Reconhecimento de espécies de aves por meio da vocalização utilizando *Python*

Fernando Fantinel da Silva[1], Reiner Franthesco Perozzo[1]

[1]Ciência da Computação – Universidade Franciscana (UFN)
Santa Maria – RS – Brasil

{fantinel89, reiner.perozzo}@gmail.com

Abstract. *This paper presents the proposal of a system for the identification of bird species through vocalization. The project has a mobile application, developed on the Android platform, which will allow the user to record and upload a bird's vocalization. A server application was also developed, using Python, which will be in charge of processing and classifying the information received. For the elaboration of this work, the Feature Driven Development methodology was chosen.*

Resumo. *Este trabalho apresenta a proposta de um sistema para identificação de espécies de aves por meio da vocalização. O projeto conta com uma aplicação móvel, desenvolvida na plataforma Android, que permitirá ao usuário realizar a gravação e o envio da vocalização de uma ave. Também foi desenvolvida uma aplicação servidor, utilizando Python, que será encarregada do processamento e classificação das informações recebidas. Para a elaboração deste trabalho foi escolhida a metodologia Feature Driven Development.*

1. Introdução

As aves têm sido matéria de observação e pesquisa desde muito tempo atrás, devido, dentre alguns motivos, ao seu canto e sua beleza [Sick 2001]. Quando se trata do assunto de reconhecimento e registro de espécies surgem alguns desafios, como citam também Sick e Barruel (1984), que algumas espécies de aves são tão semelhantes morfologicamente, que o som pode ser utilizado como critério para classificá-las.

Existem muitos estudos sobre reconhecimento de aves por meio de imagens, porém esse tipo de abordagem depende do contato visual com o animal, que em determinada situação, tende a dificultar o registro devido ao local onde a ave se encontra. Em contrapartida, uma abordagem através do som evitaria a questão do contato visual, pois tem a capacidade de evitar obstáculos e consegue atingir amplas distâncias [Márquez et al. 2011].

Diante disso, a proposta deste trabalho se trata do desenvolvimento de uma ferramenta que utilize o canto das aves como objeto de análise, auxiliando o processo de reconhecimento de espécies.

1.1. Objetivo geral

Desenvolver um sistema de reconhecimento de espécies de aves por meio da análise de vocalizações.

1.2. Objetivos específicos

Os objetivos específicos deste trabalho são:

- Desenvolver uma aplicação cliente, por meio da plataforma *Android*, que possibilite a gravação de áudio de vocalizações de aves, envio do arquivo de áudio e exibição dos resultados obtidos;
- Elaborar a aplicação servidor utilizando a linguagem de programação *Python*;
- Aplicar técnicas de processamento e classificação de sons utilizando a biblioteca *pyAudioAnalysis*;
- Projetar as funcionalidades do sistema com base nas diretrizes da metodologia *Feature Driven Development*.

2. Referencial Teórico

Esta seção apresenta uma revisão bibliográfica, denotando importantes conceitos e trabalhos relacionados, com o intuito de fundamentar a proposta do presente trabalho.

2.1. Vocalização das aves

De acordo com Dawkins (1989), os animais, para poderem sobreviver e procriar, necessitam de comunicação, essa que se dá a partir de um emissor, um receptor, uma informação e um sinal que possa transmiti-la de forma apropriada. No que diz respeito às aves, elas utilizam os sons como meio de comunicação, para várias finalidades, como alerta de predadores e para localizar filhotes [Vielliard 1987]. Catchpole e Slater (2008) citam que embora as aves possuam um sistema visual desenvolvido, fazem largo uso de sinais sonoros na comunicação. Silva (2001) acrescenta que, em ambientes florestais, os sinais sonoros se tornam ainda mais eficazes, pois podem se propagar em todas as direções.

A forma como os sinais sonoros são organizados para possibilitarem a comunicação entre as aves é estabelecida através de processos evolutivos e pelas condições que o ambiente impõe a cada espécie [Tolentino 2015]. Deste modo, Vielliard (2004) aponta que cada espécie elabora o seu próprio sistema de comunicação, de acordo com o que é exigido no meio em que ela habita, principalmente em relação ao comportamento dos sons emitidos no ambiente.

Além disso, de acordo com características e finalidades, os sinais sonoros das aves podem ser classificados como chamados e cantos. Sobre o chamado, em geral, esse se apresenta como um sinal sonoro menos elaborado, com poucas notas (unidades básicas que compõem o canto) e ocorre ao longo de todo o ano. É empregado, sobretudo, na comunicação entre pais e filhotes, como alarme contra predadores e no restabelecimento de contato visual entre membros de um grupo. Em relação ao canto, esse apresenta uma estrutura geralmente mais complexa do que o chamado, sendo composto por muitas notas e uma maior duração. O canto das aves é utilizado para comunicação em longas distâncias e tem sido o principal motivo de estudos [Tolentino 2015]. Ele pode ser emitido pelo macho, no intuito de atrair a fêmea, defender seu território e como recurso em conflito com outros machos [Sick 2001].

