UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
FACULDADE DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE SISTEMAS DE INFORMAÇÃO

DENISE APARECIDA DE SOUZA

DESENVOLVIMENTO DE APLICATIVO MÓVEL PARA TREINAMENTO ESPORTIVO

Diamantina - MG
2014

DENISE APARECIDA DE SOUZA

DESENVOLVIMENTO DE APLICATIVO MÓVEL PARA TREINAMENTO ESPORTIVO

Monografia apresentada ao curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri – UFVJM, como pré-requisito para obtenção do grau de bacharel, sob orientação do Prof. Dr. Alessandro Vivas Andrade.

Diamantina - MG
2014

DENISE APARECIDA DE SOUZA

DESENVOLVIMENTO DE APLICATIVO MÓVEL PARA TREINAMENTO ESPORTIVO

Monografia apresentada ao curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como pré-requisito para obtenção do grau de bacharel em Sistemas de informação.

COMISSÃO EXAMINADORA

_____

Prof. Dr. Alessandro Vivas Andrade (Orientador)
Universidade Federal dos Vales do Jequitinhonha e Mucuri

_____

Prof. Dr. Rafael Santin
Universidade Federal dos Vales do Jequitinhonha e Mucuri

_____

Prof. MSc. Fernando Joaquim Gripp Lopes
Universidade Federal dos Vales do Jequitinhonha e Mucuri

Aprovado em: _____/_____/_____

À minha família.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado saúde e força para superar as dificuldades e a distância longe de casa. Agradeço aos meus pais Geni de Souza Oliveira e José Oliveira de Souza, pelo amor incondicional, o carinho e a força no momentos difíceis. Agradeço aos meus irmão Deividi José de Souza e Ueliton Oliveira de Souza pelo apoio. Agradeço o meu namorado Elias Vieira da Silva que me deu apoio, incentivo nas horas difíceis, de desânimo e cansaço. Aos amigos que estiveram sempre presentes nos momentos importantes e descontraídos. Agradeço também a minha grande amiga Luana Aparecida Leite de Almeida por sempre ter me apoiado e estar ao meu lado. Ao meu orientador Alessandro Vivas Andrade e ao professor Fernando Gripp pela confiança, paciência e apoio na elaboração deste trabalho. E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“A Tecnologia Move o Mundo.”

Felipe Bilato and Steve Jobs

RESUMO

Esta monografia aborda o desenvolvimento de um aplicativo para sistema operacional Android. Tem como objetivo o desenvolvimento de um aplicativo para dispositivo móvel com foco em treinamento intervalado de alta intensidade que teste o usuário e auxilie o treinamento esportivo. Para o desenvolvimento de aplicações Android, é necessário conhecimentos relativos ao ambiente de desenvolvimento, linguagem de programação e kits de desenvolvimento Android. O aplicativo foi intitulado de Beep e suas principais funcionalidades é cadastramento de pessoas e a realização de teste, o qual é baseado no teste de vai-e-vem de Léger e Lambert e permitir a visualização dos teste de uma pessoa específica. Esta versão 1.0 do aplicativo Beep permite cadastrar pessoas com o nome, cpf, idade, sexo, massa corporal, estatura e modalidade. Também permite a realização do teste que fornecerá o progresso da pessoa, velocidade, distância percorrida, tempo de realização, nível, shuttle e VO_2máx. Além disso, fornece os teste realizados por uma determinada pessoa. Através deste aplicativo os usuários poderão realizar testes, posteriormente analisar as informações salvas e verificar o desempenho da pessoa.

Palavras-chave: Android. Aplicativo. Dispositivos. HIIT. Java. Teste. VO_2máx. XML.

ABSTRACT

This monograph approaches the development of an application for Android operating system. Aims to develop an application for mobile with focus on high intensity interval training the user to test and assist the sports training. To develop Android applications, you need knowledge of the development environment, programming language and Android development kits. The application was entitled Beep and its main functions is registration of persons and the achievement test, which is based on the test back-and-forth Leger and Lambert and allow viewing of test a specific person. This 1.0 version of Beep application allows registering people with the name, cpf, age, sex, body weight, height and type. It also allows the test that will provide the development of the person, speed, distance, exercise time, level, shuttle and VO_2máx. It also provides the test performed by a particular person. Through this application users can perform tests subsequently analyze the saved information and verify the person's performance.

Keywords: Android. Application. Devices. HIIT. Java. Test. VO_2máx. XML.

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADT - Android Development Tools

AMM - Artes Marciais Mistas

API - Application Programming Interface

AVD - Android Virtual Device

CPF - Cadastro de Pessoas Físicas

DDM - Dalvik Debug Monitor Server

GPS - Global Positioning System

HIIT - High Intensity Interval Training

IDC - International Data Corporation

IDE - Integrated Development Environment

JDK - Java Development Kit

JVM - Java Virtual Machine

MMA - Mixed Martial Arts

MMS - Multimedia-Messaging Service

OHA - Open Handset Alliance

PDA - Personal Digital Assistant

PHP - Personal Home Page

QEMU - Quick EMUlator

RIM - Research in Motion

SD Card - Secure Digital Card

SDK - Software Development Kit

SMS - Short Message Service

UNESCO - United Nations Educational Scientific and Cultural Organization

USB - Universal Serial Bus

VO_2 - Consumo de oxigênio

VO_2máx - Consumo máximo de oxigênio obtido nos testes aeróbios

VPN - Virtual Private Network

XML - Extensible Markup Language

XMPP - Extensible Messaging and Presence Protocol

LISTA DE SÍMBOLOS

Cal – caloria

Kg – Quilograma

KJ – kilojoule

Km/h – Quilômetros por hora

m – metro

SUMÁRIO

1. INTRODUÇÃO

A presente monografia aborda o desenvolvimento de um aplicativo móvel capaz de auxiliar o treinamento esportivo e testar o usuário. A princípio, na Seção 1.1, será apresentado o objeto de estudo que referencia a temática do aplicativo. Nas Seções 1.2 e 1.3 são apresentados os objetivos a serem alcançados e a estrutura do trabalho.

1.1. Apresentação

O treinamento esportivo passou por muitas evoluções e transformações nos últimos 20 anos. A principal delas foi a mudança da quantidade para a qualidade das repetições de uma determinada atividade física, tendo como objetivo capacitar os atletas para que atinjam a excelência no seu desempenho esportivo.

Com isso, surgem discussões a respeito dos aspectos da preparação física dos atletas e sua aptidão física. Dentre os vários componentes que caracterizam a aptidão física de um indivíduo, a capacidade cardiorrespiratória tem sido considerada uma das mais importantes, tanto para a grande maioria dos atletas das diferentes modalidades esportivas, como também para os indivíduos não atletas, que necessitam de uma atividade física como meio de promoção de saúde (AMERICAN COLLEGE OF SPORTS MEDICINE, 1991).

A aptidão cardiorrespiratória pode ser realizada através da mensuração do consumo máximo de oxigênio (VO_2máx). Para medir o VO_2máx, pode-se utilizados métodos diretos como a utilização de ergômetros e métodos indiretos (predições) como o teste vai-e-vem de Léger e Lambert.

Os métodos diretos são mais precisos, no entanto possuem um alto custo e necessitam de pessoal especializado para aplicação dos testes e um tempo maior na execução. Por isso, os testes indiretos tem sido frequentemente propostos por vários autores visto que sua aplicação é mais simples, de menor custo e pode ser aplicado a grandes populações.

Portanto, a presente monografia baseia-se no teste vai-e-vem de Léger e Lambert, dado seu baixo custo, simplicidade de aplicação, não necessita de pessoal especializado e proposto por vários autores.

1.2. Objetivos

O objetivo geral desta monografia é desenvolver um aplicativo para dispositivo móvel com foco em treinamento intervalado de alta intensidade.

Como objetivos específicos, estão:

- Apresentar as ferramentas utilizadas para o desenvolvimento do aplicativo móvel com foco em treinamento intervalado de alta intensidade;
- Propor e implementar um aplicativo móvel com soluções tecnológicas para testar o usuário e auxiliar a prática esportiva;

1.3. Estrutura do trabalho

No Capítulo 2 são apresentados conceitos relacionados a plataforma do Android e treinamento esportivo, necessários para um melhor entendimento da plataforma proposta bem como o teste para treinamento escolhido. São apresentados conceitos específicos da plataforma do Android e treinamento esportivo de alta intensidade, pelo fato deste trabalho abordar um problema com estas características.

No Capítulo 3 é apresentada a metodologia utilizada para desenvolvimento de aplicativo móvel para treinamento esportivo, descrevendo as ferramentas computacionais utilizadas e o método escolhido.

O Capítulo 4 apresenta o aplicativo desenvolvido, mostrando passo a passo o processo de desenvolvimento da aplicação Beep.

No Capítulo 5 é apresentado o produto final.

O Capítulo 6 apresenta as conclusões e sugestões para trabalhos futuros.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta a revisão bibliográfica conduzida no presente trabalho. De início é realizada uma explanação acerca dos dispositivos móveis. No decorrer do capítulo são expostos conceitos relacionados a plataforma Android e treinamento esportivo. A seção 2.2.1 e 2.2.2 apresentam o contexto histórico do Android e sua definição. Nas seções 2.2.3 e 2.2.4 são apresentados conceitos básicos acerca da arquitetura da plataforma Android e seus módulos, bem como os componentes de uma aplicação, que se mostram relevantes para a condução do trabalho. Já a seção 2.3 expõe o uso de dispositivos móveis no treinamento esportivo. Por fim, a seção 2.4 apresenta o Treinamento Intervalado de Alta Intensidade com o uso de sinal sonoro.

2.1. Dispositivos Móveis

Com o avanço das tecnologias de Informação, o mercado de dispositivos móveis está crescendo cada vez mais. Segundo West & Chew (2014, p.16) em uma publicação da UNESCO, em menos de uma década a tecnologia móvel se espalhou por todo o planeta. Também relata que, dos cerca de 7 bilhões de pessoas no mundo, 6 bilhões têm acesso a um dispositivo móvel. Isso corresponde a aproximadamente 85% da população mundial.

Atualmente os dispositivos móveis estão se tornando mais poderosos não apenas em relação ao número aquisições dos aparelhos, mas ao mesmo tempo em relação às suas capacidades de armazenamento, de processamento e comunicação. Dentre os dispositivos móveis mais comuns estão inclusos os smartphones, Personal Digital Assistant (PDA), celulares, console portátil (Videogame portátil), ultra mobile PC, ultrabook, notebook e coletor de dados.

Através das tecnologias de informação os dispositivos móveis estão cada vez mais inovadores possibilitando o acesso móvel, instantâneo e permanentemente conectado à informação, seja no âmbito corporativo ou privado.

Com a tecnologia móvel tão disseminada e presente na maioria das classes sociais, o mercado corporativo também está crescendo, buscando incorporar aplicações móveis em seu cotidiano para agilizar seus negócios e proporcionar maior lucratividade.

Já no que tange o âmbito privado, usuários buscam celulares com um visual agradável e moderno, de fácil navegação e uma infinidade de recursos como câmeras, Sistema de Posicionamento Global (do inglês *Global Positioning System - GPS*), TV Digital, entre outros.

As estatísticas mais recentes demonstram uma queda de venda de PCs (desktops) com a vinda de smartphones, ultrabooks, tablets, híbridos de tablet e smartphone, etc.

A 13ª edição do F/Radar (2013), tradicional estudo sobre internet realizado pela F/Nazca Saatchi & Saatchi em parceria com o Instituto Datafolha, constatou a evolução da penetração da tecnologia móvel na população brasileira.

Segundo a F/Radar (2013), o acesso via celular cresceu, nada mais, nada menos, do que impressionantes 78,5% em 20 meses. Hoje, 43 milhões de brasileiros (com mais de 12 anos) navegam pela internet por meio de dispositivos móveis. A possibilidade de ir, vir e, principalmente compartilhar instantaneamente o que quer que seja vem reunindo fãs para os dispositivos móveis.

De acordo com Kuszka (2014), diretor dos Arquitetos de Solução da Red Hat[1], o acesso aos aplicativos via dispositivos móveis pode ser analisado por meio de duas visões: a corporativa e a privativa. Apesar do dispositivo ser o mesmo para acesso a empresa e dados pessoais, as formais como isso se desenvolveu em cada um dos casos foram bem distintas.

A evolução e o impacto em termos de inovação na visão corporativa é bem perceptível. Recordando, nos anos 60 até 80, na era dos mainframes, o acesso era feito por meio dos terminais de tamanhos e peso gigantescos, sem gráficos e com pouquíssima flexibilidade. No final dos anos 80, começou a migração do mainframe para equipamentos baseados em padrões abertos (*Open Systems*) por razões de custos e possibilidade de minimizar a dependência do fabricante. Isso durou até o final dos anos

[1] http://www.redhat.com/

90, quando veio a internet e a plataforma Java possibilitando rodar o aplicativo de qualquer dispositivo, lugar, com segurança.

Atualmente, na era do "*Could Computing*", o acesso via internet e via navegadores já é um padrão e os dispositivos móveis estão extremamente evoluídos e sofisticados. Assim, com a disponibilidade de tecnologias maduras de VPN (acesso via rede com criptografia) para dispositivos móveis, todo o meio corporativo começou a entender que acessar as ferramentas de trabalho pelo dispositivo trará um aumento de produtividade, motivação e acesso a qualquer hora, lugar e com segurança.

Porém, na visão de uso privado, foi presenciado uma explosão dos sites sociais como uma ferramenta para alavancar o uso intenso do celular e outros dispositivos móveis para o usuário final. Estes, que estão cada vez mais utilizando os dispositivos móveis para trabalhar, se comunicar, buscar informações e se divertir.

No entanto, para acompanhar o avanço tecnológico e satisfazer os usuários de modo geral, as maiores empresas do mundo em tecnologia móvel travam uma disputa em busca de conquistar esse nicho de mercado. Consequentemente, criando novas plataformas e ambientes de desenvolvimento poderosos, ousados e flexíveis. Logo, o Android é a resposta da Google a essa conjuntura.

2.2. Android

Esta seção apresenta um breve histórico acerca do Android, sua definição, arquitetura e componentes de uma aplicação Android.

2.2.1. Histórico

A história do Android iniciou em outubro de 2003, na cidade de Palo Alto na Califórnia - EUA, quando Andy Rubin, Rich Miner, Nick Sears e Chris White fundaram a Android Inc., a qual desenvolvia secretamente projetos de sistemas operacionais para celulares.

Em agosto de 2005, a Google viu na Android Inc. a oportunidade de se alavancar e ocupar um espaço no mercado de tecnologia móvel. Dois anos depois, em novembro

de 2007 a Google e mais de 30 empresas líderes em tecnologia móvel formaram a *Open Handset Alliance* (OHA)[2].

Atualmente a OHA é formada por um grupo de 84 empresas. Esse grupo foi criado com a intenção de padronizar uma plataforma de código aberto e livres para celulares visando atender as expectativas e tendências do mercado atual. Dessa aliança fazem parte operadoras de celular, companhias de semicondutores, fabricantes de celulares, empresas de software e empresas de comercialização.

Ao mesmo tempo que foi anunciado a formação da OHA, a mesma revelou a plataforma Android, baseada no sistema operacional Linux.

O HTC[3] Dream G1 (também conhecido como T - Mobile G1), primeiro aparelho com a tecnologia “Android 1.0”, foi lançado em outubro de 2008. O HTC Dream G1 é apresentado na Figura 2.1. Desde então, as versões Android passaram a ser desenvolvidas sob o codinome de um doce e em ordem alfabética: Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat e a versão mais recente 5.0 Lollipop, lançada em 03 de novembro de 2014.

Figura 2.1 - HTC Dream G1 ou T - Mobile G1.

Fonte: Site blogoscoped.com[4].

[2] http://www.openhandsetalliance.com/index.html.
[3] **HTC**, do inglês ***High Tech Computer Corporation***, é uma fabricante taiwanesa de dispositivos portáteis baseados nas plataformas do Windows Mobile e Android.
[4] Disponível em: <http://blogoscoped.com/files/t-mobile-g1-large.jpg>. Acesso em Maio de 2014.

2.2.2. Definição

O Android é um sistema operacional para dispositivos móveis como *smartfones* e *tablets*. Contêm um sistema operacional baseado na plataforma de código aberto Linux, uma interface visual rica, com diversas aplicações já instaladas, integração com Google Maps, um browser para navegação web, suporte a multimídia, GPS, banco de dados integrado e ainda um ambiente de desenvolvimento bastante poderoso, inovador e flexível como o Eclipse[5]. Assim, desenvolvedores podem criar aplicações para a plataforma utilizando o kit de desenvolvimento SDK (*Software Development Kit*) do Android.

