

UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

Programa de Pós-Graduação em Tecnologia, Ambiente e Sociedade

Linha de Pesquisa: Tecnologia & Inovação

Sublinha de Pesquisa: Novas Tecnologias e Ciências

Cássio Gonçalves Sena

**DESENVOLVIMENTO DE APLICATIVOS NA PLATAFORMA ANDROID PARA O
ENSINO DE FÍSICA E MATEMÁTICA – RELATÓRIO TÉCNICO**

**Teófilo Otoni
2018**

Cássio Gonçalves Sena

**DESENVOLVIMENTO DE APLICATIVOS NA PLATAFORMA ANDROID PARA O
ENSINO DE FÍSICA E MATEMÁTICA – RELATÓRIO TÉCNICO**

Relatório técnico apresentado ao Programa de Pós-Graduação *Stricto Sensu* Mestrado Profissional em Tecnologia, Ambiente e Sociedade (PPGTAS) da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do título de mestre.

Orientador: Prof. Dr. Mauro Lúcio Franco.

Coorientador: Prof. Dr. Wederson Marcos Alves.

**Teófilo Otoni
2018**

Ficha Catalográfica
Preparada pelo Serviço de Biblioteca/UFVJM
Bibliotecário responsável: Gilson Rodrigues Horta – CRB6 nº 3104

S474d Sena, Cássio Gonçalves.
2018 Desenvolvimento de aplicativos na plataforma android para o ensino de física e matemática – relatório técnico. / Cássio Gonçalves Sena. Teófilo Otoni, 2018.
54 p. ; il.

Dissertação (Mestrado) – Universidade Federal dos Vales do Jequitinhonha e Mucuri. Programa de Pós-Graduação em Tecnologia, Ambiente e Sociedade, 2018.

Orientador: Prof. Dr. Mauro Lúcio Franco.

Coorientador: Prof. Dr. Wederson Marcos Alves.

1. Aplicativos Educacionais. 2. Android. 3. Tecnologia. 4. Física.
5. Matemática. I. Título.

CDD: 005


CÁSSIO GONÇALVES SENA

**Desenvolvimento de aplicativos na plataforma Android para o ensino
de Física e Matemática – Relatório Técnico**

Trabalho de Conclusão apresentado ao
PROGRAMA DE PÓS-GRADUAÇÃO
EM TECNOLOGIA, AMBIENTE E
SOCIEDADE – STRICTO SENSU,
nível de MESTRADO, como parte dos
requisitos para obtenção do título de
MAGISTER SCIENTIAE EM
TECNOLOGIA, AMBIENTE E
SOCIEDADE.

Orientador: Prof. Dr. Mauro Lucio
Franco

Data da aprovação: 23/03/2018


Prof. Dr. MAURO LUCIO FRANCO - UFVJM


Prof. Dr. GERALDO MOREIRA DA ROCHA FILHO - UFVJM


Prof. Dr. JAIRO LISBOA RODRIGUES - UFVJM


Prof.ª Dr.ª JANAINNE NUNES ALVES - IFNMG

TEÓFILO OTONI

AGRADECIMENTOS

Primeiramente a Deus que me deu energia e benefícios para concluir todo este trabalho.

Aos meus pais e irmãos que sempre me incentivaram em todas as minhas escolhas.

À minha Querida Esposa, pelo apoio incondicional durante toda essa trajetória.

Ao professor orientador Prof. Dr. Mauro Lúcio Franco, pelo apoio e encorajamento contínuos na pesquisa e pelos conhecimentos construídos.

Aos amigos Yvssa e Oséas que fizeram parte dessa trajetória, dividindo momentos de descontração, estudos, discussões, experiências e conquistas.

RESUMO

Os professores têm dificuldade de encontrar tecnologias que possam auxiliar no ensino aprendizagem e que promova formas diversas e atraentes para que os alunos compreendam melhor a Física e a Matemática. Uma alternativa é o desenvolvimento de aplicativos educacionais para *smartphones* com o sistema operacional Android que permite explorar a variedade de sensores presentes nos dispositivos móveis, assim potencializar a sua utilização em experimentos e observações em sala de aula. Por isso, o objetivo deste trabalho consiste na demonstração técnica da construção dos aplicativos: Mate Código de Barras, Física Lab Resistores, Física Lab Óptica e o Física Lab Cinemática e estes podem ser utilizados tanto em sistemas de coleta de dados quanto nos sistemas de processamento de dados através de seus sensores incorporados (o acelerômetro, magnetômetro, giroscópio e a câmera). Com a ajuda da Engenharia de Software e recursos da Plataforma de Desenvolvimento Android foi possível criar aplicativos didáticos como ferramentas tecnológicas que podem ser utilizados na sala de aula para auxiliar os métodos tradicionais de ensino de Física e Matemática, uma vez que a inclusão de tecnologias modernas criam inúmeras possibilidades para os professores trabalharem com seus alunos.

Palavras-chave: Aplicativos Educacionais; Android; Java; Tecnologia; Ciências; Física; Matemática.

LISTA DE ILUSTRAÇÕES

Figura 1 - Alguns dispositivos que usam Java.....	12
Figura 2 – Aplicativos representados através do fluxograma.....	15
Figura 3 – Etapas da Visualização.....	16
Figura 4 – Diagrama <i>ViewPager Screen Slides</i>	17
Figura 5 – Exemplo <i>layout</i> declarado em XML.....	18
Figura 6 – Códigos do <i>MainActivity</i> escrito em Java.	19
Figura 7 – Método utilizado no aplicativo para calcular o dígito verificado.	21
Figura 8 – Tela Principal do Aplicativo.....	21
Figura 9 – Inicialização da leitura do Código de Barras	22
Figura 10 – Realizando a leitura do código de barras através da câmera do dispositivo	22
Figura 11 – Tela de resultados com os cálculos do dígito verificador do código de barras.	23
Figura 12 – Telas com as fundamentações teóricas auxiliares.....	24
Figura 13 – Tela Principal do Aplicativo – Resistores de 4 e 5 faixas de cores.....	26
Figura 14 – Fórmula para cálculo da resistência equivalente de uma associação em série.....	27
Figura 15 – Código utilizado para cálculo da resistência equivalente de uma associação em série.....	27
Figura 16 – Fórmula para cálculo da resistência equivalente de uma associação em paralelo.....	28
Figura 17 – Código Java com o Método	28
Figura 18 – Telas com as opções para cálculo das associações em série e paralelo.	28

Figura 19 – Tela com Conteúdo Teórico - LEI DE OHM	29
Figura 20 – Tela com a tabela de cores dos resistores.	30
Figura 21 – Autores do Aplicativo	30
Figura 22. Sistema de Coordenadas Relativo a um Dispositivo	31
Figura 23. Tela do Cronômetro inteligente	32
Figura 24. Simulação de Início e Pause do Cronômetro.	33
Figura 25 - Código em Java, Início e Pause do Cronômetro.....	33
Figura 26. Inclinação do <i>Smartphone</i> – Eixo Z.....	34
Figura 27. Simulação em um Plano Inclinado	35
Figura 28 - Código Java que Calcula a Inclinação em Relação ao Eixo Z Inclinado. 35	
Figura 29. – Tela com Conteúdo Teórico - Velocidade Média.	36
Figura 30 – Autores do Aplicativo.	36
Figura 31 – Diagrama de Raios na Formação de Imagens.....	37
Figura 32 – Decomposição da Luz Através do Prisma.....	37
Figura 33 – Lentes Convergente e Divergente com Diagrama de Raios.	39
Figura 34 – Espelhos Côncavo e Convexo com Diagrama de Raios.....	39
Figura 35 - Código Java – Método de Reposicionamento dos Raios nas Lentes e Espelhos.....	40
Figura 36 - Código Java – Método que Desenha os Raios nos Espelhos e Lentes ..	40
Figura 37 - Código Java – Método que Desenha os Raios nos Espelhos e Lentes ..	41
Figura 38 - Código Java – Método que Desenha os Raios do Prisma.....	42
Figura 39 – Tela Saiba Mais com a Fundamentação e Manual de Utilização	42
Figura 40 – Autores do Aplicativo.	43

LISTA DE ABREVIATURAS E SIGLAS

ABNT - Associação Brasileira de Normas Técnicas

EAN13 - European Article Number

XML - eXtensible Markup Language

GPS - Global Positioning System

IDC - International Data Corporation

IDE - Ambiente de Desenvolvimento Integrado

IHC - Interação Humano-Computador

INPI - Instituto Nacional da Propriedade Industrial

JDK - Java SE Development Kit

NBR - Norma Brasileira

PCs - Personal Computer

QR-CODE - Quick Response Code (Código de Resposta Rápida)

SDK - Kit de Desenvolvimento de Software

UI – User Interface

UPC-A - Código Universal de Produtos

UX - User Experience

UML - Linguagem de Modelagem Unificada

SUMÁRIO

1 INTRODUÇÃO	9
2 REVISÃO DE LITERATURA	11
2.1 Linguagem Java.....	11
2.2 Android - O sistema operacional móvel líder mundial.....	12
2.3 Engenharia de Software: Modelos Incremental e iterativo.....	13
2.4 Processo de Desenvolvimento de Aplicativos Educacionais	14
3 VISÃO GERAL DOS APLICATIVOS	15
3.1 Fluxograma de usabilidade	15
3.2 Etapa de Visualização.....	16
3.3 Processo de Desenvolvimento dos Aplicativos: Material Designer	16
3.3.1 Os <i>Layouts</i> dos Aplicativos	17
3.3.2 Codificação.....	18
4 ETAPAS DO DESENVOLVIMENTO DO APLICATIVO MATE CÓDIGOS DE BARRAS	20
4.1 – Objetivo do Aplicativo.....	20
4.2 Telas e <i>Layouts</i> do Aplicativo Mate Códigos de Barras.....	21
4.2.1 Tela Principal.....	21
4.2.2 Tela de Leitura do Código de Barras.....	22
4.2.3 Tela de Resultados do Dígito Verificador	23
4.2.4 Telas de Conceitos e Fundamentos.....	23
5 ETAPAS DO DESENVOLVIMENTO DO APLICATIVO FÍSICA LAB RESISTORES	25
5.1 – Objetivo do Aplicativo.....	25
5.2 Telas e <i>Layouts</i> do Aplicativo Física Lab Resistores	25
5.2.1 Tela Principal.....	25
5.2.2 Telas de Associação em Série e Paralelo	27
5.2.3 Tela com Conteúdo Teórico.	29
5.2.4 Tela de Autores.	30
6 ETAPAS DO DESENVOLVIMENTO DO APLICATIVO FÍSICA LAB CINEMÁTICA	31
6.1 – Objetivo do Aplicativo.....	31

6.2 Telas e <i>Layouts</i> do Aplicativo Física Lab Resistores	31
6.2.1 Tela Cronômetro Inteligente e Experimento	32
6.2.2 Tela Ângulo, Nível e Experimento	34
6.2.3 Tela com Conteúdo Teórico.	35
6.2.4 Tela de Autores.	36
7 ETAPAS DO DESENVOLVIMENTO DO APLICATIVO FÍSICA LAB ÓPTICA	37
7.1 – Objetivo do Aplicativo.....	37
7.2 Telas e <i>Layouts</i> do Aplicativo Física Lab Óptica	38
7.2.1 Tela de Lentes, Espelhos e Comportamentos	38
7.2.2 Tela do Prisma e Comportamento.....	41
7.2.3 Tela com Conteúdo Teórico.	42
7.2.4 Tela de Autores.	43
8 CONCLUSÃO	44
REFERÊNCIAS	46
ANEXOS	48
ANEXO I – Artigo publicado na revista Vozes dos Vales 2017.	49
ANEXO II – Documentos de registro do aplicativo Mate Código de Barras junto ao INPI.	50
ANEXO III – Resultados da utilização do Aplicativo Física Lab Resistores.....	52

CAPÍTULO 1

INTRODUÇÃO

Este relatório técnico segue as normas da ABNT NBR 10719, e tem como principal propósito descrever as diversas fases do desenvolvimento dos aplicativos: Mate Código de Barras, Física Lab Resistores, Física Lab Óptica e o Física Lab Cinemática, sendo este último registrado pelo INPI (Instituto Nacional da Propriedade Industrial) sob o número BR51201170004528. Este documento inclui diversos diagramas e os principais códigos que explicam o funcionamento dos aplicativos, linguagens, ferramentas utilizadas e também recursos da Plataforma Android.

Segundo Aguiar e Vieira (2016, p.8), o desenvolvimento dos aplicativos é justificado pelo fato dos *smartphones* atuais possuírem processadores rápidos e telas sensíveis ao toque e por virem de fábrica com uma diversidade de sensores que medem grandezas físicas de alto valor, como: GPS responsável por posição, giroscópio que mede a velocidade angular do aparelho, magnetômetro que verifica a intensidade do campo magnético, barômetro utilizado para aferir a pressão atmosférica, higrômetro para medir a umidade do ar, luxímetro para checar a intensidade luminosa, sensor de proximidade, microfone, câmera fotográfica e de vídeo, possibilitando a criação aplicações potenciais para o estudo da Física e Matemática que os utilizam para processamento e interpretação de dados.