2.2. Análise e classificação dos sons

De acordo com Batista (2015), o reconhecimento automático de sons é algo recente, no qual apresenta uma importância nas últimas duas décadas. Isto porque a maior relevância nas pesquisas encontra-se no reconhecimento de fala [Giannoulis et al. 2013]. O pesquisador afirma que o interesse no reconhecimento de sons torna-se mais recente, devido à complexidade do reconhecimento do mesmo e sua menor utilidade, quando comparado ao reconhecimento automático de fala [Batista 2015]. "Nos sons urbanos (sons acústicos produzidos num ambiente urbano) a complexidade é ainda maior devido ao ruído de fundo característico de um ambiente urbano presente nos sons [Batista 2015, p. 21].

No entanto, Batista (2015), em sua dissertação de mestrado, cita que existem muitos trabalhos com reverência na área de reconhecimento de sons acústicos, que utilizam profusos métodos para executar o reconhecimento. Nesse sentido, de um modo geral, o pesquisador apresenta que uma metodologia para classificar os sons sustenta-se por duas fases: extração de atributos dos sons e criação de modelos de classificação utilizando os atributos extraídos. Além dessas fases, Cruz (2016) acrescenta uma fase anterior à extração de atributos reconhecida como: pré-processamento de sons.

A primeira tarefa no reconhecimento de áudio é o pré-processamento dos sons, a fim de prepará-los para a etapa de extração de atributos [Petry et al. 1999] [Carvalho e Machado 2011]. Esse procedimento, geralmente, inclui tarefas como eliminação da componente contínua do sinal, normalização do sinal e aplicação de filtros [Carvalho e Machado 2011].

A próxima fase, extração de atributos, visa buscar esses dados importantes (atributos), os quais contém informações importantes para identificar uma espécie [Mendes 2011]. Esse processo baseia-se na caracterização de um segmento de áudio e sua representação numérica [Cruz 2016]. Segundo Gunasekaran e Revathy (2011), existem diversas técnicas que podem ser empregadas no processo de caracterização de um sinal de áudio.

Posteriormente, já extraídos os atributos do som, avança-se para a etapa de extração de conhecimentos dos dados ou *data mining*, que a partir de um grande volume de dados procura extrair informações importantes e não triviais, apoiando-se no uso de algoritmos específicos [Cruz 2016]. Esses algoritmos são chamados de algoritmos de classificação. O objetivo deles é obter modelos de classificação a partir de exemplos de treino etiquetados. Cada exemplo pertence a uma classe conhecida, sendo que essa informação é fornecida ao algoritmo de classificação [Batista 2015]. A partir de um novo exemplo, ainda não etiquetado, o algoritmo de classificação tem a função de determinar a classe a que o exemplo pertence, possivelmente, com algum grau de incerteza. Esse processo de criação de um modelo com base em dados etiquetados também é chamado de "aprendizagem supervisionada" [Han et al. 2011].

2.3. Tecnologias utilizadas

2.3.1. *Python*

A linguagem de programação *Python* foi criada em 1990 por Guido Van Rossum. Inicialmente, era focada para profissionais da área da Física e Engenharia. Atualmente, a linguagem vende sendo utilizada na indústria tecnológica por empresas, tais como:

Google, *Microsoft* e *Disney*. A sua sintaxe inclui muitas estruturas de alto nível, diversos módulos prontos e *frameworks* de terceiros para a utilização de desenvolvedores(as). *Python* é um software de código aberto (com uma licença compatível com a *General Public License (GPL)*, entretanto menos restritiva, possibilitando que o *Python* seja vinculado em produtos proprietários) [Borges 2014].

2.3.2. *PyAudioAnalysis*

PyAudioAnalysis é uma biblioteca *Python* de código aberto que fornece uma ampla variedade de procedimentos de análise de áudio, incluindo: extração de características, classificação de sinais de áudio, segmentação supervisionada e não supervisionada e visualização de conteúdo. Esta biblioteca tem sido utilizada em várias aplicações de pesquisa de análise de áudio: funcionalidades de casa inteligente por meio da detecção de eventos de áudio, reconhecimento de emoção na fala, classificação de depressão com base em recursos audiovisuais, segmentação de música, recomendação de filmes multimodais com base em conteúdo e aplicações na área da saúde (por exemplo, monitoramento de hábitos alimentares) [Giannakopoulos 2015].

2.3.3. *Android*

Uma das vantagens de desenvolver aplicativos *Android* é a franqueza da plataforma. O sistema operacional é gratuito e de código fonte aberto, permitindo a visualização do código fonte da plataforma e de como seus recursos são implementados [Deitel et al. 2015].