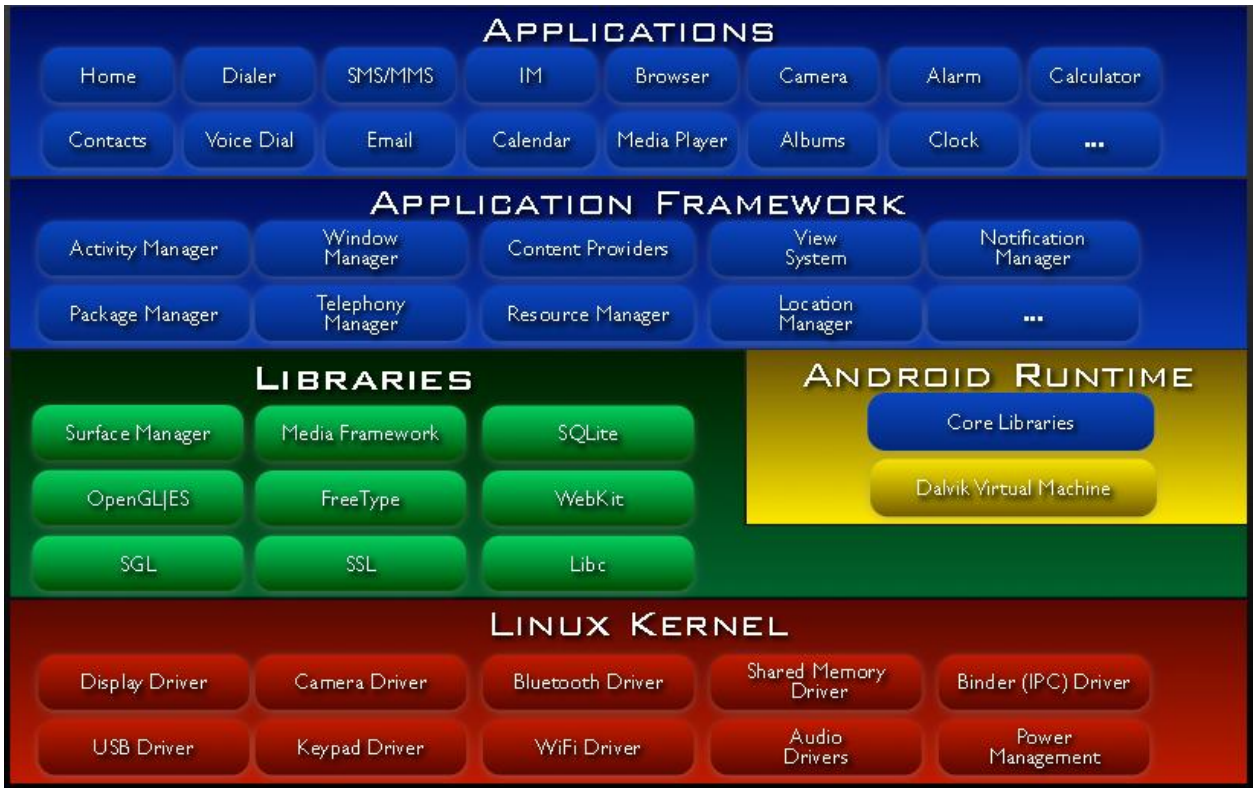
As aplicações para essa plataforma são escritas utilizando a linguagem de programação Java, são compiladas em bytecodes Dalvik e executadas usando a Máquina Virtual Dalvik (*Dalvik Virtual Machine*) especializada desenvolvida para uso em dispositivos móveis. Apesar de as aplicações Android serem escritas na linguagem Java, a Dalvik não é uma máquina virtual Java, já que não executa bytecode JVM.

2.2.3. Arquitetura

Segundo Lecheta (2010), a arquitetura do sistema operacional Android é dividida em camadas, onde cada parte é responsável por gerenciar os seus respectivos processos. A Figura 2.2 mostra os principais componentes do sistema operacional Android.

[5] http://www.eclipse.org

Figura 2.2 - Arquitetura do Android.



Fonte: Site Androidteam[6].

Nas seções seguintes, realiza-se uma explicação sobre os módulos da arquitetura Android.

2.2.3.1. Kernel do Linux

A arquitetura do Android foi baseada no *Kernel* 2.6 do Linux, herdando diversas características dessa plataforma. O *kernel* do sistema funciona como uma camada de abstração entre o *hardware* e o restante da pilha de *software* da plataforma. É responsável pelo gerenciamento da memória, dos processos, threads e a segurança dos arquivos e pastas, além de redes e drives.

---

[6] ____. **Anatomy Physiology of an Android.** Androidteam. Disponível em: <http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf>. Acesso em Abril de 2014.

2.2.3.2. Bibliotecas

O Android inclui nesta camada um conjunto de bibliotecas C/C++ usados por vários componentes do sistema. As funcionalidades são expostas através do *framework* do Android na seção 2.1.3.4. Algumas das bibliotecas do núcleo da arquitetura são listadas abaixo:

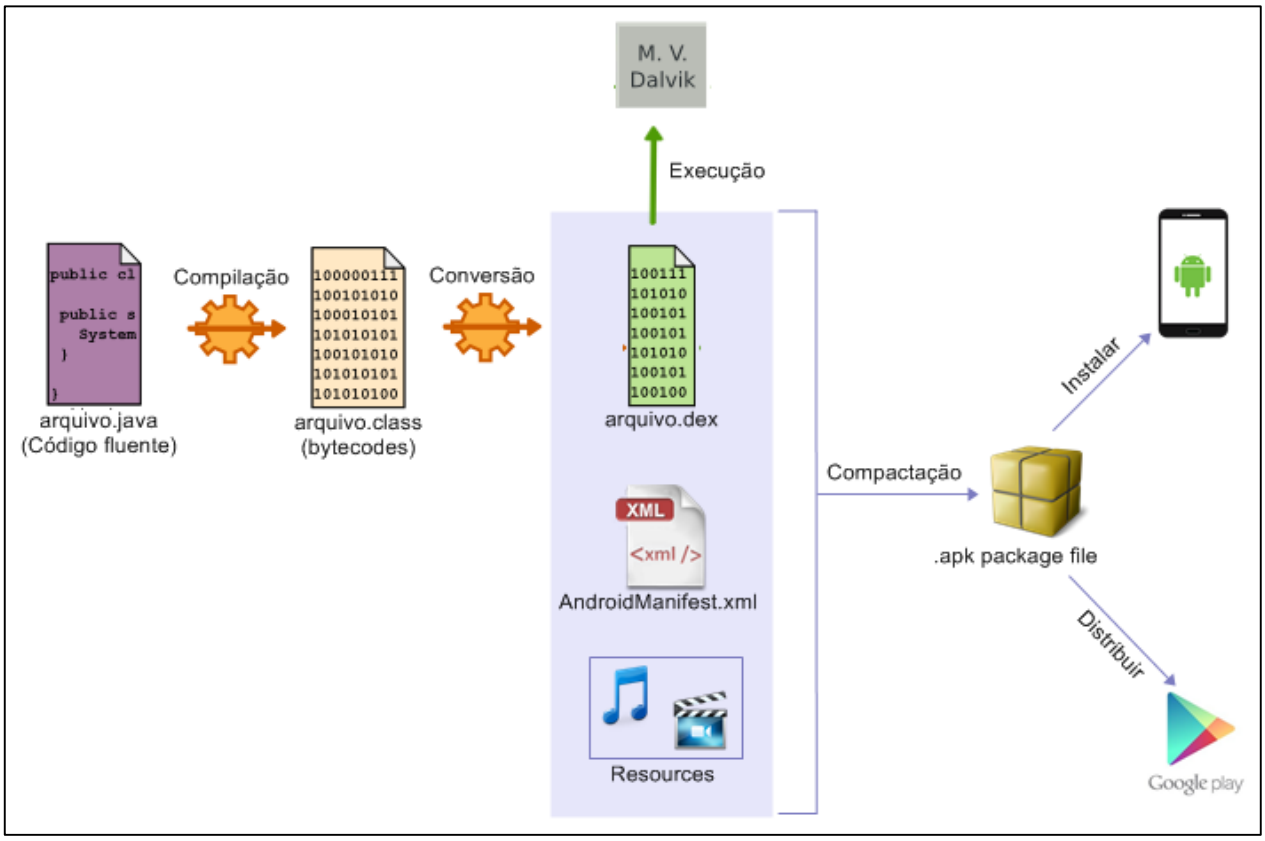
- **Surface Manager:** Controla e gerencia o acesso ao subsistema de *display*. Compõe transparentemente camadas gráficas 2D e 3D de várias aplicações.
- **3D libraries:** Implementação baseada a especificação do OpenGL 1.0.
- **SGL:** Biblioteca usa para compor gráficos 2D.
- **Media libraries:** Essas bibliotecas suportam playblack e gravação de muitos formatos de áudio e vídeo, bem como imagens estáticas, incluindo MPEG4, H.264, MP3, AAC, AMR, JPG e PNG.
- **FreeType:** É uma biblioteca usada para renderizar fontes.
- **SSL:** Fornece encriptação de dados enviados pela Internet.
- **SQLite:** É uma biblioteca C que implementa um banco de dados SQL embutido. Programas que usam a biblioteca SQLite podem ter acesso ao banco de dados SQL sem executar um processo RDBMS em separado. SQLite não é uma biblioteca de cliente usada para conectar com um servidor de banco de dados. SQLite é o servidor.
- **LibWebCore:** Biblioteca base para o navegador do Android e visões da *web*.
- **System C Library:** Uma implementação do libc derivada do BSD.

2.2.3.3. Android Runtime

Na camada Runtime é instanciada a máquina virtual Dalvik. Para cada aplicação executada no Android é criada uma Dalvik. As aplicações Android são construídas utilizando a linguagem Java, porém no sistema operacional Android não existe uma Máquina Virtual Java (JVM). O que se tem na verdade é a Dalvik otimizada para execução em dispositivos móveis.

O arquivo *.dex* (Dalvik Executable) representa a aplicação do Android compilada. Gerado a partir da conversão do bytecode (*.class*) compilado. Depois que os arquivos *.dex* serem gerados, eles são compactados juntos com outros recursos como imagens em um único arquivo com extensão *.apk* (Android Package File), que representa a aplicação final, que será distribuída e instalada. A Figura 2.3 exemplifica os processos para execução de uma aplicação.

Figura 2.3 – Processos para execução de uma aplicação.



Fonte: Elaborada pelo autor.

A camada Runtime possui além da Dalvik outro componente chamado *Core Libraries* que contém um conjunto de bibliotecas que fornece a maioria das funcionalidades disponíveis no núcleo das bibliotecas da linguagem Java, incluindo classes para manipulação de arquivos, entrada e saída, entre outras.

2.2.3.4. Framework de Aplicações

Na camada Framework de Aplicações (*Application Framework*) é encontrado as APIs[7] do Android, normalmente utilizadas pelas aplicações que são executadas sobre a plataforma. Este framework disponibiliza por exemplo, gerenciadores de serviços de telefonia e notificação.

O fundamento de todas as aplicações do framework é um conjunto de serviços e sistemas que inclui:

- **Activity Manager** (Gerenciador de Atividade): gerencia o ciclo de vida das aplicações.
- **Package Manager** (Gerenciador de Pacotes): mantém quais aplicações estão instaladas no dispositivo.
- **Windows Manager** (Gerenciador de Janelas): gerencia as janelas das aplicações.
- **Telephony Manager** (Gerenciador de telefonia): componentes para acesso aos recursos de telefonia.
- **Content Providers** (Provedores de conteúdo): permitem que as aplicações acessem os dados de outras aplicações ou compartilhem os seus próprios dados.
- **Resource Manager** (gerenciador de recursos): fornece acesso a recursos gráficos e arquivos de *layout*.
- **View System** (Visão do Sistema): conjunto rico e extensível de componentes de interface de usuário. As visões podem ser usadas para construir uma aplicação, elas incluem listas, grids, caixas de texto, botões, dentre outras.
- **Location Manager** (Gerenciador de Localização): gerencia a localização do dispositivo.
- **Notification Manager** (Gerenciador de Notificações): permite que todas as aplicações exibam alertas na barra de status.
- **XMPP Service** (Serviço XMPP): suporte para uso do protocolo XMPP (*Extensible Messaging and Presence Protocol*)[8].

---

[7] **API**, do inglês ***Application Programming Interface*** ou **Interface de Programação de Aplicações**.
[8] http://xmpp.org/xmpp-protocols/xmpp-extensions/

2.2.3.5. Aplicações

A camada de Aplicações fornece as aplicações básicas que executam sobre a plataforma do Android, tais como, cliente de SMS e MMS, cliente de e-mail, navegador, mapas, calculadora, entre outras aplicações que foram desenvolvidas por terceiros. Esta característica garante a ela o alto grau de flexibilidade e extensibilidade da plataforma.

2.2.4. Componentes de uma Aplicação Android

As aplicações Android são construídas através de uma combinação de componentes distintos. Deste modo, há quatro componentes principais para a construção de uma aplicação:

- Activity;
- Intents e Intent Filters, e Broadcast Receivers;
- Services;
- Content Providers.

Uma aplicação Android deve ser escrita com um ou mais desses componentes de construção. No entanto, nem toda aplicação necessita ter todos os quatro componentes. Uma vez que definido quais componentes serão indispensável à aplicação, é preciso listá-los em um arquivo chamado *AndroidManifest.xml*.

O Arquivo *AndroidManifest.xml* é a base de uma aplicação Android. Ele é obrigatório e deve estar na pasta raiz do projeto, contento os componentes necessários à aplicação e todas as configurações necessárias para executar a aplicação.

2.2.4.1. Activity

A *Activity* representa uma tela da aplicação, do português uma atividade, ação ou funcionalidade que o usuário pode realizar na aplicação. Sendo uma especialização da classe *android.app.Activity* mais importante no *android:*. Uma Activity define métodos,

controla estados e eventos, ou seja, controla o estado de uma tela e a passagem de parâmetros de uma tela para outra.

Normalmente, cada Activity está associada a uma *view*. Esta por sua vez, é responsável pela exibição de elementos visuais na tela, tais como botões, imagens, entre outros.

Ao gerenciar eventos de telas e também coordenar o fluxo da aplicação, uma activity apresenta um ciclo de vida bem definido. Isto é, cada activity possui um ciclo de vida que varia desde a sua criação até o momento do término da aplicação. Uma atividade pode assumir os seguintes estados: *onCreate (), onStart (), onResume (), onPause (), onStop (), onDestroy () e onRestart ().*

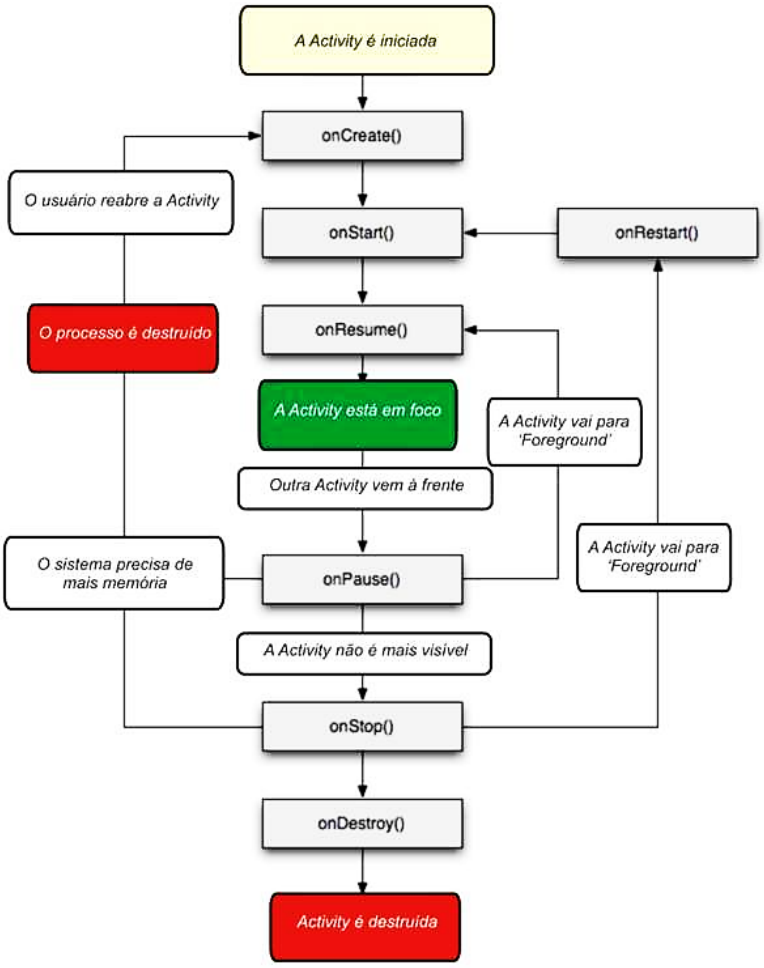
Cada activity deve obrigatoriamente implementar o método *onCreate(Bundle)*, responsável por realizar a inicialização vital para a execução da aplicação. Ele é chamado apenas uma vez, quando o usuário clicar no ícone da aplicação. É dentro deste método que deve-se chamar o método *setContentView(view)* para informar a view responsável por desenhar a interface gráfica da tela. O método setContentView(view) é o que faz a ligação entre a activity e a view e recebe como parâmetro a view que será exibida na tela.

O método *onStart ()* é chamado imediatamente após o *onCreate ()* ou quando o usuário inicia uma activity outra vez. O método *onRestart ()* chama o método *onStart ()* automaticamente. O próximo método chamado depois do *onStart ()* é sempre o *onResume ().* Representa o estado de que a activity está executando.

Uma activity pode alternar entre os estados executando e pausado. Por exemplo, se outra activity iniciar a atual é parada temporariamente. Logo, o método *onPause ()* é chamado. Isto pode acontecer também se o usuário atender uma ligação ou quando o celular entrar para o modo de espera/dormir para economizar energia e, quando o usuário o ativa, o método *onResume ()* é chamado para continuar a aplicação.

O método *onStop ()* é chamado quando a activity está sendo encerrada e não está mais visível ao usuário. Por fim, o método *onDestroy ()* encerra a execução de uma activity e pode ser chamado automaticamente pelo próprio sistema operacional para liberar recursos. Vale ressaltar que o processo da activity pode ser destruído (*kill*) pelo sistema operacional caso as condições de memória estejam críticas. A Figura 2.4 demonstra o ciclo de vida completo de uma activity, exibindo seus estados possíveis.

Figura 2.4 – Ciclo de vida de uma activity.



Fonte: Site do Java Rock Exception[9].

2.2.4.2. Intents, Intent Receiver, e Broadcast Receivers

Intents, *Intent Receiver* e *Broadcast Receivers* são componentes que estão diretamente relacionados a eventos propagados pelo sistema operacional. Cada um desempenha um papel fundamental no tratamento deste eventos e estão de certa forma interligados.

