UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA ELÉTRICA
DEPARTAMENTO ACADÊMICO DE ELÉTRICA
ENGENHARIA ELÉTRICA

MATHEUS MAFFATO CANHADAS GENVIGIR

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DIDÁTICO PARA TREINAMENTO DE PROFISSIONAIS EM FORMAÇÃO, NA DETECÇÃO DE CARACTERISTICAS SUBJETIVAS DA VOZ

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2019

MATHEUS MAFFATO CANHADAS GENVIGIR

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL DIDÁTICO PARA TREINAMENTO DE PROFISSIONAIS EM FORMAÇÃO, NA DETECÇÃO DE CARACTERISTICAS SUBJETIVAS DA VOZ

Trabalho de Conclusão de Curso apresentado ao curso Superior de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito para a obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dra. Maria Eugenia Dajer.
Coorientador: Prof. Dr. Danilo Hernane Spatti.

CORNÉLIO PROCÓPIO
2019

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia Elétrica

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

FOLHA DE APROVAÇÃO

Matheus Maffato Canhadas Genvigir

Desenvolvimento de um aplicativo móvel para treinamento em detecção de características subjetivas da voz

Trabalho de conclusão de curso apresentado às 10:30hs do dia 02/05/2019 como requisito parcial para a obtenção do título de Engenheiro Eletricista no programa de Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

_____
Prof(a). Dr(a). María Eugenia Dajer - Presidente (Orientador)

_____
Prof(a). Dr(a). Danilo Hernane Spatti - (Coorientador)

_____
Prof(a). Dr(a). Wagner Endo - (Membro)

_____
Engenheiro(a) Vinicius Sutério - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

RESUMO

GENVIGIR, M.M.C. **Desenvolvimento de um aplicativo móvel didático para treinamento de profissionais em formação, na detecção de características subjetivas da voz.** 2019. 62 f. Trabalho de Conclusão de Curso – Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

Este trabalho busca desenvolver um aplicativo para dispositivos móveis, baseado na plataforma *Android,* para auxiliar no treinamento de futuros profissionais da área da fonoaudiologia e otorrinolaringologia. Para isso, propõe-se o uso de um banco de trinta vozes previamente classificadas em predominância de três tipos de características subjetivas: rugosidade, soprosidade e tensão. Também fará uso do *software Android Studio* utilizando as linguagens de programação Java e de marcação XML para desenvolver tal aplicativo. Este trabalho resultou em uma ferramenta que disponibiliza um reprodutor de amostras vocais e quatro métodos comparadores para as características auditivas - perceptivas e grau do desvio vocal. Esta ferramenta didática permite ao usuário ouvir, comparar e classificar amostras de acordo com sua avaliação perceptual. Por fim, o aplicativo se mostrou viável ao realizar sua função evidenciando ser uma excelente ferramenta para treinamento e possíveis trabalhos futuros.

Palavras-chave: ferramenta didática. treinamento. características vocais.

ABSTRACT

GENVIGIR, M.M.C. **Development of a mobile educational application for training professionals, in the detection of subjective characteristics of the voice.** 2019. 62 f. Trabalho de Conclusão de Curso – Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

This work seeks to develop an application for mobile devices, based on the Android platform, to assist in the training of future professionals in the fields of speech therapy and otorhinolaryngology. For this purpose we propose the use of a bank with thirty voices previously classified in predominance of three types of subjective characteristics: roughness, breathiness and tension. It will also make use of the Android Studio software using Java programming language and XML markup language to develop such an application. This work resulted in a tool that provides a reproducer of vocal samples and four comparison methods for auditory - perceptual characteristics and degree of vocal deviation. This didactic tool allows the user to listen, compare and classify samples according to their perceptual evaluation. Finally, the application proved to be feasible when performing its function proving to be an excellent tool for training and possible future work.

Keywords: education tool. training. vocal characteristics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS, ABREVIATURAS E ACRÔNIMOS

API	*Application Programming Interface*
IDE	*Integrated Development Environment*
JDK	*Java Development Kit*
JRE	*Java Runtime Environment*
OHA	*Open Handset Alliance*
RNA	*Rede Neural Artificial*
SDK	*Software Development Kit*
SGML	*Standard Generalized Markup Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

É inegável o constante aumento do uso de dispositivos móveis portáteis no dia a dia das pessoas. *Smarthphones*, *Tablets e notebooks* redefiniram o mercado mundial de tecnologia móvel, sendo mais fácil sua utilização comparada a computadores pessoais que tem a necessidade de um local fixo (MAHMUD; ABDULLAH, 2014). Segundo dados da mais recente Pesquisa Anual de Administração e Uso de Tecnologia da Informação nas Empresas, o Brasil tinha uma densidade de 1,5 *smartphones* por habitante no final de maio de 2018. Além disso, somados, *smartphones*, *tablets* e *notebooks* chegam aos 306 milhões de dispositivos nesta mesma data (MEIRELLES, 2018).

Devido ao grande aumento da quantidade de usuários destes dispositivos e a agilidade ao se conectar com a internet móvel, desde 2007, os aplicativos para *smartphones* cresceram rapidamente em número e funcionalidade, oferecendo aos usuários uma gama de ferramentas para suprir suas necessidades. No início, essas aplicações, em sua maioria, eram voltadas ao setor de entretenimento. Atualmente este cenário abrange um leque de opções maiores, que podem ser alocadas em quatro grandes áreas: serviços, informações, comunicação e o já então difundido entretenimento digital (MUCCINI et al., 2012; VIDALE, 2016).

Estes aplicativos ajudam no controle de dietas, rotinas para a prática de atividades físicas, previsão do tempo, navegação de mapas (GPS), seguro veicular, guia de compras, consulta de produtos, redes sociais, setor financeiro e aplicativos voltados para a área da saúde (VIDALE, 2016). De acordo com Amorim (2011) acompanhar essa nova tendência à mobilidade, nos leva a pensar na diversidade de aplicações que ainda podem ser inseridas no meio profissional em qualquer setor. Na área da saúde, mais especificamente na fonoaudiologia, a busca pelo aperfeiçoamento de diagnósticos clínicos traz novas tendências em pesquisas e ferramentas para análise acústica e perceptivo-auditiva da voz (ANDRADE, 2003).

A voz é uma ferramenta de comunicação de valor inestimável, é a principal forma de interação entre o locutor e seu público. Segundo Behlau (2001), a voz se destaca pela sua eficiência em transmitir ideias e emoções dentro do contexto social que o indivíduo está inserido.

Sendo produzida de forma complexa, a voz é suscetível a alterações devido a variações emocionais, assim como, pelo desarranjo orgânico ou funcional

do aparelho fonador do indivíduo (ROSA, 1998). De acordo com Köhle (2004) o uso inadequado da voz em conjunto com distúrbios vocais tem recebido grande atenção pelos profissionais da fonoaudiologia.

Atualmente, o padrão – ouro é iniciar uma avaliação vocal pela avaliação perceptivo-auditiva, uma técnica considerada subjetiva e que dependente de vasta experiência adquirida pelo avaliador em seus anos de graduação e especialização (VALENTIM; CÔRTES; GAMA, 2010). A formação de novos profissionais fonoaudiólogos assumiu maior importância nos últimos anos, dessa forma, é natural o surgimento de uma demanda para novas referências literárias e ferramentas de ensino para graduação, a fim de complementar as futuras competências de tais profissionais (SILVA; SAMPAIO; BIANCHINI, 2010).

Considerando essa linha de desenvolvimento, este trabalho tem como objetivo criar um aplicativo móvel para o sistema operacional *Android*, para auxiliar no treinamento e aperfeiçoamento das capacidades perceptuais do profissional fonoaudiólogo e otorrinolaringologista na identificação das características vocais de rugosidade, soprosidade e tensão. Esta ferramenta fará uso de um banco de vozes, onde, cada amostra vocal foi previamente classificada por fonoaudiólogo especialista em voz de acordo com a predominância das características de rugosidade, soprosidade e tensão.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é desenvolver um aplicativo de aparelho móvel para plataforma *Android*, para auxiliar no treinamento de identificação de parâmetros subjetivos da voz como rugosidade, soprosidade e tensão de profissionais da área de fonoaudiologia e otorrinolaringologia.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos fundamentais utilizados para executar o presente projeto. São eles: dispositivos móveis, ambiente de desenvolvimento *Android Studio*, linguagem de programação Java e linguagem de marcação XML, sistema de produção da voz e os métodos de avaliação invasiva e não invasiva.

2.1 TECNOLOGIAS E DISPOSITIVOS MÓVEIS

A tecnologia móvel, considerada a última tecnologia do século 21 é constituída por dispositivos portáteis, que permitem sua utilização com o usuário em movimento, tendo como exemplo *notebooks*, telefones celulares, *smathphones* e outros (PINTO, 2016). Esses dispositivos se destacam pela sua capacidade de conexão com uma variedade de métodos de transmissão de dados como Wi-Fi, Terceira e Quarta Gerações de Padrões e Tecnologias de Telefonia Móvel (3G e 4G), Bluetooth entre outros (RODRIGUES, 2009).

Devido a esta alta mobilidade os dispositivos móveis tornaram-se uma ferramenta indispensável para acessar informações, receber e enviar documentos, fotos, músicas, mensagens de texto; efetuar ligações telefônicas, utilizar câmera fotográfica, dentre diversas outras opções, em qualquer hora e lugar (MAHMUD; ABDULLAH, 2014; PINTO, 2016). Com o surgimento de novas formas de comunicação sem fio, o espaço social passou por uma etapa de modificações. O desenvolvimento do acesso à internet sem fio (*wireless*) programou uma nova dinâmica dentro destas práticas sociais e introduziu novas ferramentas no cotidiano dos usuários de dispositivos móveis (MOURA, 2009).

2.2 APLICAÇÕES MÓVEIS

A popularização dos *Smarthphones* no passar dos anos tem sido considerada por muitos a revolução tecnológica de maior impacto após a inserção da própria internet e a difusão das populares redes sociais (TIBES; DIAS; MASCARENHAS, 2014). Essa nova tendência à mobilidade atraiu atenção dos desenvolvedores de softwares para o desafio de criar sistemas operacionais e aplicativos capazes de atender as necessidades dos usuários de aparelhos móveis (DELIA et al., 2015).

Ainda em concordância com Delia et al. (2015) os dispositivos móveis possuem diversos sistemas operacionais e, como pode ser observado na Tabela 1, os dois sistemas operacionais (SOs) mais utilizados no final do ano de 2017 até Setembro de 2018 são o *Android* (Desenvolvido pela Google) e iOS (Desenvolvido pela Apple).

O *Android* é a resposta do grupo *Open Handset Alliance* (OHA) junto à empresa Google para ocupar esse espaço de mercado. Esta plataforma de desenvolvimento para aplicativos móveis foi criada baseada no sistema operacional Linux e foi concebida com o objetivo de padronizar uma plataforma de código aberto e livre que atendesse a maior parcela possível do mercado de dispositivos móveis (LECHETA, 2013).

Tabela 1 – Mercado Mundial de Sistemas Operacionais de Smarthphones

Período	Android	iOS
Dezembro 2017	80,3%	19,6%
Setembro 2018	86,8%	13,2%

Fonte: IDC, Março 2019.

Aplicativos móveis são softwares que fazem a conexão homem – dispositivo, são capazes de gerenciar as ferramentas contidas em tais dispositivos como: câmera, microfone, saídas de áudio, tela, conexão com a internet, armazenamento interno, dentre outras. Esses aplicativos oferecem aos usuários uma gama de possibilidades com o objetivo de atender cada tipo de necessidade,

seja fazer uma chama, enviar mensagens de texto, acessar informações via rede móvel, enviar e receber documentos e mídias, gravar vídeos, etc. (SOARES, 2016).

2.3 KIT DE DESENVOLVIMENTO DE SOFTWARE (SDK)

O *Android SDK* é uma biblioteca com exemplos de funções, conjuntos de rotinas, padrões de programação, versões do sistema operacional *Android* e opções de emuladores para testes, que possibilita a compilação de um código Java para plataforma alvo. No caso dos dispositivos móveis, cada sistema operacional tem seu próprio Kit de Desenvolvimento, tendo suas diferenças vistas em suas linguagens de programação, Java para *Android, Objetive-C* e/ou *Swift* para *iOS* (BRINKHEDEN; ANDERSSON, 2015).

Para iniciar a codificação de um novo aplicativo, faz-se uso de um Kit de Desenvolvimento de Software (SDK) para *Android* chamado *Android SDK*. Este kit está disponível para download juntamente com a IDE *Android Studio*, ou em caso de se optar por utilizar outro software de desenvolvimento, também é possível fazer o download do SDK separadamente para vários sistemas operacionais (PEREIRA; SILVA, 2009).

2.4 INTERFACE DE PROGRAMAÇÃO DE APLICATIVOS - API

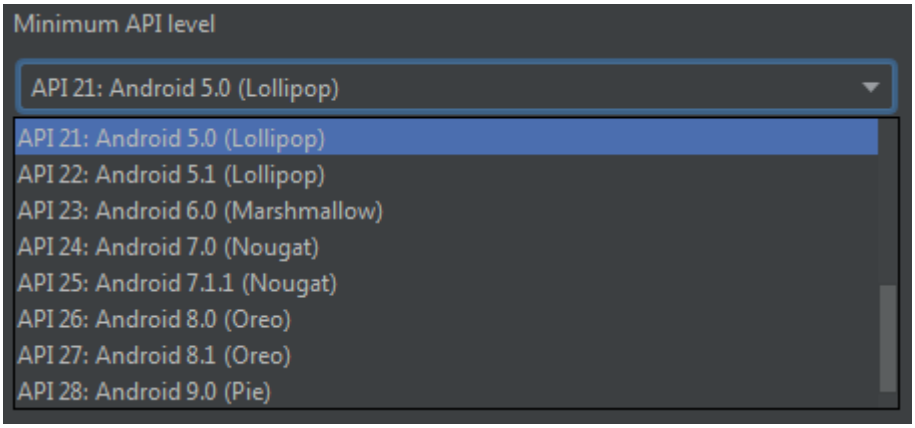
Segundo Sousa (2000) uma Interface de Programação de Aplicativos - API é composta por um conjunto de funções primitivas que integram a definição e manipulação de objetos em uma representação compatível com uma linguagem de programação orientada a objetos. Em outras palavras as APIs são um conjunto de interfaces de chamada para a realização de tarefas que retornam resultados específicos para que a aplicação possa processa-las da forma desejada. Isso significa que uma API é a ponte entre todos os códigos de diversos desenvolvedores e o sistema operacional *Android* (PINTO, 2016; SILVA, 2011).