Outro ponto importante é que grande parte dos dispositivos possuem o sistema operacional Android, que segundo o IDC (*International Data Corporation*) (2017), se faz presente em 85.0% dos celulares do mercado contra 14,7% da iOS da Apple e 0,3% de outros sistemas. Mais uma vantagem é o *Google*¹, empresa proprietária do Android, que disponibiliza para os desenvolvedores de forma gratuita ferramentas para criar e testar os softwares; além de facilitar a disponibilização dos mesmos em sua loja online por um preço acessível comparado a outras plataformas.

Os aplicativos foram desenvolvidos utilizando a linguagem de programação Java para Android e com base na engenharia de software seguiu os requisitos do método incremental e os *layouts* foram construídos baseados em *Material Designer Android ViewPager e Screen Slides*.

¹ *Google*, empresa multinacional americana de serviços online e software.

Os programas desenvolvidos e relatados neste trabalho têm como objetivo contribuir com o ensino de Física e Matemática através de qualquer *smartphone* que utilize o sistema operacional Android que possam ser integrados no processo didático onde os alunos consigam facilmente estudar os materiais disponíveis nos aplicativos, sendo assim, mostrar que as duas disciplinas estão presentes no cotidiano.

Este relatório está organizado da seguinte maneira. No capítulo 2 é discutido a importância da linguagem de programação Java, o crescimento do sistema operacional Android e sua contribuição para o mercado de celulares. Em seguida os modelos e processos da engenharia de softwares para construção dos aplicativos educacionais para *smartphones*. No capítulo 3 é apresentado a visão geral dos softwares através de fluxograma, a padronização de *layouts* e códigos adotados. Do capítulo 4 ao capítulo 7, estão expostas as telas, o manual de uso e algumas sugestões de práticas e os códigos mais importantes dos aplicativos. O capítulo 8 finaliza com as conclusões sobre o trabalho realizado.

O relatório técnico é parte integrante do projeto apresentado ao Programa de Pós-Graduação Stricto Sensu Mestrado em Tecnologia, Ambiente e Sociedade (PPGTAS) da Universidade Federal dos Vales do Jequitinhonha e Mucuri na Linha de Pesquisa: Tecnologia & Inovação e Sublinha de Pesquisa: Novas Tecnologias e Ciências intitulado de “Desenvolvimento de Softwares Educacionais em Plataforma Android para o Ensino de Matemática e Física”.

CAPÍTULO 2

REVISÃO DE LITERATURA

2.1 Linguagem Java

Conforme Schildt (2015), a Linguagem Java foi desenvolvida pela Sun Microsystems, em 1991, como parte do projeto de investigação Greem, que tinha por objetivo criar um software para dispositivos eletrônicos (televisores, videocassetes, torradeiras e outros tipos de utensílios), para que estes pudessem se comunicar com outros do mesmo tipo (classe).

Segundo Deitel (2016), a linguagem de programação Java é distribuída gratuitamente tornando-a, a mais utilizada no mundo. Essa é a linguagem adotada por muitas organizações por atender as necessidades de desenvolvimento de projetos de softwares corporativos, sendo também amplamente utilizado para implementar aplicativos, softwares baseados na internet e para dispositivos que se comunicam através de uma rede.

Somera (2006), descreve que a linguagem Java é uma linguagem orientada a objetos para o desenvolvimento de softwares, que podem ser transportados para qualquer plataforma; por exemplo: Windows, Linux, Free BSD ou Mac OS.

A Forrester Research presumiu que mais de dois bilhões de PCs estariam em uso até 2015². De acordo com a Oracle, 97% dos desktops corporativos, 89% dos desktops PC, 3 bilhões de dispositivos (Figura 1) e 100% de todos os players Blu-ray Disc™ executam o Java, e ainda existem mais de 9 milhões de desenvolvedores Java³.

De acordo com um estudo realizado pela Gartner, os dispositivos móveis continuarão a ultrapassar os PCs como os dispositivos de computação principais dos usuários; estimou-se que 1,96 bilhão de *smartphones* e 388 milhões de *tablets* foram distribuídos em 2015 - 8,7 vezes o número de PCs.⁴ Em 2018, o mercado de softwares para dispositivos móveis deverá alcançar US\$ 92 bilhões⁵. Isso está criando

² <http://www.worldometers.info/computers>.

³ <http://www.oracle.com/technetwork/articles/java/javaone12review-1863742.html>.

⁴ <http://www.gartner.com/newsroom/id/2645115>.

⁵ <https://www.abiresearch.com/press/tablets-will-generate-35-of-this-years-25-billion-/>

oportunidades profissionais significativas para pessoas que programam aplicativos móveis, muitos dos quais são programados em Java.

Figura 1 - Alguns dispositivos que usam Java.

Dispositivos		
Blu-ray Disc™	Caixas automáticos	Canetas inteligentes
Cartões de crédito	Consoles de jogos	Celulares
Cartões inteligentes	Dispositivos médicos	Copiadoras
Decodificadores de TV (set-top boxes)	Estações de pagamento de estacionamento	Desktops (computadores de mesa)
e-Readers	Medidores inteligentes	Impressoras
Eletrodomésticos	Robôs	Interruptores de luz
Imagens por ressonância magnética (MRIs)	Scanners de tomografia computadorizada	Roteadores
Passes de transporte	Sintonizadores de TV a cabo	Sistemas de diagnóstico veicular
Sistemas de aviação	Sistemas de segurança residencial	Sistemas de informação e entretenimento para automóveis
Smartphones	Tablets	Sistemas de navegação GPS
Terminais lotéricos	Termostatos	Televisões

Fonte: Deitel, 2016

2.2 Android - O sistema operacional móvel líder mundial

De acordo com um relatório do IDC (2017), no final do primeiro trimestre de 2017, o Android tinha 85,0% de participação no mercado global de *tablets*, comparados com 14,7% do iOS da Apple e 0,3% dos outros sistemas como Windows Phone.

Segundo Deitel(2015, p.3), as vendas de aparelhos portando o sistema operacional Android aumentaram rapidamente, criando enormes oportunidades para os desenvolvedores de aplicativos Android.

Com esta expansão, conforme Ableson et al. (2016, p.4), os fabricantes de dispositivos móveis necessitam de uma plataforma robusta e rica em funcionalidades para lançar no mercado seus produtos. Atualmente o sistema operacional para dispositivos móveis mais popular do mundo é o Android, que é uma plataforma moderna e ágil para desenvolvimento de aplicativos, que foi criado e é mantido pela empresa *Google Inc.*, tendo sua primeira versão lançada em outubro de 2008.

Uma característica da plataforma Android é que não tem distinção entre os aplicativos nativos para os softwares criados posteriormente através dos recursos que são disponibilizados pelo ambiente para os desenvolvedores. Isso equivale dizer que você pode escrever aplicativos poderosos que acessam recursos disponíveis no dispositivo. Outro atributo do Android é que o mesmo possui código-fonte aberto, isso permite que várias funcionalidades possam ser fornecidas sem depender do *Google*,

que é benefício oferecido pela plataforma de código-fonte aberto no mercado móvel (ABLESON et. al, 2016, p.4).

Atualmente os aplicativos que são desenvolvidos nessa plataforma são distribuídos ou compartilhados através de diversos catálogos ou repositórios de conteúdo na internet. O nome do serviço de repositório fornecido pelo *Google Inc.* é o *Google Play*, onde o usuário pode utilizar para localizar e/ou transferir o conteúdo para seu dispositivo móvel, ou para outras tecnologias compatíveis. A disponibilidade dos conteúdos varia entre os países, podendo alguns aplicativos não estarem disponíveis no país do usuário. Alguns temas pesquisados podem ser ofertados pelo *Google*, enquanto outros podem ser cedidos por terceiros, não se responsabilizando por conteúdos originados de fontes desconhecidas (GOOGLE, 2016).

2.3 Engenharia de Software: Modelos Incremental e iterativo

Existem modelos de processo sequenciais para desenvolvimento de aplicativos e softwares para PCs, como os modelos cascata e V, que são os mais antigos paradigmas da engenharia de software. Eles sugerem um fluxo de processos linear que, muitas vezes, é inadequado para os sistemas modernos (por exemplo, alterações contínuas, sistemas em evolução, prazos apertados). Entretanto, eles têm aplicabilidade em situações em que os requisitos são bem definidos e estáveis.

Modelos de processo incremental são iterativos por natureza e produzem rapidamente versões operacionais do aplicativo. Modelos de processos evolucionários reconhecem a natureza iterativa e incremental da maioria dos projetos de engenharia de software e são projetados para se adequar às mudanças. Esses modelos, como prototipação e o modelo espiral, produzem rapidamente artefatos de aplicativo incrementais (ou versões operacionais do aplicativo). Podem ser adotados para serem aplicados por todas as atividades de engenharia de software - desde o desenvolvimento de conceitos até a manutenção do sistema em longo prazo. (PRESSMAN, 2016 p. 64)

O modelo de processo concorrente permite que uma equipe de software represente elementos iterativos e concorrentes de qualquer modelo de processo. Modelos especializados incluem o modelo baseado em componentes (que enfatiza a montagem e a reutilização de componentes), o modelo de métodos formais (que estimula uma abordagem matemática para o desenvolvimento e a verificação de software) e o modelo orientado a aspectos (que considera preocupações transversais que se estendem por toda a arquitetura do sistema). O Processo Unificado é um processo de software "dirigido a casos de uso, centrado na arquitetura, iterativo e

incremental, desenvolvido uma metodologia para os métodos e ferramentas da UML” (AZEVEDO;CAMPOS, 2008).

2.4 Processo de Desenvolvimento de Aplicativos Educacionais

Segundo a proposta de Benitti et al. (2005) que diz que o processo de desenvolvimento de aplicativos educacionais baseia-se em quatro etapas de forma iterativa e incremental: concepção, elaboração, finalização e viabilização.

Concepção é a fase que define as etapas do aplicativo educacional onde o ponto de partida são os objetivos de aprendizagem que foram estruturados em requisitos computacionais, momento no qual os profissionais da área de computação e da educação são determinantes. É nesta fase que o conteúdo didático do aplicativo deve ser definido por professores baseado na concepção de software levando em conta as etapas de implementação. Na fase de elaboração se cria um protótipo funcional do aplicativo direcionado pela etapa de concepção.

A finalização não se resume na última etapa, mas é a fase no qual o aplicativo é testado pelos utilizadores. Nesta etapa também é realizado pelos autores ajustes que foram observados no processo de validação.

A viabilização é o processo em que os profissionais da educação, neste caso os professores, recebem o treinamento de interação de que forma as atividades didáticas disponibilizadas pelo aplicativo poderão ser utilizadas nas aulas.

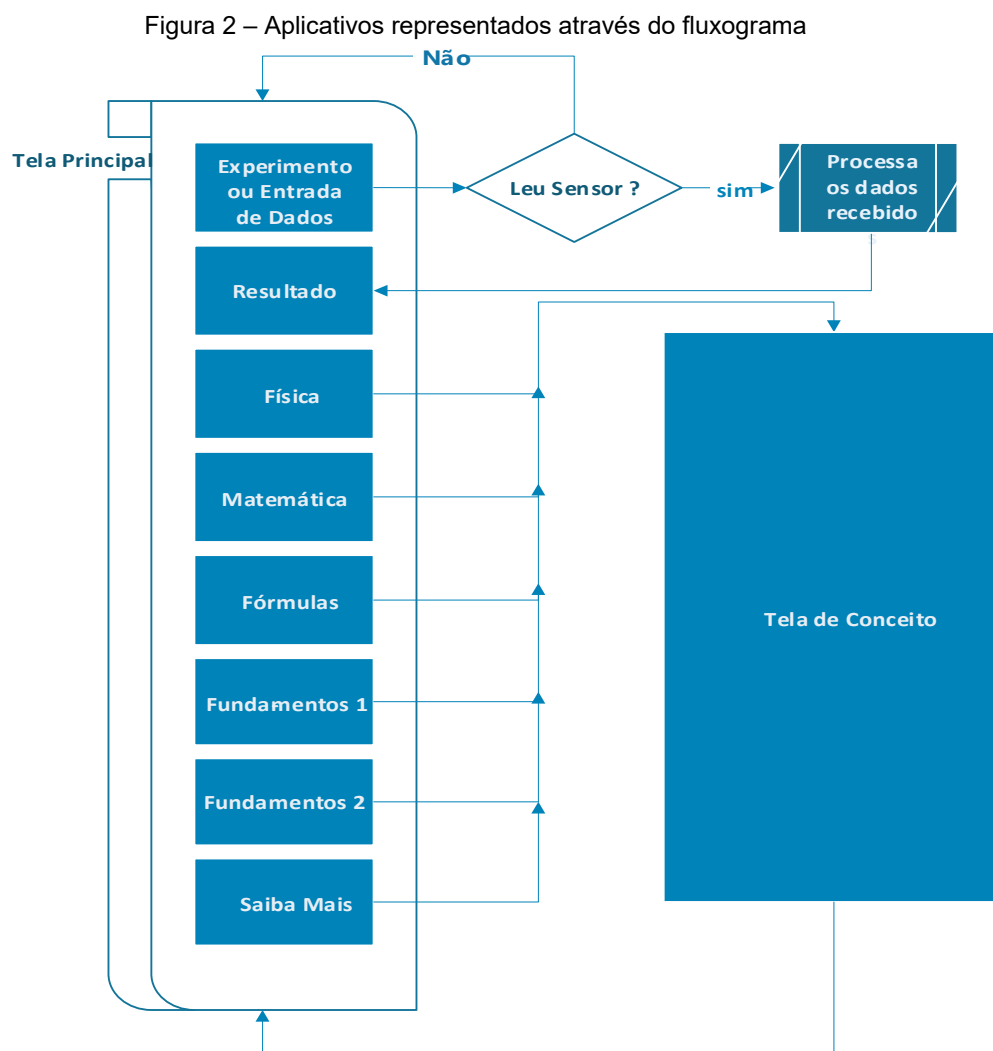
CAPÍTULO 3

VISÃO GERAL DOS APLICATIVOS

3.1 Fluxograma de usabilidade

O cenário de funcionamento dos aplicativos será demonstrado no fluxograma da Figura 2 que contém o fluxo de ações esperadas pelas aplicações.