2.4. Trabalhos relacionados

Nesta seção, são expostos alguns trabalhos que apresentam características semelhantes ao sistema aqui proposto, norteando o planejamento e desenvolvimento pretendidos.

2.4.1. Plataforma para recolha e identificação automática de espécies de pássaros utilizando registros de áudio

No trabalho de Cruz (2016) é apresentado o desenvolvimento de um sistema que possibilita a identificação de espécies de pássaros por meio dos sons emitidos por eles, a partir de gravações de áudio. Também traz como objetivo, a construção de uma base de dados etiquetada derivada dos registros de áudio capturados e submetidos ao sistema.

A respeito da extração de atributos do som, o trabalho utiliza duas metodologias, uma baseia-se na descoberta de padrões mais frequentes no som, chamada *Motifs*. A outra, bastante usada em reconhecimento de sons, baseia-se em *Mel-Frequency Cepstral Coefficients*. No que tange à classificação dos sons, são utilizados algoritmos de árvore de decisão e *Support Vector Machines*.

O trabalho contempla a implementação de uma aplicação móvel que será utilizada pelo usuário do sistema, uma aplicação *web* que será responsável pelo processamento dos dados, ou seja, a lógica que é proposta pelo trabalho e, tendo como uma terceira parte, uma aplicação *desktop* que terá a função de controlar e testar os serviços disponibilizados pela aplicação *web*.

No que se refere às tecnologias empregadas no desenvolvimento do trabalho, o autor fez uso da linguagem de programação Java na implementação da aplicação servidor (*web*) e da aplicação cliente (*Android*). Na aplicação servidor foi utilizada a

API Weka, implementada em Java, que apresenta diversos algoritmos de aprendizado de máquina e mineração de dados.

2.4.2. Reconhecimento automático de espécies de aves com base na sua vocalização

O artigo de Stastny et al. (2018) traz como objetivo reconhecer automaticamente espécies de aves por meio da análise da vocalização. Foram analisadas dezoito espécies diferentes de aves, classificadas entre seis famílias. De acordo com os autores, quando se trata do reconhecimento da vocalização das aves, é de suma importância levar em consideração o processo de reconhecimento da fala, tendo em vista que essas duas áreas apresentam semelhanças em um contexto geral.

Além disso, Stastny et al. (2018) consideraram a não linearidade acústica ao escutar as aves, tentando extrair as principais características para a descrição e modelagem da vocalização das aves através de um método de parametrização apropriado.

Neste seguimento, os autores citam que as aves, mediante suas expressões vocais, conseguem trocar uma variedade de informações. As vocalizações, que podem ser divididas em chamados e cantos, podem assumir várias funções como, por exemplo: alertar sobre perigo iminente ou indicar a conquista de um novo território. De modo geral, o canto tem maior duração e é mais complexo que o chamado.

A respeito da metodologia aplicada pelos autores, o sistema apresenta vários módulos independentes, em que o primeiro deles é responsável pela extração de parâmetros de um sinal, resultante de um arquivo de áudio. Posteriormente, os dados percorrem o módulo de *Voice Activity Detection*, com o intuito de selecionar apenas as partes avaliadas como vocalização. Por fim, o *Hidden Markov Model* é empregado no processo de reconhecimento das espécies de aves. A Figura 13, presente no Anexo A, expõe uma visão geral da proposta do trabalho, ilustrando os módulos do sistema.

2.4.3. Uso de técnicas de reconhecimento de fala para identificar espécies de aves

O trabalho de Tsai e Xue (2014) tem como propósito a implementação de um sistema de identificação automático dos sons de aves, por meio do emprego de técnicas de reconhecimento de fala, a fim de ajudar pessoas a poderem identificar as espécies de aves a partir do som emitido por elas.

Os autores apontam que existem mais de nove mil e setecentas espécies de aves no mundo. Ainda que muitos desses animais possam ser vistos, a maioria das pessoas não sabe reconhecer a que espécie pertencem. No trabalho em questão, foram escolhidas as dez espécies mais comuns encontradas na zona urbana de Taipé, capital de Taiwan. Os experimentos foram realizados com dados de áudio a partir de *CDs* e *websites*.

Com relação aos métodos utilizados no sistema, os autores adotaram dois sinais acústicos na análise, timbre e afinação. No processo de análise baseado no timbre, *Mel-Frequency Cepstral Coefficients (MFCCs)* foram empregados para caracterizar o som da ave. Em seguida, os *MFCCs* foram representados como um conjunto de parâmetros mediante o uso dos *Gaussian Mixture Models*. Na análise baseada em afinação, os sons emitidos pelas aves foram convertidos em uma sequência de notas *MIDI*. Posteriormente, *Bigram Models* foram utilizados para captar as informações de mudança dinâmica das notas.