A *Intent* também é uma classe muito importante para uma aplicação Android. Ela representa uma mensagem da aplicação para o sistema operacional. Em português *Intent* é “Intenção”, ou ação que a aplicação deseja executar/realizar uma determinada

[9] Disponível em:<http://javarockexception.wordpress.com/2012/03/12/estudando-android-ciclo-de-vida-activity/>. Acesso em Outubro de 2014.

tarefa. Entre muitas coisas, uma *intent* é a forma utilizada para que aplicações e processos diferentes se comuniquem através de mensagens.

Quando uma mensagem é enviada para o sistema operacional através de uma intent, é necessário configurar a classe *IntentFilter* para interceptar essa mensagem, com base em seu conteúdo e executar aplicações de acordo com a mensagem recebida. Neste caso, pode-se dizer que o *IntentFilter* nada mais é do que um filtro de uma intent(intenção), onde as mensagens são filtradas por ação e categoria e é ele quem decide se a mensagem daquela intent lhe interessa ou não, caso sim, a activity é executada. Os filtros podem estar configurados para executar uma activity ou um *Broadcast Receiver.*

A classe Broadcast *Receiver* não faz uso de uma interface gráfica e é utilizada para responder a determinados eventos enviados por uma intent. Seu objetivo é receber uma mensagem intent e processá-la em segundo plano sem atrapalhar o usuário. Uma aplicação pode, por exemplo, utilizar um broadcast receiver para ser avisada quando o dispositivo ao receber uma mensagem SMS, uma ligação telefônica ou qualquer outra ação customizada e, com base nessa informação, realizar algum tipo de processamento.

2.2.4.3. Services

Os S*ervices* (serviços) são códigos executados em segundo plano e que não dispõem de interface gráfica. Normalmente são utilizados para tarefas que podem consumir muito tempo para execução, sem comprometer a interação do usuário com alguma activity. Tocar uma música ou fazer o download de um arquivo são exemplos de funcionalidades que podem ser implementadas utilizando *services*.

2.2.4.4. Content Provider

O Android permite o desenvolvimento de aplicações com banco de dados para armazenamento de informações. Contudo, não existe uma forma de o banco de dados ser compartilhado entre diferentes aplicações (pacotes).

Para isso, foi criada a classe *Content Providers* ou Provedores de Conteúdo que permite que outras aplicações tenham acesso aos dados armazenados em um banco de dados, onde podem consultar, inserir, alterar e excluir tais informações.

A plataforma Android possui uma série de provedores de conteúdo nativos, como, consultar os contatos da agenda, visualizar os arquivos, imagens e vídeos disponíveis no dispositivo. Um exemplo bem claro disto é a aplicação nativa de gerenciamento de contatos do Android. Aplicações de terceiros podem utilizar um content provider para compartilhar os contatos armazenados no celular.

2.3. Uso dos Dispositivos Móveis no Treinamento Esportivo

Ao se falar em treinamento, logo vem à mente a imagem de alguém realizando muitas repetições de uma determinada atividade ou tarefa. Essa situação ou imagem da repetição passou por muitas evoluções e transformações nos últimos 20 anos. A principal delas foi a mudança da quantidade para a qualidade das repetições, tendo como consequência os resultados obtidos pelos praticantes.

Barbanti (1997) define o treinamento esportivo como um conjunto de normas organizadas que visam ao desenvolvimento e ao aperfeiçoamento individual, com o objetivo de aumentar os rendimentos físico, psicológico e cognitivo.

Segundo Nabil Ghorayeb (2013),

> Superar vários recordes mundiais parecia impossível, agora acontece seguidamente, as marcas registradas anteriormente, simplesmente viram “pó”. O que está acontecendo? Sem dúvida tudo mudou os novos materiais, os pisos das arenas, a vestimenta, o calçado esportivo, varas do salto, bolas de muitos esportes (futebol, basquete, vôlei, tênis), às novíssimas e incrementadas piscinas que absorvem as pequenas ondas que se formam na passagem dos nadadores... o esporte mudou!

Mas não foi somente o esporte que mudou, a ciência progrediu e o treinamento esportivo está cada vez mais atrelado a tecnologia, que vem trazendo benefícios essenciais para os atletas e principiantes. No futebol de 30 anos atrás, por exemplo, um

jogador percorria cinco a seis quilômetros por partida, atualmente chega ao dobro! Quando Rivelino lançavam uma bola, quem a recebia tinha tempo de “matar a bola” e sair jogando. Hoje após um lançamento onde a bola vai cair, chegam dois a três atletas para dividir a jogada com maior vigor possível.

“Num mundo cada vez mais *high tech*[10] o treinamento esportivo não poderia ficar imune a este processo. E a tecnologia cada vez mais está presente em praticamente todos os esportes.” (LENZI, 2014).

Treinadores e atletas estão cada vez mais empenhados em alcançar o desempenho máximo através de inovações que melhorem as práticas. Assim, as facilidades tecnológicas tornam possíveis a treinadores e atletas obter, analisar e integrar informações e recursos de maneira eficiente e efetiva aperfeiçoando o treinamento e a tomada de decisões. Portanto, é muito comum ver dispositivos móveis como notebook, tablet ou outras ferramentas tecnológicas nas mãos de treinadores e preparadores físicos.

Alguns esportes fazem mais uso de tal tecnologias do que outros. Porém, praticamente todos eles fazem uso de alguma ferramenta tecnológica ao se falar em alto rendimento.

Por fim, atualmente a tecnologia caminha lado a lado com o esporte. No contexto do treinamento esportivo, os dispositivos móveis tem sido usados para auxiliar o educador físico no gerenciamento de treinamentos. Isso permite que o educador não fique limitado a um lugar específico e que possa coletar e analisar informações do aluno durante a execução dos treinos utilizando por exemplo, um smartphone ou tablet.

2.4. Treinamento Intervalado de Alta Intensidade com o uso de Beep

O Treinamento Intervalado de Alta Intensidade (*High Intensity Interval Training*) ou HIIT é uma forma de exercício cardiovascular, caracterizado pela alternância de períodos curtos de exercícios anaeróbicos intensos com períodos menos intensos de recuperação. Sessões de HIIT podem variar de 7 a 20 minutos. É geralmente aplicado em muitos

[10] ***High tech*** (em português, alta tecnologia) refere-se à tecnologia considerada de ponta.

esportes como o ciclismo, corrida, remo, artes marciais mistas (AMM), mais conhecidas pela sigla MMA (do inglês: *mixed martial arts*), escalada, entre outros.

Conforme Stoppani (2012), o HIIT foi desenvolvido há décadas por técnicos de atletismo para preparar melhor os corredores e era conhecido pelo nome sugestivo de treinamento "Fartlek", junção das palavras suecas para velocidade (*fart*) e brincar (*lek*). Então, "Fartlek" quer dizer "brincar de velocidade", o que descreve bem o método HIIT.

De acordo com Ross e Leveritt (2001), dependendo da intensidade do treino, um esforço pode durar de poucos segundos até vários minutos, seguidos de alguns minutos de descanso ou de exercício de baixa intensidade. Diferentemente do treinamento de força, no qual se realizam esforços breves e intensos contra uma resistência pesada para aumentar a massa muscular.

Assim sendo, Isa Bragança, da Sociedade Brasileira de Cardiologia, citada na revista VEJA (2013, p.81) acrescenta, "Caíram por terra nossas tradicionais orientações dos 45 minutos de caminhada diária. Já está provado que o treinamento intenso e intervalado traz os mesmos resultados que os outros, que demandam muito tempo".

Com isso, o HIIT vem despertando o interesse da maioria semi-sedentária que entende a importância de se exercitar, mas que dispõem de pouco tempo, paciência ou fôlego. "Mais forte, mais rápido, mais saudável e em menos tempo." Esses são apenas alguns benefícios que o HIIT pode resultar no corpo humano.

De acordo com Astorino (2012), fazer treino intervalado de alta intensidade irá resultar em melhorias significativas no seu VO_2máx.

O termo VO_2máx, significa a capacidade máxima do corpo para transportar e utilizar o oxigênio durante o exercício. É uma grande medida de condicionamento utilizada por todo tipo de profissional da saúde para com seus atletas. Assim, quanto maior VO_2 máximo, melhor o condicionamento físico. Um maior VO_2máx também significa a possibilidade de treinar em maiores intensidades por períodos mais longos de tempo.

Os treinos curtos e intensos trabalham perto da frequência cardíaca máxima, proporcionando melhor capacidade atlética e condicionamento, melhora o metabolismo da glicose e auxilia na queima de gordura.

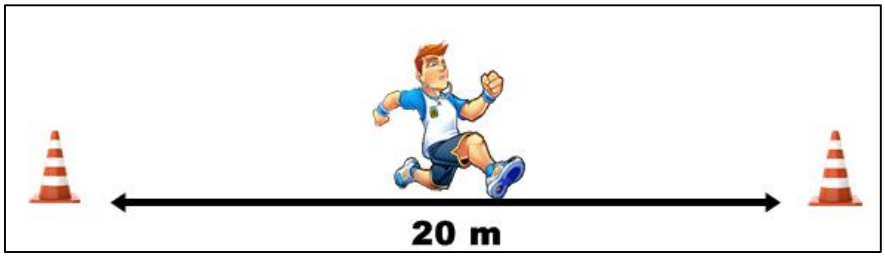
Além disso, Gibala et al. (2012) afirmam que o HIIT é um potente estímulo de treinamento. Pois, somente seis sessões de HIIT durante duas semanas, ou um total de

apenas aproximadamente 15 minutos de exercício muito intenso (gasto de energia acumulado de ~600 kJ ou ~143 cal), podem aumentar a capacidade oxidativa do músculo esquelético e melhorar o desempenho durante tarefas que dependem fundamentalmente do metabolismo energético aeróbico.

Atualmente, o treinamento Intervalado de Alta Intensidade vem sendo praticado com o auxílio do teste com sinal sonoro mais conhecido como vai-e-vem de Léger e Lambert, também denominado como "*Beep test*", yoyo test, 20 m suttle-run test, entre outros. Foi criado por Léger e Lambert em 1982 e tem como objetivo caracterizar a aptidão cardiorrespiratória de indivíduos atletas e não atletas. A mensuração se dá através da medida indireta do VO_2máx. Esse teste é possivelmente, o teste de aptidão de resistência geralmente mais usado ao redor do mundo, dado que é simples de realizar, requer o mínimo de equipamento e de baixo custo.

Segundo Duarte (2001), o teste de vai-e-vem de Léger e Lambert consiste em uma prova de ida e volta, onde uma área de 20 metros é demarcada, como ilustra a Figura 2.5.

Figura 2.5 – Protocolo do teste vai-e-vem de Léger e Lambert.



Fonte: Elaborada pelo autor.

O teste possui 21 estágios que correspondem aproximadamente a 21 minutos de teste. O indivíduo deverá percorrer este trajeto sendo a velocidade do exercício controlada por um sinal sonoro. Grandes grupos podem ser testados ao mesmo tempo e o controle se dá através de um áudio gravado para este fim, que emite "bips" com intervalos específicos para cada estágio do teste, como mostra a Tabela 2.1.

Tabela 2.1 – Especificações para realização do teste.

Estágios N.º	Velocidade (Km/h)	Tempo entre BIPS (em segundos)	N.º de idas e voltas (estágio completo)
1	8,5	9,000	7
2	9,0	8,000	8
3	9,5	7,579	8
4	10,0	7,200	8
5	10,5	8,858	9
6	11,0	6,545	9
7	11,5	6,261	10
8	12,0	6,000	10
9	12,5	5,760	11
10	13,0	5,538	11
11	13,5	5,333	12
12	14,0	5,143	12
13	14,5	4,966	12
14	15,0	4,800	13
15	15,5	4,645	13
16	16,0	4,500	13
17	16,5	4,364	14
18	17,0	4,235	14
19	17,5	4,114	15
20	18,0	4,000	15
21	18,5	3,892	15

Fonte: DUARTE; DUARTE (2001, p. 10).

O avaliado deve cruzar com pelo menos um dos pés a linha do cone ao ouvir o “bip” e voltar em sentido contrário com o mesmo objetivo (DUARTE; DUARTE, 2001). Se ao tocar o bip o indivíduo não estiver cruzado com um dos pés a linha do cone ele deverá acelerar o ritmo, caso consiga recuperar, poderá permanecer no teste até que falhe por duas vezes consecutivas, mas se caso não consiga acompanhar o ritmo do bip, e não chegue à linha do cone do outro lado antes do próximo bip o teste será finalizado e se anotará o último estágio obtido, para que possa ser mensurado o VO_2máx.

No primeiro estágio a velocidade é de 8,5 Km/h e o tempo entre os bips é de 9 segundos. A cada estágio a velocidade é acrescida 0,5 Km/h, e o tempo entre os bips é cada vez menor. O indivíduo realiza em cada estágio de 7 a 15 idas e vindas de 20 metros. A duração do teste depende da aptidão física de cada indivíduo, iniciando com uma intensidade baixa e se tornando mais intensa.

Para a mensuração do VO_2máx existem diferentes fórmulas e protocolos dependendo da população a ser estudada. Este trabalho utiliza a proposta de se obter o VO2máx em ml/kg/min, através das equações publicadas por Léger et al (1988), citado por Duarte (2001, p. 10) que estão descritas na Tabela 2.2.

Tabela 2.2 – Equações de predição do VO_2máx. em (ml/kg/min) no teste aeróbico de corrida de Vai-e-Vem de 20 metros de Léger.

População	Fórmula
Pessoas de 6 a 18 anos	y = 31,025 + (3,238 X) – (3,248 A) + (0,1536 AX).
Pessoas de 18 anos ou mais	y = - 24,4 + 6,0 X.

Onde y= $V0_2$ em ml/kg/min.; X = velocidade em km/h (no estágio atingido); A= idade em anos.
Fonte: DUARTE; DUARTE (2001, p. 10).

O *Team Test Beep* desenvolvido pela Bitworks[11] em parceria com a Topend Sports[12] é um exemplo de software desenvolvido para treinamento com uso de beep.

[11] www.bitworks.com.br
[12] http://www.topendsports.com/

3. MÉTODO E FERRAMENTAS UTILIZADAS

Neste capítulo são apresentados a metodologia, ferramentas e tecnologias utilizadas para o desenvolvimento de aplicativo móvel para treinamento esportivo. Tais ferramentas e tecnologias podem ser instaladas e utilizadas em qualquer sistema operacional: Windows, Mac OS X e Linux.

A seção 3.1 apresenta a metodologia de desenvolvimento Scrum[13]. Já na seção 3.2 são apresentadas as ferramentas utilizadas, são elas: o Java Development Kit (JDK), as ferramentas do kit de desenvolvimento SDK (*Software Development Kit*) do Android, o IDE[14] Eclipse, o plug-in Android Development Tools (ADT) e o Banco de Dados SQLite. Por fim, é abordada a configuração do ambiente de desenvolvimento.

3.1. Método

Para o desenvolvimento do sistema, foi utilizada a metodologia de desenvolvimento de software Scrum, por ser uma metodologia ágil e ser mais adequada na melhoria contínua dos processos, com organização diária para o alcance da meta definida, evita falhas na elaboração permitindo respostas rápidas às mudanças e alta produtividade, como no caso dessa proposta.

Desenvolvido por Ken Schwaber e Jeff Sutherland, o Scrum é um *framework* estrutural que está sendo usado para gerenciar o desenvolvimento de produtos complexos desde o início de 1990.

O Scrum não é um processo ou uma técnica para construir produtos; em vez disso, é um framework dentro do qual você pode empregar vários processos ou técnicas. O Scrum deixa claro a eficácia relativa das práticas de gerenciamento e desenvolvimento de produtos, de modo que você possa melhorá-las (SCHWABER e SUTHERLAND, 2013).

No *framework* Scrum, os projetos são divididos em ciclos ou interações com duração de 1 a 4 semanas chamados de Sprints. O Sprint representa um tempo definido

[13] https://www.scrum.org/
[14] **IDE**, do inglês ***Integrated Development Environment*** ou **Ambiente Integrado de Desenvolvimento.**

dentro do qual um conjunto de atividades devem ser executadas. No caso deste trabalho, foram definidos 3 Sprints com duração de 4 semanas cada.

No início de cada Sprint, faz-se uma reunião de planejamento (Sprint Planning Meeting), na qual o representante dos envolvidos (*Product Owner*) prioriza todos os itens da lista de funcionalidades a serem implementadas (*Product Backlog*). No caso deste trabalho, os envolvidos foram o professor do curso de Sistemas de Informação, Alessandro Vivas Andrade e o professor do curso de Educação Física, Fernando Joaquim Gripp Lopes.

Deste modo, os itens funcionais elencados foram cadastrar uma pessoa (atleta ou principiante) pelo seu nome, com informações de cadastro de pessoas físicas (CPF), idade, sexo, massa corporal (kg), estatura (m) e modalidade. Além deste, os itens listar as pessoas cadastradas, consultar pessoa, alterar dados da pessoa, excluir pessoa, testar a pessoa incluindo informações sobre nome, nível, shuttle, tempo, velocidade, distância percorrida, VO_2máx. Também os itens selecionar pessoa para o teste, listar testes, listar testes por pessoa, salvar teste e excluir teste. O teste segue ou obedece uma sequência de botões pressionados (iniciar e pausar).

A seguir, são selecionadas as funcionalidades que o desenvolvedor do aplicativo será capaz de implementar durante o Sprint que se inicia. As funcionalidades selecionadas da lista são transferidas para a lista de tarefas (Sprint Backlog).

Inicialmente, para o primeiro Sprint foram selecionadas e transferidas para a lista de tarefas as seguintes funcionalidades: cadastrar pessoa, listar as pessoas cadastradas e consultar pessoa. Para o segundo Sprint foram selecionadas alterar dados da pessoa e excluir pessoa. Já para o terceiro Sprint foi selecionada a funcionalidade testar a pessoa, selecionar pessoa para o teste, listar testes, listar testes por pessoa, salvar teste e excluir teste.