No diagrama foram expostas as representações dos recursos suportadas pelas aplicações seus acessos e interações. Após a inicialização as aplicações exibem a tela principal com as opções de entrada. Devido à particularidade de alguns aplicativos foi necessária uma variação neste padrão.

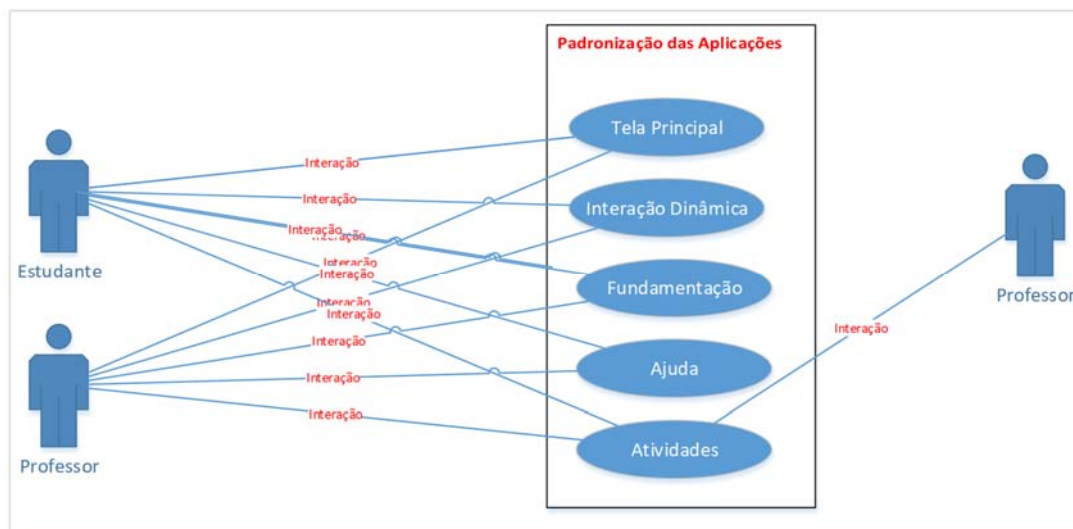


Fonte: O autor (2017)

3.2 Etapa de Visualização

A etapa de visualização, contempla o acompanhamento da utilização dos aplicativos, tendo em vista que alguns erros ainda poderão ocorrer, falhas que não foram detectadas nas fases anteriores. Outro aspecto importante neste estágio é a manutenção e suporte, tanto na área didática, quanto na computacional. Na Figura 3 exibe-se um diagrama que mostra a interação entre os atores, no caso professores e alunos interagindo com o aplicativo.

Figura 3 – Etapas da Visualização



Fonte: O autor (2017)

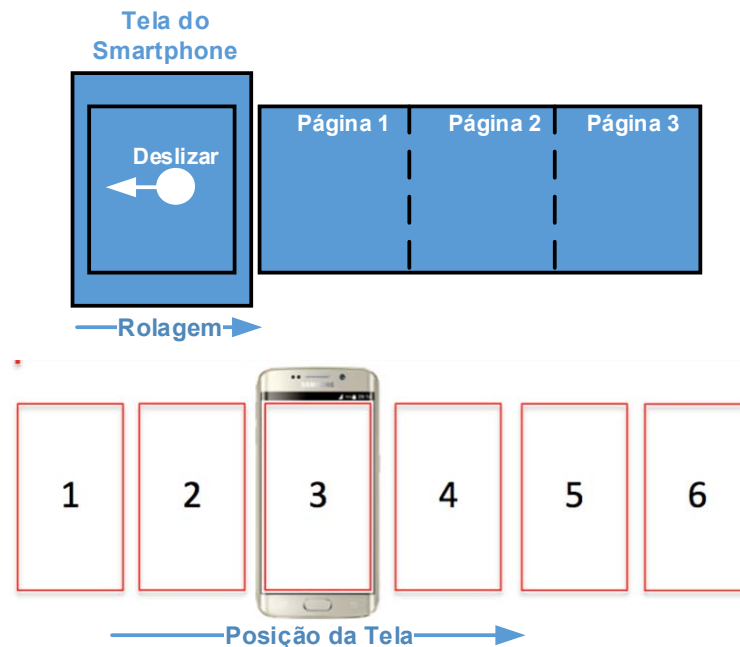
O fluxograma acima não exibe todas as telas, pois em algumas interações entre as telas auxiliares são exibidas através de recurso da linguagem Java para Android denominadas de *intents*, que é “um objeto de mensagem que pode ser usado para solicitar uma ação de outro componente de aplicativo”. (Google Developers, 2016).

3.3 Processo de Desenvolvimento dos Aplicativos: Material Designer

As telas de interface com o usuário foram construídas baseadas em *Material Designer Android ViewPager Screen Slides*, onde suas trocas são feitas deslizando o dedo sobre a tela do dispositivo, como mostra a Figura 4, baseado nos objetivos da Interação Humano–Computador (IHC), que define a experiência do usuário (EU), do

inglês *User Experience* (UX) onde determina como o usuário deverá interagir com o aplicativo até mesmo o design emocional, como por exemplo, as cores de botões, texto e telas. Os “*ViewPager Screen Slides*” são transições entre uma tela para outra e são comuns com a interface do usuário (UI) com assistentes de configuração ou apresentações de slides”. (Google Developers, 2016).

Figura 4 – Diagrama *ViewPager Screen Slides*



Fonte: O autor (2017)

3.3.1 Os *Layouts* dos Aplicativos

Para codificar os *layouts* do aplicativo, que definem a estrutura visual para a interface do usuário, foi utilizada a linguagem XML (*eXtensible Markup Language*) em português Linguagem Extensível de Marcação Genérica. O XML é o vocabulário do que corresponde às classes e subclasses de *View*, como as de *widgets* e *layouts*.

A Figura 5 mostra um exemplo do arquivo `layout_fisica.xml` do aplicativo que utiliza um *Layout Linear Vertical* para conter um exibidor de texto, Visualizador de Imagem e Visualizador Web que exibe o texto como páginas Web em HTML — com uma série de elementos aninhados.

Cada arquivo de *layout* deve conter exatamente um elemento raiz, que deve ser um objeto *View* ou *ViewGroup*. Com o elemento raiz definido, é possível adicionar

objetos ou widgets de *layout* extras como elementos filho para construir gradualmente uma hierarquia de *View* que define o *layout*.

Figura 5 – Exemplo *layout* declarado em XML

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      xmlns:card_view="http://schemas.android.com/apk/res-auto"
5      android:orientation="vertical" >
6      <FrameLayout
7          android:id="@+id/topHolder"
8          android:layout_height="0dp">
9          <LinearLayout
10             android:id="@+id/LL_bandholder"
11             android:layout_height="match_parent"
12             android:orientation="horizontal" >
13             <FrameLayout
14                 android:id="@+id/FL_resistor"
15                 android:layout_height="match_parent"
16                 android:gravity="center" >
17                 <com.FisiscalabResistores.viewpager.extensions.sample.ResistorView
18                     android:id="@+id/resistor_view"
19                     android:layout_marginTop="4sp"
20                     android:layout_marginBottom="4sp" />
21             </FrameLayout>
22             <LinearLayout
23                 android:layout_width="match_parent"
24                 android:layout_height="match_parent"
25                 android:orientation="vertical" >
26                 <LinearLayout
27                     android:layout_width="match_parent"
28                     android:background="@color/gray2">
29                 </LinearLayout>
30             </LinearLayout>
31         </LinearLayout>
32 </LinearLayout>

```

Fonte: o autor (2016)

3.3.2 Codificação

Para escrever os códigos foi utilizado o IDE (Interface de Desenvolvimento) Android Studio⁶, disponibilizado e mantido pela *Google*, que utiliza a linguagem de programação *Java Development Kit* (JDK), que é totalmente adaptada para desenvolvimento de aplicativos na plataforma Android. Outro componente importante utilizado junto com o IDE foi o SDK⁷ (*Software Development Kit*), conjunto de ferramentas que auxiliam o desenvolvimento de aplicativos para pacote de software.

⁶ <https://developer.android.com/studio/index.html>

⁷ <https://developer.android.com/studio/releases/sdk-tools.html>

A Figura 6 exibe parte dos códigos dos aplicativos escrito em Java do arquivo *MainActivity.java* que é responsável pela execução principal do aplicativo.

Figura 6 – Códigos do *MainActivity* escrito em Java.

```

1 import butterknife.ButterKnife;
2 import butterknife.InjectView;
3 public class MainActivity extends AppCompatActivity {
4     @InjectView(R.id.toolbar)
5     Toolbar toolbar;
6     @InjectView(R.id.tabs)
7     PagerSlidingTabStrip tabs;
8     @InjectView(R.id.pager)
9     ViewPager pager;
10    EditText v1R1,v1R2,v1R3,v1TotalSerie;
11
12    private MyPagerAdapter adapter;
13    private Drawable oldBackground = null;
14    private int currentColor;
15    private SystemBarTintManager mTintManager;
16
17    @Override
18    protected void onCreate(Bundle savedInstanceState) {
19        super.onCreate(savedInstanceState);
20        setContentView(R.layout.activity_main);
21        ButterKnife.inject(this);
22        // cria uma instância após a exibição do conteúdo estar configurada
23        mTintManager = new SystemBarTintManager(this);
24        mTintManager.setStatusBarTintEnabled(true);
25        adapter = new MyPagerAdapter(getSupportFragmentManager());
26        pager.setAdapter(adapter);
27        tabs.setViewPager(pager);
28        final int pageMargin = (int) TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 4, getResources()
29            .getDisplayMetrics());
30        pager.setPageMargin(pageMargin);
31        changeColor(getResources().getColor(R.color.green));
32
33    private void changeColor(int newColor) {
34        tabs.setBackgroundColor(newColor);
35        mTintManager.setTintColor(newColor);
36        Drawable colorDrawable = new ColorDrawable(newColor);

```

Fonte: O autor (2017)

CAPÍTULO 4

ETAPAS DO DESENVOLVIMENTO DO APLICATIVO MATE CÓDIGO DE BARRAS

4.1 – Objetivo do Aplicativo

O objetivo principal deste aplicativo é ler e calcular o dígito verificador do código de barras padrão EAN13 que significa *European Article Number* (Número de artigo Europeu)(COBAN, 2018) que possui 13 dígitos, onde os três iniciais identificam o país de origem. Os próximos 9 dígitos representam o código da empresa fabricante e do produto por ela produzido e o último dígito serve para verificação do escaneamento.

O funcionamento do aplicativo se dá capturando a imagem de códigos de barras EAN13 através da câmera do *smartphone*, que neste caso substitui um scanner óptico tradicional. A imagem capturada é reconhecida através de um algoritmo que exibe os cálculos matemáticos utilizados para encontrar o dígito verificador do código de barras conforme Tabela 1 e Imagem 7, além de apresentar telas com conceitos que auxiliam na compreensão dos resultados.

Tabela 1 - Passos para calcular o dígito verificador do código EAN13

Posição	13	12	11	10	9	8	7	6	5	4	3	2	1
Código EAN	7	8	9	1	1	5	0	0	4	4	2	5	x
	X												
Fator Peso	1	3	1	3	1	3	1	3	1	3	1	3	x
	=												
Código X Peso	7	24	9	3	1	15	0	0	4	12	2	15	x
Soma	7 + 24 + 9 + 3 + 1 + 15 + 0 + 0 + 4 + 12 + 2 + 15 = 92												
Qual o valor que é adicionado a soma que é múltiplo de 10 ?	Próximo múltiplo de 10 é 100, então $100 - 92 = 8$ Então Dígito verificador é igual a 8.												

Fonte: O autor (2018)

Figura 7 – Método utilizado no aplicativo para calcular o dígito verificado.