Na Figura 1, podem-se verificar alguns exemplos de APIs contidos no *software Android Studio*. As chamadas API *level* - são responsáveis por fazer a conexão entre a versão do sistema operacional *Android* e a linguagem de

programação Java. Observa-se que cada nível de API leva a frente o nome de uma versão *Android* entre parênteses, referente ao sistema ao qual é compatível (JACKSON, 2017).

De acordo com os desenvolvedores *Android*, da Google, versões mais recentes de API *level* possuem as interfaces das versões mais antigas. Assim sendo, é aconselhado, escolher uma versão mais antiga para desenvolvimento de aplicativos, dessa forma é possível alcançar um número maior de usuários no quesito compatibilidade.

Figura 1 – Representação dos níveis de API Android Studio.



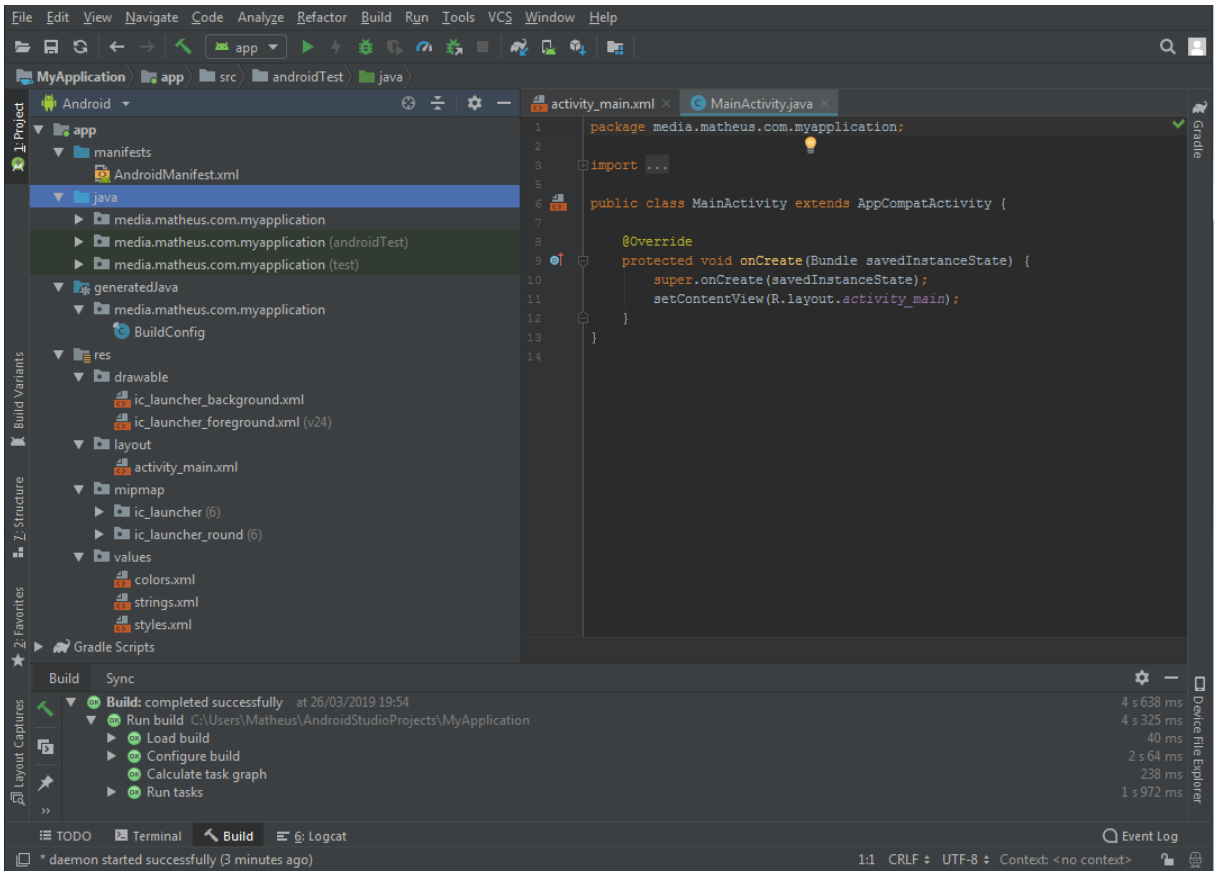
Fonte – Software Android Studio (2019)

2.5 ANDROID STUDIO

O *Android Studio* é o ambiente de desenvolvimento integrado (IDE) oficial disponibilizado pela empresa Google para o desenvolvimento de aplicativos *Android*. Como pode ser observado na Figura 2, possui um editor de código e ferramentas de desenvolvedor avançadas. Além disso, o *Android Studio* possui mais recursos para criação de aplicativos *Android* como (GOOGLE, 2019):

- Um ambiente unificado para todos os dispositivos *Android*;
- *Instant Run* para aplicar alterações a aplicativos em execução sem precisar compilar um novo APK – *Android Application Pack*;
- Ferramentas de verificação de código suspeito para detectar problemas de desempenho, usabilidade, compatibilidade com versões e outros;

Figura 2 – Representação de tela do software Android Studio.

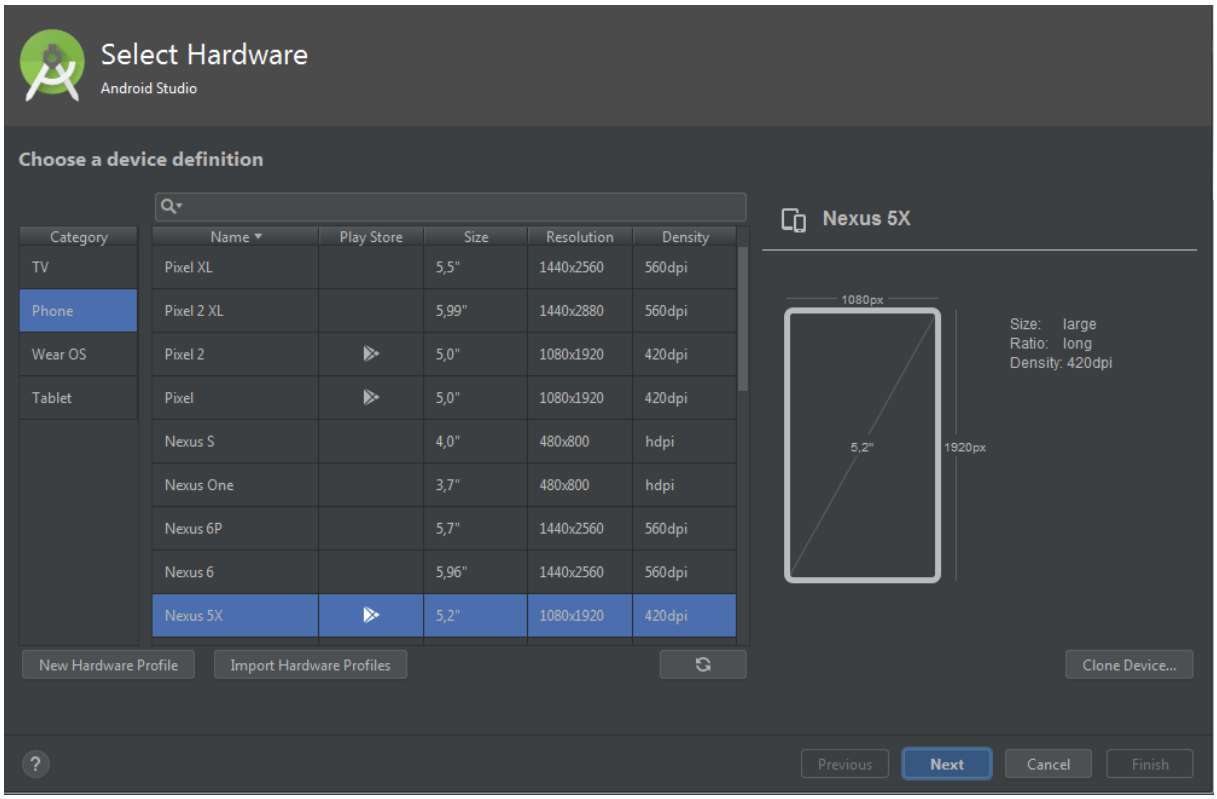


Fonte – Software Android Studio (2019)

Além das opções citadas anteriormente, o *Android Studio* possui uma série de opções para teste de aplicativos, chamados emuladores. De acordo com Pereira (2009), emuladores são dispositivos móveis virtuais que permitem instalar, executar e testar aplicações com a mesma eficiência de um hardware. Essa ferramenta permite testar os códigos assim como grande parte do sistema sem a necessidade de um aparelho móvel físico.

Como pode ser observado na Figura 3, o *Android Studio* possui uma gama de opções de emuladores, possibilitando ao desenvolvedor, escolher um modelo de aparelho móvel voltado para sua necessidade. E ainda, executar testes em vários emuladores com configurações de *hardware* diferentes a fim de buscar problemas que possam aparecer no futuro por via de compatibilidade de hardware.

Figura 3 – Tela de configuração de escolha de emuladores software Android Studio.



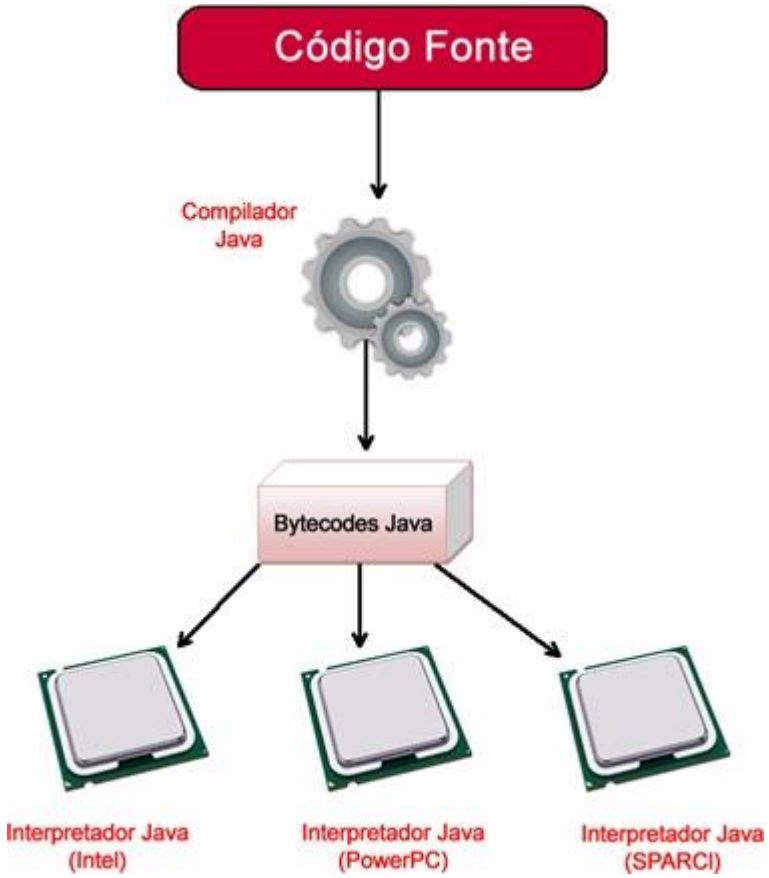
Fonte – Software Android Studio (2019)

2.6 JAVA

Java é uma linguagem de programação e plataforma (constituída por uma API e uma máquina virtual) - lançada pela primeira vez pela *Sun Microsystem* em 1995. Desde seu lançamento, a linguagem Java tem sido utilizada para criar páginas *Web,* desenvolver aplicativos e servidores corporativos, sistemas automotivos, aplicativos de dispositivos móveis, dentre outros (PALMEIRA, 2012).

Java é uma linguagem parcialmente compilada e parcialmente interpretada, o que produz a característica de adaptabilidade. Essa característica permite que um código escrito em Java seja compilado em qualquer sistema. Após a compilação o arquivo binário gerado pode ser executado em qualquer outro sistema, compreende-se melhor, observando a Figura 4. Devido a essa característica Java está presente nas mais recentes inovações, incluindo aplicativos para sistema *Android* (DEITEL, 2005).

Figura 4 – Compilação de um programa escrito em Java.



Fonte – Adaptado de PALMEIRA (2012).

A linguagem de programação Java é uma linguagem orientada a objetos, que busca modelar os objetos do mundo real para o mundo computacional utilizando-se de classes, objetos e métodos (GOLDMAN; KON; SILVA, 2006). Cada objeto possui seus atributos e comportamentos. Tais objetos são individuais e concorrentes, interagem entre si. Esses objetos podem ser uma casa, um carro, celular, uma conta bancária, uma pessoa, uma caneca, um animal, um arquivo de mídia, dentre infinitos outros (MENDES, 2009).

De maneira técnica, uma classe possui membros, dos quais seus principais tipos são os campos e os métodos. Os campos são variáveis de dados que pertencem tanto à classe quando aos objetos da classe. Os métodos são coleções de declarações que operam os estados dos campos (variáveis). Essas declarações definem o comportamento de uma classe, que podem definir o valor de uma variável, realizar operações aritméticas, chamar outros métodos e controlar a execução do programa (ARNOLD; GOSLING; HOLMES, 2005).

Na Figura 5 pode-se verificar um exemplo simples de codificação em Java onde neste exemplo destaca-se: classe (*MainActivity*) – sendo a classe principal do programa, parâmetros privados (*private ImageView* e *private Button*) – são as características da classe, seguidos pelos métodos *onCreate*, *findViewById* e *setOnClickListener* respectivamente.