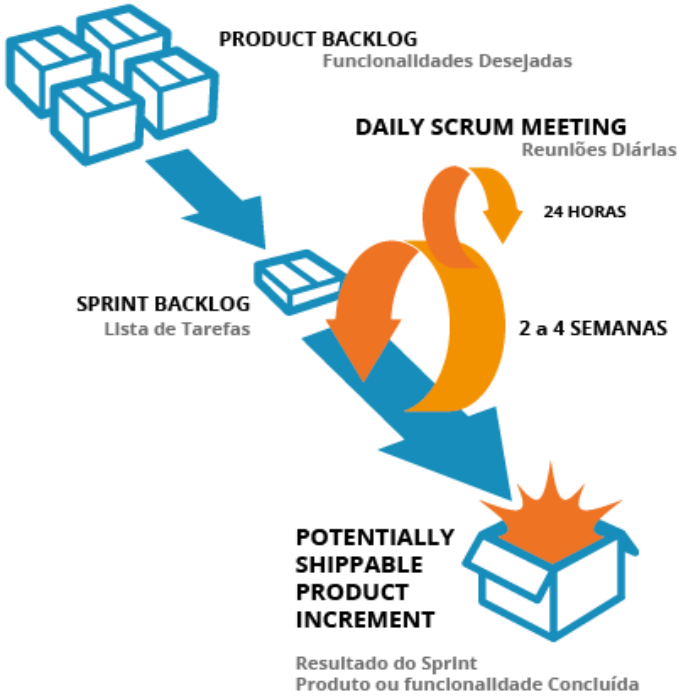
A Figura 3.1 mostra que a lista de tarefas pode ser organizadas em um “quadro de trabalho”, também chamado de *Kanban.* No quadro as tarefas são separadas basicamente em quatro estados (variando de acordo com cada projeto): a fazer, em andamento, em testes e concluído.

Figura 3.1 - Quadro de Trabalho.

Fonte: Elaborada pelo autor.

Durante cada ciclo ou Sprint foram realizadas breves reuniões de no máximo 15 minutos onde o desenvolvedor diz o que fez, o que pretende fazer e se existe algo dificultando seu trabalho. Essa reunião também pode ser chamada de *Daily Scrum*. A Figura 3.2 mostra por completo o funcionamento da metodologia Scrum.

Figura 3.2 - Funcionamento do Scrum.



Fonte: Site BRQ IT Services[15].

[15] Disponível em: <http://www.brq.com/metodologias-ageis/>. Acesso em Junho de 2014.

Além disso, ao final de cada ciclo, foram expostas as funcionalidades implementadas através de versões beta do aplicativo, mostrando o que foi alcançado neste ciclo e por fim, foi feita uma retrospectiva para identificar o que deu certo e possíveis melhorias, iniciando o próximo ciclo.

3.2. Ferramentas

Para iniciar o desenvolvimento de aplicações para o Android, a primeira ferramenta necessária é o Java Development Kit (JDK)[16]. Prefira a versão mais atual, pois o Android requer no mínimo JDK 5. Esse Kit de desenvolvimento é indispensável para se trabalhar com Java.

Java é uma linguagem de programação orientada a objeto lançada pela Sun Microsystems em 1995. Ela é compilada para um bytecode que é executado por uma máquina virtual. Esta linguagem será utilizada para o desenvolvimento da aplicação.

Além desta linguagem, também será utilizada a linguagem de marcação XML (*Extensible Markup Language*). Ela é utilizada na criação de documentos com dados organizados em uma hierarquia, como texto, banco de dados ou desenhos vetoriais. Ela trabalha com o conceito de tags (rótulos) para marcação de um determinado comando ou estrutura, como mostra a Figura 3.3. No Android o XML é muito utilizado para a criação da interface gráfica (tela) da aplicação.