```

1  int calculaDiigito(int resultado) {
2      int digV = 0;
3      while (resultado%10!=0)
4      {
5          resultado /= 10;
6          digV++;
7      }
8      String mystring = getResources().getString(R.string.desc_rodape);
9      mystring = mystring.replace("$x", "<font color='#4CAF50'> X = "+ digV + "</font>");
10     Spanned sp = Html.fromHtml(mystring);
11     ((TextView)pager.findViewById(R.id.textview893)).setText(sp);
12     return resultado;
13 }

```

Fonte: O autor (2018)

4.2 Telas e *Layouts* do Aplicativo Mate Códigos de Barras

Como já mencionado anteriormente os desenhos e cores das telas (*layouts*) foram padronizados buscando uma melhor eficiência e usabilidade do aplicativo.

Abaixo serão mostrados todos os passos para a utilização do aplicativo.

4.2.1 Tela Principal

Na tela principal o aluno ou professor, quando iniciar o aplicativo, poderá visualizar um menu superior na tela. A primeira opção é a de INÍCIO, onde possui os botões para orientações e acesso direto as opções do Menu como mostrado na Figura 8.

Figura 8 – Tela Principal do Aplicativo



Fonte: O autor (2016)

4.2.2 Tela de Leitura do Código de Barras

O primeiro passo da utilização como recurso didático para aulas de física e matemática é realizar a leitura do código de barras tocando no ícone de leitura localizado no Menu Início como exibido na Figura 9.

Figura 9 – Inicialização da leitura do Código de Barras



Fonte: O autor (2016)

Logo após a câmera do *smartphone* ser iniciada, exibirá uma linha guia vermelha, que deverá ser apontada para um código de barras EAN13 como mostrado na Figura 10.

Figura 10 – Realizando a leitura do código de barras através da câmera do dispositivo

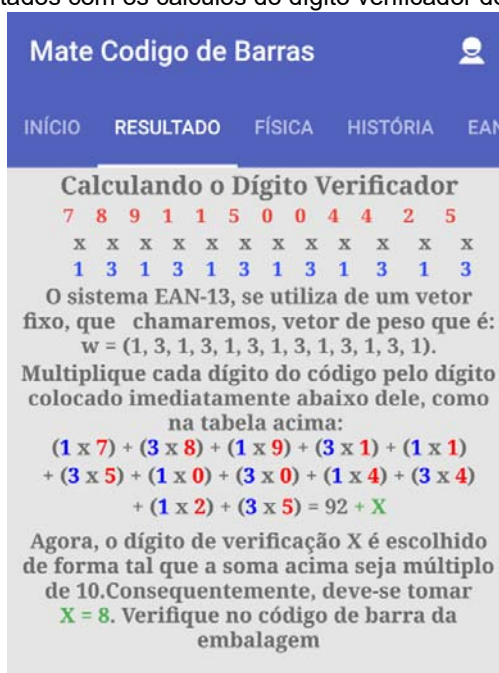


Fonte: O autor (2016)

4.2.3 Tela de Resultados do Dígito Verificador

Após a leitura o aplicativo será automaticamente direcionado para a tela “RESULTADO” contendo todos os cálculos matemáticos que foram utilizados para encontrar o dígito verificador do código de barras, mostrado na Figura 11.

Figura 11 – Tela de resultados com os cálculos do dígito verificador do código de barras.



Fonte: O autor (2016)

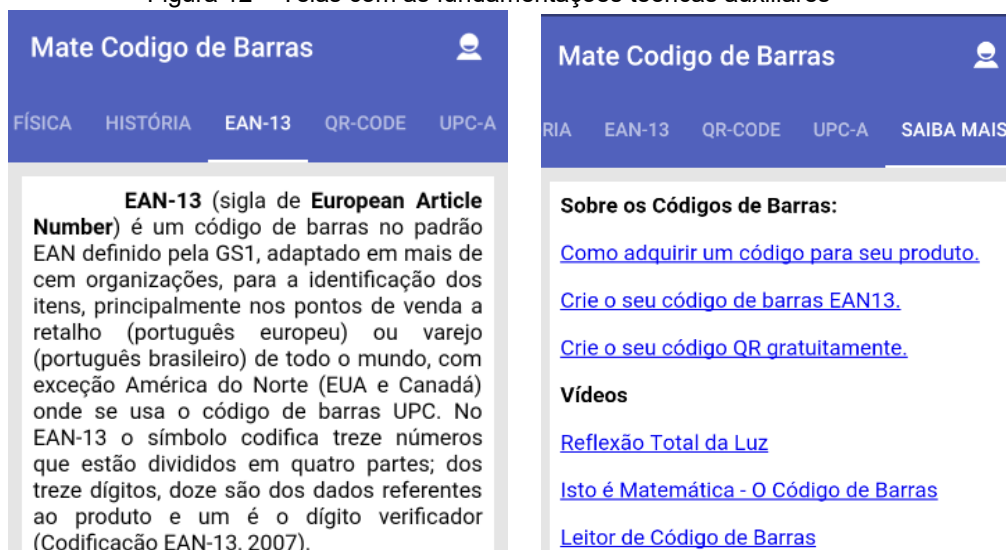
Os resultados também poderão ser acessados novamente através do ícone presente no menu “INÍCIO”.

4.2.4 Telas de Conceitos e Fundamentos

Buscando alinhar teoria e prática foi disponibilizado no aplicativo a fundamentação teórica sobre os códigos de barras distribuídos em cinco telas, procurando uma melhor utilização do aplicativo como instrumento de ensino-aprendizagem contextualizado, levando à continuidade da aprendizagem além das salas de aula e permanentemente acessível aos alunos. As telas foram dispostas da seguinte forma: uma de “FÍSICA” que apresenta a fundamentação teórica aplicada

nos leitores óticos utilizados em estabelecimento de vendas, a de “HISTÓRIA” dos códigos de barras, expondo o contexto histórico e seus inventores, outra “EAN-13” apresentando os conceitos deste tipo de código de barra, que é utilizado nos produtos brasileiros, seguida pela tela “QR-CODE” que é mais moderno capaz de armazenar pequenos textos ao invés de números. Na próxima tela exibe todos os detalhes do “UPC-A”, que é uma variação do código EAN-13 e por último a tela “SAIBA MAIS” onde são apresentados links para vídeos e sites que possibilitam tanto ao aluno quanto ao professor expandir o conhecimento na área. As telas são mostradas na Figura 12 abaixo.

Figura 12 – Telas com as fundamentações teóricas auxiliares



Fonte: O autor (2016)

CAPÍTULO 5

ETAPAS DO DESENVOLVIMENTO DO APLICATIVO FÍSICA LAB RESISTORES

5.1 – Objetivo do Aplicativo

O objetivo do aplicativo é trabalhar conteúdos da Eletricidade, mais precisamente associação de resistências utilizando resistores, que são componentes eletrônicos que integram circuitos elétricos, a sua finalidade principal é conversão de energia elétrica em energia térmica (Efeito Joule), e também alterar a diferença de potencial em determinada parte do circuito.

O valor ôhmico dos resistores é determinado por um código de cores sendo que os mais comuns possuem quatro faixas coloridas.

Em um circuito, os resistores podem se associar de três maneiras diferentes chamadas de associação em: série; paralelo e mista representadas, nas Figuras 14 e 16. Quando no mesmo circuito existem resistências associadas em série e paralelo. As associações permitem obtenção de diferentes valores de resistência elétrica.

E foi neste sentido que o aplicativo foi idealizado e desenvolvido para buscar auxiliar tanto os alunos quanto os professores de Física a identificarem o valor ôhmico dos resistores e para simular os valores resultantes das associações.

5.2 Telas e *Layouts* do Aplicativo Física Lab Resistores

Os *layouts* foram organizados buscando criar uma sequência didática para favorecer a sua utilização pelos professores e alunos em sala de aula.

Este aplicativo foi aplicado pelo Programa Institucional de Bolsas de Iniciação à Docência (Pibid) da UFVJM – Campus do Mucuri em uma turma de 2º ano do ensino médio com 28 alunos. A avaliação foi realizada pela professora de Física da turma e os resultados estão no ANEXO III.

5.2.1 Tela Principal

Foi criada uma tela para identificação do valor resistivo de um resistor, onde foi desenvolvido um layout utilizando a classe “*View*” do Android, para simular um

resistor, e com o auxílio da classe “Draw” da linguagem Java foi possível criar um novo comportamento da imagem toda vez que uma nova faixa de cor é selecionada.

A tela principal do aplicativo exibe uma imagem de um resistor com três e quatro faixas de cores, onde o usuário poderá selecionar a faixa tocando na imagem e logo após uma cor na tabela ao lado. A quantidade de faixas da resistência é escolhida acima da tabela de cores. A cada mudança de cor nas faixas, automaticamente o valor ôhmico do resistor é calculado pelo algoritmo e mostrado no campo “valor ôhmico” mostrado na Figura 13.

Outro ponto importante é o percentual de tolerância, que significa que na prática o valor do resistor pode variar para mais ou para menos dependendo do valor da cor presente na última faixa. O percentual é mostrado no centro entre os dois valores calculados, como mostra a Figura 13.

Outra funcionalidade do aplicativo é que se o usuário ao invés de escolher as cores, ele pode também informar o valor desejado da resistência no campo “valor ôhmico” e confirmar com “enter” o algoritmo irá calcular e mostrar as cores no resistor correspondente ao valor informado.

Figura 13 – Tela Principal do Aplicativo – Resistores de 4 e 5 faixas de cores.



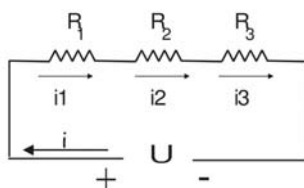
Fonte: O autor (2017)

5.2.2 Telas de Associação em Série e Paralelo

Para o cálculo da resistência total das associações foram implementadas em código Java, as fórmulas da Lei de Ohm, nome utilizado para homenagear o físico alemão Georg Simon Ohm (CAVALCANTE, 2018).

Para encontrar a resistência equivalente de um circuito em série, basta o usuário acessar a aba “SÉRIE”, e logo após informar os valores das três resistências em seus campos correspondentes e pressionar o botão “Calcular”, estas opções são mostradas na Figura 18. A fórmula utilizada para este cálculo é mostrada na Figura 14 e os códigos na Figura 15.

Figura 14 – Fórmula para cálculo da resistência equivalente de uma associação em série.



$$R_{eq.} = R_1 + R_2 + R_3$$

Fonte: O autor (2017)

Figura 15 – Código utilizado para cálculo da resistência equivalente de uma associação em série.

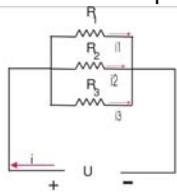
```

1 private void CalculaReistenciaTotalSerie(){
2     DecimalFormat precision = new DecimalFormat("0.00");
3     double R1 = Double.parseDouble(vlR1.getText().toString());
4     double R2 = Double.parseDouble(vlR2.getText().toString());
5     double R3 = Double.parseDouble(vlR3.getText().toString());
6     double total = R1 + R2 + R3;
7     EditText edt = (EditText) findViewById(R.id.resistTotal);
8     edt.setText(precision.format(total));
9
10 }
```

Fonte: O autor (2017)

Para calcular a resistência equivalente numa associação em paralelo o usuário deverá acessar a aba “PARALELO”, em seguida entrar com o respectivo valor ôhmico no campo indicado, e clicar no botão “calcular”; como visto na Figura 18. A fórmula aplicada para determinar a resistência total do circuito é indicada na Figura 16 e os códigos na Figura 17.

Figura 16 – Fórmula para o cálculo da resistência equivalente de uma associação em paralelo.



$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

Fonte: O autor (2017)

Figura 17 – Código Java com o Método

