virtual *Dalvik*, que gerencia a execução dos aplicativos e uma implementação baseada no Open Graphics Library Embedded System (*OpenGL ES*)(GOOGLE, 2013c).

A camada *Application Framework* é composta por (APIs) Java encarregadas de abstrair o uso das bibliotecas da segunda camada. Estas APIs invocam as bibliotecas da segunda camada através das interfaces Java Native Interface (JNI) (GOOGLE, 2013c).

Na última camada, *Applications*, estão os aplicativos disponíveis ao usuário final. Como cliente de e-mail, calendário, programa de mensagem instantânea, mapa e contatos (GOOGLE, 2013c).

2.8 Máquina Virtual *Dalvik*(MVD)

Para o desenvolvimento de aplicativos Android é utilizada linguagem Java. Entretanto o Android não possui uma Java Virtual Machine(JVM) no seu sistema operacional. Foi incluída uma máquina virtual chamada *Dalvik*, desenvolvida especialmente para execução em dispositivos móveis (LECHETA, 2009).

Ela é diferente das máquinas virtuais Java existentes em outras plataformas, especialmente por ser baseada em pilhas ao invés de registros e utilizar o Kernel do SO para tarefas básicas como controle de processos e gerenciamento de memório(DALVIKVM, 2013).

A MVD possui por padrão a execução de aplicações em *mult-tread*, permitindo que cada aplicativo seja executado separadamente. Assim a cada novo aplicativo executado, um processo contendo uma instancia da máquina virtual é aberto.

O sistema operacional então se torna encarregado por manter o processo até que seja necessário que este seja interrompido para liberar espaço na memória. Para alterar esse comportamento, é preciso criar explicitamente novas *threads* dentro de uma instância da MVD ou definir componen-

tes a serem executados em novas instâncias da máquina virtual (GOOGLE, 2013a).

A MVD possui também um *garbage collector* para evitar que objetos inutilizados memória continuem existindo (GOOGLE, 2013b).

No desenvolvimento de aplicativos todos os recursos da linguagem Java estão disponíveis normalmente. Os aplicativos são compilados para *bytecode* (class) e depois convertidos para o formato dex(*Dalvik* Executable), que representa uma aplicação Android compilada.

Os arquivos dex e outros recursos, como imagens, são compactados em um único arquivo com extensão apk (Android Package)(LECHETA, 2009) que pode ser instalado e executado no sistema operacional Android.

2.9 Sobre o Android

Os conceitos apresentados a seguir possuem como referência (OHA, 2013b).

2.9.1 Conceitos Fundamentais de uma Aplicação Android

As aplicações Android são feitas em código Java. O código assim que compilado é empacotado junto de outros recursos utilizados na aplicação em um arquivo .apk. Este arquivo é utilizado como veículo de distribuição para que os usuários possam instalar a aplicação.

Cada aplicação é executada em um processo próprio e cada processo possui sua própria máquina virtual. Sendo o Android baseado em Linux, por padrão cada aplicação executada possui um ID de usuário diferente de forma que as permissões de acesso a arquivos sejam individuais para cada aplicativo.

Cada aplicação é construída através de componentes individuais que são instanciados no momento que se tornam necessários. Existem quatro tipos de componentes básicos no Android, sendo elas *activities*, *services*,*content providers* e *broadcasts receivers*.

2.9.2 Activity

Uma *activity* corresponde a uma tela de iteração para o usuário. Uma activity pode ser apresentada de diversas maneiras, sendo as principais através de uma tela *full screen*, uma janela flutuante ou dentro de outra atividade. Toda activity possui um ciclo de vida bem definido, no qual é possível que ao longo de sua execução ela alterne entre três estados. Sendo estes, ativo, pausado e parado. A transição de estados de uma Activity é realizada através das chamadas de métodos específicos contidos na classe, conforme a Figura 5.

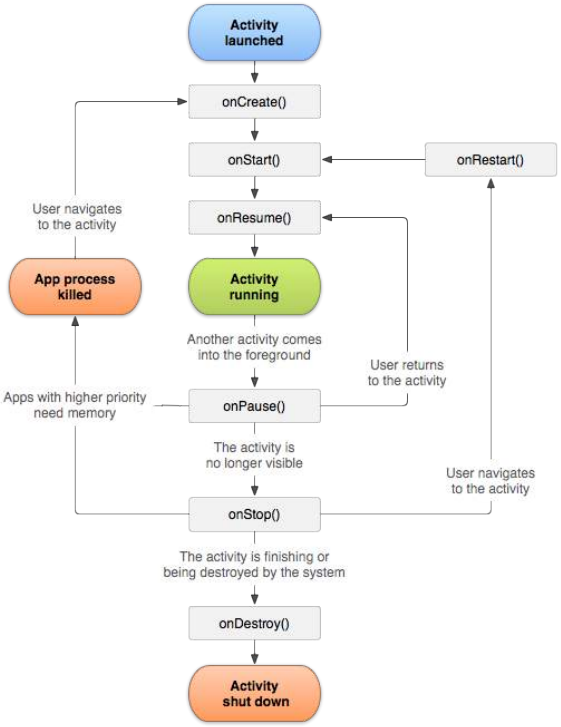


Figura 5 Ciclo de Vida Activity - Android
Fonte: <http://developer.android.com/reference/android/app/Activity.html>

Também através destes estados é possível identificar a interação que o usuário pode realizar em uma *Activity*.

2.9.3 Service

Services não possuem interface visual e não possuem interações com usuário, são componentes que podem ser executados em segundo plano durante longos períodos de tempo. Um service permanecera executando mesmo que o usuário mude aplicação. O ciclo de vida de um *service* pode ser definido como em execução e parado conforme a Figura 6.

Figura 6 Ciclo de Vida Service - Android
Fonte:
<http://www.edureka.in/blog/android-tutorials-beginners-service-component/>

2.9.4 Content Provider

Os *content providers* estão encarregados de encapsular dados e fornece-los a aplicação conforme o necessário, também através de um *content provider* é possível compartilhar dados entre aplicações.

Um content provider é considerado ativo apenas no momento que ele esta respondendo a uma mensagem, conforme a Figura 7.

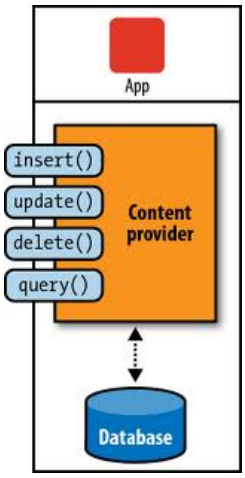


Figura 7 Execução Content Provider
Fonte: <http://dev.icybear.net/learning-android-cn/book.html>

2.9.5 Broadcast Receiver

Por último os *broadcasts receivers* são aqueles que respondem através de chamadas de determinados eventos. Através de um broadcast receiver é possível a uma aplicação enviar e receber chamadas diretas ao sistema. Exceto pelo *content provider* todos os outros três componentes são ativados através de chamadas assíncronas.

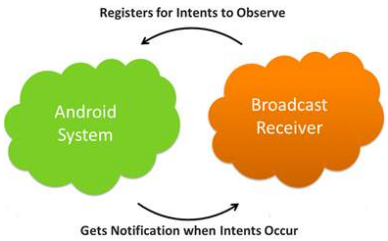


Figura 8 Broadcast Receiver
Fonte: <http://www.edureka.in/blog/what-is-android/>

2.9.6 Intent

A chamada de uma ação a ser executada em uma aplicação Android é chamada de *Intent*. Uma Intent pode ser definida de maneira explicita ou implícita. Quando definida de forma explicita, o componente a ser executado é definido explicitamente. Quando de forma implícita esse componente é escolhido pelo sistema operacional, sendo escolhido aquele que melhor responde a necessidade da aplicação.

2.9.7 Interface com o Usuário

2.9.7.1 View e ViewGroup

A interface com usuário é construída através de uma hierarquia de objetos composta por *Views* e *ViewsGroups*. Conforme a Figura 9 a hierarquia é representada através de uma árvore no qual as folhas são do tipo *View* e os ramos do tipo *ViewGroup*.

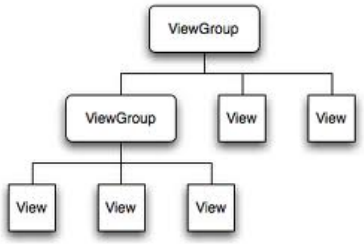


Figura 9 Hierarquia View e View Group
Fonte: <http://mobileorchard.com/android-app-development-controls-part-one-introduction-to-ui-in-android-and-text-controls/>

A *View* define uma área retangular na tela e s ervem como base para a inserção de objetos como campos de texto, botões, chamados de widgets. A classe *ViewGroup* define como esses objetos vão estar organizados na tela.

2.9.7.2 Layout

A organização dos objetos pode ser dada de forma específica ou automática. Esse tipo de organização é definido através de um *Layout*. Embora seja possível definir o tamanho e a localização específica de um componente na tela, a organização de componentes de forma não absoluta é muito mais vantajosa devido melhor adaptação da interface da aplicação a telas diferentes (tamanho e resolução) dos dispositivos no qual pode ser executada.

2.9.7.3 Widgets

A interface do Android disponibiliza diversos *Widgets* prontos, que formam um conjunto bastante rico para criar uma interface para o usuário. Também é possível a criação de componentes customizados.

2.9.8 Recursos

Em uma aplicação Android é possível referenciar elementos externos para sua utilização, como sons, imagens e layouts. Para a utilização destes recursos, todos eles devem estar dentro do diretório res/.

O acesso destes recursos através do código Java e da aplicação é feita através da classe R que disponibiliza todos os recursos disponíveis na aplicação. Essa classe contém subclasses para cada tipo de recurso suportado pelo Android com identificadores dos recursos compilados.

2.9.9 Armazenamento de Informações

O sistema operacional Android armazena informações de forma que estas sejam visíveis apenas para aplicação dona destas informações. Para que outras aplicações tenham acesso a esses dados é necessária à utilização de um *Broadcast Receiver*.

A plataforma permite que os dados sejam armazenados de diversas formas. Como através de um mecanismo de preferências através de dados primitivos, utilizada para armazenar as preferências de um usuário.

Ainda existe a possibilidade de armazenar informações diretamente na memória do dispositivo utilizando arquivos, ou banco de dados SQLite através de tabelas. Outra abordagem no armazenamento de dados se dá através do armazenamento externo utilizando a comunicação de rede.

2.10 Publicando uma Aplicação

Uma aplicação é distribuída através de um arquivo com sufixo .apk semelhante ao .jar que empacota o código compilado assim como os recursos da aplicação.

Para que uma aplicação possa ser disponibilizada e assim poder ser instalada em um dispositivo Android é necessário que ela esteja assinada digitalmente.

Entre as opções de distribuição o desenvolvedor pode disponibilizar seu aplicativo através do Android Market, Sendo o acesso ao Android Market possível através de todos os dispositivos Android, essa se torna a melhor opção para atingir o maior número de usuários.

3 Material e Métodos

A pesquisa desenvolvida nesse projeto pode ser classificada como de caráter tecnológico ou aplicada, no qual o resultado final é uma ferramenta de análise postural para dispositivos móveis.

Quanto aos requisitos coletados através da análise da ferramenta SOAP, esses podem ser considerados dados qualitativos relevantes ao processo de desenvolvimento.