Figura 3.3 - Exemplo de código em XML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bilhete>
<para>José</para>
<de>Maria</de>
<título>Lembrete</título>
<corpo>Não me esqueça neste fim-de-semana!</corpo>
</bilhete>
```

Fonte: Site da UFBA[17].

[16] Disponível em: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Acesso em Junho de 2014.
[17] Disponível em: <http://www.clem.ufba.br/tuts/xml/c03.htm>. Acesso em Julho de 2014.

Depois de instalar o Java JDK, é preciso instalar o Android SDK, que é o Kit de Desenvolvimento de aplicações para Android. Ele possui todos os recursos necessários para compilar as aplicações e emular diversas versões do Android.

A instalação do Android SDK é bem simples. Basta baixar o pacote Eclipse ADT (Android Developer Tools)[18] e descompactá-lo. Ele inclui os componentes essenciais do Android SDK para Windows e uma versão do IDE Eclipse. Caso preferir utilizar outro IDE, será necessário baixar as ferramentas do Android SDK a parte.

Estão presentes no pacote Eclipse ADT os seguintes itens:

- Eclipse + ADT plug-in
- Android SDK Tools
- Última versão da plataforma Android
- Última versão do emulador

O Android SDK fornece as bibliotecas de API (*Application Programming Interface*) e ferramentas necessárias para construir, testar e depurar aplicativos para o Android. As ferramentas do SDK mais importantes incluídas:

- **Android SDK Manager (Android SDK) e AVD[19] Manager (AVD android):** permitem gerenciar AVDs, projetos e componentes instalados do SDK.
- **Emulador (emulador):** ferramenta de emulação de dispositivo baseado em QEMU usado para criar, depurar e testar aplicativos.
- **Dalvik Debug Monitor Server (DDM):** permite depurar aplicativos Android.

Para auxiliar o desenvolvimento das aplicações, é utilizado o IDE Eclipse. O Eclipse é um Ambiente Integrado de Desenvolvimento (do inglês *Integrated Development Environment - IDE*) Java. Porém, suporta várias outras linguagens como C/C++, PHP, ColdFusion, Python, entre outras (ECLIPSE, 2014). Feito em Java, ele segue o modelo *open source* (código aberto). Sua instalação é realizada através da execução do arquivo *eclipse.exe* que se encontra no pacote Eclipse ADT, já configurado com plug-in para Android.

---

[18] Disponível em:< http://developer.android.com/sdk/index.html>. Acesso em Junho de 2014.
[19] **AVD**, do inglês ***Android Virtual Device*** ou simplesmente **configuração virtual de um celular**.

O plug-in Android Development Tools (ADT) vem encapsulado no pacote Eclipse ADT. Serve para integrar o emulador ao Eclipse, permite iniciar o emulador diretamente dentro do Eclipse, instalando automaticamente a aplicação no emulador. Também é possível instalar a aplicação em um celular real, basta instalar o drive USB do celular no computador e plugá-lo na porta USB. Além disso, o plug-in maximiza os recursos do Eclipse facilitando o desenvolvimento, os testes e a compilação do projeto.

Com o intuito de testar a aplicação desenvolvida, é utilizado o emulador que possibilita a instalação e execução de aplicações que podem ser instaladas no computador de desenvolvimento sem o uso de dispositivo móvel (PEREIRA e SILVA, 2009).

De acordo com Lecheta (2010, p. 368), o Android vem integrado com o banco de dados o SQLite[20], o que facilita o uso de banco de dados nas aplicações desenvolvidas. O SQLite é um banco de dados autocontido, compacto, com suporte nativo no android e sem necessidade de instalação ou configuração. Um banco de dados completo é armazenado em um único arquivo (*.db*) que é multiplataforma.

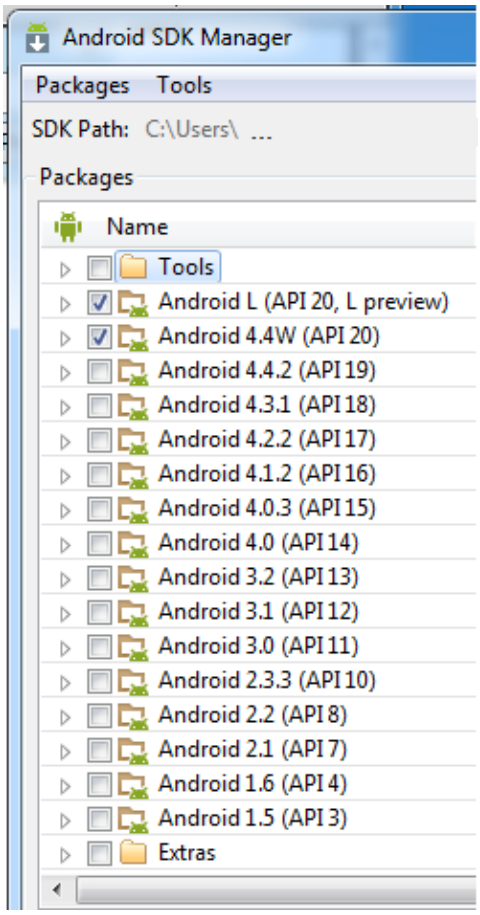
O banco de dados pode ser criado de diversas formas. Neste trabalho ele é criado utilizando a API para o SQLite, onde o banco é criado diretamente dentro do próprio código-fonte.

3.2.1. Configurações do ambiente de desenvolvimento

Antes de criar o primeiro projeto Android, é preciso fazer algumas configurações relativas ao Android SDK. A princípio, é necessário abrir o Android SDK Manager, que pode ser acessado pelo ícone  na barra de ferramentas dentro do Eclipse. O SDK Manager lista os pacotes disponíveis, como apresenta a Figura 3.4. A partir dessa lista, é preciso selecionar os pacotes referentes as versões do Android em que se deseja desenvolver, para que os mesmo sejam instalados.

[20] Disponível em: http://www.sqlite.org/

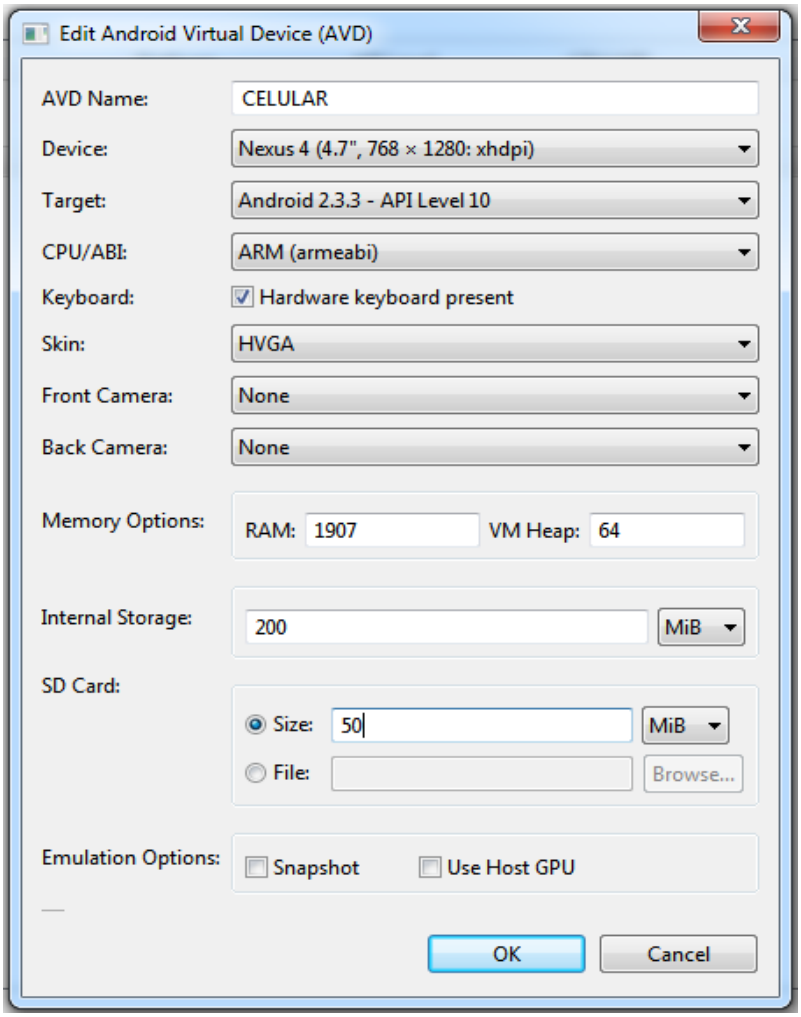
Figura 3.4 - Lista dos pacotes disponíveis com as versões do Android.



Fonte: Captura de tela do Android SDK Manager.

O próximo passo é criar uma máquina virtual do Android (AVD) ou emulador para testar a aplicação criada sem necessidade de se ter um dispositivo físico. Podem ser criadas quantas máquinas desejar. Uma AVD é criada através da aba *Tools* > *Manager AVDs* >*New*. A Figura 3.5 mostra como configurar os dados do emulador para uma AVD básica. A configuração também pode ser personalizada de acordo com o computador utilizado.

Figura 3.5 - Configuração do AVD (*Android Virtual Device*).



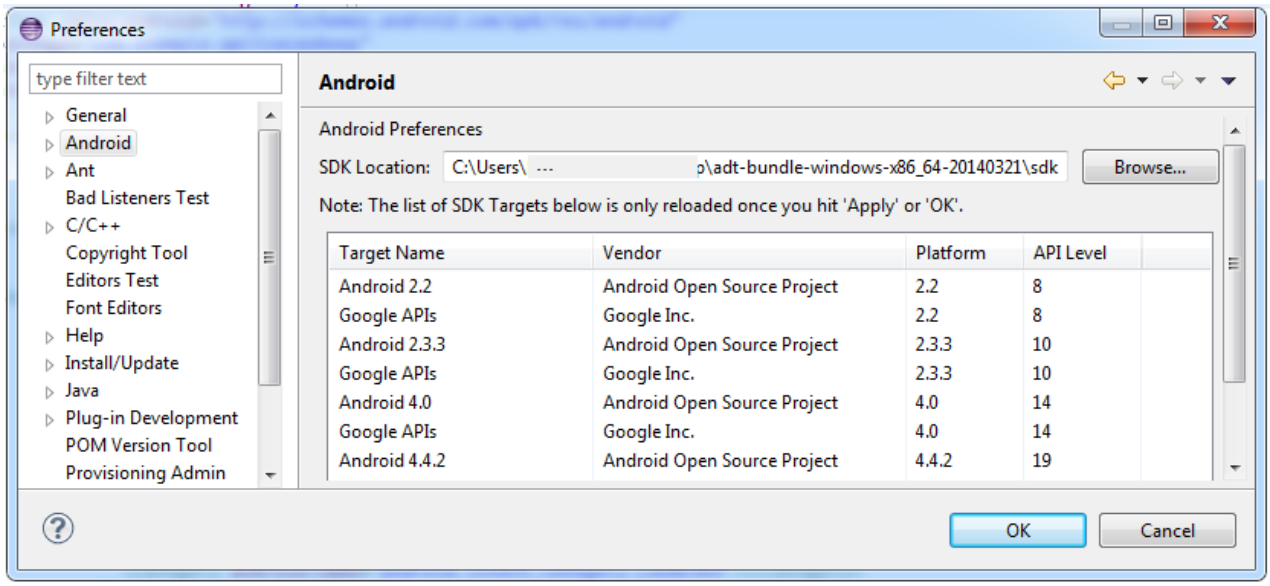
Fonte: Captura de tela do AVD.

- **AVD Name:** nome do emulador.
- **Device:** Dispositivo emulado.
- **Target:** versão do Android a ser emulada.
- **CPU/ABI:** processador.
- **Keyboard:** opção para utilizar o teclado do hardware presente.
- **Skin:** designer do dispositivo. Geralmente opta-se por HVGA, pois é uma skin média.
- **Front Camera** e **Back Camera:** opções de câmera. A princípio não é necessário.
- **Memory Options:** opções de memória. Geralmente o default é o suficiente.
- **Internal Storage:** armazenamento interno. Geralmente o default é o suficiente.

- **SD Card:** a princípio não é necessário, mas caso trabalhe com banco de dados e deseje manter os dados após reiniciar o emulador, crie um SD Card de 50 Mb (Mega), sendo suficiente para maioria dos casos.

Depois clicar em *Create AVD* e fechar a janela do *Android SDK*. A seguir, para que o plug-in do Eclipse possa encontrar o emulador do Android e todas as plataformas instaladas no SDK, é necessário fazer a configuração das preferências do Eclipse. Para isso, no Eclipse, entre no menu *Window>Preferences* e selecione a opção *Android*. Digite o caminho correto da instalação do SDK, conforme a Figura 3.6 e clique em *OK*.

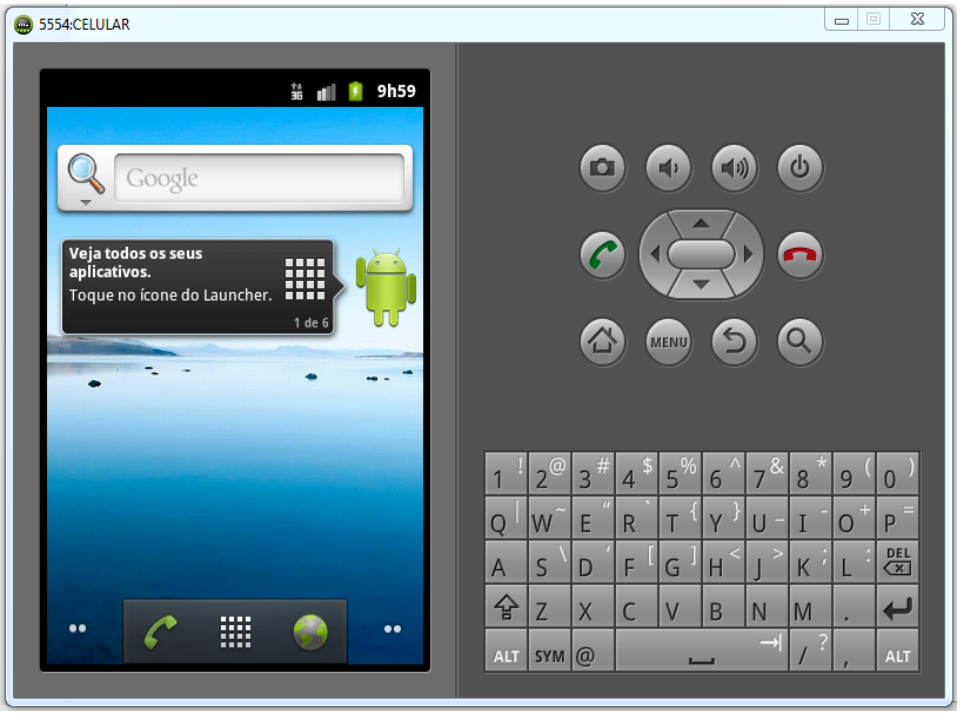
Figura 3.6 - Configuração do SDK nas preferências do Eclipse.



Fonte: Captura de tela de Preferências do Eclipse.

A Figura 3.7 mostra o emulador Android criado. Por fim, o ambiente de desenvolvimento está pronto para criação e execução de aplicações Android.

Figura 3.7 - Emulador Android.



Fonte: Captura de tela do emulador Android.

4. APLICAÇÃO PROPOSTA

Neste capítulo são apresentas as etapas para o desenvolvimento do aplicativo móvel para treinamento esportivo com foco no treinamento intervalado de alta intensidade apelidado de “Beep”. A seção 4.1 apresenta o sistema operacional utilizado e versão escolhida. Já a seção 4.2 apresenta o levantamento de requisitos. A seção 4.3 mostra de modo geral a implementação do aplicativo, dando ênfase para as principais classes.

4.1. Sistema Operacional e Versão Escolhida

Esta seção apresenta os fundamentos para a escolha do sistema operacional e versão mínima 4.0 do Android.

4.1.1. Sistema Operacional

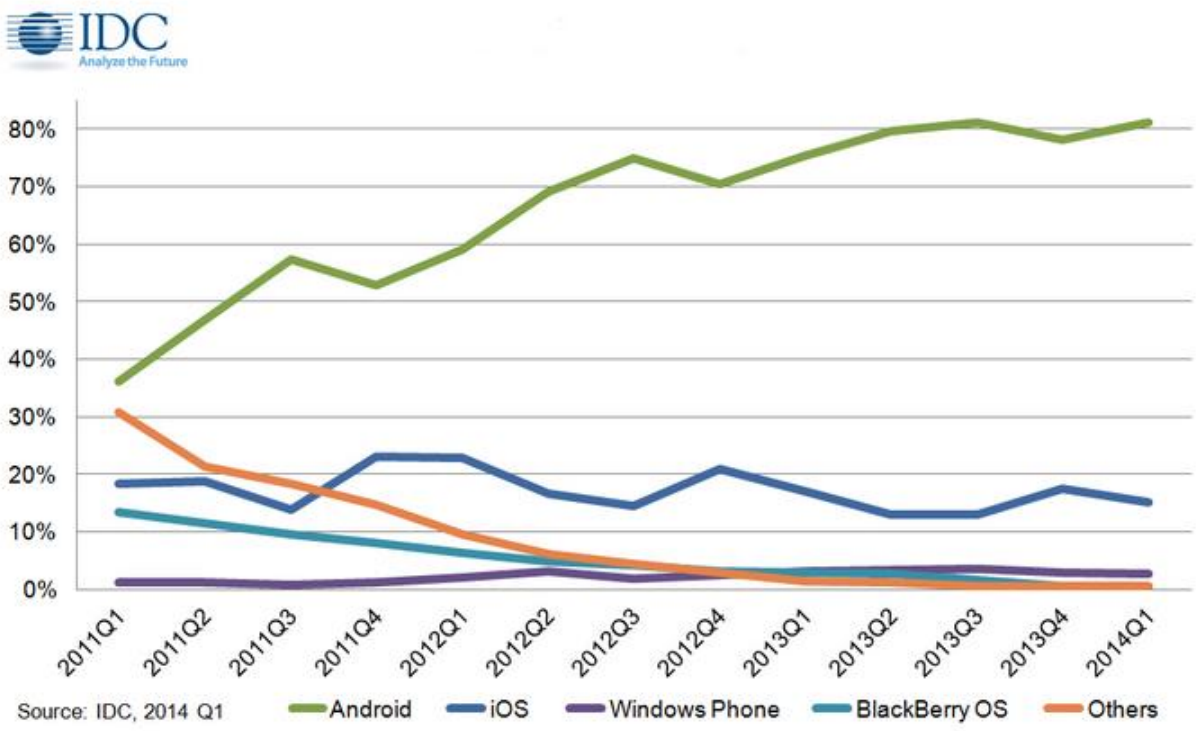
O aplicativo Beep foi desenvolvido para smartphones e tablets com base no sistema operacional Android, devido ao fato deste ser o sistema operacional de mais de um bilhão de smartphones e tablets em todo o mundo[21].

De acordo com a International Data Corporation (IDC), no primeiro trimestre de 2014 (1Q14) foram vendidos 287.8 milhões de smartphones, onde o sistema operacional Android está presente em 81.1% dos aparelhos. Já o iOS da Apple veio em segundo, com 15.2% e em terceiro lugar com 2.7% o Windows Phone seguido pelo BlackBerry com apenas 0.5%. Segundo o Gráfico 4.1, dentre os sistemas operacionais para smartphones mais expressivos (Android da Google, iOS da Apple, Windows Phone da Microsoft e BlackBerry da Research in Motion (RIM)), o Android é o que possui melhor posicionamento no mercado mundial.

Além disso, o sistema operacional foi escolhido por ser um sistema aberto em constante desenvolvimento e vasto referencial teórico.

[21] Disponível em: http://www.android.com/phones-and-tablets/. Acesso em 11 de Junho de 2014.

Gráfico 4.1 - Quota global de mercado detida pelos principais sistemas operacionais de smartphones nas vendas para usuários do primeiro trimestre de 2011 até o 1º trimestre de 2014.



Período	Andróide	iOS	Windows Phone	BlackBerry OS	Outros
Q1 2014	81,1%	15,2%	2,7%	0,5%	0,6%
Q1 2013	75,3%	17,1%	3,2%	2,9%	1,5%
Q1 2012	59,2%	23,0%	2,0%	6,3%	9,5%
Q1 2011	36,1%	18,3%	1,2%	13,6%	30,8%

Fonte: Site do IDC, 2014 Q1[22].

4.1.2. Versão

A versão escolhida para o desenvolvimento do aplicativo foi baseada na estatística de utilização de cada versão do Android, como mostra a Tabela 4.1. Esta estatística

[22] Disponível em:<http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em Julho de 2014

revela dados sobre o número de dispositivos que executam uma determinada versão do Android e que estão ativos no Google Play. Assim, foi escolhida a versão mínima 4.0 do Android, pois atenderá desde a versão 4.0 até a versão mais recente 5.0 Lollipop, totalizando mais de 89,6% dos dispositivos ativos no mercado.

Tabela 4.2 - Estatística de utilização de cada versão do Android.

Version	Codename	API	Distribution
2.2	Froyo	8	0.6%
2.3.3 - 2.3.7	Gingerbread	10	9.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	8.5%
4.1.x	Jelly Bean	16	22.8%
4.2.x		17	20.8%
4.3		18	7.3%
4.4	KitKat	19	30.2%

Fonte: Google Dashbords, 2014[23].

4.2. Levantamento dos Requisitos

Para o levantamento dos requisitos do sistema utilizou-se a técnica de *Brainstorming (*tempestade de ideias*).* Foram realizadas reuniões envolvendo o professor Alessandro Vivas Andrade e o professor do curso de Educação Física, Fernando Joaquim Gripp Lopes, onde permitiu que os mesmos pudessem dar sugestões e explorassem novas ideias para o aplicativo a ser desenvolvido.

Durante o desenvolvimento, foram mostradas ao professor Fernando Joaquim Gripp Lopes versões alfa e beta do aplicativo Beep buscando validar os requisitos sugeridos nas reuniões.

[23] Disponível em: <http://developer.android.com/about/dashboards/index.html#Platform>. Acesso em Dezembro de 2014.

4.2.1. Requisitos Essenciais

- Cadastrar uma nova pessoa (atleta ou principiante) com nome, CPF (Cadastro de Pessoas Física), idade, sexo, massa corporal (kg), estatura (m) e modalidade.
- Excluir pessoa.
- Testar a pessoa obtendo informações sobre o nível, shuttle, tempo, velocidade, distância percorrida e VO_2máx.
- Salvar teste.
- Excluir teste.
- Permitir a listagem de todos os testes realizados.
- Permitir a listagem dos testes de uma determinada pessoa.

4.2.2. Requisitos Desejáveis

- Permitir a listagem de todas as pessoas cadastradas.
- Consultar o cadastro de alguma pessoa.
- Alterar os dados de alguma pessoa.
- Selecionar pessoa para realizar o teste.

4.3. Implementação

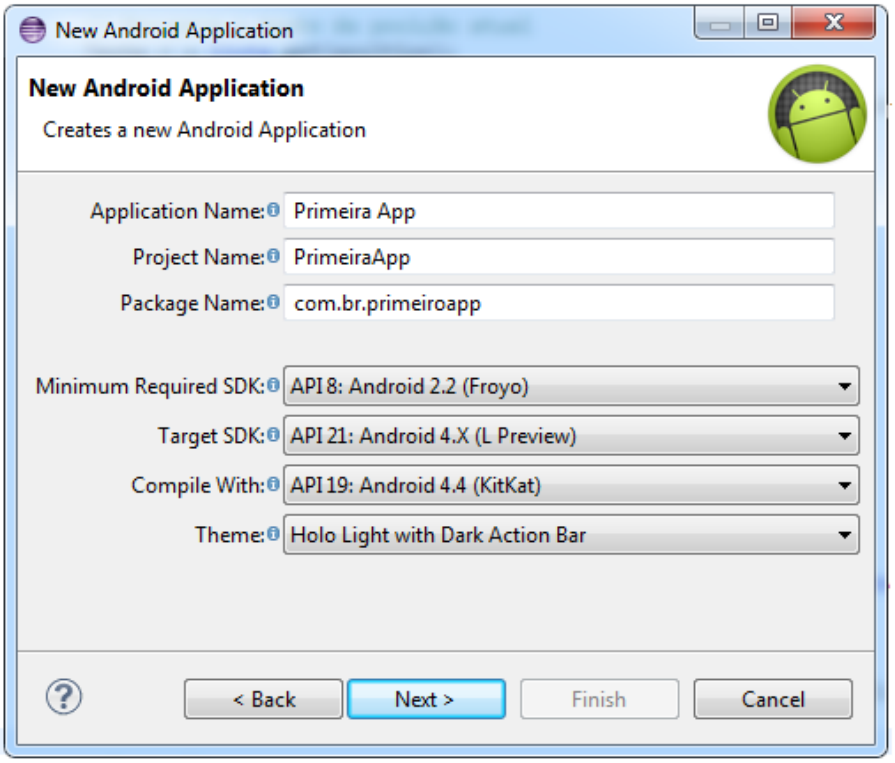
Esta seção abordará todos os passos envolvidos na construção e implementação do aplicativo móvel *Beep*.

4.3.1. Criação do Projeto Beep

A criação de um projeto Android no Eclipse é muito simples. Após a instalação e configuração de todas as ferramentas necessárias para a criação de uma aplicação Android, é importante seguir algumas etapas. Essas etapas serão demonstradas através da criação do *Projeto Beep*.

Clique no menu *File>New>Android Application Project*. Na tela seguinte serão preenchidos os dados de configuração do projeto da aplicação, conforme a Figura 4.1.

Figura 4.1 - Criação de um projeto Android.



Fonte: Captura de tela de New Android Project do Eclipse.

- **Application Name:** Nome do aplicativo. Nome que será exibido para o usuário.
- **Project Name:** Nome do projeto.
- **Package Name:** Nome do pacote. O nome do pacote é composto por um endereço invertido do seu site (caso tiver) ou um endereço fictício. O pacote deve ser único, pois é utilizado como identificador para a aplicação na Play Store.
- **Minimum Required SDK:** Menor versão do Android suportada pela aplicação. Quando menor a API, menor será o suporte a algumas funcionalidades. Por exemplo, a API 14 oferece suporte a algumas funcionalidades que não há na API 8. Isso indica também que o aplicativo só será instalado em celulares com o Android 4.0 ou superior.
- **Target SDK:** Indica a maior versão do Android na qual a aplicação é testada.

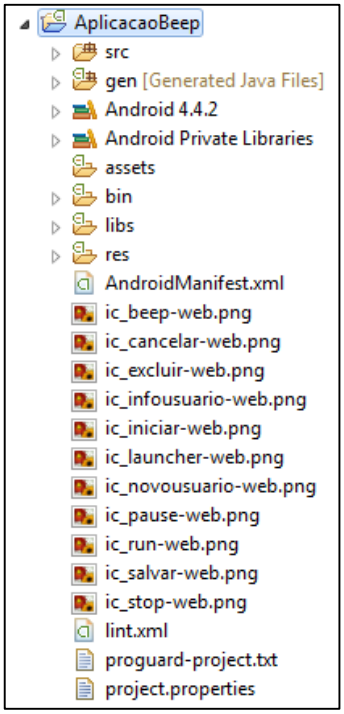
- **Compile With:** Indica com qual versão entre os pacote baixados no SDK Manager a aplicação será compilada. Embora a aplicação possa ser compilada com versões anteriores, é recomendado usar a versão mais recente.
- **Theme:** Especifica o estilo de interface do Android que será usado na aplicação. Este campo é opcional.

Após definir as configurações do projeto, clique em *Next* nas próximas telas e na última em *Finish* para finalizar as configurações e criar o projeto no workspace do Eclipse.

4.3.2. Estrutura de Diretórios

Todo projeto de aplicação para Android possui uma estrutura de diretórios. Quando o projeto é criado, é gerada automaticamente uma estrutura de pastas, onde a primeira que possui o nome do projeto representa a pasta raiz (diretório raiz). A Figura 4.2 apresenta a estrutura de diretórios compactada do projeto *Beep*.

Figura 4.2 - Estrutura de Diretórios do projeto Beep.



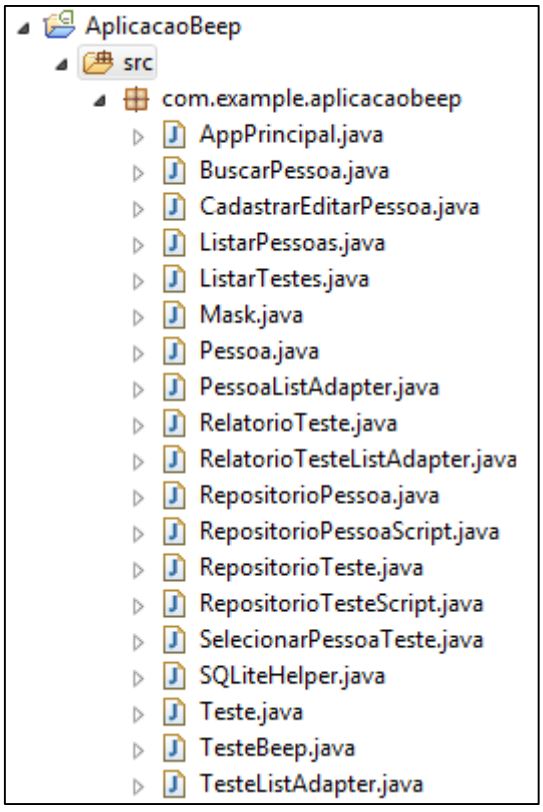
Fonte: Captura do projeto no Eclipse.

Esta pasta raiz contém todos os arquivos do projeto, desde arquivos de codificação até recursos imagens e arquivo de configuração. A pasta raiz possui os seguintes diretórios criados automaticamente: src, gen, assets, bin, libs, res e o arquivo AndroidManifest.xml. Como exemplo, foi utilizada a estrutura de diretórios do projeto Beep.

4.3.2.1. Pasta src

Esta pasta do projeto contém o pacote padrão da aplicação, que organiza todas as classes escritas em Java, ou seja, os arquivos de código fonte da aplicação. Algumas classes.java representam telas da aplicação e são responsáveis por controlar o estado e os eventos da tela. Já outras classes que oferecem alguma funcionalidade para a aplicação, como, a criação de um banco de dados. A Figura 4.3 mostra a pasta src.

Figura 4.3 - Pasta src do projeto Beep.

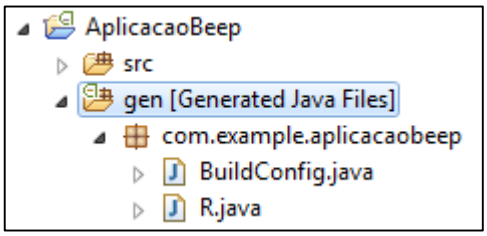


Fonte: Captura da pasta src no Eclipse.

4.3.2.2. Pasta gen

Esta pasta contém as classes e/ou arquivo geradas automaticamente pelo Eclipse. A principal classe desta pasta é a classe *R.java,* pois permite que a aplicação acesse qualquer recurso como arquivos e imagens do projeto. O conteúdo desta pasta nunca deve ser alterado manualmente. A Figura 4.4 mostra a pasta gen.

Figura 4.4 - Pasta gen do projeto Beep.

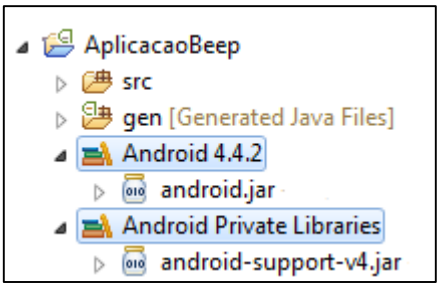


Fonte: Captura da pasta gen no Eclipse.

4.3.2.3. Bibliotecas

O local das bibliotecas é criado automaticamente pelo Eclipse. Este local contém as bibliotecas da aplicação. O aplicativo Beep utiliza a biblioteca de suporte a versão 4. A Figura 4.5 ilustra a biblioteca utilizada na aplicação.

Figura 4.5 - Bibliotecas do aplicativo Beep.

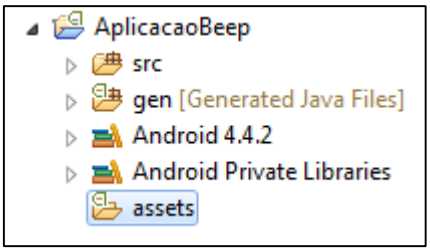


Fonte: Captura das bibliotecas no Eclipse.

4.3.2.4. Pasta assets

Esta pasta contém os arquivos opcionais ao projeto ou recursos extras da aplicação, como músicas, páginas html, arquivos de texto, uma fonte customizada, etc. O projeto beep não fez uso de nenhum arquivo externo, assim não utilizando a pasta assets. A Figura 4.6 exibe a pasta assets.

Figura 4.6 - Pasta assets do aplicativo Beep.

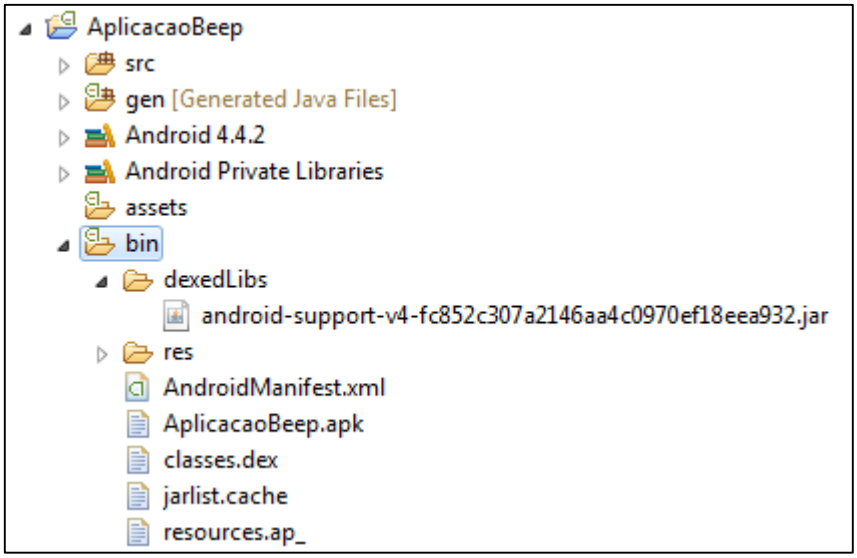


Fonte: Captura da pasta assets no Eclipse.

4.3.2.5. Pasta bin

Pasta onde ficam os arquivos compilados pelo Eclipse. Ou seja, os arquivos gerados no momento da construção (build) do projeto, como o pacote *AplicacaoBeep.apk* (pacote instalável do aplicativo no Android). Recomenda-se a não mexer nestes arquivos. A Figura 4.7 mostra a pasta bin.

Figura 4.7 - Pasta bin do aplicativo Beep.

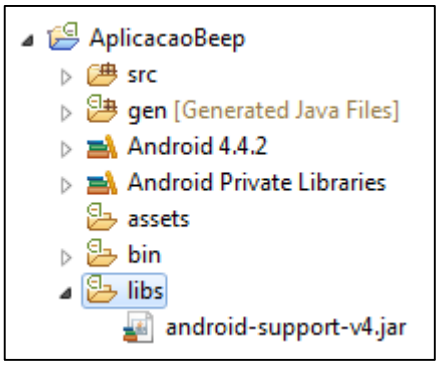


Fonte: Captura da pasta bin no Eclipse.

4.3.2.6. Pasta libs

Esta pasta contém bibliotecas .jar nativas do Android. Esta biblioteca é utilizada caso o desenvolvedor precise interagir diretamente com o sistema operacional Android. A Figura 4.8 mostra a pasta libs.

Figura 4.8 - Pasta libs do aplicativo Beep.



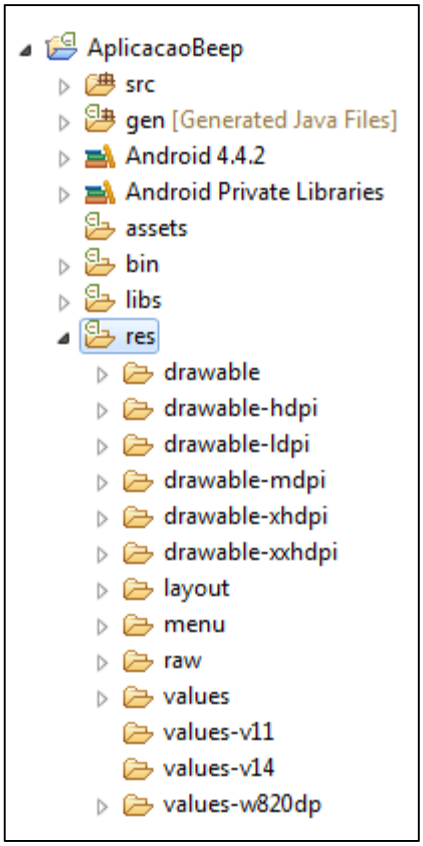
Fonte: Captura da pasta libs no Eclipse.

4.3.2.7. Pasta res

A pasta res abreviatura de *resources*, contém os recursos da aplicação, como imagens, layout de telas, arquivos de internacionalização e também pode conter recursos de áudio. Dentro desta pasta, existem subpastas para cada tipo de recurso, como as pastas drawable, layout, menu, raw e values.

Nas pastas **drawable** contém as imagens da aplicação (png, jpeg ou gif). A Figura 4.9 mostra a divisão das pastas entre drawable-hdpi, drawable-ldpi, drawable-mdpi, drawable-xhdpi e drawable-xxhdpi é usada para guardar as imagens em resoluções diferentes, devido a existência de diversos celulares Android com resolução de tela diferentes.

Figura 4.9 - Pasta res do aplicativo Beep.



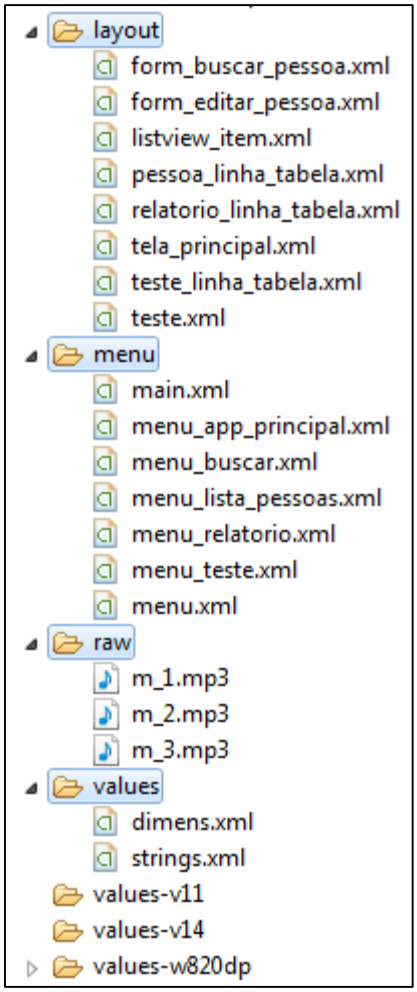
Fonte: Captura da pasta res no Eclipse.

A pasta **layout** é o local onde ficam os arquivos XML que representam as telas da aplicação. A criação dessas interfaces gráficas (telas), como botões, checkbox, entre outros, é feita no arquivo XML, pois deixa o código mais limpo e permite uma maior separação entre a parte visual da aplicação e sua lógica.

A pasta **menu** contém os arquivos XML utilizados para criar os menus da aplicação, onde são definidos os itens que cada menu terá e como são visualizados no menu. O desenvolvedor de aplicações Android além das pasta *assets*, pode optar por armazenar os arquivos de áudio na pasta *res/**raw***. Já a pasta **values** contém os arquivos XML utilizados para a internacionalização da aplicação e outras configurações.

A Figura 4.10 apresenta em *res/layout* as telas do projeto, os menus da aplicação em *res/menu*, os arquivos de áudio em *res/raw* e os arquivos de internacionalização e configuração na pasta *values*.

Figura 4.10 - Arquivo da pasta layout, menu, raw e pasta values do aplicativo Beep.



Fonte: Captura da pasta res/layout e res/values no Eclipse.

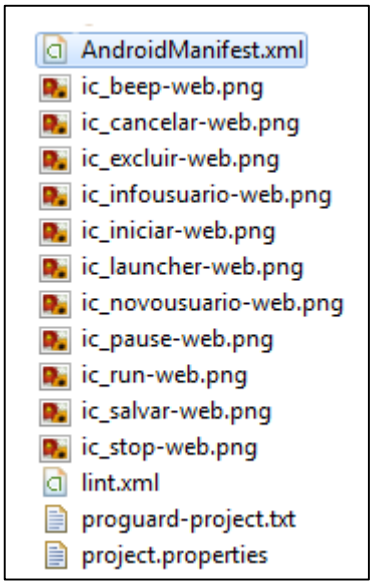
4.3.2.8. Arquivos de Configuração

Os arquivos de configuração servem para informar a versão da aplicação, nome, pacote, permissões e atividades. Dentre os arquivos de configuração, o arquivo *AndroidManifest.xml* como mostra a Figura 4.11, é o mais importante. O *AndroidManifest.xml* é um arquivo XML onde ficam as configurações necessárias para a execução da aplicação. Este arquivo é obrigatório e fica na raiz do projeto.

No *AndroidManifest.xml* são definidas informações importantes, como nome do pacote padrão do projeto, nome das atividades (*activities*) usadas, a versão mínima da

API do Android suportada pela aplicação, a versão de compilação da aplicação e várias outras configurações.

Figura 4.11 - Arquivos de configuração do aplicativo Beep.



Fonte: Captura dos arquivos de configuração no Eclipse.

4.3.3. Criação do Banco de dados

A criação do banco de dados do projeto Beep é feita diretamente pela API Java. Para isso, é criado uma nova classe dentro da pasta src chamada ***SQLiteHelper.java*** que é uma subclasse de *android.database.sqlite.SQLiteOpenHelper* que já encapsula toda a lógica de criação de um banco de dados.

Os métodos *onCreate (SQLiteDatabase db)* e *onUpgrade (SQLiteDatabase db, int versaoAntiga, int novaVersao)* como mostra a Figura 4.12, são chamados automaticamente pelo Android quando o banco de dados precisa ser criado pela primeira vez ou ser atualizado devido a uma nova versão.

Figura 4.12 - Métodos onCreate e onUpgrade.