Figura 5 – Exemplo de código Java.

```
public class MainActivity extends AppCompatActivity {

    private ImageView btn_empresa;
    private ImageView btn_servico;
    private ImageView btn_cliente;
    private ImageView btn_contato;
    private Button btn_passarDados;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btn_empresa = (ImageView) findViewById(R.id.empresaId);
        btn_servico = (ImageView) findViewById(R.id.servicoId);
        btn_cliente = (ImageView) findViewById(R.id.clientesId);
        btn_contato = (ImageView) findViewById(R.id.contatoId);
        btn_passarDados = (Button) findViewById(R.id.btn_passaDadosId);

        btn_passarDados.setOnClickListener((view) -> {

                Intent intent = new Intent( packageContext: MainActivity.this,SegundaActivity.class);
                intent.putExtra( name: "nome",  value: "Info");
                startActivity(intent);

        });

        btn_empresa.setOnClickListener((view) -> {
                startActivity(new Intent( packageContext: MainActivity.this, EmpresaActivity.class));
        });
```

Fonte – Autoria Própria

É interessante pensar numa classe como sendo uma fábrica repleta de impressoras que contém instruções para imprimir objetos. Pode-se imaginar também, uma classe como sendo o projeto de uma casa e, dentro desse projeto (classe) teremos as características desse projeto como: número de janelas, portas, cor da casa, medidas dos cômodos da casa - estes são os atributos da classe. Por fim teremos as ações que esta casa poderá realizar ou que poderão ser realizadas sobre ela - são os chamados métodos: abrir e fechar a porta, janela e garagem; acender e apagar as luzes, dentre outros (DAMASCENO, 2019).

2.7 XML

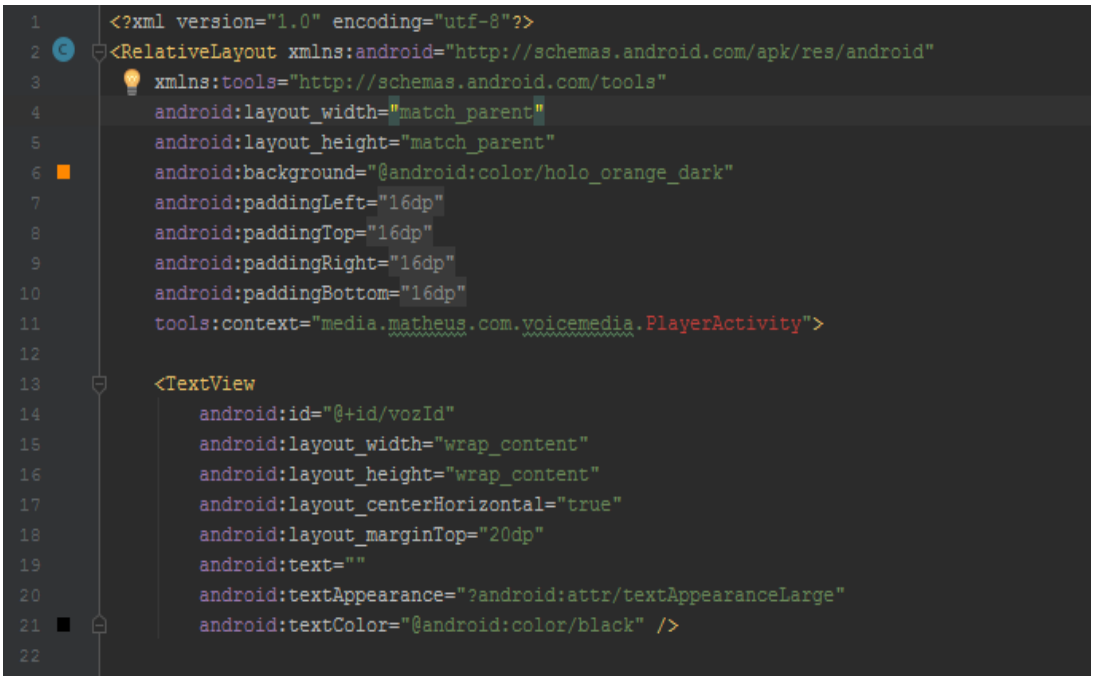
De acordo com Almeida (2002) em dezembro de 1997 o *W3 Consortium – World Wide Internet Consortium,* publicou a versão 1.0 do XML – *Extended Markup Language*, uma versão mais simples do antigo SGML – *Standard Generalized Markup Language*. A aplicação do XML tem o objetivo de obter implementações mais eficientes para manipulação de dados na Internet.

A linguagem XML é conhecida como “linguagem de marcação”, esse nome é herança do conceito de marcas textuais (RAMALHO; HENRIQUE, 2001). Anotações ou marcas em um texto tinham a função de dar instruções aos datilógrafos sobre a maneira como certa parte do texto deveria ser representada. Se determinada parte seria escrita em negrito, itálico ou sublinhado; representavam também informações estruturais como: parágrafo na linha subsequente, cabeçalho e onde uma palavra termina e outra começa (ALMEIDA, 2002).

Para processamento computacional uma linguagem de marcas deve ser utilizada para codificar um texto. Ela especifica quais marcas são permitidas, quais são exigidas, como se devem fazer distinções entre as marcas e o texto e, qual o significado da marcação. Para via de comparação, uma linguagem de marcação permite maneiras de descrever o dado para armazenamento, processamento ou transmissão por um software e difere de uma de programação como o Java, C, Basic, dentre outras, pelo fato de a programação executar ações, tratar variáveis; efetuar cálculos e tomar decisões (BAX, 2000).

Para exemplificar estes conceitos pode-se observar na Figura 6 um exemplo de codificação do *layout* de uma tela do aplicativo móvel, onde, a linguagem de marcação XML é comumente utilizada. Nessa figura também pode-se observar uma série de parâmetros definidos por tais marcações, como o tipo do objeto (*RelativeLayout*), o posicionamento (*android:padding*), altura, largura, o nome do objeto que será chamado na execução do código do aplicativo (*PlayerActivity*). Estes elementos poderão ser customizados de acordo com a necessidade do usuário ou do aplicativo.

Figura 6 – Representação de parte de layout em formato XML.



```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_orange_dark"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="16dp"
    tools:context="media.matheus.com.voicemedia.PlayerActivity">

    <TextView
        android:id="@+id/vozId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="@android:color/black" />

```

Fonte – Autoria Própria

Pode-se observar na primeira linha a declaração XML, com a versão utilizada - versão 1.0; e a codificação usada para os caracteres - utf-8; em sequência, na terceira linha se inicia o processo de criação do *layout* da página, seguindo com a declaração de parâmetros como plano de fundo, espaçamento de margens e a referência desta página quando ela for solicitada pelo aplicativo durante sua execução.

2.8 VOZ

A capacidade da fala é uma função inata, e a voz se transforma ao longo da vida de acordo com as características físicas e emocionais de casa pessoa (BEHLAU, 2001). Voz é o som produzido pela vibração das pregas vocais, modificado pelas cavidades situadas abaixo e acima dela, conhecidas como cavidades de ressonância, ou seja, voz é fonação acrescida de ressonância, sendo assim um produto resultante da união de vários elementos, que interagem de forma específica, gerando uma emissão acústica, de alta complexidade, proveniente da passagem do ar pelas pregas vocais (BEHLAU, 2001; DAJER, 2006).

Esses sistemas e sua relação com a produção do som da voz podem ser observados no Quadro 1:

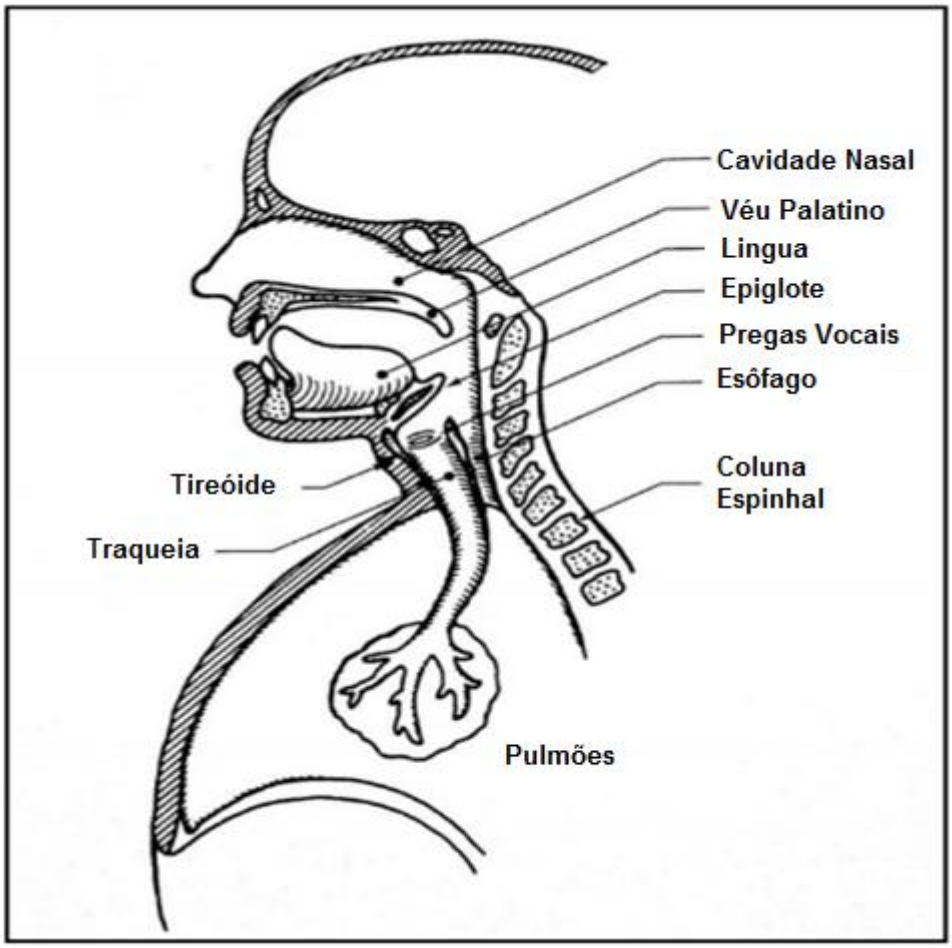
Quadro 1 – Sistemas e suas funções na produção da voz.

Sistema	Função	Estrutura
Respiratório	Fonte de Energia	Pulmões
Fonatório	Fonte de Vibração	Pregas Vocais
Ressonância	Fonte de Ressonância	Cavidade Oral e Nasal
Articulatório	Articulação	Língua, Lábios, Mandíbula Palato e Dentes
Nervoso	Coordenação	Córtex Cerebral

Fonte: Adaptado de Behlau (2001, p26)

A Figura 7 exemplifica o trato vocal, permitindo verificar a sequência dos acontecimentos para produção da voz, iniciando nos pulmões e terminando nas saídas da cavidade bucal e nasal.

Figura 7 – Representação esquemática das estruturas do trato vocal.



Fonte: Adaptado de RABINER E SCHAFER (2011, p.71)

De acordo com Behlau (2001), para que haja a produção de voz em condições normais, deve existir um trabalho conjunto e sincronizado dos sistemas respiratório, fonador, ressonante, articulatório e nervoso central. Dessa forma pode-se dizer que a voz apresenta características particulares em cada indivíduo, provenientes de sua anatomia (SANTOS, 2015).

2.8.1 Métodos de Avaliação Vocal

2.8.1.1 Métodos Invasivos

Os métodos invasivos de avaliação vocal baseiam-se na obtenção de imagens internas da laringe do paciente, permitindo a avaliação das condições anatômicas e fisiológicas das pregas vocais de forma visual. Dentre os exames

estão a Laringoscopia Direta e Laringoscopia Indireta, Endoscopia Flexível e Endoscopia Rígida e Exame Estroboscópico.

Estes exames são caracterizados pela inserção de ferramentas no comprimento da laringe, seja um pequeno espelho, como no caso da Laringoscopia Indireta ou um telescópio de 90 graus como no caso da Endoscopia Rígida (GRANATO, 2005). Tais métodos causam certo desconforto aos pacientes devido a esta inserção de instrumentos pela cavidade bucal por certo período de tempo, tal desconforto causa ânsia e pode provocar o refluxo momentâneo (TEIXEIRA; FERREIRA; CARNEIRO, 2011).

2.8.1.2 Métodos Não Invasivos

Estas análises correspondem à avaliação inicial que o especialista utiliza no diagnóstico clínico. A seguir, serão abordados dois tipos de avaliações não invasivas, priorizando a avaliação perceptivo-auditiva por ser o método escolhido para o desenvolvimento deste trabalho.

2.8.1.2.1 Avaliação Acústica

Métodos não invasivos de avaliação são utilizados como ferramentas auxiliares no diagnóstico realizado por profissionais da voz, além disso, permite o avaliador captar alterações vocais precoces, sendo um ótimo recurso para prevenção da saúde vocal. A análise acústica, por meio de processos computacionais, permite extrair medidas quantitativas das características do sinal vocal (MENDES; FERREIRA; CASTRO, 2012). Estas características são parâmetros acústicos como: periodicidade, amplitude, duração e composição espectral. Tais variações nestes parâmetros poderão ser observadas por meio de análise computacional do processo de fonação de vogais prolongadas ou amostras de fala gravadas (TEIXEIRA; FERREIRA; CARNEIRO, 2011).

Tal avaliação proporciona um complemento ao diagnóstico do processo de geração do sinal vocal, pois estes parâmetros relacionam-se com determinadas patologias da voz (DAJER, 2006; FERMINO, 2016).

2.8.1.2.2 Avaliação Perceptivo - Auditiva

Este método de avaliação se baseia em um julgamento feito por um especialista com base em uma série de comparações internas deste profissional, de uma amostra de som vocal de um indivíduo. Os resultados destas avaliações estão sujeitos a variações de diversos fatores e muda de profissional para profissional (ROSA, 1998).