```

1 - private void CalculaResistenciaTotalParalelo(){
2     DecimalFormat precision = new DecimalFormat("0.00");
3     double R1 = Double.parseDouble(vlR1.getText().toString());
4     double R2 = Double.parseDouble(vlR2.getText().toString());
5     double R3 = Double.parseDouble(vlR3.getText().toString());
6     if(R1!=0) R1 = 1/R1;
7     if(R2!=0) R2 = 1/R2;
8     if(R3!=0) R3 = 1/R3;
9     double total = Math.pow((R1 + R2 + R3),-1);
10    EditText edt = (EditText) findViewById(R.id.resistTotalPara);
11    edt.setText(precision.format(total));
12 }

```

Fonte: O autor (2017)

Figura 18 – Telas com as opções para o cálculo das associações em série e paralelo.

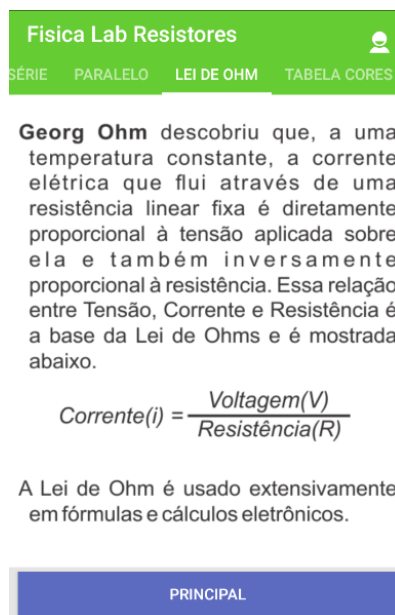


Fonte: O autor (2017)

5.2.3 Tela com Conteúdo Teórico.

Na aba “LEI DE OHM”, Figura 19, foi disponibilizado uma breve teoria que mostra a relação entre Tensão, Corrente e Resistência mostrando sua importância para a Lei de Ohm. Nesta tela o aluno poderá acessar o conteúdo para acompanhar as explicações do professor durante as aulas ou para estudar em qualquer outro momento ou alguma atividade proposta.

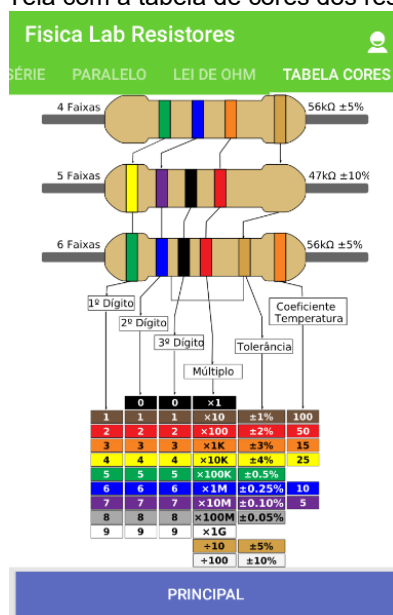
Figura 19 – Tela com Conteúdo Teórico - LEI DE OHM.



Fonte: O autor (2017)

A Figura 20 demonstra como funciona a codificação dos resistores baseado em uma tabela de cores. Esta tabela além de estar disponível no aplicativo foi também a lógica utilizada para criar a primeira tela, “PRINCIPAL”, onde encontra-se o resistor virtual dinâmico. A tela “TABELA CORES” possui a imagem contendo três resistores com 4, 5 e 6 faixas com seus respectivos valores. Para um resistor de quatro faixas a primeira e a segunda são os dígitos, a terceira o multiplicador, e a última o valor de tolerância. No de cinco faixas, a primeira, segunda e terceira são os dígitos, a quarta o multiplicador e a última refere-se ao valor de tolerância. Para o de seis faixas, a primeira, segunda, terceira e quarta representam os dígitos, a quinta representa a tolerância e a última o coeficiente de temperatura.

Figura 20 – Tela com a tabela de cores dos resistores.

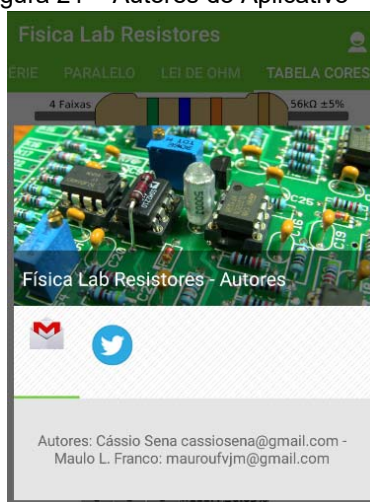


Fonte: O autor (2017)

5.2.4 Tela de Autores.

Para acessar a tela de autores basta tocar no ícone presente na parte superior do layout como mostra a Figura 21. Ao acessar são mostrados os dados de contatos dos idealizadores e criadores do aplicativo.

Figura 21 – Autores do Aplicativo



Fonte: O autor (2017)

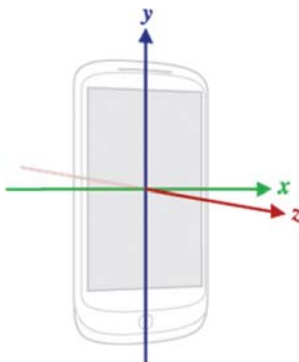
CAPÍTULO 6

ETAPAS DO DESENVOLVIMENTO DO APLICATIVO FÍSICA LAB CINEMÁTICA

6.1 – Objetivo do Aplicativo

O objetivo deste aplicativo é realizar experimentos utilizando o acelerômetro. Esse sensor monitora o movimento ou posicionamento tridimensional do dispositivo com alta precisão baseado na força da gravidade atuante sobre os eixos mostrado na Figura 22.

Figura 22. Sistema de Coordenadas Relativo a um Dispositivo



Fonte: *Sensors Overview*, Google Developers (2017)

A primeira finalidade do aplicativo é ser utilizado como cronômetro inteligente. O aluno ou professor podem medir com grande precisão o tempo que um objeto esférico percorre uma determinada distância em um plano inclinado, assim medindo com exatidão a sua velocidade média da origem até o destino final.

O segundo propósito é utilizar o aplicativo para determinar o ângulo de inclinação do dispositivo em todos os seus eixos. Utilizando este recurso o docente e o discente poderão determinar o coeficiente de atrito estático entre duas superfícies em um plano inclinado.

6.2 Telas e *Layouts* do Aplicativo Física Lab Resistores

No desenvolvimento das telas e *layouts* teve-se uma atenção especial quanto ao tamanho e disposição dos objetos, como os “*TextViews*”, que são os exibidores de

texto, devido a quantidade de dados a serem mostrados. Outra alteração foi no tamanho e posicionamento do botão “Preparar” presente na tela principal, na disposição e tamanho dos “*Gauges*” que são os visualizadores gráficos presentes na tela “Ângulo”.

6.2.1 Tela Cronômetro Inteligente e Experimento

A primeira tela propõe o cronômetro inteligente que consegue ler as mínimas vibrações produzidas sobre uma superfície. Essa percepção se dá através da variação da gravidade. O *smartphone* deverá ficar com a tela para cima sobre a superfície inclinada, então o eixo Z é o ponto de referência onde o aplicativo monitora as alterações na força gravitacional, destacado em vermelho na Figura 23. Outro dado complementar para cada eixo é valor máximo atingido pela gravidade em cada ponto, marcado na cor verde na Figura 23.

Figura 23. Tela do Cronômetro inteligente

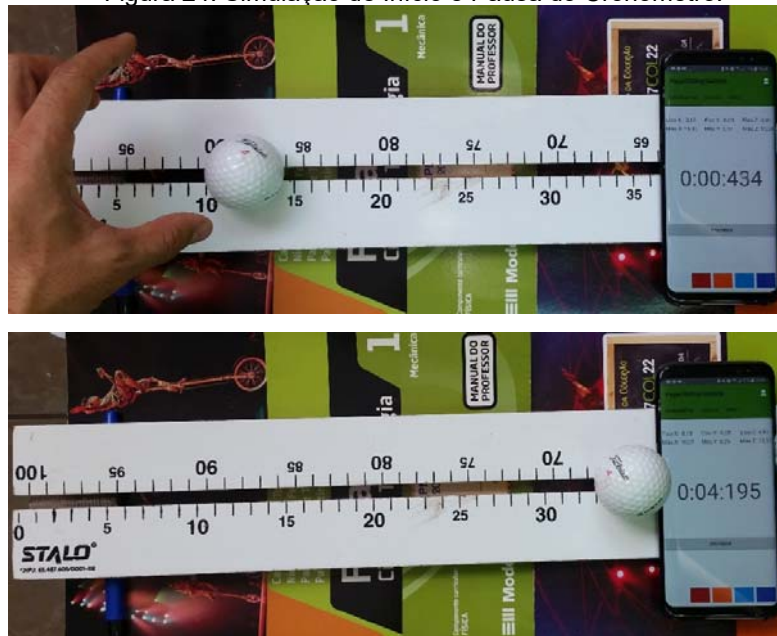


Fonte: O autor (2017)

No centro da tela se encontra o contador de tempo que possui uma precisão em milésimos de segundos. Para que o contador seja disparado é necessário que o usuário posicione o dispositivo sobre o plano e pressione o botão “Preparar”, então a partir deste momento, o aplicativo entrará em modo de espera, aguardando qualquer

mudança significativa sobre a força atuante no eixo Z, logo após basta soltar um objeto esférico sobre o plano inclinado, onde o aplicativo posicionado perceberá a vibração provocada pelo impacto do corpo, que dará início a contagem que só encerrará quando o objeto em movimento chocar-se contra o *smartphone*, como mostra a Figura 24. Para reiniciar e zerar o contador basta pressionar o botão “Preparar” novamente e se necessário repetir o experimento.

Figura 24. Simulação de Início e Pausa do Cronômetro.



Fonte: O autor (2017)

O código em Java para iniciar e pausar o cronômetro é mostrado abaixo na Figura 25, onde os dados são capturados dentro do método sobrescrito “*onSensorChanged*” que é uma implementação da interface “*SensorEventListener*”, que um recurso da plataforma Android.

Figura 25 - Código em Java, Início e Pausa do Cronômetro.

```

1  @Override
2  public void onSensorChanged(SensorEvent event) {
3
4      //Parando o Cronômetro.
5      if( event.values[0] < -1.5 && timeState)
6          StopTime();
7      //Iniciando o Cronometro.
8      if( event.values[2] > 11.0 && !timeState){
9          StartTime();
10         timeState = true;
11     }
12

```

Fonte: O autor (2017)

6.2.2 Tela Ângulo, Nível e Experimento

Na segunda tela do aplicativo o aluno ou o professor poderá aferir a inclinação de um plano e também seu nivelamento, uma vez que a superfície nivelada permitirá maior precisão dos resultados.

Os instrumentos utilizados para medir ângulos e inclinações em experimentos mecânicos requerem alta precisão. As simulações nos planos inclinados com instrumentos tradicionais não são satisfatórios, então o aplicativo é uma boa alternativa para melhoria de tais medidas. O medidor de ângulo é mostrado na Figura 26, onde exibe os ângulos formados em relação a todos os eixos, sendo eixo Z a referência, pois o dispositivo está sobre o plano com a tela para cima. A tela mostra em tempo real o valor angular de uma inclinação e a recalibração periódica do smartphone garante uma boa qualidade das medições.

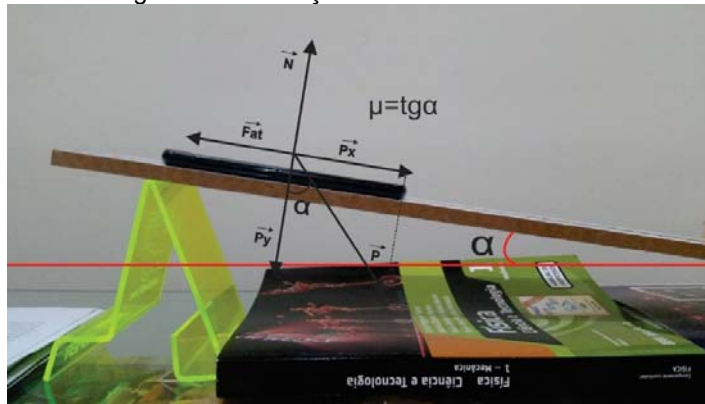
Figura 26. Inclinação do *Smartphone* – Eixo Z



Fonte: O autor (2017)

Para calcular o coeficiente de atrito o aluno ou professor deverá montar um plano inclinado, conforme Figura 27, colocar o *smartphone* sobre a superfície, e na sequência iniciar a inclinação até que o dispositivo comece a deslizar. Neste momento basta verificar o ângulo mostrado na tela do aparelho. O coeficiente de atrito é encontrado calculando a tangente do ângulo exibido pelo aplicativo, que é exatamente o ângulo de inclinação do plano. O experimento e as decomposições das forças atuantes sobre o aparelho são mostrados na Figura 27.

Figura 27. Simulação em um Plano Inclinado



Fonte: O autor (2017)

A Figura 28 exibe o código em Java utilizado para calcular o ângulo de inclinação do dispositivo em relação eixo Z.

Figura 28 - Código Java que Calcula a Inclinação em Relação ao Eixo Z Inclinado.