```
public void onCreate(SQLiteDatabase db) {

public void onUpgrade(SQLiteDatabase db, int versaoAntiga, int novaVersao) {
```

Fonte: Captura de parte da classe SQLiteHelper no Eclipse.

4.3.4. Principais Classes

O Android é bastante flexível em relação à criação da interface gráfica, permitindo que a tela seja feita em XML ou escrita na linguagem Java. Porém, recomenda-se a separação da parte visual da aplicação (botões, imagens, etc.) em um arquivo XML e a parte lógica (banco de dados, adaptadores, controle do estado e eventos, etc.) em classes escritas na linguagem Java. Ao fazer tal separação, o desenvolvedor deixa o código mais limpo e mais fácil manutenção.

O projeto AplicacaoBeep utilizou as seguintes classes para representação das suas funcionalidades: AppPrincipal.java, RepositorioPessoaScript.java, CadastrarEditarPessoa.java, BuscarPessoa.java, PessoaListAdapter.java, TesteListAdapter.java, RelatorioTesteListAdapter.java, ListarPessoas.java, ListarTestes.java, SelecionarPessoaTeste.java, TesteBeep.java e RelatorioTeste.java.

4.3.4.1. AppPrincipal.java

A classe AppPrincipal é o ponto de partida da aplicação. Esta é a classe principal do programa e representa a tela (activity) principal do aplicativo. A mesma deve ser declarada com a ação e categoria no arquivo AndroidManifest.xml, conforme o trecho de código mostrado na Figura 4.13.

Figura 4.13 – Declaração da classe AppPrincipal no arquivo AndroidManifest.xml.

```
<activity
    android:screenOrientation="portrait"
    android:name="AppPrincipal">
     <intent-filter>
         <action android:name="android.intent.action.MAIN"></action>
         <category android:name="android.intent.category.LAUNCHER"></category>
     </intent-filter>
</activity>
```

Fonte: Captura de parte do arquivo AndroidManifest.xml.

A ação MAIN significa que essa classe será o ponto inicial a aplicação e a categoria LAUNCHER define que a aplicação estará disponível (ícone e rótulo) na tela

que aplicativos instalados e que poderá ser inicializada pelo usuário clicando no seu ícone. Vale ressaltar que em uma aplicação existirá apenas uma classe com essa configuração.

No projeto AplicacaoBeep, a classe AppPrincipal é responsável por mostrar na tela do aplicativo o menu principal do aplicação. Para cada botão presente no menu, a classe executa determinadas ações.

Inicialmente, a tela da aplicação é criada automaticamente através do método *onCreate* e seus componentes visuais, como botões e imagens são desenhados/exibidos pelo método *setContentView (R.layout.tela_principal).* Este método carrega o arquivo de tela no formato XML localizado na pasta layout, como mostra a Figura 4.14.

Figura 4.14 – Sobrescrita do onCreate e chamada de tela XML.

```
@Override
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.tela_principal);
```

Fonte: Captura de parte da classe AppPrincipal.java.

Posteriormente, são criados e inicializados os ImageButtons, que serão acessados manipulados pela classe conforme a Figura 4.15.

Figura 4.15 – Criação e inicialização dos botões do menu principal da aplicação

```
//Criação e inicialização dos ImageButton
ImageButton btCadastro = (ImageButton) findViewById(R.id.bttelacadastrar);
ImageButton btTeste = (ImageButton) findViewById(R.id.bttelateste);
```

Fonte: Captura de parte da classe AppPrincipal.java.

Após isso, a aplicação precisa interagir com o usuário. A interação acontece quando o usuário clica em um botão ou seleciona a opção de um menu, por exemplo. No Android, para controlar os eventos/ações de um botão é utilizado o método *setOnClickListener*. Na Figura 4.16, o método é chamado sobre o ImageButton. Ao clicar no botão o método *onClick* é chamado e uma ação “cadastrar” é tomada. Nesta ação é implementado o código fonte que abre outra tela para cadastrar uma nova pessoa.

Figura 4.16 – Chamada do método setOnClickListener sobre o botão Cadastro.

```
btCadastro.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        cadastrar();
    }
});
```

Fonte: Captura de parte da classe AppPrincipal.java.

4.3.4.2. RepositorioPessoaScript.java

Depois de criar o banco de dados, a classe RepositorioPessoaScript invoca a criação e do banco de dados e inserção de dados no mesmo. Também usa a classe SQLLiteHelper para se conectar ao banco. A seguir, a classe RepositorioPessoaScript executa comandos SQL para a criação das tabelas “pessoa” e “teste”, conforme a Figura 4.17.

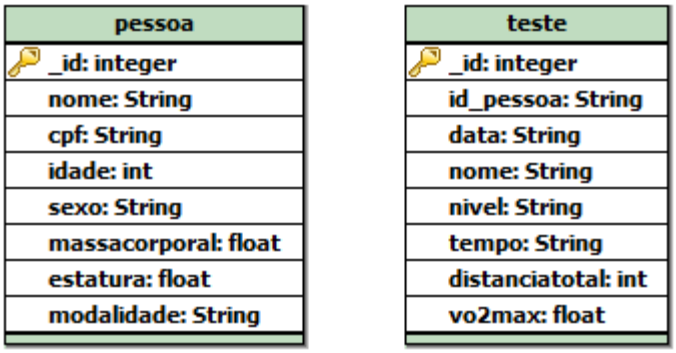
Figura 4.17 – Comandos SQL para criação das tabelas de pessoa e teste.

```
// Cria a tabela com o "_id" sequencial
private static final String[] SCRIPT_DATABASE_CREATE = new String[] {

    // Cria a Tabela pessoa
    "create table pessoa( "
    + "_id integer primary key autoincrement,"
    + "nome text not null,"
    + "cpf text not null,"
    + "idade text not null,"
    + "sexo text not null,"
    + "massacorporal text not null,"
    + "estatura text not null,"
    + "modalidade text not null );",

    // Cria a Tabela teste
    "create table teste( "
    + "_id integer primary key autoincrement,"
    + "id_pessoa text not null,"
    + "data text not null,"
    + "nome text not null,"
    + "nivel text not null,"
    + "tempo text not null,"
    + "distanciatotal text not null,"
    + "vo2max text not null);",
```

Fonte: Captura de parte da classe RepositorioPessoaScript.java.

Ao executar estes comandos, é gerada a tabela “pessoa” com os atributos _id usado para identificar o registro na tabela pessoa, nome da pessoa, o cpf, a idade, o sexo, a massa corporal, a estatura e a modalidade que a pessoa pratica. A Figura 4.18 exibe a representação lógica das tabelas pessoa e teste.

Figura 4.18 – Representação lógica das tabelas pessoa e teste.



Fonte: Elaborada pelo autor.

Também é gerada a tabela “teste” com os atributos _id para identificar o registo na tabela teste, o id_pessoa para identificar a pessoa que realizou o teste, a data de realização, o nome da pessoa, o nível final obtido, o tempo de realização do teste, a distância total percorrida e o VO_2máx.

4.3.4.3. CadastrarEditarPessoa.java

A classe CadastrarEditarPessoa.java é responsável por criar um formulário para inserir uma pessoa no banco de dados e para alterar os dados de uma pessoa. A classe trata as duas situações: inserir e editar.

A primeira situação consiste em inserir uma nova pessoa. Essa situação é tratada quando o usuário clicar no botão de cadastro na tela principal do aplicativo, invocando a classe CadastrarEditarPessoa.java. Quando a classe é chamada para inserir uma nova pessoa, o arquivo *form_editar_pessoa.xml* é carregado, exibindo a tela de cadastro com o formulário que contém os campos para ler as entradas relacionadas ao nome, cpf,

idade, massa corporal, estatura, um RadioButton para ler sexo e um Spinner para ler a modalidade. As entradas são armazenadas em variáveis temporárias.

O RadioButton permite que o usuário selecione o sexo. São criados dois botões, sendo um para o sexo masculino e outro para o sexo feminino. A Figura 4.19 mostra a leitura do seleciona pelo usuário.

Figura 4.19 – Leitura do sexo selecionado pelo usuário.