A avaliação perceptivo-auditiva utiliza escalas e protocolos que fornecem parâmetros para identificação das características da voz. Algumas destas escalas possuem reconhecimento internacional como é o caso da GRBAS, proposta por Hirano em 1981. A escala GRBAS possui 5 parâmetros estabelecidos: G - *Grade* (Grau geral do desvio vocal), R - *Roughness* (Rugosidade), B - *Breathiness* (Soprosidade), A - *Asthenia* (Astenia) e S – *Strain* (Tensão). Estes parâmetros são classificados com uma escala de quatro graus de intensidade: G0 (ausência de disfonia); G1 (disfonia leve); G2 (disfonia moderada) e G3 (disfonia intensa) (PINHO; PONTES, 2002).

A voz possui características vibratórias subjetivas que são mais comumente observadas nestas escalas de avaliação. Esses parâmetros são: rugosidade, soprosidade e tensão e podem ser definidos de acordo com Pinho e Pontes (2002) da seguinte forma:

- **Rugosidade:** é caracterizado como sendo uma irregularidade vibratória da mucosa das pregas vocais durante a fonação. Esta situação gera ruídos imprevistos de baixa frequência. O sinal da voz irregular possui um ruído similar ao som de uma bandeira tremulando ao vento forte.

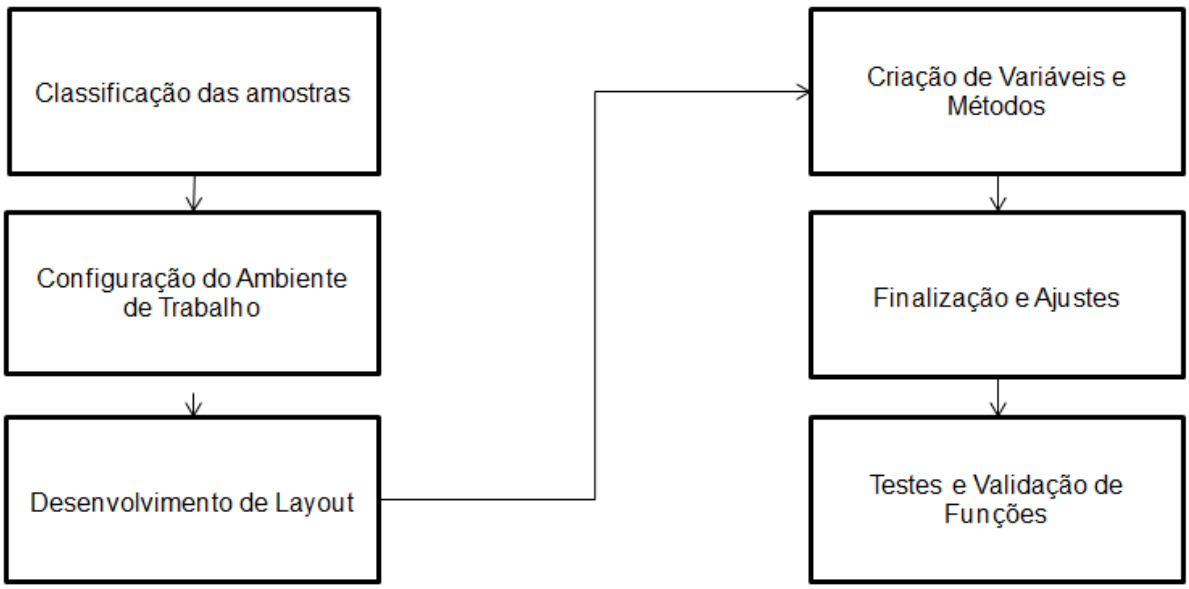
- **Soprosidade:** é perceptível quando há o escape de ar na fala e gera a impressão de ruído audível de fundo. O mau funcionamento de qualquer músculo, principalmente da musculatura intrínseca da laringe, que é responsável pelo fechamento da glote, fará com que a característica da soprosidade apareça no sinal vocal.

- **Tensão:** é caracterizada pelo aumento da rigidez da musculatura glótica. Apresenta ruídos provenientes de vibrações com alta frequência. Trás junto ao sinal vocal a impressão de voz seca e sem capacidade de projeção.

3 METODOLOGIA

Este capítulo contém informações sobre o banco de amostras vocais utilizado neste trabalho bem como a descrição detalhadas dos procedimentos realizados no desenvolvimento do mesmo, procedimentos observados na Figura 8.

Figura 8 – Fluxograma de procedimentos.



Fonte: Autoria própria

3.1 BANCO DE DADOS

O banco de dados utilizado neste trabalho foi cedido pela Fonoaudióloga Dra. Gláucya Madazio do Centro de Estudos da Voz – CEV. Esta base de dados é composta por trinta amostras vocais de trinta pacientes diferentes, que apresentaram características subjetivas predominantes de ou rugosidade, ou soprosidade ou tensão. Todas as amostras são gravações da vogal sustentada /e/.

Para gravação das amostras, os indivíduos foram alocados individualmente em ambiente silencioso e permaneceram sentados durante todo o processo. Para captura de áudio foi utilizado um microfone de cabeça da marca Plantronics, modelo Audio 20, posicionado cerca de 1 centímetro do canto da boca do paciente em posição diagonal. O microfone estava conectado a uma placa de som *Sound Blaster Audigy* 7.1, conectada a um computador.

O registro dos arquivos de áudio no formato .WAV foi feito utilizando-se o *software Sound Forge*, versão 4.5, com taxa de amostragem de 44,1 kHz e quantização de 16 bits. Cada uma das trinta amostras foi submetida à edição, utilizando-se do mesmo *software* para tal procedimento, descartando o início de cada emissão devido à instabilidade causada pela fase de ataque vocal. Os arquivos foram então avaliados e classificados de acordo com suas características subjetivas por três avaliadoras fonoaudiólogas especialistas em voz.

Os arquivos estão igualmente distribuídos entre gênero e parâmetro onde existem cinco amostras de cada característica subjetiva para cada gênero conforme Tabela 2.

Tabela 2 – Distribuição das amostras do banco de vozes de acordo com gênero e característica subjetiva.

Característica Predominante	Masculino	Feminino
Rugosidade	5	5
Soprosidade	5	5
Tensão	5	5
Amostras por Gênero	**15**	**15**

Fonte: Adaptado de Pavoni (2017, p. 31)

Para realização deste trabalho foi necessária à classificação das amostras vocais de acordo com a escala GRBAS por grau de desvio vocal: G0 (ausência de disfonia); G1 (disfonia leve); G2 (disfonia moderada) e G3 (disfonia intensa).

Esta classificação foi realizada com 20% de repetição por um fonoaudiólogo especialista em voz que apresentou consistência interna de 83,3%. Esta classificação de acordo com o grau de desvio vocal pode ser vista na Tabela 3, onde estão listados os nomes originais das amostras e o grau de desvio vocal de cada uma das amostras referente a essa escala.

Os nomes são escritos com caracteres que representam a primeira letra referente ao gênero do indivíduo ao qual a amostra foi coletada: homem (H) e mulher (M), seguido pela primeira letra referente à característica subjetiva: rugosidade (R), soprosidade (S) e tensão (T).

Tabela 3 – Distribuição das amostras do banco de vozes de acordo com nome e grau característico - G0, G1, G2 e G3.

Nome	Grau	Nome	Grau
12 - H R	G1	16 - M R	G2
7 - H R	G2	36 - M R	G2
15 - H R	G3	55 - M R	G2
17 - H R	G2	97 - M R	G3
37 - H R	G2	104 - M R	G2
20 - H S	G3	15 - M S	G2
49 - H S	G3	0 9 - M S	G2
60 - H S	G3	21 - M S	G2
79 - H S	G3	28 - M S	G2
90 - H S	G2	64 - M S	G2
9 - H T	G2	6 - M T	G2
27 - H T	G3	23 - M T	G1
30 - H T	G2	38 - M T	G2
48 - H T	G1	45 - M T	G2
93 - H T	G3	59 - M T	G2

Fonte: Autoria própria

3.2 FERRAMENTAS E METODOLOGIA - CONFIGURAÇÃO DO AMBIENTE DE TRABALHO.

Todo o desenvolvimento prático do trabalho foi feito utilizando um ambiente de desenvolvimento integrado (IDE), o *software Android Studio*, que contém as ferramentas necessárias para tal, como: bibliotecas de funções, exemplos de funções pré-programadas, classes, métodos e diversas funcionalidades para teste prático de simulação. Serão utilizados também o JDK – *Java Development Kit* e o JRE – *Java Runtime Environment.* O JDK possui as ferramentas necessárias para desenvolver e testar uma aplicação escrita na linguagem orientada a objetos e o JRE possui ferramentas que o sistema utiliza para executar tal aplicação, ele serve

como uma ponte para que o computador entenda a codificação do aplicativo na linguagem Java e execute de forma correta as funções que o desenvolvedor deseja.

3.3 PROCEDIMENTOS

Para começar com o desenvolvimento da aplicação primeiro é necessário configurar o ambiente de trabalho, e para tal verifica-se no computador onde o aplicativo será desenvolvido se já existe alguma versão do Java instalado.

3.3.1 Configuração das Variáveis do Sistema

Para executar tal tarefa no Windows 7, o caminho a seguir é percorrido: Menu Iniciar/ Executar/ digitar: “cmd” e pressionar a tecla Enter.

Seguindo este caminho, o prompt de comando será aberto. Dentro do prompt de comando digitar: “java -version”; este comando informa ao desenvolvedor se existe alguma versão do Java instalada como pode ser observado na Figura 8.

Figura 9 – Visualização do prompt de comando.

Fonte: Adaptado vídeo aula "Instalação Java, JDK e JRE" (DAMASCENO, 2019)

Agora é necessário verificar se o ambiente do Windows já foi configurado para executar o Java, pois existe a possibilidade do computador, no passado, ter recebido uma instalação e configuração de ambiente do Windows para uma versão menos atualizada. Para tal verificação, ainda no prompt de comando, digitar: "javac", como pode ser visto na Figura 9.

Figura 10 – Visualização do prompt de comando.

Fonte: Adaptado vídeo aula "Instalação Java, JDK e JRE" (DAMASCENO, 2019)

Após essas duas verificações e a confirmação de ausência da ferramenta Java, é necessário fazer o download do JDK e JRE.

Para obter estas ferramentas é necessário acessar o link: <https://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html >, diretamente do desenvolvedor da plataforma. A página vista na Figura 10 será carregada e então, seleciona-se a opção de download correspondente ao sistema operacional do computador, neste caso, Windows x64.

Ambas as ferramentas JDK e JRE são baixadas e instaladas juntas por este link de download.

Figura 11 – Página de download do JDK.

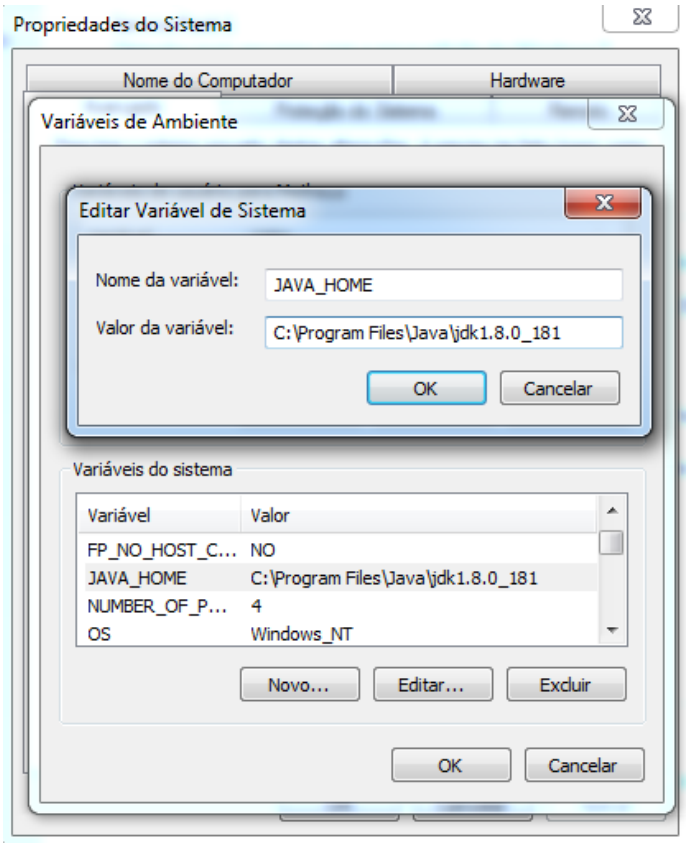
Java SE Development Kit 8u201
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.98 MB	jdk-8u201-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u201-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.98 MB	jdk-8u201-linux-i586.rpm
Linux x86	185.77 MB	jdk-8u201-linux-i586.tar.gz
Linux x64	168.05 MB	jdk-8u201-linux-x64.rpm
Linux x64	182.93 MB	jdk-8u201-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u201-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.33 MB	jdk-8u201-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.31 MB	jdk-8u201-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.99 MB	jdk-8u201-solaris-x64.tar.Z
Solaris x64	92.16 MB	jdk-8u201-solaris-x64.tar.gz
Windows x86	197.66 MB	jdk-8u201-windows-i586.exe
Windows x64	207.46 MB	jdk-8u201-windows-x64.exe

Fonte: Adaptado de Oracle

Após a instalação dessas ferramentas, configuram-se as variáveis de ambiente do Windows. Para tal segue-se o procedimento de configuração: Menu Iniciar/ Clicar com botão direito do mouse em Computador/ Propriedades/ Configurações Avançadas do Sistema/ Variáveis de Ambiente. Nas variáveis do sistema, seleciona-se a opção: “Novo” como visto na Figura 11. Na aba “Editar Variáveis do sistema” o nome escolhido para a variável foi “JAVA_HOME” e em “Valor da variável” coloca-se o caminho onde o JDK foi instalado no disco rígido.