```

1 public double CalculaInclinacao(float[] event) {
2
3     float[] g = new float[3];
4     g = event.values.clone();
5     double norm_of_g = Math.sqrt(g[0] * g[0] + g[1] * g[1] + g[2] * g[2]);
6     // Normalizando o Vetor Aceleração
7     g[0] = (float) (g[0]/norm_of_g);
8     g[1] = (float) (g[1]/norm_of_g);
9     g[2] = (float) (g[2]/norm_of_g);
10
11     double inclinacao = (double) Math.round(Math.toDegrees(Math.acos(g[2])));
12     return inclinacao;
13 }

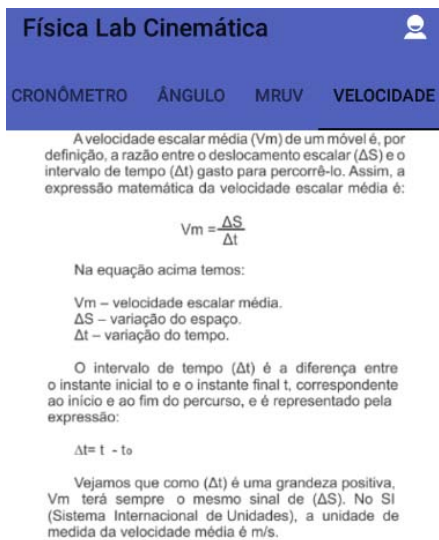
```

Fonte: O autor (2017)

6.2.3 Tela com Conteúdo Teórico.

Na aba “VELOCIDADE”, Figura 29, foi disponibilizada a teoria relacionada ao cálculo da velocidade média de um objeto em movimento. Mostra a relação entre distância percorrida e o tempo. Essa tela pode ser utilizada para ajudar o aluno em atividades complementares propostas pelo professor aos alunos, até mesmo auxiliar para estudos extraclasse.

Figura 29. – Tela com Conteúdo Teórico - Velocidade Média.

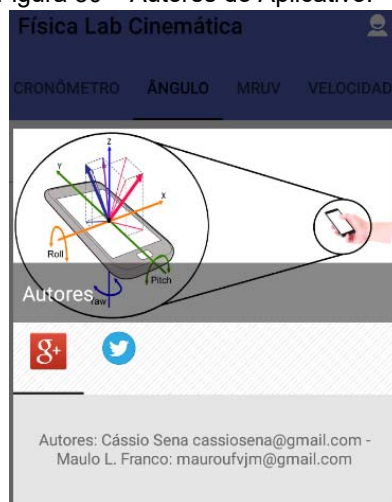


Fonte: O autor (2017)

6.2.4 Tela de Autores.

Para acessar a tela de autores basta tocar no ícone presente na parte superior do layout como mostra a Figura 30. Assim é mostrado os dados de contatos dos idealizadores e criadores do aplicativo.

Figura 30 – Autores do Aplicativo.



Fonte: O autor (2017)

CAPÍTULO 7

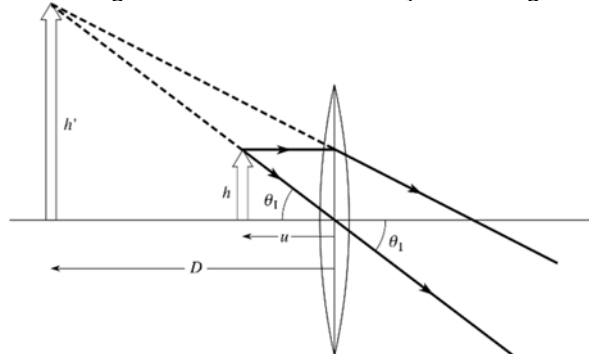
ETAPAS DO DESENVOLVIMENTO DO APLICATIVO FÍSICA LAB ÓPTICA

7.1 – Objetivo do Aplicativo

Este aplicativo foi desenvolvido para demonstrar o conceito básico do diagrama de raios no processo de formação de imagens nos espelhos esféricos, em lentes convexas, côncavas e a simulação da dispersão da luz através de um prisma.

Através da tela, tanto o aluno quanto o professor poderão arrastar um objeto variando sua distância em relação à lentes e espelhos esféricos e em tempo real acompanhar os raios que dão origem a formação geométrica da imagem (Figura 31).

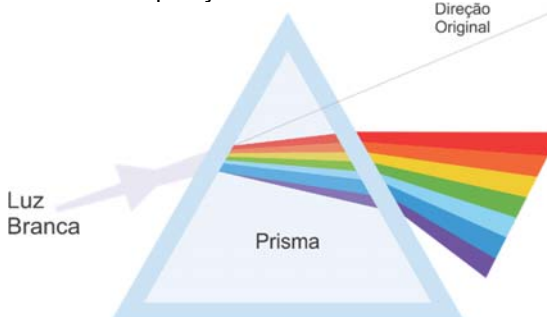
Figura 31 – Diagrama de Raios na Formação de Imagens



Fonte: *Optical instruments, The Open University* (2018)

Na tela “Prisma” existe a simulação de uma luz policromática, aquela composta por uma combinação de duas ou mais cores monocromáticas, incidindo no prisma em um de seus lados e no outro mostra a dispersão dessa luz em várias cores buscando demonstrar a decomposição da luz branca como mostra a Figura 32.

Figura 32 – Decomposição da Luz Através do Prisma



Fonte: O autor (2017)

7.2 Telas e *Layouts* do Aplicativo Física Lab Óptica

Houve uma atenção especial no desenvolvimento dos *layouts* para que estes ocupassem o maior espaço possível da tela do dispositivo para melhor visualização, já que os recursos apresentados pelo aplicativo tratam de imagens dinâmicas, que redimensionam a todo momento. Este aproveitamento de espaço também contribuiu para melhor percepção do efeito da dispersão no prisma. Uma das alterações em relação aos outros aplicativos anteriormente descritos foi a retirada do botão “Principal”, presente em todas as telas para facilitar o retorno a primeira tela do aplicativo.

7.2.1 Tela de Lentes, Espelhos e Comportamentos

As primeiras quatro telas do aplicativo apresentam lentes e espelhos esféricos respectivamente. Possuem o eixo de simetria com o centro de curvatura representado pela letra “C” e o foco simbolizado pela letra “F”. Em frente às lentes e os espelhos encontra-se uma pena que simula o objeto principal que tem a sua imagem formada pelo diagrama de raios, presente nas Figuras 33 e 34. O objeto (pena), pode ser arrastado e posicionado em qualquer lugar em cima do eixo de simetria. À medida que ele se afasta ou se aproxima da lente ou do espelho, o diagrama de raio forma a sua imagem com propriedades diferentes. Essa mudança nas características da imagem é prevista pela óptica geométrica. Quando o professor ensinar essa abordagem da Física os alunos poderão utilizar o aplicativo para verificar e responder questões durante as aulas.

Quando a imagem formada pelos raios for virtual, isto é, quando vista no ponto de encontro dos prolongamentos dos raios refletidos, aparece um desenho (destacado em vermelho na Figura 33) de um fantasma colorido embaixo da imagem para simbolizar a característica.

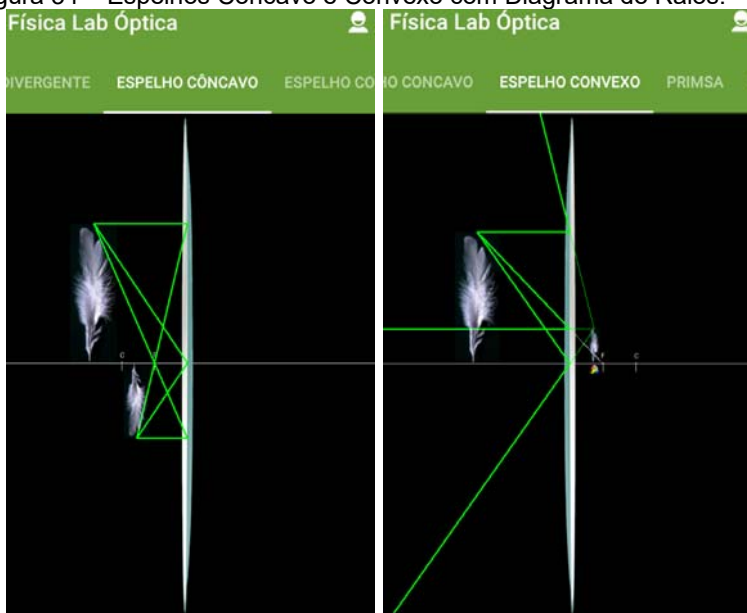
O método utilizado para criar e reposicionar os raios durante o movimento do objeto sobre o eixo de simetria é mostrado na Figura 35 e na Figura 36.

Figura 33 – Lentes Convergente e Divergente com Diagrama de Raios.



Fonte: O autor (2017)

Figura 34 – Espelhos Côncavo e Convexo com Diagrama de Raios.



Fonte: O autor (2017)

Figura 35 - Código Java – Método de Reposicionamento dos Raios nas Lentes e Espelhos

```

1 public void moveimagem() {
2     final float rI = imagem.getDistance();
3     final float rO = objeto.getDistance();
4     final float hO = objeto.getHeight();
5
6     switch (state) {
7     case CONVERGENTE_ESPELHO:
8         imagem.x = lente.x - (1.0f / (1.0f / rF - 1.0f / rO));
9         if (objeto.y < lente.y) {
10             imagem.y = lente.y - ((rI / rO) * hO);
11         } else {
12             imagem.y = lente.y + ((rI / rO) * hO);
13         }
14         break;
15     case DIVERGENTE_ESPELHO:
16         imagem.x = (lente.x - (1.0f / ((1.0f / (-rF)) - (1.0f / rO)))));
17         if (objeto.y < lente.y) {
18             imagem.y = lente.y - ((rI / rO) * hO);
19         } else {
20             imagem.y = lente.y + ((rI / rO) * hO);
21         }
22         break;
23     case CONVERGENTE_lente:
24         if (objeto.x < lente.x) {
25             imagem.x = lente.x + (1.0f / (1.0f / rF - 1.0f / rO));
26         } else {
27             imagem.x = lente.x + (1.0f / (1.0f / -rF - 1.0f / rO));
28         }
29         if (objeto.y < lente.y) {
30             imagem.y = lente.y + ((rI / rO) * hO);
31         } else {
32             imagem.y = lente.y - ((rI / rO) * hO);
33         }
34         break;
35     case DIVERGENTE_lente:
36         if (objeto.x < lente.x) {
37             imagem.x = lente.x + (1.0f / (1.0f / -rF - 1.0f / rO));
38         } else {
39             imagem.x = lente.x + (1.0f / (1.0f / rF - 1.0f / rO));
40         }
41         if (objeto.y < lente.y) {
42             imagem.y = lente.y + ((rI / rO) * hO);
43         } else {
44             imagem.y = lente.y - ((rI / rO) * hO);
45         }
46     }
47 }

```

Fonte: O autor (2017)

Figura 36 - Código Java – Método que Desenha os Raios nos Espelhos e Lentes