```
campoSexo.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup campoSexo, int checkedId) {
        boolean sim = R.id.radioSim == checkedId;
        boolean nao = R.id.radioNao == checkedId;

        if (sim) {
            sexo = "M";
            return;
        } else{
            if (nao) {
                sexo = "F";
                return;
            }
        }
    }
});
```

Fonte: Captura de parte da classe CadastrarEditarPessoa.java.

As modalidades são apresentadas por uma lista. Para a leitura da modalidade, é criada um vetor onde são adicionadas as modalidades. O método *setOnItemSelectedListener ()* é chamado para capturar a modalidade selecionada na lista. Após capturar a modalidade, a mesma é na variável temporária modalidade, conforme a Figura 4.20.

Figura 4.20 – Leitura da modalidade selecionada pelo usuário.

```
campoSexo.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup campoSexo, int checkedId) {
        boolean sim = R.id.radioSim == checkedId;
        boolean nao = R.id.radioNao == checkedId;

        if (sim) {
            sexo = "M";
            return;
        } else{
            if (nao) {
                sexo = "F";
                return;
            }
        }
    }
});
```

Fonte: Captura de parte da classe CadastrarEditarPessoa.java.

Depois da ler a entrada de dados, o usuário dever pressionar o botão salvar. Este, chama o método salvar ().

A segundo situação consiste em alterar os dados de uma pessoa. Essa classe é invocada quando o usuário clicar em uma das pessoas da lista exibida pela classe ListarPessoas.java.

A classe CadastrarEditarPessoa.java recebe da classe ListarPessoas.java o id da pessoa selecionada para recuperar e exibir na tela o nome, idade, sexo, massa corporal, estatura e modalidade da pessoa carregando o mesmo arquivo XML utilizado para inserir uma pessoa. Após carregar os dados da pessoa na tela, o usuário pode alterar e salvar ou pode pressionar o botão “Excluir”, que está visível apenas para essa situação.

Por fim, as funcionalidades de listar pessoa e buscar uma pessoa pelo nome são apresentadas na tela de cadastro através de ícones na barra de ação. Suas respectivas classes são chamadas no método *onMenuItemSelected ()* da ActionBar implementada na classe.

4.3.4.4. BuscarPessoa.java

A classe BuscarPessoa.java é responsável por exibir os dados de uma pessoa na tela, onde a busca dos dados é feita fornecendo o nome da pessoa. Para isso, a classe

exibe na tela um formulário com campos de nome, cpf, idade, sexo, massa corporal, estatura e modalidade e também um botão “buscar” a esquerda do campo nome. Os dados são buscados no banco de dados e carregados na tela depois que o usuário digitar nome da pessoa e clicar no botão “buscar”. A Figura 4.21 apresenta o trecho de código que recupera o nome fornecido pelo usuário, busca a pessoa no banco de dados e atualiza com os valores os campos do formulário na tela.

Figura 4.21 – Atualização dos campos com o resultado da busca.

```
// Recupera o nome da pessoa
String nomePessoa = nome.getText().toString();

// Busca a pessoa pelo nome
Pessoa pessoa = buscarPessoa(nomePessoa);

if (pessoa != null) {
    // Atualiza os campos com o resultado

    cpf.setText(pessoa.cpf);
    idade.setText(String.valueOf(pessoa.idade));
    sexo.setText(pessoa.sexo);
    massacorporal.setText(String.valueOf(pessoa.massacorporal));
    estatura.setText(String.valueOf(pessoa.estatura));
    spinner.setText(pessoa.modalidade);

} else {
    // Limpa os campos
    cpf.setText("");
    idade.setText("");
    sexo.setText("");
    massacorporal.setText("");
    estatura.setText("");
    spinner.setText("");

    Toast.makeText(BuscarPessoa.this, "Nenhuma pessoa encontrada", Toast.LENGTH_SHORT).show();
```

Fonte: Captura de parte da classe BuscarPessoa.java.

4.3.4.5. PessoaListAdapter.java, TesteListAdapter.java e RelatorioTesteListAdapter.java

Às vezes é preciso exibir na tela informações complexas, como uma lista com vários itens na vertical com barra de rolagem(*scroll*). Existem meios que possibilitam a exibição dessa lista, de forma que o programador não precise se preocupar com a interface gráfica da tela e sim, apenas informar os dados que irão preencher a lista.

Assim, as classes PessoaListAdapter.java, TesteListAdapter.java e RelatorioTesteListAdapter.java são responsáveis pela criação dos adaptadores para gerar a lista de todas as pessoas cadastradas, a lista de todos os testes e a lista dos teste de determinada pessoa, respectivamente. O construtor de cada uma destas classes recebe como parâmetro os valores de cada linha da tabela e armazena em uma lista.

Para desenhar cada linha na tela é chamado o método *getView ()* automaticamente. Para isso, o arquivo XML destinado a determinada lista é acessado. No caso da classe PessoaListAdapter.java, este método desenha cada linha com nome da pessoa e cpf acessando o arquivo *pessoa_linha_tabela.xml*. Para a classe TesteListAdapter.java, o método *getView ()* desenha cada linha com a data, o nome, nível, tempo, distância total e VO_2máx acessando o arquivo *teste_linha_tabela.xml*.

Por fim, para a classe TesteListAdapter.java, o método *getView ()* desenha cada linha com o nome, a data, nível, tempo, distância total e VO_2máx acessando o arquivo *listview_item.* A Figura 4.22 mostra parte da estrutura do método *getView ()* implementado na classe PessoaListAdapter.java.

Figura 4.22 – Acesso e atribuição do valores para cada componente gráfico.

```
// Atualiza o valor do TextView
TextView nome = (TextView) view.findViewById(R.id.nome);
nome.setText(p.nome);

TextView cpf = (TextView) view.findViewById(R.id.cpf);
cpf.setText(p.cpf);
```

Fonte: Captura de parte da classe PessoaListAdapter.java.

4.3.4.6. ListarPessoas.java e ListarTestes.java

As classes ListarPessoas.java e ListarTestes.java são usadas exibição de informações carregadas do banco de dados em telas na forma de listas. A classe ListarPessoas.java exibe o nome e cpf de todas as pessoas cadastradas no banco de dados na tela e a classe ListarTestes.java exibe a data, nome, nível, tempo, distância total e VO_2máx de todos os testes realizados em outra tela. Ambas as classes possuem a mesma lógica para exibir os dados.

Em ambas as classes, a leitura das informações contidas no banco de dados é feita através da inicialização de uma variável para auxiliar a recuperação da lista de pessoas ou lista de testes do banco de dados de acordo com cada classe. Na classe ListarPessoas.java por exemplo, o método *atualizarLista ()* contém o código que recupera a lista de pessoas e o código de um adaptador para a lista. Depois de recuperar a lista de pessoas é chamado o adaptador (classe PessoaListAdapter.java) para preencher a lista de pessoas na tela, conforme o trecho de código da Figura 4.23.

Figura 4.23 – Recuperação e exibição da lista de pessoas do banco de dados.

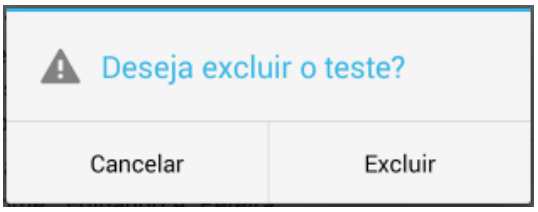
```
        //Inicialização da variável
        repositorio = new RepositorioPessoaScript(this);

        // Atualiza a lista de pessoas
        atualizarLista();
    }

    public void atualizarLista() {

        // Recupera a lista do banco de dados
        pessoas = repositorio.listarPessoas();

        // Adaptador de lista customizado para exibir cada linha da tabela pessoa
        setListAdapter(new PessoaListAdapter(this, pessoas));
    }
```

Fonte: Captura de parte da classe ListarPessoas.java.

A classe ListarPessoa.java exclusivamente, implementa a ação de *onListItemClick* que oferece a funcionalidade de alterar os dados da pessoa quando a mesma for clicada na lista. O usuário ao clicar em uma pessoa da lista, chama o método *editarPessoa ()*. Este recupera o id (identificador) da pessoa clicada, cria uma Intent (intenção) para abrir a tela de editar e passa o id da pessoa para a classe CadastrarEditarPessoa.java que por sua vez, abre o formulário com os dados para edição.

A classe ListarTeste.java também implementa a ação de *onListItemClick*. No entanto, oferece a funcionalidade de excluir o teste quando o mesmo for clicado na lista de testes. Ao clicar no teste contido na lista, é exibida uma janela de alerta onde o usurário decide se deseja excluir ou não o teste, conforme a Figura 4.24.

Figura 4.24 – Mensagem de alerta para excluir teste.



Fonte: Captura de tela Captura de tela de dispositivo Android.

4.3.4.7. SelecionarPessoaTeste.java

A classe SelecionarPessoaTeste.java é chamada quando o usuário clica no botão de selecionar pessoa localizado na tela de teste. Esta classe lista as pessoas cadastradas no banco de dados utilizando o adaptador PessoaListAdapter.java. Também implementa o método *onListItemClick*, que pega a posição da pessoa clicada e chama um método para recuperar o id da pessoa e passa o id como parâmetro para a tela de teste.

4.3.4.8. TesteBeep.java

A classe TesteBeep.java é responsável pela realização do teste Beep. Esta classe exibe na tela durante o teste uma barra de progresso (ProgressBar), um cronometro, informações de nível corrente, shuttle, distancia total percorrida, velocidade e VO_2máx. Além, dos botões de selecionar uma pessoa para o teste, Pausar, Play e Stop/Reset.

Primeiramente, a Figura 4.25 expõem a linha de código para desabilitar o desligamento automático de tela. Ou seja, não permite que a tela de teste “durma”.

Figura 4.25 – Código que desabilita o desligamento automático de tela.

```
//Não permitir que a tela durma
//Desabilitar o desligamento automático de tela
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

Fonte: Captura de parte da classe TesteBeep.java.

Além disso, as informações de VO_2máx são colocadas em modo de invisibilidade, conforme a Figura 4.26. Assim, tornando invisíveis durante o teste e aparecendo na tela somente quando o teste for interrompido pelo usuário.

Figura 4.26 – Ocultando informações de VO_2máx na tela.

```
// Invisibilidade para as informações de VO2max
tvcampoVo2max.setVisibility(View.INVISIBLE);
tvVo2max.setVisibility(View.INVISIBLE);
```

Fonte: Captura de parte da classe TesteBeep.java.

A classe TesteBeep.java implementa o método *data (),* que apanha a data atual do sistema. Esta informação é necessária para registrar no banco de dados a período em o teste foi realizado. A Figura 4.27 exibe o trecho de código que realiza esta função e retorna a data atual no formato “dia/mês/ano”.

Figura 4.27 – Trecho de código para apanha data atual do sistema.

```
// Pegando a data atual do sistema
Date dataAtual = new Date(System.currentTimeMillis());
SimpleDateFormat formatador = new SimpleDateFormat("dd/MM/yyyy");

return formatador.format(dataAtual);
```

Fonte: Captura de parte da classe TesteBeep.java.

Para a realização do teste, o usuário é obrigado a selecionar uma pessoa para o teste acionando o botão, respectivo. Ao acionar este botão, a classe SelecionarPessoaTeste.java é chamada para listar as pessoas cadastradas. Após a seleção da pessoa, o usuário tem a permissão para começar o teste. Basta acionar o botão Play, que a classe inicia o cronometro e chama o método *contador ().*

O teste foi implementado no método *contador ().* Este método possui um ProgressBar que irá setar a barra a cada estágio concluído. Também utiliza um *mCountDownTimer (),* que é uma contador para contagem regressiva de tempo. Ele foi utilizado para definir os intervalos que os bips eram tocados. Ele é inicializado com o tempo total do teste em milissegundos e o intervalo em milissegundos que é decrementado.

Para cada estágio é feita a alteração de algumas informações na tela do teste, também a leitura de outras e a execução do áudio que simula o “bip”, conforme a Figura 4.28. Para a execução do áudio, foi utilizada a biblioteca MediaPlayer.

Figura 4.28 – Alteração e leitura de informações na tela de teste a cada estágio e nível corrente.

```
mProgressBar.setProgress(1);
nivel.setText("2." + p);
tvNivel.setText("Nível 1 completo");
tvshuttle.setText("Shuttle de 8");
velocidade = (float) 9.0;
tvcampoVelocidade.setText("9.0 Km/h");
play(3);
```

```
Distanciatotal = distancia;
nivel.setText("2." + p);
Nivel = nivel.getText().toString();
tvdistancia.setText(distancia + " m");
play(2);
```

Fonte: Captura de parte da classe TesteBeep.java.

Quando o usuário acionar o botão Pause, é chamado o método *stop ()* para o cronometro e cancelada a contagem regressiva. Depois, é realizado o cálculo do VO_2máx e coloca-se as informações de VO_2máx visíveis na tela. A seguir, é aberta uma janela AlertDialog que solicita que o usuário tome a decisão de salvar ou não o teste.

4.3.4.9. RelatorioTeste.java

A classe RelatorioTeste.java permite que o usuário consulte no banco de dados, todos os testes realizados por uma determinada pessoa. Ela é responsável por exibir o nome, a data, nível, tempo distancia total, VO_2máx de todos testes na tela em forma de lista. Inicialmente, a esta classe recupera a lista de pessoas do banco de dados e lista todas as pessoas cadastradas na aplicação, segundo a Figura 4.29.

Figura 4.29 – Método para recuperar e exibir as pessoas cadastradas.

```
public void atualizarLista() {

    // Recupera a lista de pessoas e exibe na tela
    pessoas = repositorio.listarPessoas();

    // Adaptador de lista customizado para cada linha de uma pessoa
    setListAdapter(new PessoaListAdapter(this, pessoas));
}
```

Fonte: Captura de parte da classe RelatorioTeste.java.

A partir dessa lista, o usuário pode selecionar a pessoa que deseja exibir os testes. Para isso, basta o usuário clicar em um dos nomes da lista. A clicar no nome, é chamado o método *recupera_dados_pessoa (int posicao),* que recupera as informações de seu cadastro, como sexo, idade, entre outros.

Após os dados serem recuperados, o id da pessoa é passado como parâmetro para o método *atualizarListaTestesPessoa (String id_pessoa)*. Este método é responsável por recuperar os testes do banco de dados e listar os testes que contém o mesmo id que foi passado como parâmetro, como mostra a Figura 4.30.

Figura 4.30 – Trecho do código para recuperar e exibir os testes por id_pessoa.

```
// Pega a lista de testes e exibe na tela
testes = repositorio_teste.listarTestesPorId_pessoa(id_pessoa);

// Adaptador de lista customizado para cada linha de um teste
setListAdapter(new RelatorioTesteListAdapter(this, testes));
```

Fonte: Captura de parte da classe RelatorioTeste.java.

Além da exibição dos testes, esta classe um ícone na ActionBar que ao ser acionado, exibe um Alert Dialog com as informações cadastrais da pessoa.

4.3.5. Bibliotecas utilizadas

Foram incorporadas as bibliotecas Action Bar, Alert Dialog, Chronometer, Media Player e Progress Bar à aplicação. Algumas bibliotecas foram utilizadas ter em vista a padronização em relação a outros aplicativos. Já outras, foram utilizadas para auxiliar o teste Vai-e-vem.

4.3.5.1. Action Bar

Buscando a padronização foi utilizado no aplicativo Beep a biblioteca *android.app.ActionBar*. O Action Bar é uma barra de ação localizada na parte superior da tela que geralmente é persistente em todo o aplicativo. Também é considerando um dos elementos de design mais importantes que o desenvolvedor deve implementar.

A barra de ação é dividida em quatro áreas funcionais diferentes que se aplicam a maioria dos aplicativos. São elas: o ícone do aplicativo, um espaço para exibir o título do aplicativo ou conteúdo não-interativo, botões de ação (como *Pesquisa*). As ações que não couberem na barra de ação são movidas automaticamente para o ícone overflow (ícone para visualizar o nome das ações). A Figura 4.31 mostra um aplicativo que possui um Action Bar.

Figura 4.31 – Uma Action Bar que inclui o ícone do aplicativo [1], [2] dois ícones de ação, e [3] ícone overflow.

Fonte: Site Google Developers[24].

Os Action Bar estão disponíveis apenas para a versão 3.0 e superior, dado que foram adicionados pela primeira vez no Android 3.0 (API nível 11).

4.3.5.2. Alert Dialog

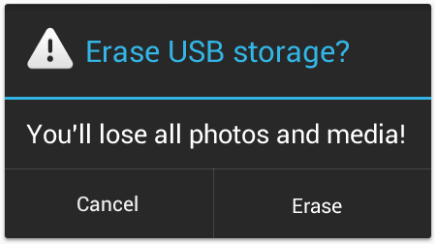
Uma caixa de diálogo é uma janela que solicita ao usuário a tomar uma decisão ou inserir informações adicionais exigidas pelo aplicativo para continuar a tarefa. Um diálogo não preenche toda a tela e requer ao usuário uma interação antes de prosseguir.