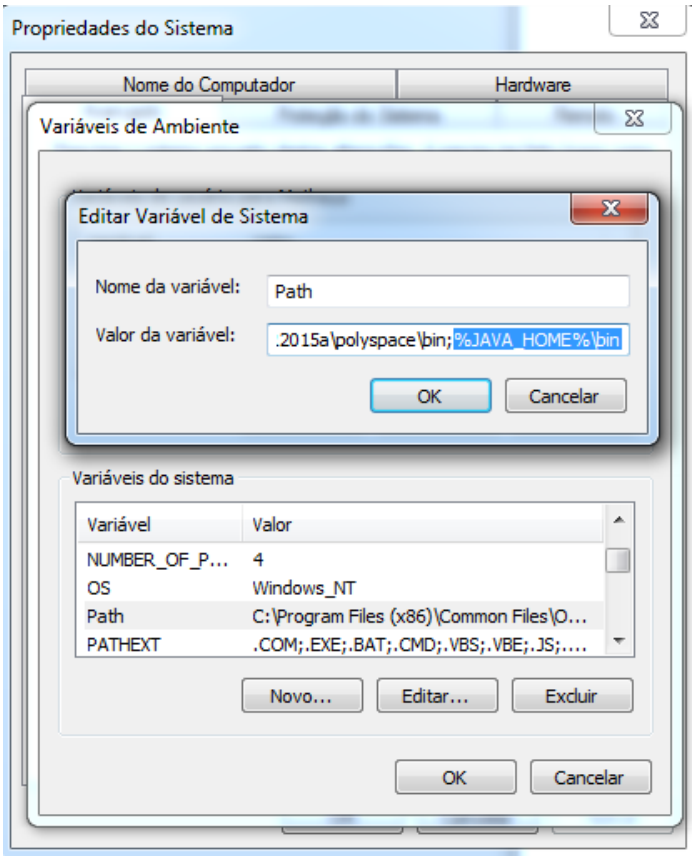
Figura 12 – Configuração de Variáveis de Ambiente.



Fonte – Autoria própria

Em sequência, ainda nas variáveis do sistema procura-se a variável chamada “Path”, clica-se em Editar e no final da sentença do campo “Valor da Variável” escreve-se: “%JAVA_HOME%\bin” como pode ser visto na Figura 12.

Figura 13 – Instanciando Variáveis de Ambiente.



Fonte – Autoria própria

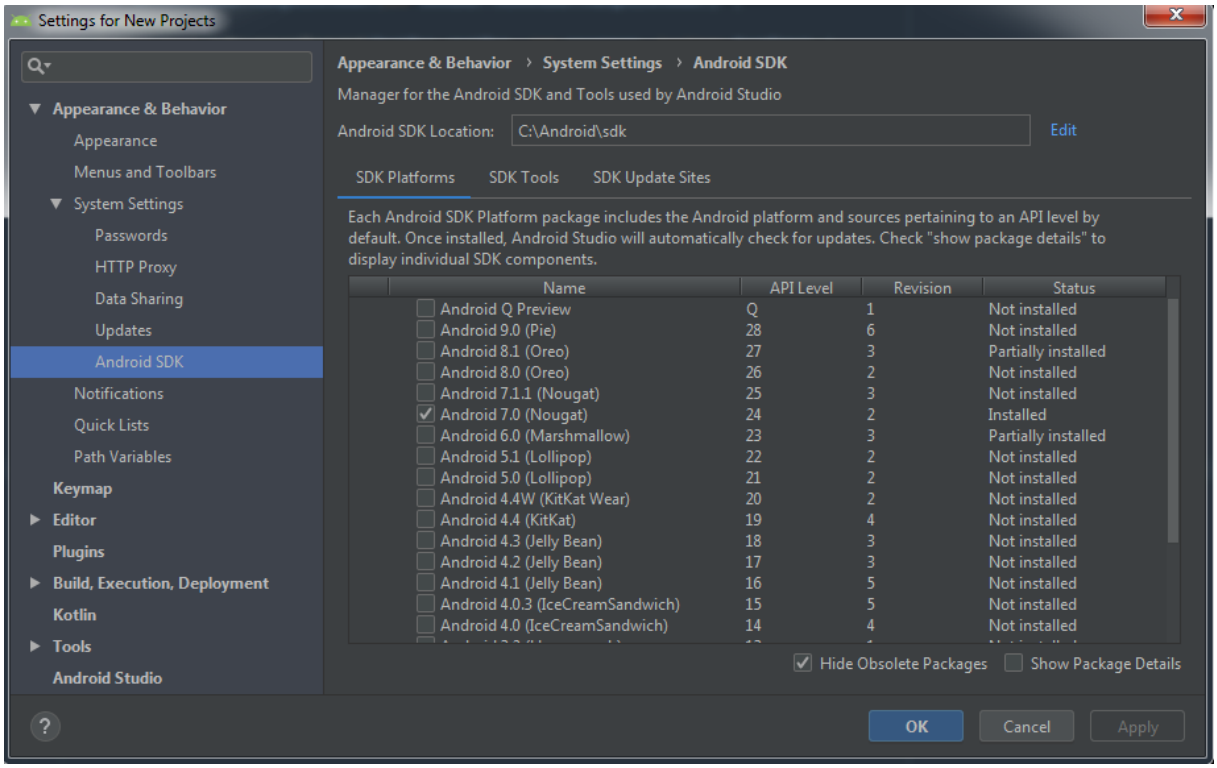
Estas variáveis nada mais são do que os caminhos que o sistema Windows irá percorrer quando o JDK for executado ou solicitado pelo usuário ou pelo *Android Studio* em qualquer situação.

3.3.2 Instalação Android Studio e Configuração SDK Android

Depois de efetuar o download do *software* Android Studio pelo link: < https://developer.android.com/studio/?hl=pt-br> e concluir os passos de instalação é necessário configurar a ferramenta SDK *Manager*. Na opção “SDK *Manager*”, uma nova tela se abre e pode ser vista na Figura 13.

Nesta tela é possível selecionar qual versão do SDK será instalada, sendo que, dentro desta instalação está à versão compatível do sistema operacional e sua API correspondente. Na Figura 13, a versão escolhida foi o Android 7.0 Nougat.

Figura 14 – Tela de configuração SDK.

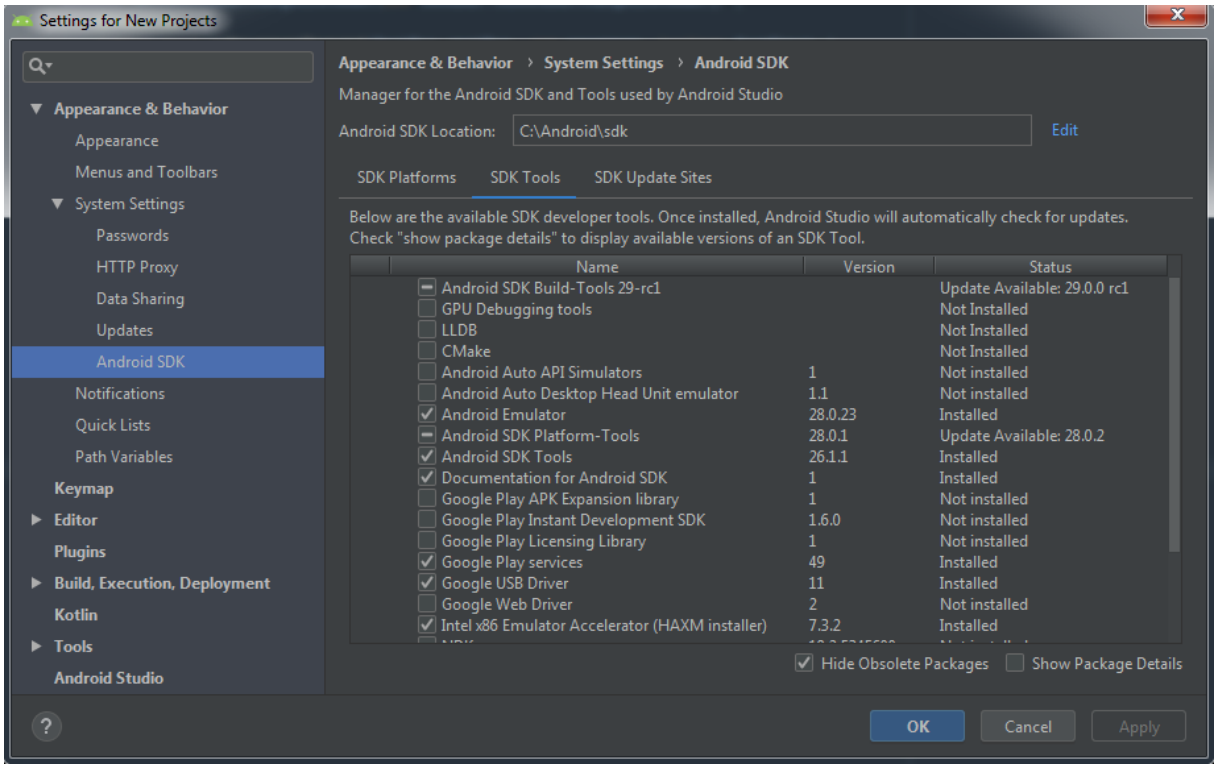


Fonte – Software Android Studio (2019)

Após efetuar o passo anterior, uma configuração mais refinada é necessária, onde serão definidas quais ferramentas em específico serão instaladas no computador. Na Figura 14 é possível observar a tela de ferramentas do SDK e uma listagem com todas as opções disponíveis para uso. As opções vistas na Figura 14 com as *checkbox* selecionadas foram às escolhidas para utilização deste trabalho e são listadas da seguinte forma:

- Android Emulator
- Android SDK Tools
- Documentation for Android SDK
- Google Play Services
- Google USB Driver
- Intel x86 Emulator Accelerator
- Support Repository

Figura 15 – Tela de configuração das ferramentas do SDK.



Fonte – Software Android Studio (2019)

Esta seleção de opções traz consigo os recursos de testes para o aplicativo que será desenvolvido, a documentação das ferramentas instaladas no computador, opções de criação e desenvolvimento de *layout* para o aplicativo e o driver de conexão USB. Este driver, além de emular o aplicativo virtualmente, pode estabelecer conexão com hardware externo para realização de testes.

3.4 DESENVOLVIMENTO DE LAYOUT

Para o trabalho proposto se faz necessário como primeiro passo, desenvolver o layout do aplicativo. Esse processo define a localização de cada elemento na tela do dispositivo móvel bem como a quantidade de elementos ativos contidos no aplicativo e que serão instanciados posteriormente durante a programação.

Com essas informações em mãos e o esboço visual o risco de perder funções importantes para o usuário durante a codificação praticamente desaparece.

Estes elementos ativos referentes à primeira tela do aplicativo que se abre logo após a inicialização estão listados na Tabela 4 e estão acompanhados de suas respectivas funções.

Tabela 4 – Elementos contidos no primeiro layout.

Elemento	Função
TextView	Caixa de texto – Título da primeira tela do aplicativo.
TextView	Caixa de texto – Instruções para iniciar uso do aplicativo.
TextView	Caixa de texto – Título da função de exemplos.
TextView	Caixa de texto – Título da lista de vozes.
ImageView	Caixa de Imagem – Sinal visual de reprodução de amostra vocal rugosa.
ImageView	Caixa de Imagem – Sinal visual de reprodução de amostra vocal soprosa.
ImageView	Caixa de Imagem – Sinal visual de reprodução de amostra vocal tensa.
Button	Botão – Inicia áudio de amostra vocal rugosa na sessão de exemplos.
Button	Botão – Inicia áudio de amostra vocal soprosa na sessão de exemplos.
Button	Botão – Inicia áudio de amostra vocal tensa na sessão de exemplos.
ListView	Lista – Seleção da amostra vocal e inicio da segunda tela.

Fonte: Autoria própria

Os elementos da segunda tela do aplicativo, referentes à tela de reprodução de amostras vocais, controle de mídia, classificação de característica subjetiva e classificação de grau característico estão listados na Tabela 5, também seguidos de suas funções.

Tabela 5 – Elementos contidos no segundo layout.

(continua)

Elemento	Função
TextView	Caixa de texto – Nome da amostra vocal que está em execução.
TextView	Caixa de texto – Instruções para seleção de característica subjetiva dominante.
TextView	Caixa de texto – Resultado da classificação de característica.

Tabela 5 – Elementos contidos no segundo layout.

(conclusão)

Elemento	Função
TextView	Caixa de texto – Informa Grau característico escolhido e resultado da classificação.
TextView	Caixa de texto – Informa progresso da Seekbar para grau.
TextView	Caixa de texto – Informa valor inicial da Seekbar.
TextView	Caixa de texto – Informa posição inicial da Seekbar.
TextView	Caixa de texto – Informa valor final da Seekbar.
TextView	Caixa de texto – Informa posição final da Seekbar.
SeekBar	Barra de Seleção – Escala de grau subjetivo.
Button	Botão – Seleciona característica Rugosidade para comparação com amostra.
Button	Botão – Seleciona característica Soprosidade para comparação com amostra.
Button	Botão – Seleciona característica Tensão para comparação com amostra.
ImageView	Caixa de Imagem – Botão reproduz amostra seguinte.
ImageView	Caixa de Imagem – Botão reproduz amostra anterior.
ImageView	Caixa de Imagem – Botão reproduz amostra vocal.
ImageView	Caixa de Imagem – Botão para reprodução de amostra vocal.

Fonte: Autoria própria

3.5 VARIÁVEIS E MÉTODOS

Nesta etapa, foram criadas as variáveis do aplicativo, isto é, variáveis que vão receber os valores dos elementos ativos das Tabelas 4 e 5, para que, durante a execução pelo usuário, possam ser realizadas as operações necessárias.

Estas variáveis podem ser vistas na Tabela 6, onde estão descritos seus nomes, a qual objeto de *layout* está instanciada e/ou sua função.

Também na Tabela 6, estão listadas algumas variáveis globais de execução e operação. Algumas linhas podem conter mais de um nome de variável, são variáveis que estão instanciadas a objetos semelhantes no *layout* e que possuem a mesma função.

Tabela 6 – Variáveis globais e de instanciamento de objetos.