```

1 public void drawRaios(Canvas g) {
2     int s = imagem.x < lente.x ? 1 : -1;
3
4     drawRay(g, objeto.x, objeto.y, lente.x, objeto.y, true);
5     drawRay(g, lente.x, objeto.y, imagem.x, imagem.y);
6     drawRay(g, objeto.x, objeto.y, lente.x, imagem.y, true);
7     drawRay(g, imagem.x, imagem.y, lente.x, imagem.y);
8     drawRay(g, objeto.x, objeto.y, lente.x, lente.y, true);
9     drawRay(g, lente.x, lente.y, imagem.x, imagem.y);
10
11     if (state == Reflection.LENTE_DIVERGENTE
12         || state == Reflection.ESPELHO_DIVERGENTE) {
13         rextend(imagem.x, imagem.y, lente.x, objeto.y, g);
14         rextend(imagem.x, imagem.y, lente.x, imagem.y, g);
15         rextend(imagem.x, imagem.y, lente.x, lente.y, g);
16     } else if (state == Reflection.LENTE_CONVERGENTE
17         && Math.abs(lente.x - objeto.x) < rF) {
18         rextend(imagem.x, imagem.y, lente.x, lente.y, g);
19     } else if (state == Reflection.ESPELHO_CONVERGENTE
20         && Math.abs(lente.x - objeto.x) < rF) {
21         rextend(imagem.x, imagem.y, lente.x, objeto.y, g);
22         rextend(imagem.x, imagem.y, lente.x, imagem.y, g);
23         rextend(imagem.x, imagem.y, lente.x, lente.y, g);
24     }
25
26     // Desenhe a continuação do raio que cruza o ponto focal.
27     Paint p = new Paint();
28     p.setColor(Color.LTGRAY);
29
30     switch (state) {
31     case LENTE_DIVERGENTE:
32         g.drawLine(lente.x + s * rF, lente.y, lente.x, imagem.y, p);
33         break;
34     case LENTE_CONVERGENTE:
35         if (Math.abs(lente.x - objeto.x) < rF) {
36             g.drawLine(lente.x - s * rF, lente.y, objeto.x, objeto.y, p);
37             drawRay(g, lente.x, objeto.y, lente.x + s * rF, lente.y, true);
38             rextend(lente.x, objeto.y, lente.x + s * rF, lente.y, g);
39             rextend(imagem.x, imagem.y, lente.x, imagem.y, g);
40             rextend(lente.x, objeto.y, rF, 0, g);
41         }
42         break;
43     case DIVERGENTE_MIRROR:
44         g.drawLine(lente.x - s * rF, lente.y, lente.x, imagem.y, p);
45     }
46     default:
47     }
48 }

```

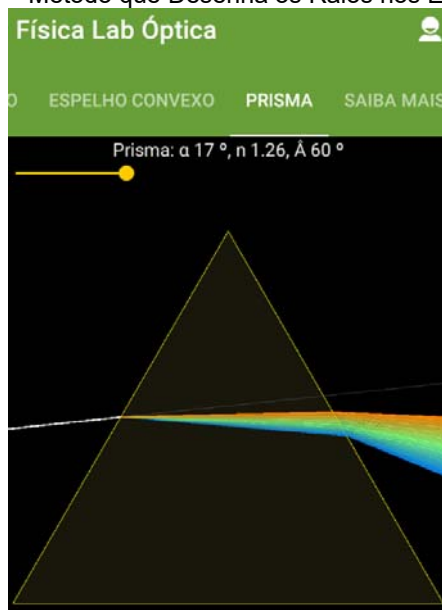
Fonte: O autor (2017)

7.2.2 Tela do Prisma e Comportamento

A tela “PRISMA” do aplicativo possui um “*SeekBar*”, componente deslizante que altera o índice de refração da luz, que é um fenômeno relacionado com sua mudança de velocidade quando passa de um meio para outro, devido suas densidades ópticas. Esta grandeza é adimensional, isto é, não possui unidade de medida. Quando o usuário altera o índice de refração o efeito prismático sofre alterações significativas, que são facilmente visualizadas no aplicativo. O ângulo de incidência da luz é alterado quando a luz branca é arrastada para cima ou para baixo, mostrado na Figura 37.

Parte do código escrito em Java que desenha os raios do prisma é apresentado na Figura 38.

Figura 37 - Código Java – Método que Desenha os Raios nos Espelhos e Lentes



Fonte: O autor (2017)

Figura 38 - Código Java – Método que Desenha os Raios do Prisma

```

1 - private void desenhaRAIOS(Canvas g, int width, double a1, boolean continuacao) {
2     double a2 = a2(a1);
3     double t1 = Math.PI / 2 - beta1(a1);
4     double t2 = Math.PI / 2 - (aa - beta1(a1));
5     double ab = distance(E, A);
6     double ad = Math.sin(t1) * ab / Math.sin(t2);
7
8     O.x = A.x + (int) (ad * Math.sin(aa / 2));
9     O.y = A.y + (int) (ad * Math.cos(aa / 2));
10
11     double ro = (a2 - aa / 2);
12     if (Double.isNaN(ro))
13         return;
14
15     double r = 2 * (width - O.x);
16     double dh = (r * Math.tan(ro));
17     ll.x = (int) (O.x + r);
18     ll.y = (int) (O.y + dh);
19
20 - if (O.y < 82.y && !Double.isNaN(dh)) {
21 -     if (ltop.x > 0) {
22         Path path = new Path();
23         path.moveTo(O.x, O.y);
24         path.lineTo(E.x, E.y);
25         path.lineTo(ltop.x, ltop.y);
26         path.close();
27         g.drawPath(path, bold);
28         path.reset();
29         path.moveTo(O.x, O.y);
30         path.lineTo(ltop.x, ltop.y);
31         path.lineTo(l2.x, l2.y);
32         path.lineTo(ll.x, ll.y);
33         path.close();
34         g.drawPath(path, bold);
35 -     } else {
36         line(g, E, O, bold);
37         g.drawLine(O.x, O.y, ll.x, ll.y, bold);
38     }
39 }
40 ltop.x = O.x;
41 ltop.y = O.y;
42 l2.x = ll.x;
43 l2.y = ll.y;
44 }