O resultado dessa análise, assim como a construção dos diagramas de interface e de casos de uso são dados qualitativos no qual podem ser observados pontos importantes para o desenvolvimento da ferramenta.

Os materiais e ferramentas utilizadas para desenvolver o aplicativo móvel de análise postural foram: Android SDK, Banco de dados SQLite e IDE (Integrated Development Environment) de Desenvolvimento Eclipse como apresentados na Figura 10.

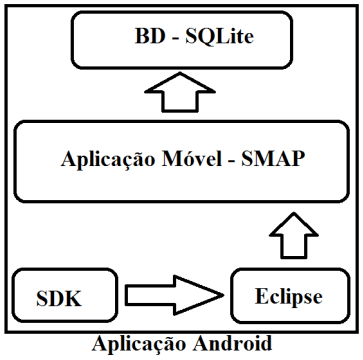


Figura 10 Diagrama de Ferramentas Utilizadas

Durante seu desenvolvimento foi utilizado o Android SDK em conjunto com o Eclipse através do *plug-in* Android ADT. A aplicação móvel faz a manipulação de dados através do Sqlite utilizando a API ORM lite.

A seguir são apresentadas as informações relevantes sobre cada ferramenta utilizada.

- **Android SDK**: O Android SDK(2013), na versão 4.1 utilizada neste trabalho, é uma ferramenta *open-source* que prove bibliotecas, ferramentas e funcionalidades necessárias para construir, depurar, testar e disponibilizar aplicativos Android. A ferramenta pode ser obtida através do site <https://developer.android.com/sdk/index.html>.

- **Sqlite**: É um banco de dados transacional nativo no sistema do Android que não depende de instalação de servidores e não necessita de configuração. É um banco de dados muito utilizado em sistemas que necessitam de um banco de dados eficiente e veloz. Ele pode ser obtido gratuitamente através do site <http://www.sqlite.org/>.

- **IDE Eclipse**: A IDE Eclipse (2013) foi desenvolvida pela *International Business Machines* (IBM), e posteriormente disponibilizada para a comunidade sobre as licenças *Common Public License* (CPL) e *Eclipse Public License* (EPL). O eclipse é um ambiente de desenvolvimento de aplicações feito em Java, que permite aos seus utilizadores desenvolverem aplicações em várias linguagens.

  O ambiente eclipse proporciona ao desenvolvedor tudo aquilo necessário para o desenvolvimento e codificação de uma aplicação. Suas funcionalidades também podem ser facilmente estendidas através da utilização de plug-ins.

- **Android ADT**: O plug-in Android ADT, é um plug-in para o eclipse disponibilizado e mantido pelo Google. Ele possui integração direta com o SDK do Android e diversas funcionalidades, como o designer de

interfaces através de uma ferramenta visual, depuração de aplicações, e também assinatura de aplicações através de certificados.

- **Astah**: É uma ferramenta que permite a manipulação de diversos modelos de diagramas para modelagem de software baseado em UML. Ele possui diversos recursos que tornam o trabalho mais prático.

- **Balsamiq**: É uma ferramenta proprietária que permite a criação de diagramas de interface de maneira prática e intuitiva. Ela possui diversos componentes pré-definidos que podem ser utilizados na criação dos diagramas.

3.1 Método

O desenvolvimento do sistema para análise postural foi dividido em etapas de acordo com o modelo sequencial de (PRESSMAN, 2006). Essas etapas são:

- **Requisitos**: O levantamento de requisitos foi realizado através da análise das funcionalidades existentes na ferramenta SOAP, conforme a Figura 11

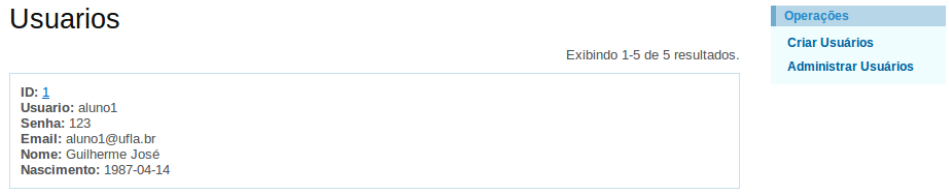


Figura 11 Exemplo - Coleta de Requisitos

Requisitos identificados

  - Manutenção de alunos
  - Listar alunos

- **Análise**: Com os principais requisitos coletados, uma análise sobre eles foi realizada buscando identificar os principais elementos que compõem determinado requisito e a maneira como cada requisito interage entre si.

  Exemplo de Análise de Requisitos:

  Listar Alunos

  - A lista deve apresentar informações importantes sobre o aluno
  - A manutenção de alunos deve estar acessível a partir da lista de alunos

  Através disto o diagrama de casos de uso foi criado para exemplicar melhor a interação de cada requisito e permitir a melhor vizualização do fluxo de uso da aplicação.

- **Projeto**: A partir da análise e coleta dos requisitos o projeto do software foi definido e os diagramas de interface foram criados. Com a criação dos diagramas de interface foi possível vizualizar os principais elementos da interface da ferramenta de forma a torná-la intuitiva e prática para o usuário.

- **Desenvolvimento**: O desenvolvimento foi realizado utilizando as ferramentas apresentadas anteriormente através de prototipação.

- **Teste**: Os testes realizados na aplicação, foram testes de caixa preta(testes funcionais), no qual a medida que a aplicação era desenvolvida seus recursos eram testados.

4 Desenvolvimento

4.1 Modelagem do Software

Esta seção apresenta a análise para o sistema móvel de análise postural, abordando requisitos funcionais e não funcionais que definem as principais funcionalidades do sistema, seu diagrama de casos de uso e seu diagrama de interfaces.

4.1.1 Requisitos Funcionais

Os requisitos funcionais foram coletados com base no sistema de análise postural online já existente desenvolvido através de um projeto de iniciação científica da UFLA, SOAP. Através destes é possível visualizar as principais funcionalidades do software. Sendo elas, a manutenção de indivíduos no sistema, (aqueles que serão avaliados), a realização da análise postural através de imagens e a geração de um relatório postural.

A seguir são apresentados os principais requisitos funcionais:

Requisito Funcional		Requisitos não funcionais		
Nome	**Descrição**	**Nome**	**Tipo**	**Obrigatório**
Manutenção de Aluno	O Sistema deve inserir, atualizar, remover e listar alunos cadastrados	Validar campos	Interface	X
		Alerta de Feedback	Interface	X
		Persistência dos dados	Persistência	X
Manutenção de Avaliações	O sistema deve permitir diversas avaliações por aluno, bem como a possibilidade de remoção e atualização das imagens relacionadas a avaliação	Alerta de feedback	Interface	X
		Persistência dos dados	Persistência	X
Captura de Fotos	O sistema deve permitir a captura de imagens através de camêra do aparelho	Alerta de Feedback	Interface	X
		Persistência dos dados	Persistência	X
Análise das Fotos	O sistema deve permitir a análise postural através das fotos	Alerta de Feedback	Interface	X
		Persistência dos dados	Persistência	X
Geração de Relatórios	O sistema deve permitir a geração de relatórios através da análise postural das fotos	Alerta de Feedback	Interface	X
		Persistência dos dados	Persistência	X

Figura 12 Requisitos

4.1.2 Diagrama de Casos de Uso

Conforme os requisitos funcionais o seguinte diagrama de casos de uso foi criado. Através do casos de uso criados foi possível definir o fluxo de uso da aplicação, no qual os diagramas de interface foram feitos posteriormente.

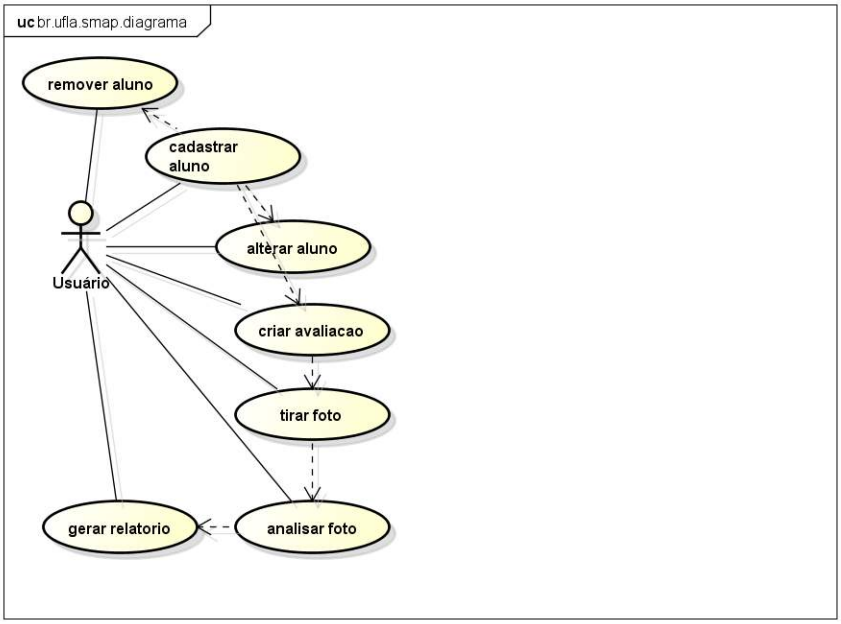


Figura 13 Diagrama de Casos de Uso

4.1.3 Diagramas de Interface

Através da coleta e análise de requisitos, foram criados diagramas de interface com objetivo de exemplificar o funcionamento do software.

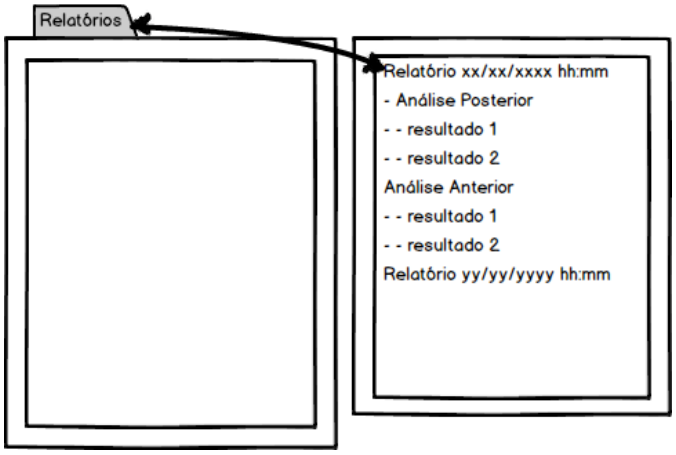


Figura 14 Diagrama de Interface - Relatórios

Os diagramas de interface criados, possibilitaram uma melhor vizualização do fluxo de utilização do software, de forma a identificar os elementos mais importantes a serem desenvolvidos no software e a maneira como o usuário interage com eles.

4.2 Execução e Depuração do Aplicativo

Para o desenvolvimento deste aplicativo, foi utilizado o Moto G, com a versão 4.3 do Android, com o intuito de realizar testes reais da aplicação, principalmente pelo fato do Emulador SDK Android não possuir um suporte simplificado de acesso à câmera e possibilitar apenas ocorrências fictícias do envio de coordenadas GPS.

A execução e depuração do aplicativo do dispositivo podem ser feitas apenas conectando o dispositivo ao computador, permitindo que seus drivers sejam instalados. Ao executar a aplicação através da IDE, o aparelho móvel é detectado, apresentando as opções de executar a aplicação através do emulador ou do dispositivo móvel.

Também através do aparelho móvel é possível realizar a depuração do código da mesma forma como realizada no emulador. Adicionando pontos de parada, com a finalidade de encontrar erros de codificação.