(continua)

Nome	Objeto instanciado e/ou função.
listaMusicas	Lista de amostras vocais, seleciona a amostra vocal e inicia segunda tela.
SoundR, soundT, soundS	Caixa de Imagem – Sinal visual de reprodução das amostras vocais.
Rugosidade	Botão – Inicia áudio de amostra vocal rugosa na sessão de exemplos.
Sopro	Botão – Inicia áudio de amostra vocal soprosa na sessão de exemplos.
Tensao	Botão – Inicia áudio de amostra vocal tensa na sessão de exemplos.
Exemplos	Variável que recebe função Media Player do Android para executar mídias.
Musicas	Lista com nova nomenclatura para amostras vocais.
Btn_rug	Botão – Classificar amostra vocal em Rugosa.
Btn_sop	Botão – Classificar amostra vocal em Soprosa.
Btn_ten	Botão – Classificar amostra vocal em Tensa.
Btn_play	Botão – Executa mídia de amostra vocal.
Btn_pause	Botão – Pausa execução de mídia de amostra vocal.
Btn_next	Botão – Executa amostra vocal subsequente a que estiver em execução.
Btn_prev	Botão – Executa amostra vocal anterior a que estiver em execução.
vozId	Caixa de Texto – Apresenta nome da amostra vocal em execução.
textoResultado	Caixa de Texto – Apresenta resultado de classificação característico.
textoGrau	Caixa de Texto – Apresenta valor de grau selecionado e resultado de classificação.
seekBar	Barra de Seleção – Instancia escala de grau subjetivo
j	Variável de contagem – Recebe posição da amostra vocal contida dentro da lista de amostras.
valorProgresso	Variável de contagem – Recebe posição da barra de seleção.
Numero	Recebe posição da barra de seleção para realizar operações de classificação.
mediaPlayer	Instancia método de execução de mídias Android.
arrayList	Lista contendo nomes das amostras vocais.
Texto	Variável de classificação de Rugosidade.
Texto2	Variável de classificação de Soprosidade.

Tabela 6 – Variáveis globais e de instanciamento de objetos.

(conclusão)

Nome	Objeto instanciado e/ou função.
Texto3	Variável de classificação de Tensão.
Caracteristicas	Lista contendo características subjetivas das amostras vocais.
Grau	Lista contendo grau característico das amostras vocais.

Fonte: Autoria própria

Após criar as variáveis, foram criados os métodos. Estes métodos são funções em blocos de código que, quando chamadas, executam uma ou mais ações nas variáveis vistas anteriormente.

Na Tabela 7 estão dispostos os métodos contidos no código deste aplicativo, bem como sua função.

Tabela 7 – Métodos e suas funções.

Método	Função
playMusic	Método que inicia a reprodução de amostra vocal.
pauseMusic	Método que pausa a reprodução de amostra vocal.
nextMusic	Método que carrega a próxima amostra vocal e inicia sua reprodução.
previousMusic	Método que carrega a amostra vocal anterior e inicia sua reprodução.
comparaRug	Método que compara amostra vocal com característica subjetiva.
comparaSop	Método que compara amostra vocal com característica subjetiva.
comparaTen	Método que compara amostra vocal com característica subjetiva.
comparaSeekBar	Método que compara grau da amostra vocal.
onCreate	Método que inicia o aplicativo.
onDestroy	Método que fecha tela do aplicativo e reseta Media Player do Android.
onClick	Método de acionamento de clique do aplicativo.

Fonte: Autoria própria

Na Figura 15 um dos métodos listados na Tabela 7, chamado “comparaSeekBar” pode ser visto de forma detalhada, este método recebe o valor que o usuário escolheu para o grau característico da amostra vocal e compara com

o valor real do grau da amostra, se estiver correto uma mensagem aparece na tela informando que o grau está correto, caso contrário, uma mensagem aparece na tela informando que está incorreto e o usuário tem a possibilidade de refazer a análise.

Figura 16 – Método de comparação de valor de grau.

```
250     public void comparaSeekBar(){
251         int numero = seekBar.getProgress();
252         String numero2 = Integer.toString(numero);
253         if(grau[j].equals(numero2)){
254             textoGrau.setTextColor(getResources().getColor(R.color.correto));
255             textoResultado.setText("O grau está correto!");
256
257         }else{
258             textoGrau.setTextColor(getResources().getColor(R.color.incorreto));
259             textoResultado.setText("O grau está incorreto!");
260
261         }
262     }
```

Fonte: Autoria própria

3.6 FINALIZAÇÃO E AJUSTES

Como última fase do projeto, após a criação dos dois layouts e das variáveis e métodos, foi feita a concatenação de tudo. Nessa fase o primeiro passo foi mudar os nomes das amostras vocais, pois o *Android Studio* tem regras para a utilização de mídias – nomes de arquivos de áudio não podem começar com número e nem com letra maiúscula – portanto, foi necessário atualizar os nomes das mídias como pode ser visto na Tabela 8.

Tabela 8 – Nomenclatura das amostras vocais antes e depois da atualização.

(continua)

Antes	Depois	Antes	Depois
12 – H R	hr01	16 – M R	mr19
7 – H R	hr02	36 – M R	mr23
15 – H R	hr04	55 – M R	mr26
17 – H R	hr05	97 – M R	mr29
37 – H R	hr09	104 – M R	mr30
20 – H S	hs06	15 – M S	ms16

Tabela 8 – Nomenclatura das amostras vocais antes e depois da atualização.

(conclusão)

Antes	Depois	Antes	Depois
49 – H S	hs11	0 9 – M S	ms17
60 – H S	hs12	21 – M S	ms20
79 – H S	hs13	28 – M S	ms22
90 – H S	hs14	64 – M S	ms28
9 – H T	ht03	6 – M T	mt18
27 – H T	ht07	23 – M T	mt21
30 – H T	ht08	38 – M T	mt24
48 – H T	ht10	45 – M T	mt25
93 – H T	ht15	59 – M T	mt27

Fonte: Autoria própria

Após atualização dos nomes, utiliza-se a variável “ArrayList” como visto na Figura 16, para criar uma lista que recebe como parâmetro os nomes das amostras em ordem alfabética e relaciona cada nome dentro dessa lista com uma Id (uma posição na lista que varia de 0 a 29, totalizando 30 posições).

Figura 17 – Comando para desenvolvimento de *ArrayList*.

```
89          arrayList = new ArrayList<String>();
90          Field[] fields = R.raw.class.getFields();
91          for(int i = 0; i < fields.length; i++){
92              arrayList.add(fields[i].getName());
93          }
```

Fonte: Autoria própria

O resultado obtido por essa codificação pode ser visualizado na Tabela 9, que apresenta a lista de nomes com suas respectivas posições.

Tabela 9 – Nomenclatura das amostras e Id de localização no *ArrayList.*

Nome	Id – Posição	Nome	Id – Posição
hr01	0	mr19	15
hr02	1	mr23	16
hr04	2	mr26	17
hr05	3	mr29	18
hr09	4	mr30	19
hs06	5	ms16	20
hs11	6	ms17	21
hs12	7	ms20	22
hs13	8	ms22	23
hs14	9	ms28	24
ht03	10	mt18	25
ht07	11	mt21	26
ht08	12	mt24	27
ht10	13	mt25	28
ht15	14	mt27	29

Fonte: Autoria própria

Para dar continuidade, codificou-se o aplicativo para que a classe Media Player contida no sistema operacional *Android* recebesse o arquivo de mídia correspondente a posição que o usuário vai selecionar na lista de amostras vocais, deixando o aplicativo pronto para receber o próximo comando do usuário. Esse trecho de codificação pode ser visto na Figura 17.

Figura 18 – Comando para instanciar Media Player Android.

```
148    public void playMusic(){
149
150        if(mediaPlayer == null){
151                vozId.setText("VOZ 0"+ (j+1));
152                recuperarId = getResources().getIdentifier(arrayList.get(j),
153                        defType: "raw", getPackageName());
154                mediaPlayer = MediaPlayer.create( context: Player_Activity.this, recuperarId);
155                mediaPlayer.start();
156
157        }
158        else{
159            mediaPlayer.start();
160        }
161    }
```

Fonte: Autoria própria

Em sequência foram configurados os botões de reprodução de mídia, assim como os botões de comparação de característica subjetiva e por fim foi utilizado um método para comparar o grau de característica subjetiva utilizando uma barra de rolagem *touch screen,* métodos citados anteriormente na Tabela 7. Os métodos de comparação podem ser vistos na sequência de Figuras 18 – 21.

Figura 19 – Método de comparação de rugosidade.

```
217    public void comparaRug(){
218        String texto = btn_rug.getText().toString();
219        if(caracteristicas[j].equals(texto)){
220            btn_rug.setBackgroundResource(R.color.correto);
221            textoResultado.setText("A resposta está correta!");
222        }else{
223            btn_rug.setBackgroundResource(R.color.incorreto);
224            textoResultado.setText("A resposta está incorreta!");
225        }
226    }
```

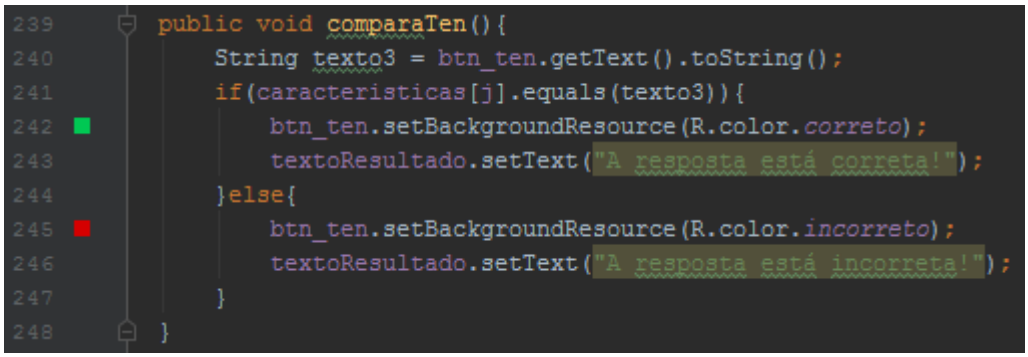
Fonte: Autoria própria

Figura 20 – Método de comparação de soprosidade.

```
228    public void comparaSop(){
229        String texto2 = btn_sop.getText().toString();
230        if(caracteristicas[j].equals(texto2)){
231            btn_sop.setBackgroundResource(R.color.correto);
232            textoResultado.setText("A resposta está correta!");
233        }else{
234            btn_sop.setBackgroundResource(R.color.incorreto);
235            textoResultado.setText("A resposta está incorreta!");
236        }
237    }
```

Fonte: Autoria própria

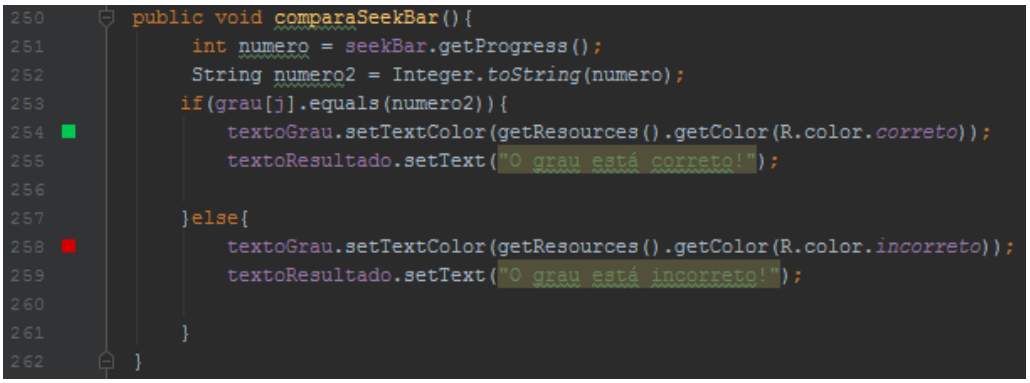
Figura 21 – Método de comparação de tensão.



```
239    public void comparaTen(){
240        String texto3 = btn_ten.getText().toString();
241        if(caracteristicas[j].equals(texto3)){
242            btn_ten.setBackgroundResource(R.color.correto);
243            textoResultado.setText("A resposta está correta!");
244        }else{
245            btn_ten.setBackgroundResource(R.color.incorreto);
246            textoResultado.setText("A resposta está incorreta!");
247        }
248    }
```

Fonte: Autoria própria

Figura 22 – Método de comparação de grau característico.



```
250    public void comparaSeekBar(){
251         int numero = seekBar.getProgress();
252         String numero2 = Integer.toString(numero);
253        if(grau[j].equals(numero2)){
254            textoGrau.setTextColor(getResources().getColor(R.color.correto));
255            textoResultado.setText("O grau está correto!");
256
257        }else{
258            textoGrau.setTextColor(getResources().getColor(R.color.incorreto));
259            textoResultado.setText("O grau está incorreto!");
260
261        }
262    }
```

Fonte: Autoria própria

4 RESULTADOS

A busca pela criação de uma ferramenta de utilização intuitiva, simples e capaz de atender as necessidades de treinamento de jovens graduandos e profissionais fonoaudiólogos resultou em um aplicativo móvel para sistema operacional *Android* capaz de proporcionar ao usuário uma experiência inovadora e eficiente. No decorrer deste capitulo serão demonstradas todas as funções intrínsecas a utilização deste aplicativo e o porquê cada uma delas é necessária para a realização de seu propósito.

4.1 VALIDAÇÕES E FUNÇÕES DO APLICATIVO

Para realização de todos os testes das funções propostas foi utilizado o emulador nativo do *software Android Studio*. Este emulador tem as características de um *Smarthphone* Nexus 5X com API level 26, possui sistema operacional compatível a este API *level* – SO *Android* 8.0 Oreo.

Este emulador pode ser visualizado na Figura 22, ele possui tela de 5.2 polegadas, resolução 1080x1920, a 420dpi (*Dots per Inch*), memória RAM de 2Gb e um disco de armazenamento interno de 64Gb com um cartão de memória removível de 512 Mb.

Figura 23 – Emulador Nexus 5X.

Fonte: Autoria própria

Neste aplicativo foi criada uma tela inicial funcional e simples, com as devidas instruções para proporcionar ao usuário uma rápida imersão ao conteúdo disponível. Logo na primeira tela vista na Figura 23 são apresentados, ao lado direito, três exemplos de amostras vocais, cada um dos exemplos possui uma característica subjetiva distinta e disponível a visualização. Tal ferramenta possibilita calibrar a audição do usuário para dar continuidade ao treinamento.