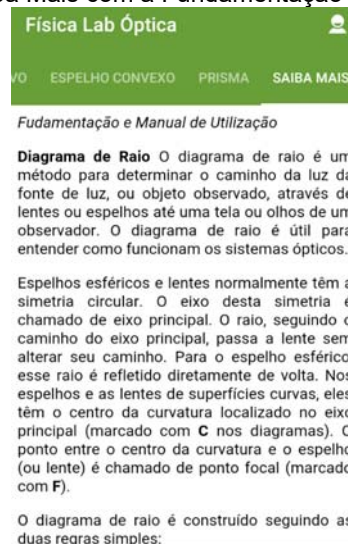
```

Fonte: O autor (2017)

7.2.3 Tela com Conteúdo Teórico.

A tela “SAIBA MAIS” mostrada na Figura 39, foi disponibilizada para auxiliar tanto o aluno quanto o professor na utilização do aplicativo. Esta apresenta a teoria conceitual sobre as lentes e espelhos esféricos utilizados no aplicativo, e seus processos de formação de imagens, além de um breve manual de utilização para melhor aproveitamento da ferramenta como um recurso didático.

Figura 39 – Tela Saiba Mais com a Fundamentação e Manual de Utilização

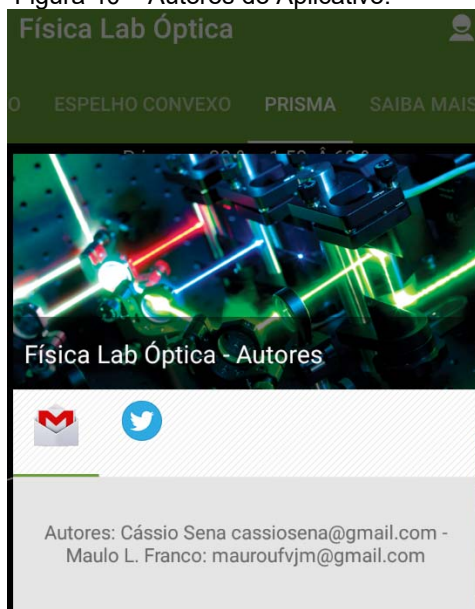


Fonte: O autor (2017)

7.2.4 Tela de Autores.

Para acessar os dados de contatos dos idealizadores e criadores do aplicativo, basta tocar no ícone presente na parte superior do layout (Figura 40).

Figura 40 – Autores do Aplicativo.



Fonte: O autor (2017)

CAPÍTULO 8

CONCLUSÃO

O desenvolvimento do presente estudo possibilitou expor os aspectos da criação de aplicativos educacionais para smartphones, que buscam enriquecer o ensino e aprendizagem da Física e Matemática com uma perspectiva diferenciada, que possibilite ao professor buscar uma aproximação do aluno com o conteúdo ensinado. Além disso, permitiu a utilização das metodologias da engenharia de software para qualidade do produto final e avaliar os recursos técnicos disponibilizados pela plataforma Android que contribuiu para a confecção dos softwares de uma forma a torná-lo acessível a todos.

Seguindo os modelos de Engenharia de Software e os recursos da plataforma de desenvolvimento Android com a linguagem Java foi possível construir aplicativos com funcionalidades didáticas expondo de forma clara o conteúdo proposto e ainda várias sugestões para o uso educacional e sendo compatíveis com grande parte dos dispositivos com Android presentes no mercado. Cumprindo os padrões adotados nos projetos preliminares de softwares foi possível desenvolver os aplicativos de uma forma coerente e produtiva, alcançando a experiência do usuário e a facilidade em realizar as ações pretendidas.

A exposição dos fragmentos de códigos, explica de forma técnica o funcionamento interno dos softwares representando suas ações quando o usuário interage com os programas. As principais linhas da linguagem desenvolvida não resumem o produto como um todo, mas apenas as finalidades específicas de cada aplicativo.

A possibilidade de processar dados experimentais inseridos manualmente ou adquiridos através dos sensores, coloca os aplicativos a um nível parecido com um mini laboratório móvel de Física, uma vez que a disposição das telas e a portabilidade dos dispositivos permitem o acesso rápido aos recursos educacionais disponibilizados.

Devido a acessibilidade ao *Google Play*, loja oficial de aplicativos para o sistema operacional Android, foi possível disponibilizar os aplicativos para serem baixados e instalados de forma gratuita, alcançando assim a proposta inicial deste projeto de pesquisa.

Deve ser ressaltado que dos aplicativos idealizados e desenvolvidos, apenas o Física Lab Resistores foi testado em sala de aula, mostrando um engajamento dos alunos nas práticas realizadas com resultados positivos. Se bem analisada, a viabilidade de o aluno utilizar o *smartphone* no estudo da Física e Matemática abre muitas possibilidades de atuação, inclusive extraclasse, associando situações do dia a dia aos conteúdos abordados em sala de aula.

REFERÊNCIAS

ABLESON W. Frank; SEM, Robi; KING, Chris; ORTIZ, C. Enrique. *Android em Ação*. 3 ed. Rio de Janeiro: Elsevier - Campus, 2012

AGUIAR,E.C.;VIEIRA,P.V. Mecânica com o acelerômetro de smartphones e tablets. *Física na Escola*, v.14, n.1, 2016. Disponível em :< www.sbfisica.org.br/fne/Vol14/Num1/fne-14-1-a03.pdf>

AZEVEDO,D.P.;CAMPOS,R. Definição de requisitos de software baseada numa arquitetura de modelagem de negócios. *Prod.* [online]. 2008, vol.18, n.1, pp.26-46. ISSN 0103-6513. Disponível em: <<http://dx.doi.org/10.1590/S0103-65132008000100003>>

BENITTI, F. B. V., Seara, E. F. R., & Schlindwein, L. M. (2005). Processo de Desenvolvimento de Software Educacional: proposta e experimentação CINTED-UFRGS. *Novas Tecnologias na Educação*.

CAVALCANTE, Kleber G. "Georg Simon Ohm"; *Brasil Escola*. Disponível em <<https://brasilecola.uol.com.br/fisica/georg-simon-ohm.htm>>. Acesso em 10 de janeiro de 2018.

COBAN – Código de Barras Nacional. Disponível em: < <http://codigodebarrasnacional.com.br/blog/o-que-e-ean-13-do-produto>>. Acesso em: 13 Jan. 2018.

DEITEL, H. M.; DEITEL, P. J. *Java Como Programar*: 10 ed. São Paulo: Bookman, 2016. Cap.1. p. 2-5.

DEITEL, Paul J.; DEITEL, Harvey M.; DEITEL, Abbey; MORGANO, Michael. *Android para Programadores: Uma abordagem baseada em aplicativos*. 2 ed. Porto Alegre: Editora Bookman, 2015.

GOOGLE - Termos de Serviço do *Google Play* Disponível em: <https://play.google.com/intl/pt-BR_br/about/play-terms.html> acessado Em: 03 Março. 2017.

IDC - *Smartphone OS Market Share*, 2017 Q1. Disponível em:<<https://www.idc.com/pro mo/smartphone-market-share/os>>. Acesso em: 15 Agosto. 2017

Intents e Filtros de *Intents*, *Google Developers* disponível em : <<https://developer.android.com/guide/components/intents-filters.html?hl=pt-br>> Acesso em: 11 de março 2017.

Optical instruments, *The Open University* disponível em : <http://www.met.reading.ac.uk/pplato2/h-flap/phys6_4.html> Acesso em: 13 janeiro. 2018.

PRESSMAN, Roger S; MAXIM, Bruce R.. *Engenharia de Software: uma abordagem profissional*. 8. ed. São Paulo: Pearson Makron Books, 2016.

SCHILDT, H. Java para iniciantes. 6ª ed. São Paulo:Ediora Bookman, 2015

Sensors Overview, *Google Developers* disponível em : <https://developer.android.com/guide/topics/sensors/sensors_overview.html> Acesso em: 12 dezembro. 2017.

SOMERA, Guilherme. Treinamento Profissional em Java: Aprenda a programar nesta poderosa linguagem. São Paulo: Digerati Books, 2006. Cap. 1. p. 5-8.

ANEXOS

ANEXO I – Artigo publicado na revista Vozes dos Vales 2017⁸.



Utilização da Plataforma Android para Desenvolvimento do Aplicativo Mate Código de Barras para o Ensino de Física e Matemática

Prof. Dr. Mauro Lúcio Franco

Doutor em Química pela Universidade Federal de Minas Gerais – UFMG
Docente da Faculdade de Ciências Sociais Aplicadas e Exatas – FACSAC da
Universidade Federal dos Vales do Jequitinhonha e Mucuri UFVJM - MG/Brasil
<http://lattes.cnpq.br/5529582752535382>
E-mail: ml.franco@ufvjm.edu.br

Prof. Dr. Wederson Marcos Alves

Doutor em Engenharia Agrícola pela Universidade Federal de Viçosa – UFV
Docente da Faculdade de Ciências Sociais Aplicadas e Exatas – FACSAC, da
Universidade Federal dos Vales do Jequitinhonha e Mucuri – UFVJM - MG/Brasil
<http://lattes.cnpq.br/8599448364867450>
E-mail: wederson.alves@ufvjm.edu.br

Cássio Gonçalves Sena

Mestrando em Tecnologia Ambiente e Sociedade -TAS - UFVJM/MG
Docente da Faculdade Presidente Antônio Carlos de Teófilo Otoni/MG - Brasil
<http://lattes.cnpq.br/3314411776560208>
E-mail: cassiosena@gmail.com

Resumo: O presente trabalho foi elaborado tendo em vista o desenvolvimento de um aplicativo para dispositivos móveis com Android para calcular o dígito verificador dos códigos de barras buscando favorecer o ensino de física e matemática. Para um produto final que atendesse os requisitos foi seguido as metodologias previstas na engenharia de software. Foi utilizado para criação a interface de desenvolvimento (IDE) Android Studio junto com Software Development Kit (SDK) e a linguagem java,

Revista Vozes dos Vales – UFVJM – MG – Brasil – Nº 11 – Ano VI – 05/2017
Reg.: 120.2.095–2011 – UFVJM – QUALIS/CAPES – LATINDEX – ISSN: 2238-6424 – www.ufvjm.edu.br/vozes


- 2 -

pois esta plataforma favorece a criação de aplicativos de forma produtiva, padronizada e dinâmica. Com a finalização do aplicativo pode-se observar uma boa disponibilização do conteúdo didático abordado como também uma interação com ambiente externo.

Palavras-chave: Física, Matemática, Android, Java, Aplicativo Educacional, Códigos de Barras

⁸ <http://site.ufvjm.edu.br/revistamultidisciplinar/files/2017/03/Mauro0602.pdf>

ANEXO II – Documentos de registro do aplicativo Mate Código de Barras junto ao INPI




protocolo

INPI INSTITUTO NACIONAL DE PROPRIEDADE INTELECTUAL

05/05/2017 020170001620

VP25/04/2017



BR 51 2017 000452 8

PEDIDO DE REGISTRO DE PROGRAMA DE COMPUTADOR

IDENTIFICAÇÃO DO PEDIDO (Para uso do INPI)

Número do Pedido _____ Protocolo, Data e Hora _____

DADOS DO AUTOR DO PROGRAMA

Nº de Autores 3 Se mais de um, preencha a "Continuação", com todos os dados solicitados neste Quadro. Date e assine.

CPF* 924.442.026-00

Nome MAURO LÚCIO FRANCO

Nome Abreviado, pseudônimo ou sinal convencional (se houver) _____

Data de Nascimento 20/02/1972 Nacionalidade BRASILEIRO

Endereço RUA RACHID HANDERE, 961 – BAIRRO BELA VISTA

Cidade Teófilo Otoni UF MG País BRASIL

CEP 39800-961 Telefone 3388287506 FAX _____

E-mail ml.franco@ufvjm.edu.br

DADOS DO TITULAR DOS DIREITOS PATRIMONIAIS

Nº de Titulares 1 Se mais de um, preencha a "Continuação", com todos os dados solicitados neste Quadro. Date e assine.

CPF/CNPJ* 16888315000157

Nome/Razão Social UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

Nome abreviado, pseudônimo ou sinal convencional (se houver) UFVJM

Data de Nascimento 06/09/2005 Nacionalidade/Origem -

Endereço CAMPUS JK- RODOVIA MGT 367 - KM 583, Nº 5000 - ALTO DA JACUBA

Cidade DIAMANTINA UF MG País BRASIL

CEP 39.100-000 Telefone 3835326097 FAX -

E-mail nitec@ufvjm.edu.br

☒ **SIM**, este Titular é Pessoa Jurídica. Caso afirmativo, assinale a melhor classificação:

☐ Órgão Público ☐ Sociedade com Intuito não Econômico ☐ Microempresa ☐ Software House

☒ Instituição Pública de Ensino ou Pesquisa ☐ Instituição Privada de Ensino ou Pesquisa ☐ Outras

ENDEREÇO PARA CORRESPONDÊNCIA E CONTATO (Preencha apenas o necessário)

Toda correspondência será enviada para:

☐ Escaninho nº _____ ☐ O Procurador ou ☒ O Titular acima ou

☐ Representação INPI em: _____ ☐ O Endereço abaixo:

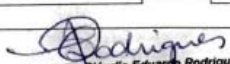
Nome _____

Endereço _____

Cidade _____ UF _____ País _____

CEP _____ Telefone _____ FAX _____

E-mail _____



Prof. Dr. Claudio Eduardo Rodrigues
Vice-Reitor/UFVJM

Modelo I (folha 1/2) E

REGISTRO DE PROGRAMA DE COMPUTADOR - CONTINUAÇÃO

Utilize este ANEXO, em quantas folhas forem necessárias, para complementar as informações dos formulários "Pedido de Registro de Programa de Computador" e "Petição - Programa de Computador".

DADOS DO AUTOR DO PROGRAMA

CPF* 045.901.706-39

Nome Cássio Gonçalves Sena

Nome Abreviado, pseudônimo ou sinal convencional (se houver)

Data de Nascimento 12/04/1981 Nacionalidade BRASILEIRO

Endereço Adalberto Hollerbach, 220 AP 302

Cidade Teófilo Otoni UF MG País BRASIL

CEP 39801-258 Telefone (33)988061118 FAX

E-mail CASSIOSENA@GMAIL.COM

DADOS DO AUTOR DO PROGRAMA

CPF* 053.864.166-50

Nome Jordana Henrique Pereira

Nome Abreviado, pseudônimo ou sinal convencional (se houver)

Data de Nascimento 06/11/1981 Nacionalidade BRASILEIRO

Endereço Adalberto Hollerbach, 220 AP 302

Cidade Teófilo Otoni UF MG País BRASIL

CEP 39801-258 Telefone (33)988210050 FAX

E-mail JORDANASEMPRE@GMAIL.COM

DADOS DO TITULAR DOS DIREITOS PATRIMONIAIS

CPF/CNPJ*

Nome/Razão Social

Nome abreviado, pseudônimo ou sinal convencional (se houver)

Data de Nascimento Nacionalidade/Origem -

Endereço

Cidade UF País

CEP Telefone FAX -

E-mail

Prof. Dr. Claudio Eduardo Rodrigues
Vice-Reitor/PRM

ANEXO III – Resultados da utilização do Aplicativo Física Lab Resistores Realizado pela Professora de Física da escola pública.

Gráfico 1 - Compreensão dos conteúdos trabalhados após o uso do Aplicativo.

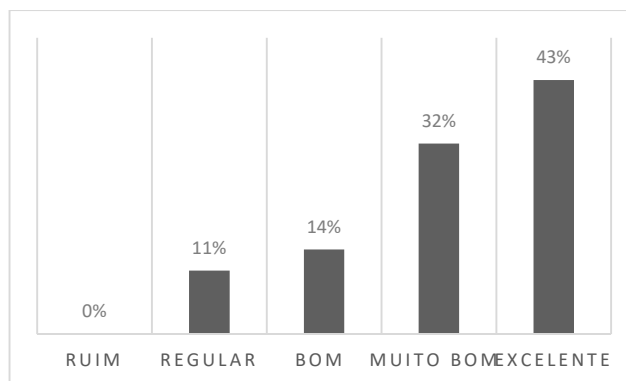


Gráfico 2 - Interesse na utilização de aplicativos para o ensino da Física.

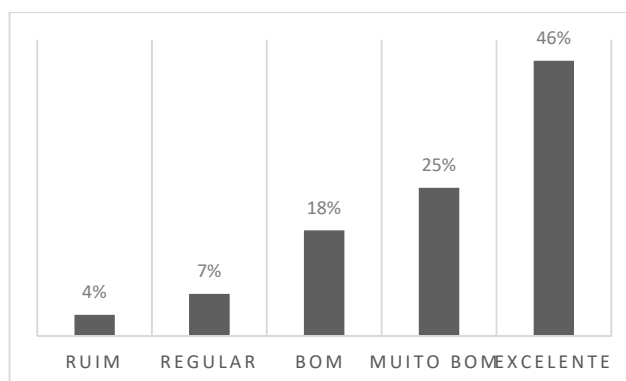


Gráfico 3 - Avaliação da abordagem de conteúdos da Física e suas aplicações tecnológicas.

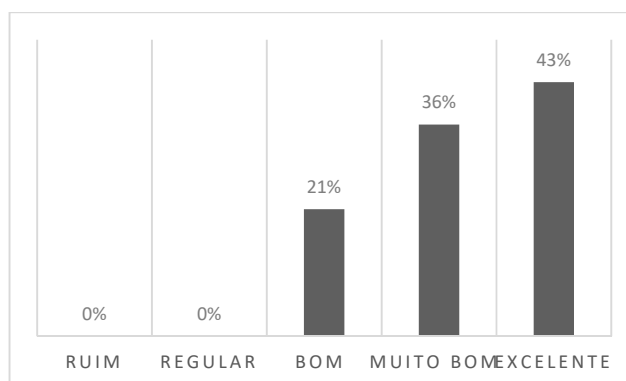


Gráfico 4 - O tempo das atividades quando realizadas com o Aplicativo.

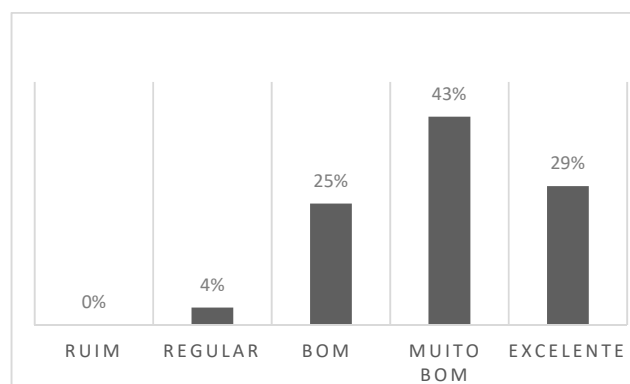


Gráfico 5 - Avaliação da clareza na linguagem utilizada no Aplicativo.

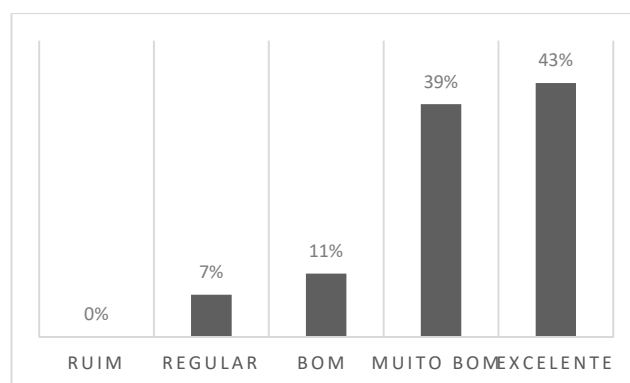


Gráfico 6 - Avaliação do interesse e atenção pelo contexto abordado na oficina.

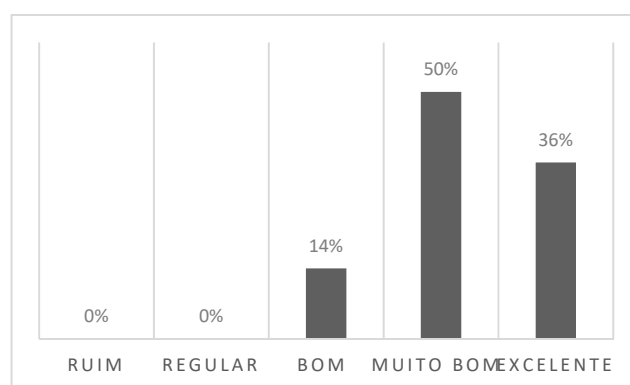


Gráfico 7 - Avaliação sobre o grau de aprendizagem do aluno com relação às atividades envolvendo a utilização do Aplicativo.

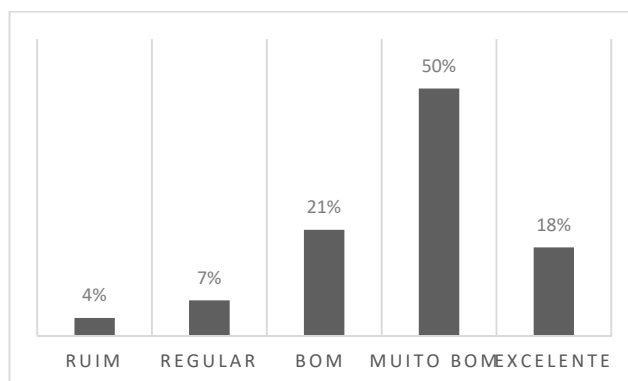


Gráfico 8 - Avaliação da oficina de uma forma geral