Esse modo é muito útil, devido à possibilidade de simular um ambiente mais próximo do real, permitindo solucionar problemas não encontrados no ambiente do emulador.

4.3 Testes Funcionais

Os testes foram realizados à medida que cada funcionalidade era desenvolvida. Eles foram realizados para verificar se o funcionamento do software ocorria como o esperado e se cada nova funcionalidade desenvolvida não gerava erros em funcionalidades desenvolvidas anteriormente.

Através dos testes diversos problemas foram identificados, possibilitando realizar melhorias e correções, como a utilização de um ORM para a persistência dos dados e a correção de um problema ao carregar imagens salvas na aplicação.

O problema do carregamento de imagens ocorria devido ao fato do dispositivo móvel possuir recursos limitados em relação ao emulador utilizado, fazendo com que a imagem não pudesse ser carregada e utilizada na aplicação.

A solução encontrada foi utilizar um método que possibilite a aplicação carregar a imagem utilizando uma menor quantidade de memória, evitando que a aplicação apresentasse um erro inesperado para o usuário.

5 Implementação

Para o desenvolvimento do aplicativo móvel, foi utilizada como base a versão 4.1 do sistema Android, esta sendo chamada de *Jelly Bean*. A escolha dessa versão foi feita devido a determinados recursos fornecidos pela API do Android estarem disponíveis a partir desta versão.

Na programação Android, qualquer versão superior do sistema em relação a versão no qual o aplicativo foi desenvolvido é suportada. Sendo assim qualquer dispositivo que rode uma versão superior a 4.1 do Android irá suportar a aplicação.

5.1 Interface

Para a criação das interfaces visuais de cada tela, foi utilizado um assistente visual contido no SDK, que é baseado em XML, em conjunto com o plug-in ADT na IDE eclipse. Através desse assistente, a criação de interfaces é dada de maneira interativa, *drag and drop*, organizando os componentes na tela de acordo com o necessário, enquanto o código XML é gerado automaticamente. A criação de telas é exemplificada na Figura 15.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:text="@string/nome"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/edt_nome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView2"
        android:ems="10"
        android:inputType="textPersonName" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/edt_nome"
```

Figura 15 Código de layout - XML

5.2 Permissões

Para a utilização de recursos externos a aplicação, como câmera, GPS, armazenamento no cartão de memoria, é necessário adicionar permissões no arquivo *AndroidManifest.xml*, contido em todos os aplicativos desenvolvidos para Android. Dessa forma sempre que um aplicativo é instalado no Android os recursos a serem utilizados são informados durante a instalação e esta só é concluída após a confirmação do usuário.

Esse método de segurança evita que aplicações mal intencionadas utilizem recursos do aparelho sem o conhecimento do proprietário, sendo exemplificada na Figura 16.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.ufla.smap"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="14" />

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <meta-data
            android:name="DATABASE"
            android:value="SMAP.db" />

        <activity
            android:name="br.ufla.smap.activity.PrincipalPessoaActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

Figura 16 Android manifest

5.3 Captura da Foto

Para realizar a captura de fotos o aplicativo utiliza a própria funcionalidade do dispositivo, chamando esta através de um *Intent* , encarregado de capturar a imagem. Para esta chamada apenas o caminho onde a imagem irá ser salva é passado.

```
1 Uri uriSavedImage = Uri.fromFile(new File("/sdcard/" +
      AvaliacaoSingleton.getInstance().getPessoaBean().
      getNome() + "_frontal.png"));
2 Intent intent = new Intent(MediaStore.
      ACTION_IMAGE_CAPTURE);
3 intent.putExtra("output", uriSavedImage);
      localizacao.setFoto(AvaliacaoSingleton.getInstance().
      getAvaliacaoSelecionada().getImgAnterior());
4 startActivityForResult(intent, TIRAR_FOTO);
```

O código apresentado específica onde a foto será salva, no caso no cartão SD do dispositivo, bem como com qual nome ela será salva realizando uma chamada a camêra padrão do Android através de um *intent*.

5.4 GPS

Para obtenção das coordenadas do GPS, foram utilizados dois recursos nativos do SDK Android, *Location Manager* e *Location Listener*, destinados ao tratamento do recurso GPS.

O recurso *LocationManager* é responsável por gerenciar o serviço do sistema de GPS, através deste é possível adicionar *Listeners* de status do serviço, como alerta de proximidade de pontos de referencia. Também através deste é possível definir a atualização da posição através do método requestLocationUpdates() que permite definir o tempo das atualizações e também se necessário que a atualização ocorra após uma distancia mínima percorrida.

5.5 Persistência de Dados

A persistência dos dados foi feita utilizando o ORM Lite. Através desse ORM é possível persistir objetos no banco de dados sem a necessidade de escrever SQLs. Ele possibilita a ligação do Objeto criado com um objeto criado no banco de dados, mapeado através de anotações.

Também utilizando este framework é possível criar um factory que gerencie as conexões com o banco de dados de forma simples e eficiente.

5.6 Rotação de Linha

Uma das funções mais importantes do software é poder determinar o desvio de postura de um individuo em relação à postura correta. Para que esse desvio pudesse ser visualizado foi necessário criar uma função capaz de rotacionar uma reta em torno de seu próprio eixo. O código a seguir apresenta parte dessa função.

```
1    float endXPos2 = (float) (centerX + (Math.cos(Math.
        toRadians(sum)) * (endXPos - centerX) + Math.sin(
        Math.toRadians(sum)) * (endYPos - centerY)));
2    float endYPos2 = (float) (centerY + (-Math.sin(Math.
        toRadians(sum)) * (endXPos - centerX) + Math.cos(
        Math.toRadians(sum)) * (endYPos - centerY)));
```

Esse método é encarregado de realizar a rotação e o desenho das retas, que possibilitam a análise postural. Sempre quando chamado o método irá receber os pontos iniciais que determinam uma reta no eixo cartesiano.

Através desses pontos um ponto médio será definido, e duas retas serão desenhadas a partir deste ponto médio. Através disto o usuário irá vizualizar a reta ser desenhada como uma única reta que gira em torno do próprio eixo, embora duas retas estejam sendo desenhadas simultaneamente.

5.7 Manutenção da Sessão

Durante a execução da aplicação o software necessita que algumas informações sejam mantidas, entretanto tais informações têm de ser descartadas quando a execução é interrompida. Para manter essas informações durante a execução do software, foi criada uma classe utilizando o padrão

Singleton. Essa classe existe durante toda a execução do software, entretanto é instanciada apenas uma única vez no inicio da execução do software.