Ao lado dos exemplos, no lado esquerdo da tela, encontra-se disponível para execução o acesso a *ListView* contendo todas as amostras vocais disponíveis no banco de dados. Tais amostras já estão classificadas dentro do aplicativo de acordo com sua característica perceptiva e o grau de desvio vocal.

Figura 24 – Tela Inicial do aplicativo.

Fonte: Autoria própria

Nesta tela foram colocadas instruções para o usuário, caso julgue necessário, executar inicialmente os três exemplos disponíveis. Ao executar tais exemplos um sinal visual é mostrado na tela sinalizando qual exemplo está em reprodução, isto pode ser visto na Figura 24 e, assim que selecionado um novo exemplo, o sinal visual do exemplo anterior se apaga.

Figura 25 – Tela Inicial do aplicativo – exemplos em execução.

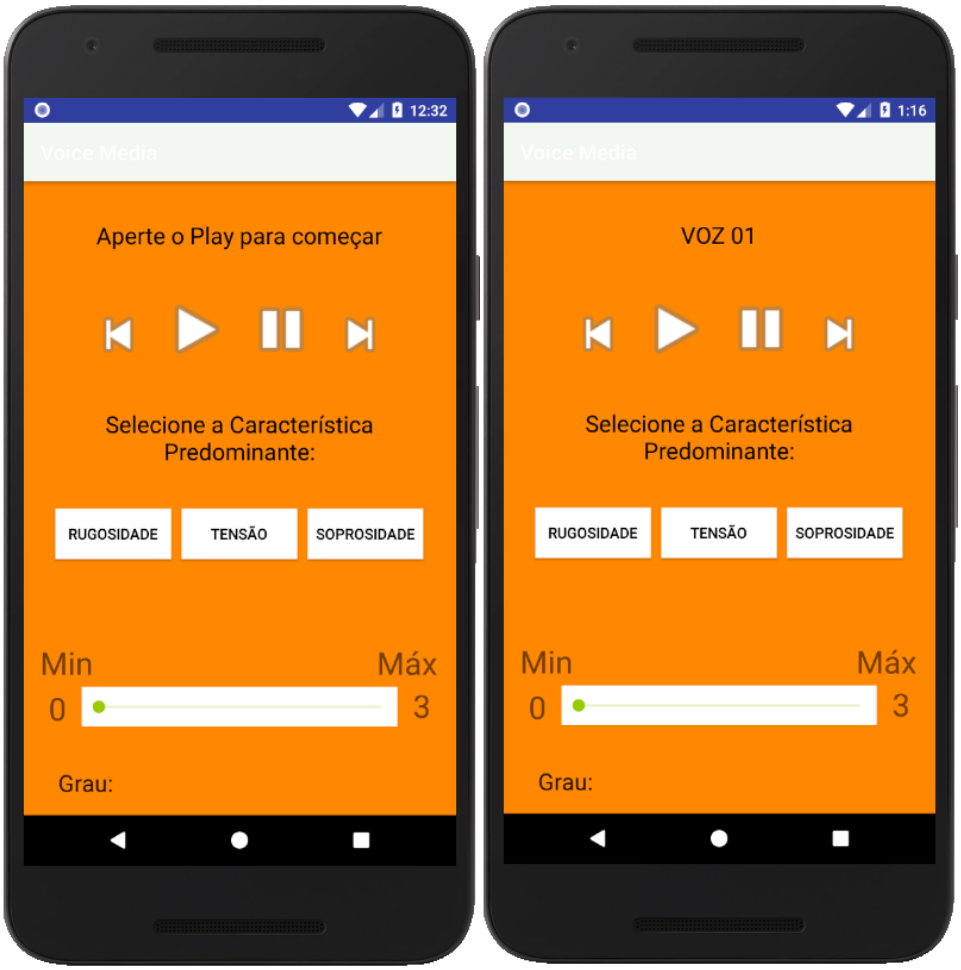
Fonte: Autoria própria

Após executar os três exemplos o usuário já tem disponível a lista de amostras vocais à esquerda, na tela, e intuitivamente, seleciona a seu critério, uma das amostras vocais para reprodução.

Tais amostras apresentam nomes modificados se comparados aos nomes que possuem no banco de dados do aplicativo, essa mudança tem a função de não repassar ao usuário qual característica subjetiva essa amostra contém.

Ao selecionar uma das opções da lista uma nova tela se abre. Esta tela vista na Figura 25, apresenta ao usuário no topo da tela, uma instrução para dar início à reprodução da amostra vocal. Em seguida ao executar a amostra, esta instrução é substituída pelo nome da amostra em reprodução e, na parte inferior do nome aparecem: quatro botões de controle de reprodução de mídia, três botões de classificação de característica subjetiva e uma barra de rolagem para classificação de desvio vocal de acordo com a escala GRBAS.

Figura 26(a) – Tela de reprodução e comparação de características subjetivas.



Fonte: Autoria própria

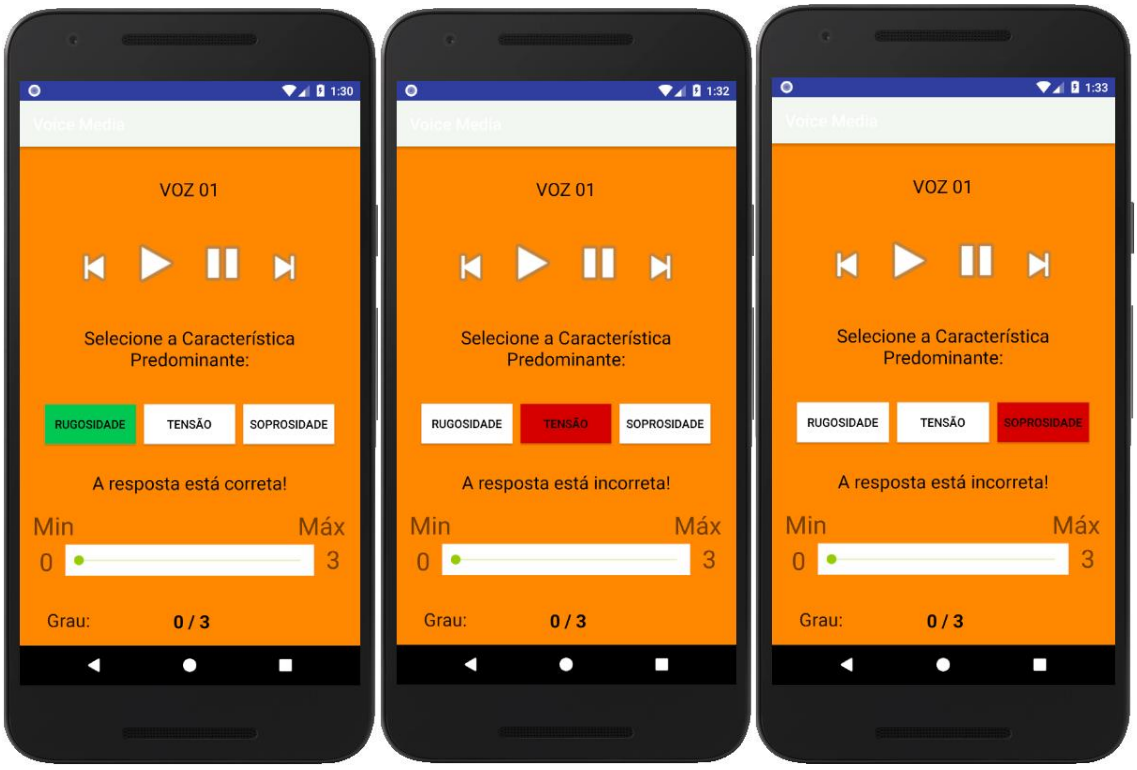
Ao apertar o botão de Play para reproduzir a amostra, o usuário escuta o áudio da amostra vocal. Ao fim da reprodução o áudio é interrompido, porém, o usuário pode reproduzir a mesma amostra quantas vezes julgar necessário.

Para dar continuidade ao treinamento, assim que sentir-se capaz, o usuário seleciona, utilizando um dos três botões nomeados com as características rugosidade, soprosidade e tensão, qual das três características subjetivas está presente na amostra vocal escutada.

Após a escolha e clique no botão, o aplicativo compara se a resposta do usuário está correta e sinaliza ao usuário visualmente mudando a cor do botão para verde caso esteja correta ou para vermelha caso esteja incorreta. Além dessa mudança de cor, é mostrada na tela uma mensagem textual informando o resultado da comparação.

A Figura 26 mostra essa comparação característica para a amostra vocal número 1.

Figura 26(b) – Tela de reprodução e comparação de características subjetivas.

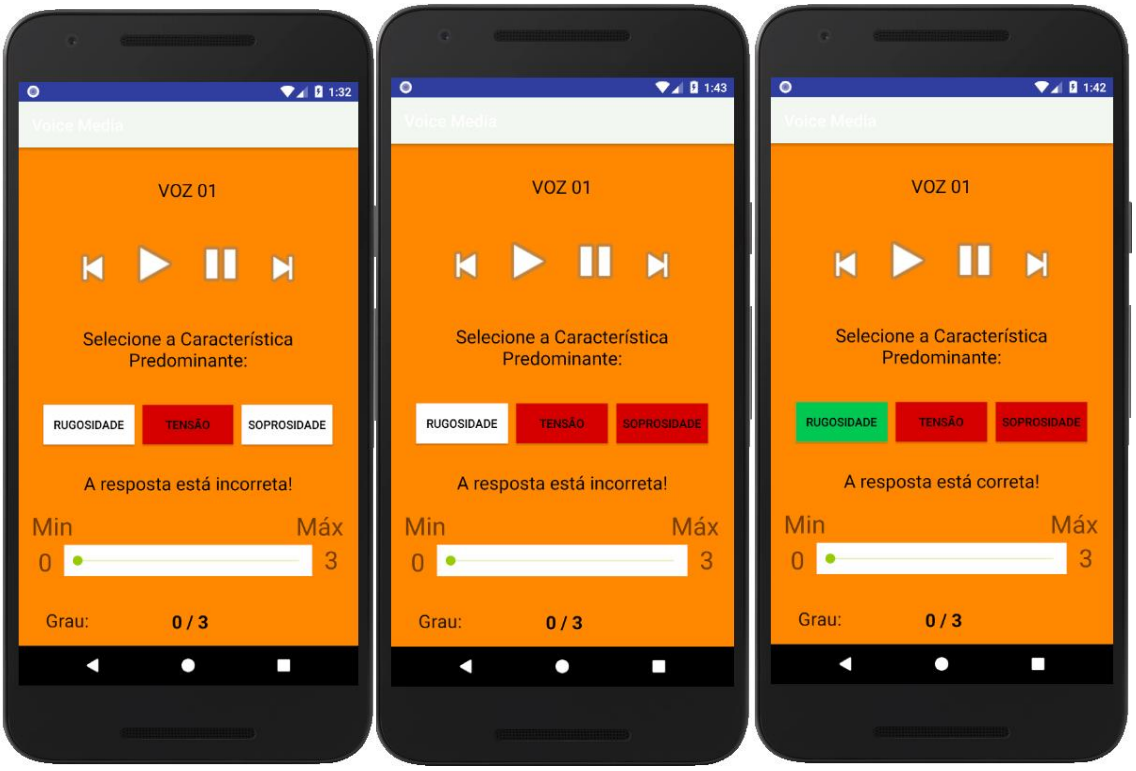


Fonte: Autoria própria

Caso o usuário selecione um botão e resulte em uma classificação incorreta o botão receberá a coloração vermelha como citado anteriormente, porém, ainda será possível continuar com a classificação.

O botão de característica incorreta permanecerá na cor vermelha e o usuário poderá selecionar novamente outro botão com uma característica até que faça a classificação de forma correta como visto na Figura 27, onde foi selecionada a opção Tensão e classificada incorreta, em seguida selecionada a opção Soprosidade e também classificada incorreta e por fim, na última tentativa, selecionada a opção Rugosidade e classifica como correta.

Figura 27 – Comparação de características subjetivas.



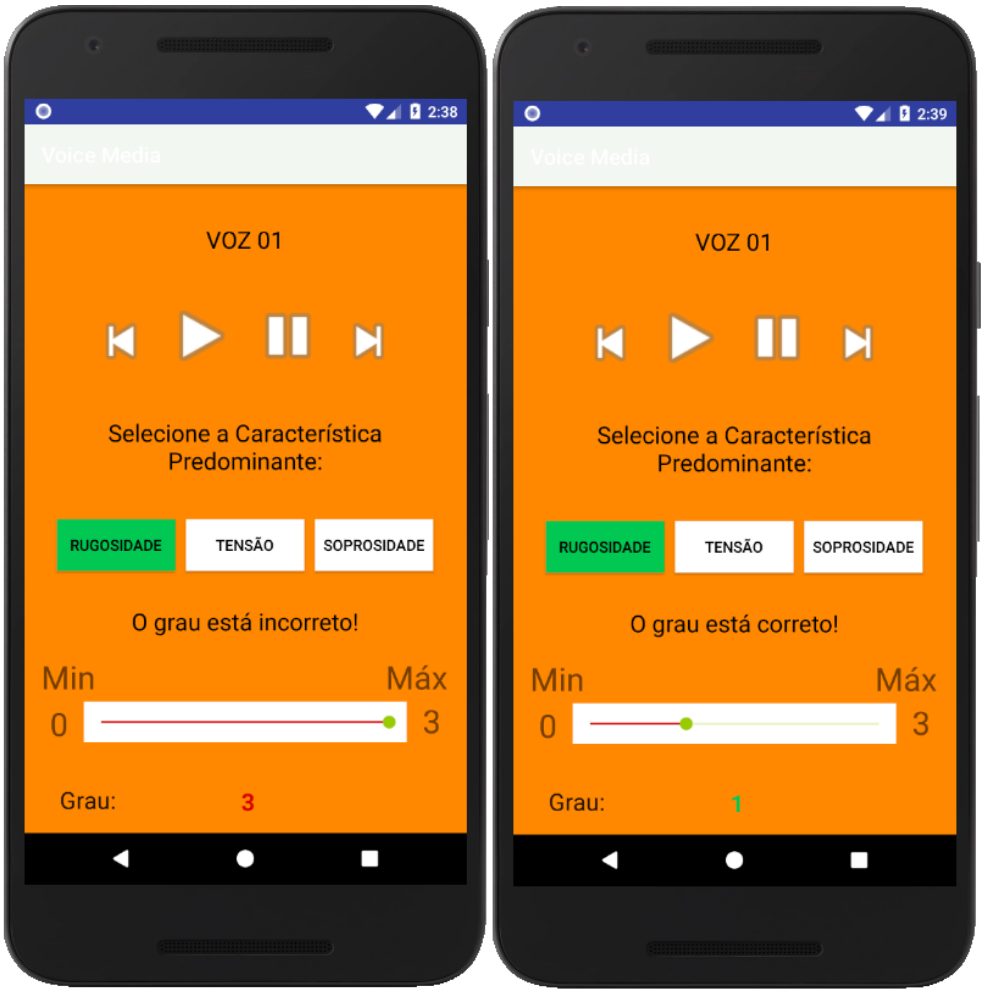
Fonte: Autoria própria

Após a classificação da característica subjetiva, o usuário tem a opção de se aprofundar mais classificando a amostra em seu grau característico. Como opção de classificação, o aplicativo disponibiliza uma barra de rolagem com escala mínima em zero e máxima em três.