Da mesma forma, o Alert Dialog exige a confirmação ou aceitação do usuário antes de prosseguir. Pode mostrar um título, até três botões, uma lista de itens selecionáveis ou não, ou um layout personalizado.

[24] Disponível em: < http://developer.android.com/guide/topics/ui/actionbar.html>. Acesso em Novembro de 2014.

A biblioteca *android.app.AlertDialog* foi utilizada no aplicativo Beep para que o usuário após realizar o teste possa tomar a decisão de salvar o teste ou não. Figura 4.32 mostra um Alert Dialog básico com título, mensagem e dois botões.

Figura 4.32 – Exemplo de um diálogo básico.



Fonte: Site Google Developers[25].

4.3.5.3. Chronometer

O Chronometer ou cronometro é um componente no Android que serve para fazer o controle de tempo. É fácil e simples de implementar, além de oferecer uma interface simples e funcional. Por padrão, ele irá exibir o valor atual do temporizador no formato "MM:SS". A biblioteca *android.widget.Chronometer* foi utilizada no aplicativo para controlar o tempo de realização do teste.

4.3.5.4. Media Player

A biblioteca *android.media.MediaPlayer* serve para controlar e executar arquivos de áudios, vídeos e streams. Os principais formatos de áudio suportados são mp3, midi, 3gp, ogg, mp4, wav. Já os principais formatos de vídeo suportados são mp4 e 3gp.

A biblioteca foi utilizada no aplicativo para emitir o som do “bip” ao reproduzir um arquivo de áudio. Também foi utilizada para emitir determinado som ao clicar nos botões de play, pause e stop/reset.

[25] Disponível em: < http://developer.android.com/design/building-blocks/dialogs.html>. Acesso em Novembro de 2014.

4.3.5.5. ProgressBar

O ProgressBar é um indicador visual de progresso em alguma operação. Exibe uma barra de progresso na tela que representa o quanto a operação tem progredido. A biblioteca *android.widget.ProgressBar* foi usada no aplicativo para melhorar a interatividade da aplicação com o usuário, fornecendo uma ideia de quando o teste será concluído. A barra de progresso é preenchida à medida que o usuário for completando os estágios/ níveis. A Figura 4.33 demonstra uma aplicação em execução, onde pode-se visualizar a barra de progresso.

Figura 4.33 – Barra de progresso no Holo Dark e Light.

Fonte: Site Google Developers[26].

[26] Disponível em: < http://developer.android.com/design/building-blocks/progress.html>. Acesso em Novembro de 2014.

5. APLICAÇÃO DESENVOLVIDA

Nesta seção é apresentada a aplicação desenvolvida. O aplicativo Beep possui basicamente sete telas que exercem a iteração com o usuário. São elas: Tela do Menu Principal, Tela de Cadastrar/Editar, Tela de Buscar Pessoa, tela de Listar Pessoas Cadastradas, Tela de Teste, Tela de Listar Testes e Tela de Listar Testes por Pessoa.

5.1. Tela do Menu Principal

A Figura 5.1 apresenta a *tela Inicial* do aplicativo. Esta tela apresenta ao usuário um menu principal, no qual o usuário têm acesso as duas principais funcionalidades do aplicativo que consistem em cadastro de pessoas e o teste. Ao pressionar com um único clique o botão respectivo a sua opção, o mesmo levará para a tela correspondente. A tela principal pode ser observada na Figura 5.1.

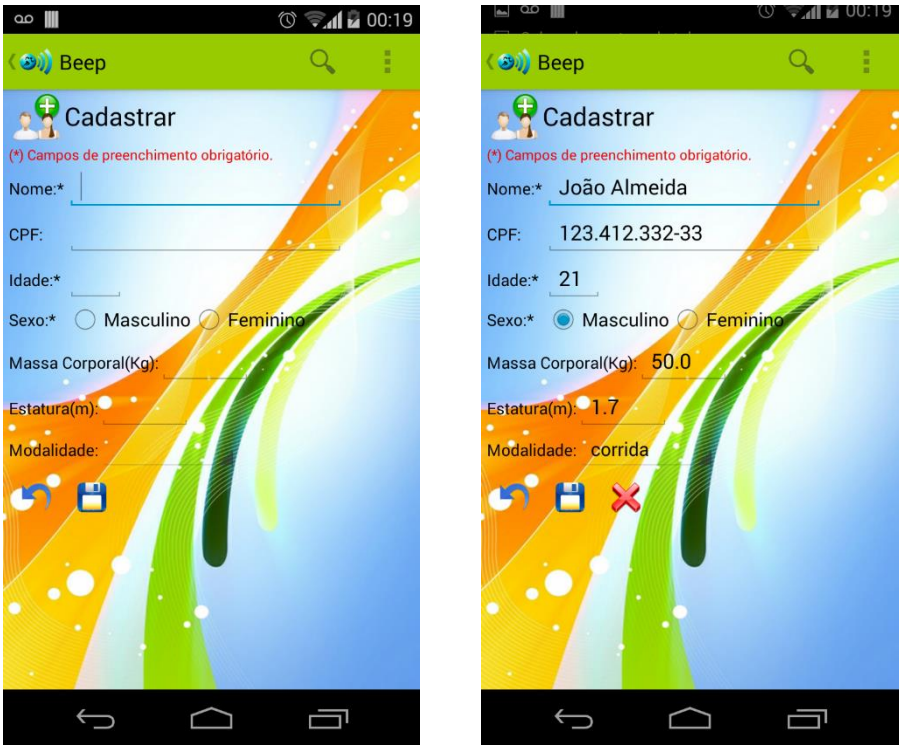
Figura 5.1 – Menu principal do Aplicativo Beep.

Fonte: Captura de tela de dispositivo Android.

5.2. Tela de Cadastrar/Editar

A tela de *Cadastrar/Editar* permite o cadastro de pessoas que deseja praticar o treinamento intervalado de alta intensidade. Esta tela fornece um formulário para o usuário insira uma nova pessoa na aplicação. O usuário deve informar o nome da pessoa, o cpf, idade, sexo, massa corporal, estatura e modalidade praticada, como é observado na Figura 5.2.

Figura 5.2 - Tela de cadastrar/Editar do aplicativo.



Fonte: Captura de tela de dispositivo Android.

Esta tela possui uma barra de ação na parte superior com um ícone para buscar alguma pessoa cadastrada e um ícone overflow com a opção de listar as pessoas cadastradas. Além disso, esta tela pode ser utilizada no modo de edição, ou seja, quando a tela for aberta no modo de alteração, a mesma carrega as informações da pessoa para alteração de alguma informação e permite a exclusão da pessoa.

5.3. Tela de Buscar Pessoa

A tela de *Buscar Pessoa* permite que o usuário busque as informações de uma pessoa que foi cadastrada no aplicativo Beep. Basta inserir o nome da pessoa e clicar no ícone de busca e em seguida são carregadas todas as informações. A Figura 5.3 mostra a Tela de Buscar Pessoa.

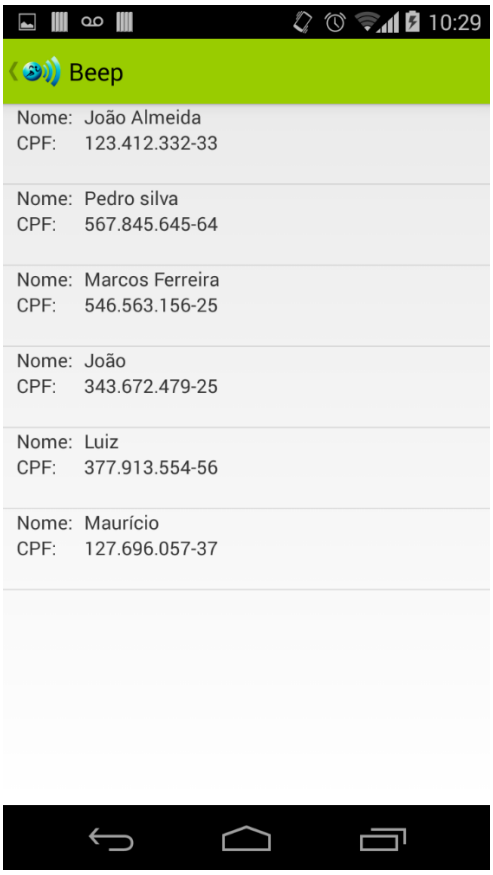
Figura 5.3 - Tela de Buscar Pessoa.

Fonte: Captura de tela de dispositivo Android.

5.4. Tela de Listar Pessoas Cadastradas

É uma tela que exibe uma lista preenchida com as pessoas cadastradas. Cada item da lista é composto pelo nome e cpf. Ao clicar em uma pessoa da lista é aberta a tela de cadastro no modo de edição. A Figura 5.4 exemplifica esta tela.

Figura 5.4 - Tela de Listar pessoas cadastradas.



Fonte: Captura de tela de dispositivo Android.

5.5. Tela de Teste

Esta tela é destinada a realização do teste vai-e-vem de Léger e Lambert. Contém informações de progresso, nível corrente, shuttle, distância percorrida, tempo de realização do teste, velocidade, VO_2máx e nome de quem está realizando o teste. A Figura 5.5 exibe a tela de teste.

Figura 5.5 - Tela de Teste.

Fonte: Captura de tela de dispositivo Android.

Ao clicar no botão selecionar uma pessoa, o usuário é direcionado para uma tela que contém uma lista preenchida com as pessoas cadastradas. Nesta outra tela, o usuário seleciona a pessoa que será submetida ao teste e retorna para a tela de teste.

A seguir, o usuário pode pressionar o botão Play para começar o teste e Pause para interromper o teste. Ao interromper o teste, é exibida na tela uma janela com uma mensagem para confirmação de salvamento do teste. Posteriormente, o usuário pode realizar outro teste. Basta pressionar o botão stop/reset para limpar a tela e repetir os passos anteriormente descritos.

A tela de teste possui uma ActionBar com um ícone overflow. No ícone é fornecido as opções de listar testes e de buscar testes por pessoa.

5.6. Tela de Listar Testes

É uma tela que exibe uma lista preenchida com todos os testes realizados. Cada item da lista é composto pela data de realização do teste, nome, ultimo nível atingido, tempo total de realização, distância total percorrida e VO_2máx. Ao clicar em um teste da lista é exibida uma mensagem na tela que solicita a decisão de excluir o teste. A Figura 5.6 exibe esta tela.

Figura 5.6 - Tela de Listar Testes.

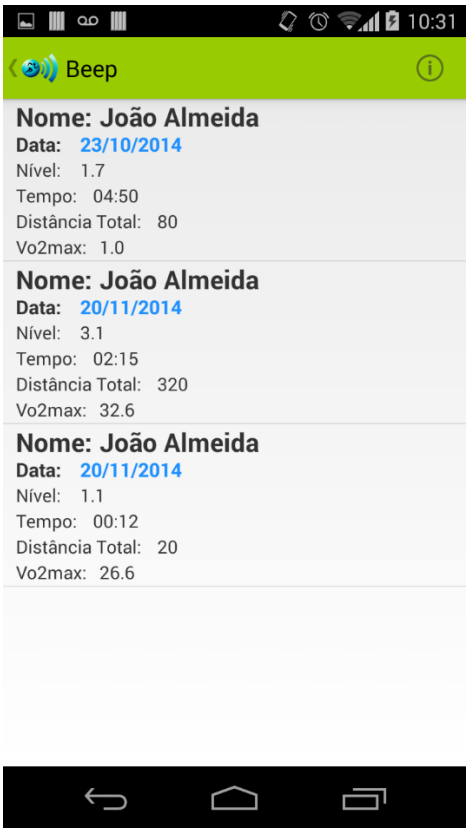
Fonte: Captura de tela de dispositivo Android.

5.7. Tela de Listar Testes por Pessoa

Ao abrir a tela listar testes por pessoa é exibida uma lista das pessoas cadastradas. O usuário deve clicar na pessoa que deseja ver os testes. A seguir, esta

tela exibe uma lista preenchida com todos os testes realizados por uma determinada pessoa. Cada item da lista é composto pelo nome, data de realização do teste, nome, ultimo nível atingido, tempo total de realização, distância total percorrida e VO_2máx. A Figura 5.7 exibe esta tela.

Figura 5.7 - Tela de Listar Testes por Pessoa.



Fonte: Captura de tela de dispositivo Android.

6. CONCLUSÃO

O desenvolvimento do presente trabalho trouxe uma significativa contribuição para os alunos do curso de Educação Física da Universidade Federal dos Vales do Jequitinhonha e Mucuri, uma vez que irá servir de apoio para o treinamento esportivo.

O sistema operacional Android é o que possui melhor posicionamento no mercado mundial e está presente na maioria dos *smartphones* e *tablets* em todo o mundo. Essa foi uma das motivações para o desenvolvimento do aplicativo Beep. Além disso, existe um grande referencial teórico acerca do sistema operacional Android.

Ao final do trabalho tem-se como resultado o aplicativo Beep, desenvolvido para o sistema operacional Android com sistema superior a versão 4.0. O aplicativo permite que o usuário não fique limitado a um lugar específico. Ao salvar as informações dos testes para ser posteriormente visualizadas, percebe-se que podem ser realizadas várias análises através dessas informações.

Como sugestões para trabalhos futuros propõem-se uma nova versão do aplicativo com a adição da funcionalidade de “modo treinamento”. O modo de treinamento permitiria o usuário a realizar seções de treinamento com intervalos e velocidade específicas para cada usuário.

REFERÊNCIAS

_____. **Team Test Beep.** Topend Sports Network, 2014. Disponível em:<http://www.topendsports.com/testing/team-beeptest/index.htm>. Acesso em 08 Jul. 2014.

[5]_____. **Anatomy Physiology of an Android.** Androidteam. Disponível em:<http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf>. Acesso em 05 Abr. 2014.

AMERICAN COLLEGE OF SPORTS MEDICINE. **Guidelines for exercise testing and prescription.** 4. ed. Philadelphia: Lea & Febiger, 1991.

ASTORINO, Todd A; ALLEN, Ryan P; ROBERSON, Daniel W e JURANCICH, Matt. **Effect of high-intensity interval training on cardiovascular function, VO2max, and muscular force.** Ann Intern MeJ Strength Cond Res, v.26 (1), p.38-45, 2012. Disponível em: < http://www.ncbi.nlm.nih.gov/pubmed/22201691>. Acesso em 22 Jun. 2014.

DUARTE, M. F. S. e DUARTE, C.R. **Validade do teste aeróbico de corrida de vai-e-vem de 20 metros**. Revista Brasileira de Ciência e Movimento. v.9, n.3, p.07-14, 2001.

F/Nasça Saatchi & Saatchi e Instituto Datafolha (2013). **F/Radar:** *panorama do Brasil na internet*. 13 ed. Nazca Saatchi & Saatchi, 2013. Disponível em: <http://www.fnazca.com.br/index.php/2013/12/20/fradar-13%c2%aa-edicao/>. Acesso em 10 Mai. 2014.

GHORAYEB, Nabil. **Desenvolvimento da tecnologia tem influência no rendimento esportivo**. São Paulo: EuAtleta.com, 2013. Disponível em: <http://globoesporte.globo.com/eu-atleta/saude/noticia/2013/07/desenvolvimento-da-tecnologia-influencia-no-desempenho-esportivo.html>. Acesso em 26 Jun. 2014.

GIBALA, Martin J.; LITTLE, Jonathan P.; MACDONALD, Maureen J. and HAWLEY, John A. **Physiological adaptations to low-volume, high-intensity interval training in health and disease.** J. Physiol, v. 590.5, p. 1077-1084, 2012.

KUSZKA, Boris. **Dispositivos móveis:** *interface com o mundo mobile*. Canaltech corporate, 2014. Disponível em: <http://corporate.canaltech.com.br/coluna/mobile/Dispositivos-moveis-a-interface-com-o-mundo/>. Acesso em 01 Jun. 2014.

LECHETA, Ricardo R Lecheta. **Google Android:** *aprenda a criar aplicações para dispositivos móveis com o Android SDK*. 2.ed. rev. ampl. São Paulo: Novatec, 2010.

LENZI, Sandro. **Tecnologia e treinamento esportivo**. Pensamento esportivo, Fev. 2014. Disponível em:<http://www.pensamentoesportivo.com.br/conhecendo-o-esporte/6303-tecnologia-e-treinamento-esportivo#sthash.qxrsVyUL.Dn42pjk1.dpbs>. Acesso em 25 Jun. 2014.

PEREIRA, Lucio Camilo Oliva; SILVA, Michel Lourenço. **Android para Desenvolvedores.** Rio de Janeiro: Brasport, 2009.

ROSS A. and EVERITT. **Long-term metabolic and skeletal muscle adaptations to short-sprint M. Ltraining:** *implications for sprint training and tapering*. Sports Med, v. 31, p.1063-1082, 2001.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Definitive Guide to Scrum:** the rules of the game. Scrum.org, p.18, 2013. Disponível em:<https://www.scrum.org/scrum-guide>. Acesso em 02 Jul. 2014.

STOPPANI, Jim. **The Ultimate 8-Week HIIT-For-Fat-Burning Program.** Bodybuilding.com, 2012. Disponível em:<http://www.bodybuilding.com/fun/ultimate-8-week-hiit-for-fat-burning-program.html>. Acessado em Junho de 2014.

VEJA**. A Revolução dos Músculos:** *os exercícios rápidos e intensos que estão mudando (quase) tudo nas academias*. ed. 2334, n.33.São Paulo: Editora Abril, Ago. 2013.

WEST, Mark & CHEW, Han Ei. **Reading in the mobile era:** *a study of mobile reading in developing countries.* Paris: UNESCO, 2014. Disponível em:<http://unesdoc.unesco.org/images/0022/002274/227436e.pdf>. Acesso em 01 Jun. 2014.