```
1  public static AvaliacaoSingleton getInstance() {
2      if (sessao == null) {
3      sessao = new AvaliacaoSingleton();
4    }
5    return sessao;
6  }
```

5.8 Fragments

A api do Android possui um grupo de componentes chamados *Fragments*. Esses componentes possibilitam ao desenvolvedor criar interfaces mais customizáveis e que possuem comportamento adequado em diversos dispositivos.

Fragments permitem que cada porção da interface possua um comportamento de execução diferente. No aplicativo foi utilizado o Fragment do tipo *DialogFragment*, que permite a criação de alertas com comportamento customizado.

5.9 Adapter

Adapter é um grupo de componentes que possibilita manipular e apresentar um conjunto de dados de forma simples e concreta.

Essa manipulação é feita forma fácil devido ao Adapter tratar as informações através de um método interno, necessitando apenas a construção de uma classe definindo o que irá ser apresentado.

No aplicativo foi utilizada a classe *BaseExpandableListAdapter*, que consiste em uma lista expansível, que quando clicada apresenta um grupo de informações.

6 Resultados

O aplicativo Android possui três funcionalidades principais,sendo elas, **cadastro de pessoas** que podem ser avaliadas, **análise postural** através de fotos e a **geração de um relatório postural** a partir da análise.

A funcionalidade **cadastrar pessoas** insere novas pessoas na base de dados do sistema, mantendo informações relevantes sobre esta. A funcionalidade **analisar foto** permite ao usuário realizar uma análise postural de uma imagem obtida pelo aplicativo e a funcionalidade de **gerar relatório** permite ao usuário gerar um relatório de análise postural com base na análise da imagem realizada anteriormente.

6.1 Cadastrar Pessoas

Ao acessar o aplicativo o usuário irá encontrar uma TAB com duas opções, cadastro e listagem de pessoas, conforme Figura 17.

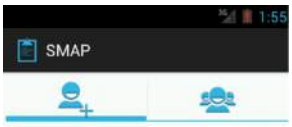


Figura 17 TAB inicial

Conforme a Figura acima, a primeira opção apresenta a tela de cadastro de pessoas, no qual o usuário poderá inserir uma nova pessoa, atualizar ou remover uma pessoa já cadastrada. A tela de cadastro contem os campos nomes, sexo, data de nascimento, altura e peso e também os botões gravar dados e remover.

Ao clicar no botão gravar uma validação dos campos obrigatórios será realizada. Caso algum campo esteja inválido uma mensagem de erro será apresentada ao usuário conforme a Figura 18.

Figura 18 Tela de Cadastro de Pessoas

Caso todos os campos estejam válidos uma nova pessoa é inserida no sistema e um alerta de notificação é apresentado ao usuário assim como uma mensagem de notificação na barra de notificações do Android conforme a Figura 19.

Figura 19 Mensagem de Alerta - Cadastro completado com sucesso

A segunda TAB apresenta a tela de listagem de pessoas em ordem alfabética. Ao clicar em uma pessoa a lista vai se expandir apresentando as informações referentes a pessoa e se esta já foi avaliada ou não. Clicando em qualquer uma dessas informações uma *popup* contendo duas opções irá ser apresentada ao usuário. Através dessa *popup* o usuário pode editar a pessoa ou iniciar/visualizar uma avaliação existente para aquela pessoa conforme a Figura 20.

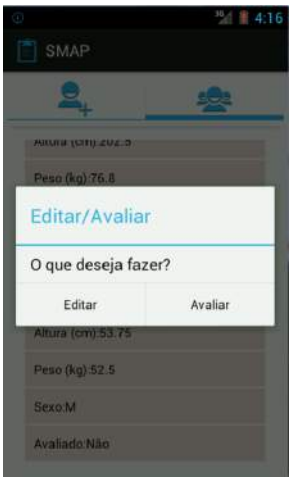


Figura 20 Tela de Listagem de Pessoas

6.2 Análise Postural por Linhas

Quando o usuário escolhe a opção avaliar no *popup* apresentado anteriormente o usuário vai ser redirecionado para outra tela. Essa tela apresenta uma TAB contendo 5 opções, sendo uma para listagem de avaliações e as demais para a captura de imagens e análise postural dos diversos tipos de imagens a serem avaliados. A tela de listagem das avaliações apresenta, informações como a data e hora das avaliações e quais imagem foram capturadas para aquela avaliação conforme a Figura 21.

Figura 21 TAB de Avaliação

As TABs referentes à imagem são compostas por uma função de captura de fotos e avaliação. Ao clicar na imagem o usuário irá ser direcionado para a tela de captura de fotos. Clicando no botão Tirar foto, o aplicativo irá executar a função câmera nativa do dispositivo, possibilitando ao usuário capturar uma foto que irá ser utilizada na avaliação postural da pessoa. Ao tirar uma foto o usuário pode escolher entre manter aquela foto ou obter uma nova imagem.

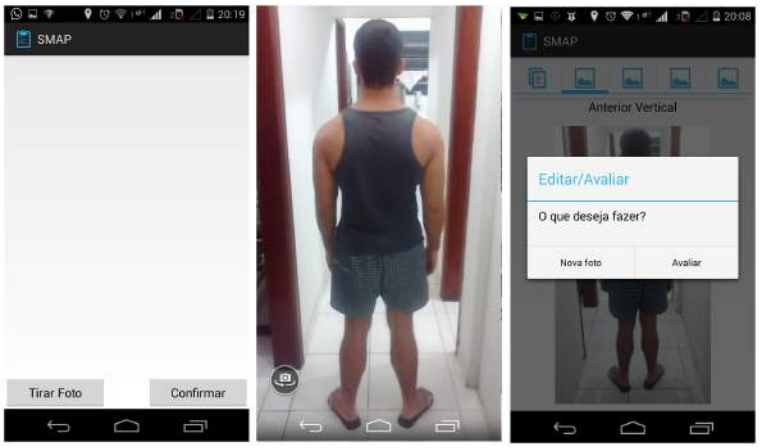


Figura 22 Sequência de Captura de Fotos

Na função de avaliar fotos o usuário pode selecionar diversas linhas e executar operações sobre estas conforme a Figura 23. Quando uma linha é selecionada ela aparece como verde, e sua rotação em relação à linha original é apresentada em vermelho, possibilitando ao usuário visualizar a interação com a imagem de uma maneira mais amigável.

A rotação daquela linha é apresentada numa caixa de texto centralizada entre os botões de rotação. Ao clicar no botão Gerar avaliação o usuário, irá gerar uma nova avaliação para aquela imagem, com base nas operações realizadas sobre a imagem.

Figura 23 Avaliação postural

6.3 Relatório de Análise Postural

Posteriormente para gerar um novo relatório o usuário, escolhe diversas opções avançadas relacionadas à postura. Após isto um relatório com base na avaliação da imagem e a escolha dessas opções é gerado. A Figura 24 apresenta quais as opções relacionadas aos padrões de postura para gerar o relatório podem ser escolhidas.

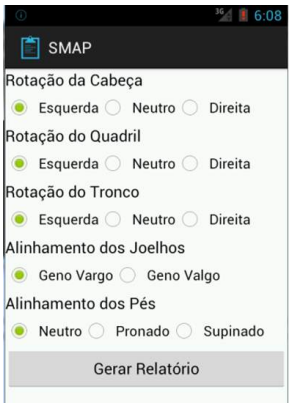


Figura 24 Opções Relatório

A Figura 25 apresenta o relatório final com base na análise postural por linhas.

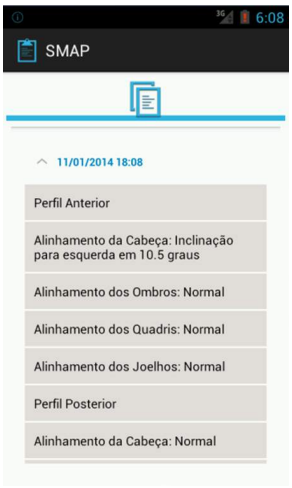


Figura 25 Relatório de Avaliação Postural

O processo de análise postural através das linhas (apresentado anteriormente) é realizado através de padrões de desvios já definidos (opções de relatório) em conjunto com os desvios apresentados através da análise da imagem.

Cada rotação de linha é dada de maneira positiva e negativa, assim como a posição em que determinadas partes do nosso corpo se encontram, como por exemplo os pés (pronado, supinado).

Através dessas duas informações uma *factory* genérica gera o relatório final.

6.4 Comparativo com a ferramenta SOAP

Com base no levantamento de requisitos, análise de requisitos e desenvolvimento do Software foi possível identificar as principais semelhanças entre as duas ferramentas conforme a Figura 26.

SOAP	**SMAP**
Manutenção de Pessoas(cadastro,alteração,remoção)	X
Lista de Pessoas	X
Carregar Imagens	A ferramenta carrega apenas imagens vinculadas a uma avaliação específica.
Manutenção de análises(cadastro,alteração,remoção)	X
Lista de análises	A ferramenta não permite a vizualização global de todas as análises realizadas, estas estão disponíveis a partir da pessoa analisada.
Análise Postural por linhas	X
Geração de Relatórios	X

Figura 26 SOAP x SMAP

A partir dessas semelhanças foram identificados os principais pontos fortes e fracos da ferramenta SMAP.

Pontos Fortes

- **Histórico de Avaliações**: o sistema mantém um histórico de todas as avaliações realizadas para uma determinada pessoa, possibilitando

uma melhor visibilidade do progresso realizado a cada avaliação postural.

- **Histórico de Imagens**: o sistema mantém todas as imagens existentes para uma pessoa.
- **Disponibilidade da ferramenta**: por ser um aplicativo móvel, ele pode ser utilizado por qualquer pessoa que tenha um disposito com o Android a partir da versão 4.1.

Pontos Fracos

- **Avaliação vinculada a uma pessoa**: as avaliações realizadas no sistema só estão disponíveis a partir de uma pessoa cadastrada, impedindo ao usuário ter uma perspectiva geral das avaliações realizadas.
- **Imagens salvas na memória do dispositivo**: as fotos tiradas pelo sistema são salvas diretamente na mémoria do dispositivo, possibilitando que elas possam ser alteradas e removidas por outros programas.

7 CONCLUSÃO

O sistema desenvolvido tem como finalidade realizar uma análise postural através de imagens capturadas por dispositivos móveis que utilizam Android, gerando um relatório postural e podendo analisar o progresso da pessoa analisada.

A análise e criação da aplicação foram feitas através da utilização de conceitos adquiridos ao longo da graduação em diversas disciplinas, como Banco de Dados, Interface Homem Máquina, Engenharia de Software e Programação Orientada a Objetos.

Durante o desenvolvimento da aplicação um estudo da plataforma Android foi realizado, mostrando algumas diferenças da plataforma, no qual a principal diferença notada foi o fato da sua *API*, canvas, ser inferior aquela encontrada no Java, tornando o desenvolvimento da aplicação mais díficil do que o esperado.

Através deste estudo também foi possível utilizar e entender como recursos disponíveis na maioria dos smarthphones, como GPS e a Câmera funcionam, bem como a facilidade que o Android trás para sua utilização.

A forma de trabalhar com o desenvolvimento de uma aplicação Android é muito similar ao desenvolvimento de qualquer outra aplicação Java, pois a síntese é a mesma e o Android possui grande parte das APIS disponíveis no Java. Bastando apenas um estudo e entendimento sobre como funcionam as classes exclusivas do Android.

7.1 Problemas Encontrados

Durante o desenvolvimento do trabalho foram encontradas algumas dificuldades, como a dificuldade em encontrar material relacionado à classe

drawble do Android, responsável por representar aquilo que pode ser desenhado na tela. O entendimento dessa classe foi importante, devido à necessidade de criar o recurso mais importante da aplicação.

Outro problema foi a dificuldade em desenvolver uma interface amigável para o usuário, devido a falta de experiência no desenvolvimento de aplicações móveis e como as informações e o fluxo de uma aplicação deve ser tratado, buscando minimizar as funcionalidades, tornado-as o mais práticas possíveis.

7.2 Trabalhos Futuros

Para trabalhos futuros serão realizados algumas melhorias na aplicação, bem como a possibilidade de uma análise postural através de pessoas em movimento. Uma maior possibilidade de análises e um software customizável conforme as necessidades de uso do usuário.

Também como trabalho futuro existe possibilidade de comercialização do software. Para isto será necessário realizar uma análise da usabilidade da aplicação em torno dos usuários do software, possibilitando assim identificar melhorias na usabilidade, corrigir possíveis erros e projetar novas funcionalidades com base nas necessidades destes.

REFERÊNCIAS

ARIMA, K. *Qual será o seu próximo celular?: Por que o Google, a Apple e a Palm têm cada vez mais chances de disputar essa resposta*. 2009. Disponível em: <http://info.abril.com.br/arquivo/2009/fev.shtml>. Acesso em: 20.12.2013.

AVALIACAO. *Avaliação Física*. 2014. Disponível em: <http://www.avaliacaofisica.com.br/si/site/0210>. Acesso em: 20.12.2013.

BATIZ, E. C.; SANTOS, A. F. dos; ANZARDO, O. E. *A postura no trabalho dos operadores de checkout de supermercados: uma necessidade constante de análises*. 2009. Disponível em: <http://www.scielo.br/pdf/prod/v19n1/12.pdf>. Acesso em: 20.12.2013.

BRACCIALLI, L. M. P.; VILARTA, R. Aspectos a serem considerados na elaboração de programas de prevenção e orientação de problemas posturais. 2000. Disponível em: <http://portalsaudebrasil.com/artigospsb/reumato092.pdf>. Acesso em: 20.12.2013.

DALVIKVM. *Dalvik virtual machine*. 2013. Disponível em: <https://code.google.com/p/dalvik>. Acesso em: 20.12.2013.

FERREIRA, F. P. M. Avaliação postural dos idosos de porto alegre-rs com o uso da técnica de moiré de sombra. 2008. Disponível em: <http://tardis.pucrs.br/dspace/bitstream/10923/3632/1/000404768-Texto%2bCompleto-0.pdf>. Acesso em: 20.12.2013.

GARGENTA, M. *Learning Android*. [S.l.]: O Reilly Media, 2011.

GOOGLE. *Application fundamentals*. 2013. Disponível em: <http://developer.android.com/guide/topics/fundamentals.html>. Acesso em: 20.12.2013.

_____. *Designing for performance*. 2013. Disponível em: <http://developer.android.com/guide/practices/design/performance.html>. Acesso em: 20.12.2013.

_____. *What is Android?* 2013. Disponível em: <http://developer.android.com/guide/basics/what-is-android.html>. Acesso em: 20.12.2013.

JR, A. A. Estilo de vida e desordem na coluna lombar: uma resposta dos componentes da aptidão física relacionada à saúde. *Revista brasileira de atividade física e saúde*, v. 1, n. 1, p. 36–56, 1995.

KENDALL, F. P.; MCCREARY, E. K.; PROVANCE, P. G. *Músculos: Provas e Funções*. [S.l.]: São Paulo: Manole, 1986.

LECHETA, R. *Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK*. [S.l.]: Novatec, 2009.

MAGEE, D. J. *Avaliação musculoesquelética*. [S.l.]: Barueri: Manole, 2002.

MEIER, R. *Professional Android Application Development*. [S.l.]: Indianapolis: Wiley Publishing, 2009.

NORIEGA, C. E. L. *Desenvolvimento de um programa computacional para avaliação postural de código aberto e gratuito*. Tese (Doutorado) — Universidade de São Paulo, 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/47/47135/tde-15062012-105633/pt-br.php>. Acesso em: 20.12.2013.

OHA. *Android Overview*. 2013. Disponível em: <http://www.openhandsetalliance.com/android_overview.html>. Acesso em: 20.12.2013.

_____. *The Developers Guide*. 2013. Disponível em: <http://developer.android.com/index.html>. Acesso em: 20.12.2013.

PRESSMAN, R. *Engenharia de software*. [S.l.]: São Paulo, SP: McGraw Hill, 2006.

SANTOS, C. I. S. C.; BRAGA, A. B. N.; SAAD, V. P.; RIBEIRO, I. A. B.; CONTI, M.; OBERG, P. B. M.; D, T. Ocorrência de desvios posturais em escolares do ensino público fundamental de jaguariúna, são paulo. 2009. Disponível em: <http://www.scielo.br/pdf/rpp/v27n1/12.pdf>. Acesso em: 20.12.2013.

SAPO. *SAPO - Software de Análise Postural*. 2005. Disponível em: <http://web.archive.org/web/20080701030305/http://sapo.incubadora.fapesp.br/portal>. Acesso em: 20.12.2013.

_____. _____. 2005. Disponível em: <http://web.archive.org/web/20080630195003/http://sapo.incubadora.fapesp.br/portal/ajuda/OProtocoloSAPODeMarca_c3_a7_c3_a3oDePontos/>. Acesso em: 20.12.2013.

_____. _____. 2014. Disponível em: <https://code.google.com/p/sapo-desktop/>. Acesso em: 20.12.2013.

SMITHL, L. K.; WEISS, E. L.; LEHMKUHL, L. D. *Cinesiologia Clínica de Brunnstrom*. [S.l.]: São Paulo Manole, 1997.

TEIXEIRA, C. S.; KOTHE, F. *Avaliação da postura corporal de violinistas e violistas*. 2012. Disponível em: <http://www.scielo.br/scielo.php?pid=S1517-75992012000200014&script=sci_arttext&tlng=es>. Acesso em: 20.12.2013.

TOSCANO, J. J. d. O.; EGYPTO, E. P. d. A influência do sedentarismo na prevalência de lombalgia. *Revista Brasileira de Medicina do Esporte*, 2001. Disponível em: <http://www.scielo.br/scielo.php?pid=S1517-86922001000400004&script=sci_arttext>. Acesso em: 20.12.2013.

UNIMEDRIO. *Problemas da Má Postura Corporal*. 2013. Disponível em: <http://www.unimedrio.com.br/unimed/filesmng.nsf/F5AC13737C26C3AB832579B4008050D9/$File/Postura.pdf>. Acesso em: 20.12.2013.

APÊNDICE

APÊNDICE A - Primeiro Apêndice

Classe - ”LinhaHorizontal.java”