Essa classificação é feita pressionando o ponto verde na barra de rolagem e arrastando o mesmo pelos níveis da escala até a posição escolhida. Quando o ponto é liberado a comparação de valores é feita pelo aplicativo e um sinal visual aparece na tela como visto na Figura 28 a classificação em grau da amostra vocal número 1. Os níveis progridem de unidade em unidade pela barra, ou seja, não é possível atribuir valores diferentes de números inteiros iniciando em zero e terminando em três.

A classificação em grau segue o mesmo pretexto da classificação anterior, onde caso o usuário efetue uma classificação incorreta é possível fazer uma nova sem perder os dados anteriores.

Figura 28 – Comparação de grau das características subjetivas.



Fonte: Autoria própria

Por fim, os botões de controle de mídia permitem ao usuário selecionar a próxima amostra vocal subsequente à amostra que foi reproduzida ou selecionar a anterior a ela.

Ao chegar à amostra trinta o aplicativo retornará à primeira amostra caso seja selecionada a opção de ir para a próxima. Semelhante a isto, na amostra número 1, caso o botão de retornar seja pressionado uma mensagem visual informa que não existem amostras anteriores aquela, como pode ser visto na Figura 29.

Figura 29 – Mensagem visual para fim do número de amostras.

Fonte: Autoria própria

5 CONSIDERAÇÕES FINAIS

A chegada dos *Smarthphones* trouxe, além de uma revolução informativa, a possibilidade de buscar novos meios de solucionar problemas e situações cotidianas e disponibilizar essas soluções para o público quase que instantaneamente. Criou-se a possibilidade de utilizar um aparelho portátil, que tem como principal característica a mobilidade.

Levando isso em consideração, durante a elaboração deste trabalho foi necessário aprender a utilizar algumas novas tecnologias como: software *Android Studio*, novas linguagens de programação linguagens de criação de layout e conceitos importantes vindos de outras áreas do conhecimento como a fonoaudiologia, assim como criar e realizar simulações que trouxessem resultados efetivos.

O objetivo central deste trabalho foi desenvolver uma ferramenta didática capaz de auxiliar estudantes em fase de graduação a desenvolver e aprimorar sua capacidade auditiva, e, profissionais fonoaudiólogos e otorrinolaringologistas a se prepararem para realizar suas avaliações e diagnósticos subjetivos, dando novas condições tão eficazes quanto as atuais para isso.

O aplicativo disponibiliza um banco de amostras vocais, classificadas em característica subjetiva e grau de desvio vocal, um reprodutor de mídia para que o usuário possa ouvir as amostras e então, utilizar dos classificadores de forma didática para aperfeiçoar sua capacidade de audição.

Dito isto, é possível afirmar que tal objetivo segue o caminho para ser concluído, visto que, o desenvolvimento deste aplicativo foi feito por alguém que não possuía capacitação profissional auditiva para classificar amostras vocais, e, ao final dos testes efetuados nos classificadores apresentados neste trabalho, o mesmo foi capaz de identificar com certa facilidade a predominância característica de algumas amostras.

Este aplicativo será um forte aliado tecnológico aos profissionais, visto ser uma ferramenta portátil e de simples manuseio.

Este trabalho, no entanto, é o primeiro passo para o desenvolvimento deste seguimento, e para dar continuidade e aperfeiçoar esta ferramenta é válido elucidar que podem ser adicionadas novas funcionalidades.

Para trabalhos futuros, destaca-se a possibilidade de um cadastro de usuários, onde, um profissional em atividade, fonoaudiólogo ou otorrinolaringologista, poderá adicionar novas amostras vocais já classificadas de acordo com a característica subjetiva e grau, pois o banco de vozes é limitado em trinta unidades amostrais. Isto elevaria o nível funcional do aplicativo visto que receberia atualizações constantes dos próprios usuários. Será necessário também realizar uma pesquisa para coleta de informações e validação das funções do aplicativo em campo, visto que, para teste real, é necessário submeter esta ferramenta a utilização por profissionais em formação e medir sua real capacidade didática.

REFERÊNCIAS

ALMEIDA, M. B. **Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares.** Revista Ciência da Informação, Brasília, v.31, n. 2, p.5-13, 2002.

AMORIN, P. MARCOS. **Mobilidade Aliada aos Projetos de TI.** 2011. Revista TI, Curitiba. Disponível em: <http://www.revistati.com.br/ti_controle/app/webroot/extras/MOBILIDADE.pdf >. Acessado em: 05 nov 2018

ANDRADE, L. M. de O. **Determinação dos Limiares de Normalidade dos Parâmetros Acústicos da Voz**. Dissertação de Mestrado, 63 f. Universidade de São Paulo, São Carlos, 2003.

ARNOLD, K.; GOSLING, J.; HOLMES, D. **THE Java Programming Language.** 4. ed. USA - Santa Clara, California, 2006.

BAX, M. P. **Introdução as Linguagens de Marcas.** Revista Ciência da Informação v. 30, n.1, p. 32 - 38, 2001.

BEHLAU, M. **Voz: o livro do especialista. v. 1.** Rio de Janeiro: Revinter, 2001.

BRINKHEDEN, D.; ANDERSSON, R. **A performance study of hybrid mobile applications compared to native applications**. Department of Engineering Science. University West - Sweden. Degree Project, 2015.

DAJER, M. E. **Padrões Visuais de Sinais de Voz Através de Técnica de Análise Não Linear.** São Carlos: EESC - USP, 2006. Dissertação de Mestrado.

DELIA, L. et al. **Multi-platform mobile application development analysis.** In: Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Con-ference on, p. 181–186, 2015.

DAMASCENO, J. **Classes e Objetos.** Curso Desenvolvimento Android, 2019.

DEITEL, H.M. “Java - Como programar”. 4. ed. 2005. São Paulo.

FERMINO. M. A. **Classificação de Distúrbios Vocais Utilizando Redes Neurais Artificiais.** 2017. 72 f. Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2017.

GRANATO, Luis Fernando. **Teoria Bayesiana na avaliação das condições da laringe**. 2005. Tese (Doutorado em Engenharia Elétrica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2005. Acesso em: 11 mar 2019.

GOLDMAN, A.; KON, F.; SILVA, P. J. S. **Introdução à Ciência da Computação com Java e Orientação a Objetos.** Editado e Revisado por Raphael Y. de Camargo. 1. ed. São Paulo: IME – USP, 2006.

GOOGLE Developers. **Android Studio.** Disponível em: <https://developer.android.com/studio>. Acessado em: 03 mar 2019.

IDC Corporate. **Smarthphone Market Share.** Disponível em: < https://www.idc.com/promo/smartphone-market-share/os>. Acessado em: 03 mar 2019.

JACKSON, W. ***Android Apps for Absolute Beginners:*** *Covering Android* 7. 4. ed. New York: Springer Science+Bussines Media, 2017.

KÖHLE, J. et al. **Ação de proteção de saúde vocal: perfil da população e correlação entre auto-avaliação vocal, queixas e avaliação fonoaudiológica perceptivo-auditiva e acústica.** Distúrbios da Comunicação. ISSN 2176-2724 16.3, 2004.

LECHETA, R. R. **Google Android:** aprenda a criar aplicações para dispositivos moveis com o Android SDK. 3. ed. Editora Novatec. São Paulo, 2013.

MAHMUD, D. M.; ABDULLAH, N. A. S. **Mobile application development feasibility studies:** A case study in universiti teknologi mara. In: Open Systems (ICOS), 2014 IEEE Com-ference on, 2014. p. 30–35.

MENDES, D. R. **Java com ênfase em orientação a objetos.** São Paulo. Novatec Editora, 2009.

MENDES, A. P.; FERREIRA, L. J. L.; CASTRO, E. **Softwares e hardwares de análise acústica da voz e da fala.** Revista Distúrbios da Comunicação. ISSN 2176-2724, v. 24, n. 3, 2012.

MOURA, A. (2009). **Geração Móvel:** um ambiente de aprendizagem suportado por tecnologias móveis para a "Geração Polegar". In P. Dias, A. J. Osório (org.) Actas da VI Conferência Internacional de TIC na Educação Challenges 2009 / Desafios 2009 (pp. 50-78). Braga: Universidade do Minho.

MUCCINI, H.; FRANCESCO, A. D.; ESPOSITO, P. **Software testing of mobile applications:** Challenges and future research directions. In: Automation of Software Test (AST), 2012 7th International Workshop on, 2012. p. 29–35.

ORACLE. **Java SE Development Kit 8 Downloads.** Disponível em: <<https://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html >. Acesso em: 11 abr 2019.

PALMEIRA, T. V. V. **Java:** História e principais conceitos. 2012. Disponível em: <https://www.devmedia.com.br/java-historia-e-principais-conceitos/25178> . Acesso em: 11 abr 2019.

PAVONI, H. E. **Classificação de Sinais Vocais em Parâmetros Não Acústicos Utilizando Redes Neurais Artificiais.** 2017. 72 f. Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2017.

PEREIRA, L. C. O.; SILVA, M. L. **Android para desenvolvedores.** Rio de Janeiro. ed. 1. Brasport, 2009.

PINHO, S. R.; PONTES, P. **Escala de avaliação perceptiva da fonte glótica:** RASAT. 2002. Disponível em:< http://www.fonovim.com.br/arquivos/0a98d8cb14ab1362d29db0110c7d4175-Disfonia-rasat-pt1.pdf>. Acesso em 03 mar 2019.

PINTO, F. de A. **Pet Care**: Uma Aplicação Móvel Híbrida para Gestão de Dados Veterinários. 2016. 58 f. Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2016.

RABINER, L. R.; SCHAFER, R. W. **Theory and Applications of Digital Speech Processing**. First Edition. New Jersey, US: John Wiley & Sons Inc., 2012.

RAMALHO, J. C. R.; HENRIQUE P. R. **XML e XSL da Teoria à Prática.** FCA – Editora de Informática, 2001.

RODRIGUES, G. **Smartphones e suas tecnologias**. São Carlos – USP, São Paulo, 2009. Trabalho de Conclusão de Curso.

ROSA, Marcelo de Oliveira. **Análise Acústica da Voz para Pré-Diagnóstico de Patologias da Laringe**. 1998. 261 f. São Carlos – USP. Dissertação de Mestrado.

SANTOS, M. O. **Análise Acústica de Desvios Vocais Infantis Utilizando a Transformada Wavelet.** 2015. Instituto Federal de Educação, Ciência e Tecnologia da Paraiba – IFPB. Dissertação de Mestrado.

SILVA, D. G. Marcelo da; SAMPAIO, T. M. M.; BIANCHINI, E. M. G. **Perseptions of newly graduated Speech and Language Pathologists regarding their academic formation, professional goals and knowledge update**. Revisão Sociedade Brasileira de Fonoaudiologia. v. 15. n.1. São Paulo , 2010.

SILVA, T. **Uso e Desenvolvimento de Aplicativos Sociais:** perspectiva da teoria ator – rede. Razón y Palabra, n. 76. 2011.

SOARES, M.A. **Aplicativo Móvel para Academia:** Estudo de Tecnologias e Desenvolvimento. 2016. 69 f. Instituto Federal de Educação, Ciência e Tecnologia, Minas Gerais, 2016.

SOUSA, E. P. Machado de. **Emulação de um Gerenciador de Dados Orientado a Objetos através de uma Interface de Programação de Aplicativos sobre um Gerenciador Relacional.** Dissertação de Mestrado. 126 f. USP - ICMC, São Paulo, 2000.

TEIXEIRA J. P.; FERREIRA D. B.; CARNEIRO S. M. **Análise Acústica Vocal –** Determinação do Jitter e Shimmer para diagnóstico de patologias da fala. IPT – Portugal, 2011.

TIBES, C. M. S.; DIAS, J. D. D.; ZEM-MASCARENHAS, S. H. **Mobile applications developed for the health sector in Brazil:** an integrative literature review. Revista Mineira de Enfermagem, v. 18, n. 2, p. 479-486, 2014. Disponível em: <http://www.reme.org.br/artigo/detalhes/940>. Acesso em 03 fev 2019.

VALENTIM, A. F.; CÔRTES, M. G.; GAMA, A. C. C. **Spectrographic analysis of the voice:** effect of visual training on the reliability of evaluation. Revista da Sociedade Brasileira de Fonoaudiologia, 2010.

VIDALE G. ***Os melhores Aplicativos para Cuidar da Saúde***. 2016. Disponível em: < http://veja.abril.com.br/saude/os-melhores-aplicativos-para-cuidar-da-saude/ >. Acesso em: 02 nov 2017.