```
1  package br.ufla.smap.abstracts.linhas;
2  import android.graphics.Bitmap;
3  import android.graphics.Canvas;
4  import android.graphics.Color;
5  import android.graphics.Paint;
6  import br.ufla.smap.abstracts.DrawLineImplements;
7
8  public class LineHorizontal implements
       DrawLineImplements {
9
10   private Integer startXPos;
11
12   private Integer startYPos;
13
14   private Integer endXPos;
15
16   private Integer endYPos;
17
18   private Integer comprimento;
19
20   private Integer angulo = 0;
21
22   private Integer lineColor;
23
24   @Override
```

```
25  public void desenharLinha(Integer startXPos, Integer
        startYPos, Integer endXPos, Integer endYPos, Bitmap
         imagem, Integer comprimento) {
26    this.startXPos = startXPos;
27    this.startYPos = startYPos;
28    this.endXPos = endXPos;
29    this.endYPos = endYPos;
30    this.comprimento = comprimento / 2;
31    Canvas canvas = new Canvas(imagem);
32    Paint paint = new Paint();
33    this.changeToBlue();
34    paint.setColor(lineColor);
35    canvas.drawLine(this.startXPos, this.startYPos, this
          .endXPos, this.endYPos, paint);
36  }
37
38  @Override
39  public void redesenharLinha(Bitmap imagem) {
40    Canvas canvas = new Canvas(imagem);
41    Paint paint = new Paint();
42    this.changeToGreen();
43    paint.setColor(lineColor);
44    canvas.drawLine(this.startXPos, this.startYPos, this
          .endXPos, this.endYPos, paint);
45  }
46
47  @Override
48  public void moverParaEsquerda(Bitmap imagem) {
49    Canvas canvas = new Canvas(imagem);
50    Paint paint = new Paint();
```

```
51      this.startXPos -= 1;
52      this.endXPos -= 1;
53      paint.setColor(this.lineColor);
54      canvas.drawLine(this.startXPos, this.startYPos, this
            .endXPos, this.endYPos, paint);
55
56    }
57
58    @Override
59    public void moverParaDireita(Bitmap imagem) {
60      Canvas canvas = new Canvas(imagem);
61      Paint paint = new Paint();
62      this.startXPos += 1;
63      this.endXPos += 1;
64      paint.setColor(this.lineColor);
65      canvas.drawLine(this.startXPos, this.startYPos, this
            .endXPos, this.endYPos, paint);
66    }
67
68    @Override
69    public void moverParaCima(Bitmap imagem) {
70      Canvas canvas = new Canvas(imagem);
71      Paint paint = new Paint();
72      this.startYPos -= 1;
73      this.endYPos -= 1;
74      paint.setColor(this.lineColor);
75      canvas.drawLine(this.startXPos, this.startYPos, this
            .endXPos, this.endYPos, paint);
76    }
77
```

```
78  @Override
79  public void moverParaBaixo(Bitmap imagem) {
80    Canvas canvas = new Canvas(imagem);
81    Paint paint = new Paint();
82    this.startYPos += 1;
83    this.endYPos += 1;
84    paint.setColor(this.lineColor);
85    canvas.drawLine(this.startXPos, this.startYPos, this
          .endXPos, this.endYPos, paint);
86  }
87
88
89
90  @Override
91  public void rotacionar(Bitmap imagem, Double sum,
       Integer startXPos, Integer startYPos, Integer
       endXPos, Integer endYPos) {
92    this.startXPos = startXPos;
93    this.startYPos = startYPos;
94    this.endXPos = endXPos;
95    this.endYPos = endYPos;
96
97    Canvas canvas = new Canvas(imagem);
98    Paint paint = new Paint();
99    float centerX = (startXPos + endXPos) / 2;
100   float centerY = (startYPos + endYPos) / 2;
101
102   float startXPos2 = (float) (centerX + (Math.cos(Math
          .toRadians(sum)) * (startXPos - centerX) + Math.
```

```
        sin(Math.toRadians(sum)) * (startYPos - centerY))
        );
103     float startYPos2 = (float) (centerY + (-Math.sin(
        Math.toRadians(sum)) * (startXPos - centerX) +
        Math.cos(Math.toRadians(sum)) * (startYPos -
        centerY)));
104
105     float endXPos2 = (float) (centerX + (Math.cos(Math.
        toRadians(sum)) * (endXPos - centerX) + Math.sin(
        Math.toRadians(sum)) * (endYPos - centerY)));
106     float endYPos2 = (float) (centerY + (-Math.sin(Math.
        toRadians(sum)) * (endXPos - centerX) + Math.cos(
        Math.toRadians(sum)) * (endYPos - centerY)));
107
108     paint.setColor(this.lineColor);
109     canvas.drawLine(startXPos2, startYPos2, endXPos2,
        endYPos2, paint);
110   }
111
112   @Override
113   public Bitmap apagarLinha() {
114     return Bitmap.createBitmap(100, 100, Bitmap.Config.
        ARGB_4444);
115   }
116
117   @Override
118   public void changeToBlue() {
119     this.lineColor = Color.BLUE;
120
121   }
```

```
122
123   @Override
124   public void changeToRed() {
125     this.lineColor = Color.RED;
126
127   }
128
129   @Override
130   public void changeToGreen() {
131     this.lineColor = Color.GREEN;
132
133   }
134
135   @Override
136   public void lineSelected(Bitmap imagem) {
137     Canvas canvas = new Canvas(imagem);
138     Paint paint = new Paint();
139     paint.setColor(this.lineColor);
140     canvas.drawLine(this.startXPos, this.startYPos, this
          .endXPos, this.endYPos, paint);
141
142   }
143
144   public Integer getStartXPos() {
145     return startXPos;
146   }
147
148   public void setStartXPos(Integer startXPos) {
149     this.startXPos = startXPos;
150   }
```

```
151
152   public Integer getStartYPos() {
153     return startYPos;
154   }
155
156   public void setStartYPos(Integer startYPos) {
157     this.startYPos = startYPos;
158   }
159
160   public Integer getEndXPos() {
161     return endXPos;
162   }
163
164   public void setEndXPos(Integer endXPos) {
165     this.endXPos = endXPos;
166   }
167
168   public Integer getEndYPos() {
169     return endYPos;
170   }
171
172   public void setEndYPos(Integer endYPos) {
173     this.endYPos = endYPos;
174   }
175
176   public Integer getComprimento() {
177     return comprimento;
178   }
179
180   public void setComprimento(Integer comprimento) {
```

```
181     this.comprimento = comprimento;
182   }
183
184   public Integer getAngulo() {
185     return angulo;
186   }
187
188   public void setAngulo(Integer angulo) {
189     this.angulo = angulo;
190   }
191
192   public Integer getLineColor() {
193     return lineColor;
194   }
195
196   public void setLineColor(Integer lineColor) {
197     this.lineColor = lineColor;
198   }
199
200 }
```

Classe - ”GpsService.java”

```
1 package br.ufla.smap.service;
2
3 import java.util.Date;
4
5 import android.content.Context;
6 import android.location.LocationListener;
7 import android.location.LocationManager;
8 import br.ufla.smap.activity.PrincipalPessoaActivity;
```

```
9  import br.ufla.smap.bean.Localizacao;
10 import br.ufla.smap.listeners.MyLocationListener;
11
12 public class GpsService {
13
14   private static GpsService gpsService;
15   private static LocationManager mLocManager;
16   private static LocationListener mLocListener;
17   private static PrincipalPessoaActivity activity;
18
19   public static boolean createGpsService(Context context
       , PrincipalPessoaActivity act) {
20     activity = act;
21     try {
22       mLocManager = (LocationManager) activity.
           getSystemService(Context.LOCATION_SERVICE);
23       mLocListener = new MyLocationListener(context);
24
25       return true;
26     } catch (Exception exc) {
27       return false;
28     }
29   }
30
31   public static GpsService getInstance() {
32     if (gpsService == null)
33       return new GpsService();
34     else
35       return gpsService;
36   }
```

```
37
38   public Localizacao getLocalizacao() {
39     mLocManager.requestLocationUpdates(LocationManager.
         GPS_PROVIDER, 20000,100, mLocListener);
40     if (mLocManager.isProviderEnabled(LocationManager.
         GPS_PROVIDER)) {
41       Localizacao localizacao = new Localizacao();
42       localizacao.setLongitude(MyLocationListener.
           longitude);
43       localizacao.setLatitude(MyLocationListener.
           latitude);
44       localizacao.setAltitude(MyLocationListener.
           altitude);
45       localizacao.setDataGps(new Date());
46       return localizacao;
47     } else {
48       return null;
49     }
50   }
51 }
```

Classe - ”MyLocationListener.java”

```
1 package br.ufla.smap.listeners;
2
3 import android.content.Context;
4 import android.location.Location;
5 import android.location.LocationListener;
6 import android.os.Bundle;
7 import android.widget.Toast;
8
```

```
9  public class MyLocationListener implements
       LocationListener {
10
11   public static double latitude;
12   public static double longitude;
13   public static double altitude;
14   private Context context;
15
16
17
18   public MyLocationListener(Context context) {
19     super();
20     this.context = context;
21   }
22
23   @Override
24   public void onLocationChanged(Location loc) {
25     loc.getLatitude();
26     loc.getLongitude();
27     loc.getAltitude();
28     latitude = loc.getLatitude();
29     longitude = loc.getLongitude();
30     altitude = loc.getAltitude();
31   }
32
33   @Override
34   public void onProviderDisabled(String provider) {
35     Toast.makeText(context, "Receptor GPS Desativado!",
           Toast.LENGTH_SHORT).show();
36   }
```

```
37
38    @Override
39    public void onProviderEnabled(String provider) {
40      Toast.makeText(context, "Receptor GPS Ativo!", Toast
          .LENGTH_SHORT).show();
41    }
42
43    @Override
44    public void onStatusChanged(String provider, int
        status, Bundle extras) {
45    }
46
47 }
```

Classe - ”CameraPreviewActivity.java”

```
1  package br.ufla.smap.activity;
2
3  import java.io.File;
4  import java.util.Random;
5
6  import android.annotation.SuppressLint;
7  import android.app.Activity;
8  import android.content.Intent;
9  import android.content.pm.ActivityInfo;
10 import android.graphics.Bitmap;
11 import android.graphics.BitmapFactory;
12 import android.net.Uri;
13 import android.os.Bundle;
14 import android.provider.MediaStore;
15 import android.view.View;
```

```
16 import android.view.View.OnClickListener;
17 import android.widget.Button;
18 import android.widget.ImageView;
19 import android.widget.Toast;
20 import br.ufla.smap.R;
21 import br.ufla.smap.bean.Localizacao;
22 import br.ufla.smap.enums.FotoEnum;
23 import br.ufla.smap.service.GpsService;
24 import br.ufla.smap.singleton.AvaliacaoSingleton;
25
26 @SuppressLint({ "ShowToast", "SdCardPath" })
27 public class CameraPreviewActivity extends Activity
       implements OnClickListener {
28   private ImageView img_foto;
29   private static final int TIRAR_FOTO = 1020394857;
30   private static final Random random = new Random();
31   private Button bt_click;
32   private Button bt_confirmar_camera;
33   private Localizacao localizacao;
34
35   /** Called when the activity is first created. */
36   @Override
37   public void onCreate(Bundle savedInstanceState) {
38     super.onCreate(savedInstanceState);
39     super.setRequestedOrientation(ActivityInfo.
           SCREEN_ORIENTATION_PORTRAIT);
40     setContentView(R.layout.camera_preview);
41
42     img_foto = (ImageView) findViewById(R.id.img_foto);
43
```

```
44      bt_click = (Button) findViewById(R.id.bt_click);
45      bt_click.setOnClickListener(this);
46      bt_confirmar_camera = (Button) findViewById(R.id.
            bt_confirmar_camera);
47      bt_confirmar_camera.setOnClickListener(this);
48      localizacao = GpsService.getInstance().
            getLocalizacao();
49
50
51  }
52
53  @SuppressLint("ShowToast")
54  @Override
55  protected void onActivityResult(int requestCode, int
        resultCode, Intent data) {
56    super.onActivityResult(requestCode, resultCode, data
          );
57    if (requestCode == TIRAR_FOTO) {
58      // Ok
59      if (resultCode == RESULT_OK) {
60        // Aqui pego a imagem
61        BitmapFactory.Options bmpFactoryOptions = new
              BitmapFactory.Options();
62        bmpFactoryOptions.inJustDecodeBounds = false;
63        // imageFilePath image path which you pass with
              intent
64        Bitmap bitmap = null;
65        if (AvaliacaoSingleton.getInstance().getTipoFoto
              () == FotoEnum.ANTERIOR) {
```

```
66            bitmap = BitmapFactory.decodeFile(
                  AvaliacaoSingleton.getInstance().
                  getAvaliacaoSelecionada().getImgAnterior(),
                   bmpFactoryOptions);
67            localizacao.setTipo(FotoEnum.ANTERIOR.name());
68            AvaliacaoSingleton.getInstance().
                  getAvaliacaoSelecionada().getLocalizacao().
                  add(localizacao);
69          } else if (AvaliacaoSingleton.getInstance().
                getTipoFoto() == FotoEnum.POSTERIOR) {
70            bitmap = BitmapFactory.decodeFile(
                  AvaliacaoSingleton.getInstance().
                  getAvaliacaoSelecionada().getImgPosterior()
                  , bmpFactoryOptions);
71            localizacao.setTipo(FotoEnum.POSTERIOR.name())
                  ;
72            AvaliacaoSingleton.getInstance().
                  getAvaliacaoSelecionada().getLocalizacao().
                  add(localizacao);
73          } else if (AvaliacaoSingleton.getInstance().
                getTipoFoto() == FotoEnum.ESQUERDA)
74            bitmap = BitmapFactory.decodeFile(
                  AvaliacaoSingleton.getInstance().
                  getAvaliacaoSelecionada().getImgEsquerda(),
                   bmpFactoryOptions);
75          else if (AvaliacaoSingleton.getInstance().
                getTipoFoto() == FotoEnum.DIREITA)
76            bitmap = BitmapFactory.decodeFile(
                  AvaliacaoSingleton.getInstance().
```

```
            getAvaliacaoSelecionada().getImgDireita(),
            bmpFactoryOptions);
77        // Seta ela no ImaView do Layout
78        img_foto.setImageBitmap(bitmap);
79        // Aqui posso salvar a foto se quizer.
80      } else if (resultCode == RESULT_CANCELED) {//
          Cancelou a foto
81        Toast.makeText(this, "Cancelou", Toast.
            LENGTH_SHORT);
82      } else { // Saiu da Intent
83        Toast.makeText(this, "Saiu", Toast.LENGTH_SHORT)
            ;
84      }
85
86    }
87  }
88
89  @SuppressLint("SdCardPath")
90  @Override
91  public void onClick(View v) {
92    if (v.getId() == R.id.bt_click) {
93      int id_foto = random.nextInt(10000);
94      if (AvaliacaoSingleton.getInstance().getTipoFoto()
            == FotoEnum.ANTERIOR) {
95        AvaliacaoSingleton.getInstance().
            getAvaliacaoSelecionada().setImgAnterior("/
            sdcard/" + AvaliacaoSingleton.getInstance().
            getPessoaBean().getNome()+id_foto+ "_frontal.
            png");
```

```
96          Uri uriSavedImage = Uri.fromFile(new File("/
                sdcard/" + AvaliacaoSingleton.getInstance().
                getPessoaBean().getNome()+id_foto+ "_frontal.
                png"));
97          Intent intent = new Intent(MediaStore.
                ACTION_IMAGE_CAPTURE);
98          intent.putExtra("output", uriSavedImage);
99          localizacao.setFoto(AvaliacaoSingleton.
                getInstance().getAvaliacaoSelecionada().
                getImgAnterior());
100         startActivityForResult(intent, TIRAR_FOTO);
101       } else if (AvaliacaoSingleton.getInstance().
              getTipoFoto() == FotoEnum.POSTERIOR) {
102         AvaliacaoSingleton.getInstance().
                getAvaliacaoSelecionada().setImgPosterior("/
                sdcard/" + AvaliacaoSingleton.getInstance().
                getPessoaBean().getNome()+id_foto + "_tras.
                png");
103         Uri uriSavedImage = Uri.fromFile(new File("/
                sdcard/" + AvaliacaoSingleton.getInstance().
                getPessoaBean().getNome() +id_foto+ "_tras.
                png"));
104         Intent intent = new Intent(MediaStore.
                ACTION_IMAGE_CAPTURE);
105         intent.putExtra("output", uriSavedImage);
106         localizacao.setFoto(AvaliacaoSingleton.
                getInstance().getAvaliacaoSelecionada().
                getImgPosterior());
107         startActivityForResult(intent, TIRAR_FOTO);
```

```
108         } else if (AvaliacaoSingleton.getInstance().
                getTipoFoto() == FotoEnum.ESQUERDA) {
109           AvaliacaoSingleton.getInstance().
                  getAvaliacaoSelecionada().setImgEsquerda("/
                  sdcard/" + AvaliacaoSingleton.getInstance().
                  getPessoaBean().getNome()+id_foto + "_esq.png
                  ");
110           Uri uriSavedImage = Uri.fromFile(new File("/
                  sdcard/" + AvaliacaoSingleton.getInstance().
                  getPessoaBean().getNome()+id_foto+ "_esq.png"
                  ));
111           Intent intent = new Intent(MediaStore.
                  ACTION_IMAGE_CAPTURE);
112           intent.putExtra("output", uriSavedImage);
113           localizacao.setFoto(AvaliacaoSingleton.
                  getInstance().getAvaliacaoSelecionada().
                  getImgEsquerda());
114           startActivityForResult(intent, TIRAR_FOTO);
115         } else if (AvaliacaoSingleton.getInstance().
                getTipoFoto() == FotoEnum.DIREITA) {
116           AvaliacaoSingleton.getInstance().
                  getAvaliacaoSelecionada().setImgDireita("/
                  sdcard/" + AvaliacaoSingleton.getInstance().
                  getPessoaBean().getNome()+id_foto + "_dir.png
                  ");
117           Uri uriSavedImage = Uri.fromFile(new File("/
                  sdcard/" + AvaliacaoSingleton.getInstance().
                  getPessoaBean().getNome()+id_foto+ "_dir.png"
                  ));
```

```
118         Intent intent = new Intent(MediaStore.
                ACTION_IMAGE_CAPTURE);
119         intent.putExtra("output", uriSavedImage);
120         localizacao.setFoto(AvaliacaoSingleton.
                getInstance().getAvaliacaoSelecionada().
                getImgDireita());
121
122         startActivityForResult(intent, TIRAR_FOTO);
123       }
124
125     } else if (v.getId() == R.id.bt_confirmar_camera) {
126       Intent intent = new Intent(this,
              PrincipalAvaliacaoActivity.class);
127       startActivity(intent);
128     }
129
130   }
131
132 }
```

Classe - ”ORMLiteHelper.java”

```
1 package br.ufla.smap.helper;
2
3 import java.sql.SQLException;
4
5 import android.content.Context;
6 import android.database.sqlite.SQLiteDatabase;
7 import android.util.Log;
8 import br.ufla.smap.bean.Avaliacao;
9 import br.ufla.smap.bean.Localizacao;
```

```
10 import br.ufla.smap.bean.Pessoa;
11 import br.ufla.smap.bean.ResultadoAvaliacao;
12
13 import com.j256.ormlite.android.apptools.
       OrmLiteSqliteOpenHelper;
14 import com.j256.ormlite.dao.Dao;
15 import com.j256.ormlite.dao.RuntimeExceptionDao;
16 import com.j256.ormlite.support.ConnectionSource;
17 import com.j256.ormlite.table.TableUtils;
18
19 public class ORMLiteHelper extends
       OrmLiteSqliteOpenHelper {
20
21   // Nome da base de dados.
22   private static final String DATABASE_NAME = "smap.db";
23
24   // Versão da base de dados.
25   private static final int DATABASE_VERSION = 10;
26
27   // Caso você queria ter apenas uma instancia da base
       de dados.
28   private static ORMLiteHelper mInstance = null;
29
30   private Dao<Pessoa, Integer> pessoaDao = null;
31   private RuntimeExceptionDao<Pessoa, Integer>
       pessoaRuntimeDao = null;
32
33   private Dao<Avaliacao, Integer> avaliacaoDao = null;
34   private RuntimeExceptionDao<Avaliacao, Integer>
       avaliacaoRuntimeDao = null;
```

```
35
36  private Dao<ResultadoAvaliacao, Integer>
        resultadoAvaliacaoDao = null;
37  private RuntimeExceptionDao<ResultadoAvaliacao,
        Integer> resultadoAvaliacaoRuntimeDao = null;
38
39  private Dao<Localizacao, Integer> localizacaoDao =
        null;
40  private RuntimeExceptionDao<Localizacao, Integer>
        localizacaoRuntimeDao = null;
41
42  private static final String TAG_LOG = "apresentacao
        orm";
43
44  public ORMLiteHelper(Context context) {
45    super(context, DATABASE_NAME, null, DATABASE_VERSION
        );
46  }
47
48  public void onCreate(SQLiteDatabase db,
        ConnectionSource connectionSource) {
49    try {
50      // TableUtils é responsábel por algumas operações
          sobre tabelas,
51      // como, por exemplo, deletar/inserir tabelas.
52      TableUtils.createTable(connectionSource, Pessoa.
          class);
53      TableUtils.createTable(connectionSource, Avaliacao
          .class);
```

```
54        TableUtils.createTable(connectionSource,
              ResultadoAvaliacao.class);
55        TableUtils.createTable(connectionSource,
              Localizacao.class);
56
57        Log.i(TAG_LOG, "create as tabelas");
58      } catch (SQLException e) {
59        throw new RuntimeException(e);
60      }
61    }
62
63    public void onUpgrade(SQLiteDatabase db,
        ConnectionSource connectionSource, int oldVersion,
        int newVersion) {
64      try {
65        TableUtils.dropTable(connectionSource, Pessoa.
              class, true);
66        TableUtils.dropTable(connectionSource, Avaliacao.
              class, true);
67        TableUtils.dropTable(connectionSource,
              ResultadoAvaliacao.class, true);
68        TableUtils.dropTable(connectionSource, Localizacao
              .class, true);
69        onCreate(db, connectionSource);
70
71        Log.i(TAG_LOG, "atualizaou as tabelas");
72      } catch (SQLException e) {
73        throw new RuntimeException(e);
74      }
75
```

```
76  }
77
78  public static ORMLiteHelper getInstance(Context
        context) {
79    if (mInstance == null) {
80      mInstance = new ORMLiteHelper(context.
            getApplicationContext());
81    }
82    return mInstance;
83  }
84
85  public Dao<Pessoa, Integer> getPessoaDao() throws
        SQLException {
86    if (pessoaDao == null) {
87      pessoaDao = getDao(Pessoa.class);
88    }
89    return pessoaDao;
90  }
91
92  public RuntimeExceptionDao<Pessoa, Integer>
        getPessoaRuntimeDao() {
93    if (pessoaRuntimeDao == null) {
94      pessoaRuntimeDao = getRuntimeExceptionDao(Pessoa.
            class);
95    }
96    return pessoaRuntimeDao;
97  }
98
99  public Dao<Avaliacao, Integer> getAvaliacaoDao()
        throws SQLException {
```

```
100     if (avaliacaoDao == null) {
101       avaliacaoDao = getDao(Avaliacao.class);
102     }
103     return avaliacaoDao;
104   }
105
106   public RuntimeExceptionDao<Avaliacao, Integer>
          getAvaliacaoRuntimeDao() {
107     if (avaliacaoRuntimeDao == null) {
108       avaliacaoRuntimeDao = getRuntimeExceptionDao(
              Avaliacao.class);
109     }
110     return avaliacaoRuntimeDao;
111   }
112
113   public Dao<ResultadoAvaliacao, Integer>
          getResultadoAvaliacaoDao() throws SQLException {
114     if (resultadoAvaliacaoDao == null) {
115       resultadoAvaliacaoDao = getDao(ResultadoAvaliacao.
              class);
116     }
117     return resultadoAvaliacaoDao;
118   }
119
120   public RuntimeExceptionDao<ResultadoAvaliacao, Integer
          > getResultadoAvaliacaoRuntimeDao() {
121     if (resultadoAvaliacaoRuntimeDao == null) {
122       resultadoAvaliacaoRuntimeDao =
              getRuntimeExceptionDao(ResultadoAvaliacao.class
              );
```

```
123     }
124     return resultadoAvaliacaoRuntimeDao;
125   }
126
127   public Dao<Localizacao, Integer> getLocalizacaoDao()
          throws SQLException {
128     if (localizacaoDao == null) {
129       localizacaoDao = getDao(Localizacao.class);
130     }
131     return localizacaoDao;
132   }
133
134   public RuntimeExceptionDao<ResultadoAvaliacao, Integer
          > getLocalizacaoRuntimeDao() {
135     if (localizacaoRuntimeDao == null) {
136       localizacaoRuntimeDao = getRuntimeExceptionDao(
              Localizacao.class);
137     }
138     return resultadoAvaliacaoRuntimeDao;
139   }
140
141 }
```

Classe - ”AvaliacaoSingleton.java”

```
1 package br.ufla.smap.singleton;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import br.ufla.smap.bean.Avaliacao;
```

```
7  import br.ufla.smap.bean.Opcao;
8  import br.ufla.smap.bean.Pessoa;
9  import br.ufla.smap.bean.Relatorio;
10 import br.ufla.smap.bean.ResultadoAvaliacao;
11 import br.ufla.smap.enums.FotoEnum;
12
13 public class AvaliacaoSingleton {
14
15   private static AvaliacaoSingleton sessao;
16   private Pessoa pessoaBean;
17   private List<Avaliacao> avaliacaoList = new ArrayList<
         Avaliacao>();
18   private Avaliacao avaliacaoSelecionada;
19   private FotoEnum tipoFoto;
20   private Relatorio relatorioCorrente;
21   private Opcao opcaoCorrente;
22   private Integer currentAvaliacaoTab;
23   private ResultadoAvaliacao resultado;
24
25
26
27
28   public ResultadoAvaliacao getResultado() {
29     return resultado;
30   }
31
32   public void setResultado(ResultadoAvaliacao resultado)
         {
33     this.resultado = resultado;
34   }
```

```
35
36    public Integer getCurrentAvaliacaoTab() {
37      return currentAvaliacaoTab;
38    }
39
40    public void setCurrentAvaliacaoTab(Integer
          currentAvaliacaoTab) {
41      this.currentAvaliacaoTab = currentAvaliacaoTab;
42    }
43
44    public Opcao getOpcaoCorrente() {
45      return opcaoCorrente;
46    }
47
48    public void setOpcaoCorrente(Opcao opcaoCorrente) {
49      this.opcaoCorrente = opcaoCorrente;
50    }
51
52    public Relatorio getRelatorioCorrente() {
53      return relatorioCorrente;
54    }
55
56    public void setRelatorioCorrente(Relatorio
          relatorioCorrente) {
57      this.relatorioCorrente = relatorioCorrente;
58    }
59
60    public FotoEnum getTipoFoto() {
61      return tipoFoto;
62    }
```

```
63
64    public void setTipoFoto(FotoEnum tipoFoto) {
65      this.tipoFoto = tipoFoto;
66    }
67
68    public Avaliacao getAvaliacaoSelecionada() {
69      return avaliacaoSelecionada;
70    }
71
72    public void setAvaliacaoSelecionada(Avaliacao
          avaliacaoSelecionada) {
73      this.avaliacaoSelecionada = avaliacaoSelecionada;
74    }
75
76    public List<Avaliacao> getAvaliacaoList() {
77      return avaliacaoList;
78    }
79
80    public void setAvaliacaoList(List<Avaliacao>
          avaliacaoList) {
81      this.avaliacaoList = avaliacaoList;
82    }
83
84    public Pessoa getPessoaBean() {
85      return pessoaBean;
86    }
87
88    public void setPessoaBean(Pessoa pessoaBean) {
89      this.pessoaBean = pessoaBean;
90    }
```

```
91
92   public static AvaliacaoSingleton getInstance() {
93     if (sessao == null) {
94       sessao = new AvaliacaoSingleton();
95     }
96     return sessao;
97   }
98
99 }
```

Solução do Problema de carregamento de imagens

```
1 this.buffer = Bitmap.createBitmap(100, 100, Bitmap.
    Config.ARGB_4444);
```

APÊNDICE B - Segundo Apêndice

Ferramentas

SDK Android: O google disponibiliza uma versão do SDK para diversas plataformas, podendo ser encontrado no seguinte endereço: <http://developer.android.com/sdk/index.html>

A instalação do SDK Android no windows é feita através de um arquivo .zip que quando extraído pode ser executado.

Eclipse ADT: O eclipse ADT é um plugin fornecido pelo google, que permite a integração do eclipse com as ferramentas disponíveis no SDK do Android.

Seu processo de instalação pode ser encontrado no seguinte endereço: <http://developer.android.com/sdk/installing/installing-adt.html>

Assinando uma aplicação Android: O processo para compilar e assinar um aplicativo Android pode ser vizualizado em um tutorial encontrado no seguinte endereço: <http://escoladeandroid.blogspot.com.br/2011/06/compilar-e-assinar-aplicativo-android.html>

Balsamiq: A instalação do Balsamiq é realizada através de um executável encontrado no seguinte endereço: <http://balsamiq.com/download/>

Astah: A instalação do Astah é realizada através de um executável encontrado no seguinte endereço: <http://astah.net/download>

APÊNDICE C - Terceiro Apêndice

Diagramas de Interface

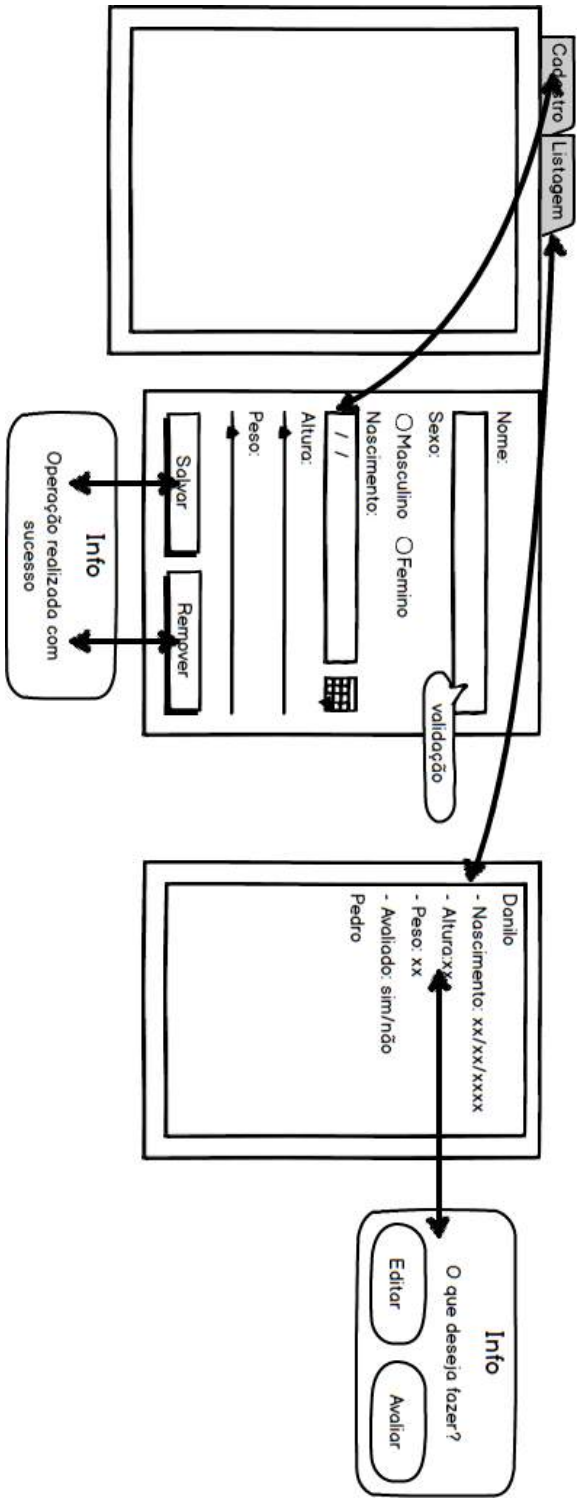


Figura 27 Diagrama de Interface - Cadastro de Pessoas

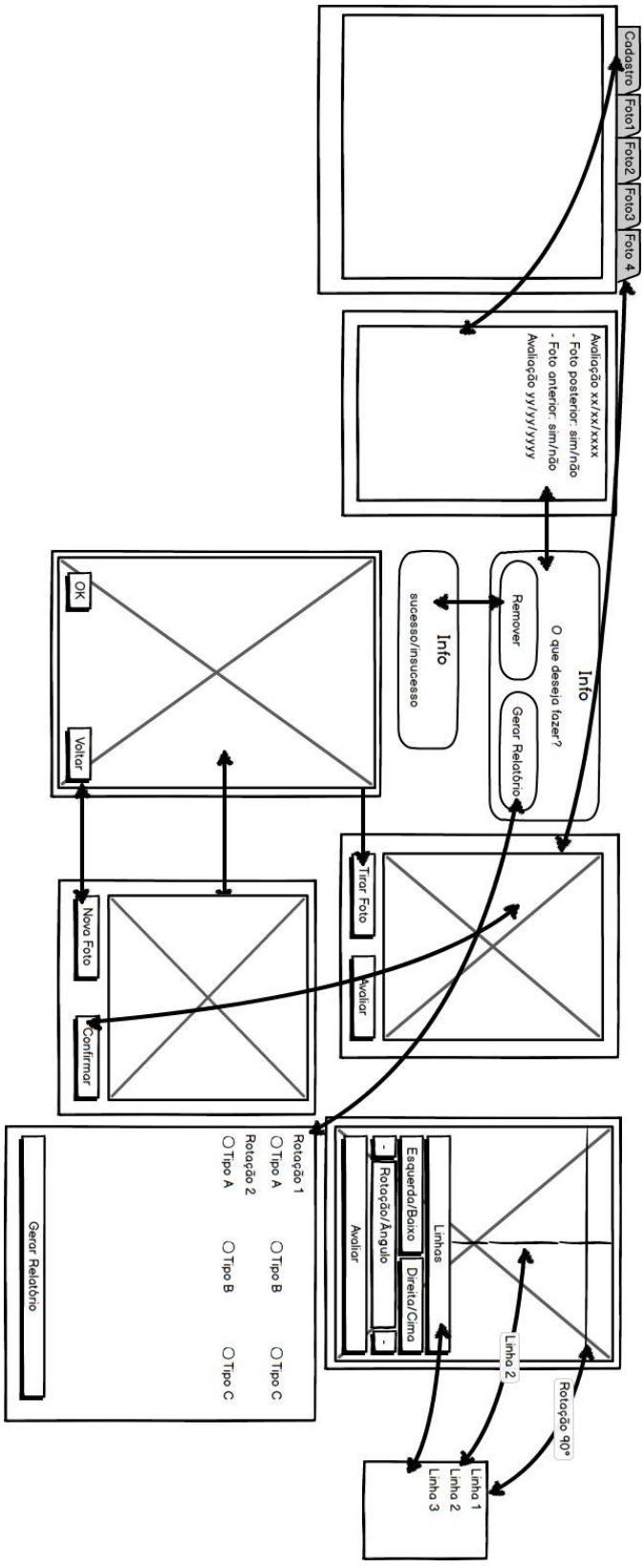
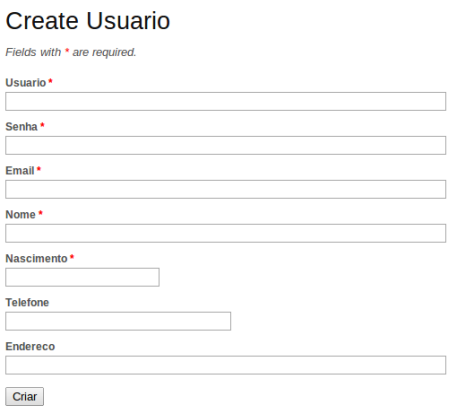


Figura 28 Diagrama de Interface - Avaliação de Pessoas

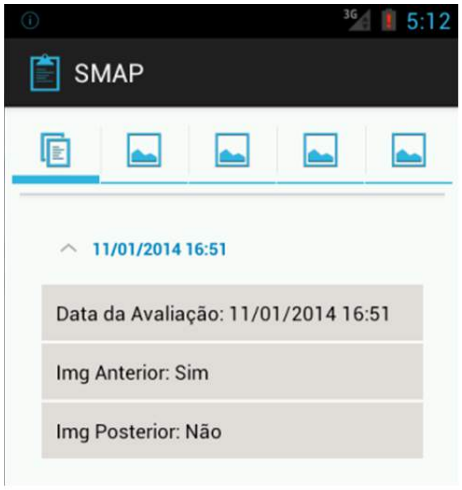
APÊNDICE D - Quarto Apêndice

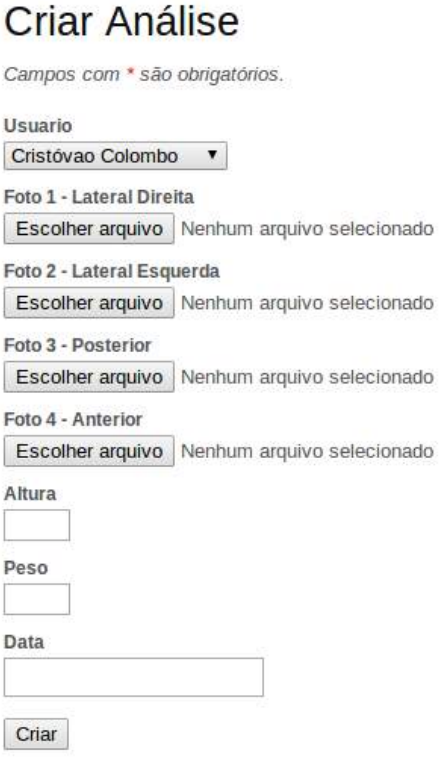
Imagens Comparativas - SMAP x SOAP



SMAP SOAP

Figura 29 Cadastro de Pessoa - SMAP x SOAP





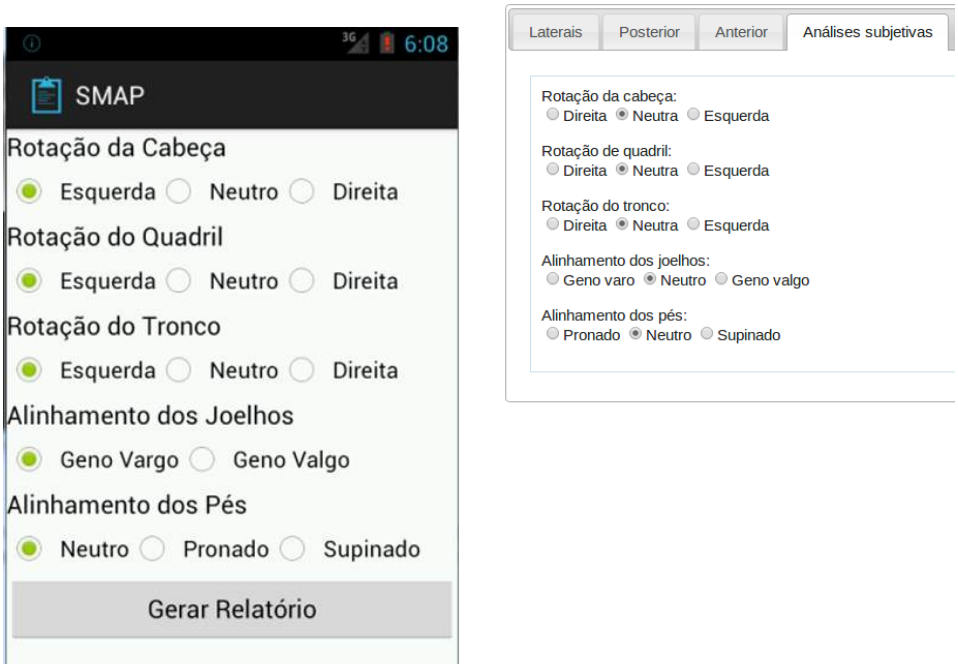
SMAP SOAP

Figura 30 Nova Análise Postural - SMAP x SOAP

SMAP

SOAP

Figura 31 Análise Postural - SMAP x SOAP



SMAP SOAP

Figura 32 Opções de Análise - SMAP x SOAP

SMAP SOAP

Figura 33 Relatório - SMAP x SOAP

APÊNDICE E - Quinto Apêndice

Sobre a Aplicação

Instalando o Aplicativo no Dispositivo

Embora a maioria dos aplicativos existentes para Android, possam ser baixados e instalados através da loja de aplicativos do google, a ”Play Store”, a ferramenta desenvolvida neste projeto não esta disponível através dessa opção.

Para instalar a aplicação é necessário transferir ela para memória do dispositivo e instalá-la de forma manual. Esse tipo de instalação requer que

a opção de instalar ”aplicativos de fontes desconhecidas”esteja ativa. Essa opção está disponível através das configurações do SO Android, na opção de ”Segurança”.

Passo a Passo - Instalar aplicação Android de forma manual

1. Transferir o arquivo ”.apk”para o dispositivo.
2. Verificar se a opção para instalar aplicações de ”fontes desconhecidas”está marcada nas opções de ”segurança”do dispositivo Android.
3. Utilizar um gerenciador de arquivos para acessar o diretório para onde o arquivo ”.apk”foi transferido.
4. Abrir o arquivo ”.apk”.
5. Concluir o processo de instalação iniciado pelo SO Android.

Utilizando o Aplicativo

Cadastrando uma nova pessoa

1. Escolha a TAB com o ícone

Figura 34 Ícone - cadastro de pessoa

2. Prencha as informações necessárias
3. Clique na opção ”Gravar Dados”

Criando uma nova avaliação

1. Escolha a TAB com o ícone

Figura 35 Ícone - lista de pessoas

2. Selecione uma pessoa da lista
3. Clique na opção ”Avaliar”

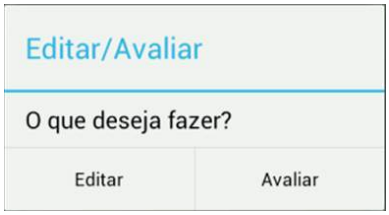


Figura 36 Opção - avaliar

Tirando uma foto

1. Escolha a TAB com o ícone

Figura 37 Ícone - foto

2. Clique na opção ”Tirar Foto”

Iniciando uma Avaliação Postural

1. Escolha a TAB com o ícone

Figura 38 Ícone - avaliação postural

2. Clique na opção foto

3. Escolha a opção ”Avaliar”

Figura 39 Opção - avaliar

Gerando um novo relatório

1. Escolha a TAB com o ícone

Figura 40 Ícone - avaliações

2. Selecione uma Avaliação da Lista

3. clique na opção ”Gerar Relatório”