2.5. Considerações sobre os trabalhos relacionados

Os trabalhos abordados apresentaram um panorama preciso a respeito do reconhecimento de sons e, especificamente, sobre vocalizações de aves. Foram detalhadas tarefas fundamentais no processo de análise e classificação de sons, tais como: pré-processamento, extração de atributos e uso de algoritmos de classificação.

Uma característica diferencial do trabalho em questão, se dá na utilização de um dispositivo móvel para interação com o usuário, semelhante ao que ocorre no trabalho de Cruz (2016), possibilitando que o sistema obtenha uma resposta em tempo de execução, visto que a gravação da vocalização da ave pode ser processada logo após o registro.

3. Metodologia

Nesta seção é descrita a metodologia de desenvolvimento a ser utilizada neste trabalho.

3.1 Boas práticas do *Feature Driven Development (FDD)*

A metodologia *Feature Driven Development* (Desenvolvimento Guiado por Funcionalidades, em português) traz como objetivo o emprego de pequenas iterações que normalmente duram em torno de duas semanas, onde ao final acontece a entrega de uma parte do *software* funcionando [Highsmith 2002]. Ela apresenta duas fases, compostas por cinco partes (Anexo B - Figura 14), que são descritas a seguir.

- Primeira fase (concepção e planejamento):
    1. Desenvolvimento do modelo abrangente: elaboração de um diagrama de domínio ou de classe a fim de visualizar a estrutura do *software*.
    2. Construção da lista de funcionalidades: determinar as funcionalidades do *software* a partir do diagrama de domínio criado na etapa passada, gerando assim uma lista de funcionalidades.
    3. Planejamento por funcionalidade: as funcionalidades pré-determinadas no passo anterior são organizadas em ordem de importância e dificuldade de implementação, determinando, assim, os prazos para serem entregues.

- Segunda fase (construção):
    1. Detalhamento por funcionalidade: é o detalhamento das funcionalidades, em que são selecionadas as funcionalidades a serem feitas. Nessa fase deve ser criado o diagrama de atividades exibindo o fluxo do sistema.
    2. Construção por funcionalidade: onde as funcionalidades são implementadas, tomando como referência o que foi detalhado nas fases anteriores.

Pelo fato da metodologia *FDD* apresentar duas fases, assim como é estruturado o Trabalho Final de Graduação (I e II) e por se tratar de uma metodologia pouco burocrática, sendo empregada em projetos em que existe um ou poucos colaboradores, foram utilizadas boas práticas da metodologia *FDD* no desenvolvimento deste trabalho.

4. Proposta

O presente trabalho traz como propósito o desenvolvimento de um sistema de reconhecimento automático de espécies de aves com base na vocalização. Por meio de

um aplicativo instalado em um dispositivo móvel, o usuário fará uma gravação do canto (vocalização) de uma ave. Esse registro será enviado ao servidor, que realizará todo o processo de análise do arquivo de áudio recebido. Após a obtenção dos resultados, a aplicação no dispositivo móvel recebe como resposta, uma mensagem exibindo o nome da espécie de ave que, provavelmente, coincida com aquela da gravação. A Figura 1 expõe uma visão geral da proposta do trabalho.

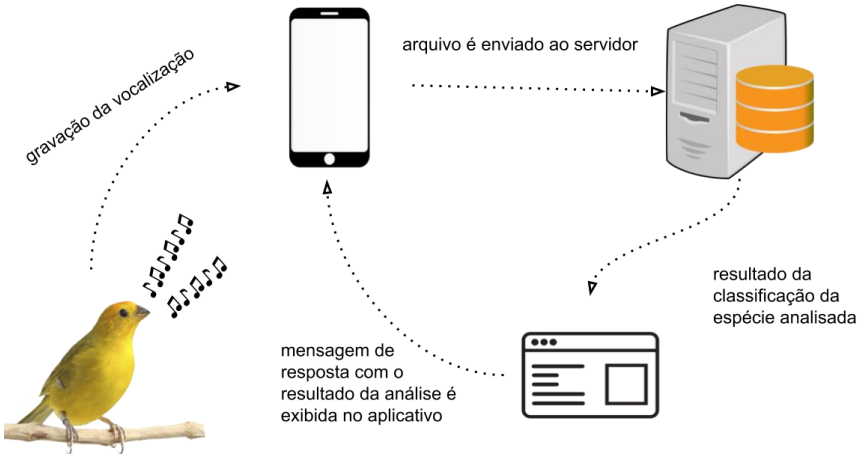

Figura 1. Visão geral do sistema

4.1. Projeto

4.1.1. Desenvolvimento do modelo abrangente

Nesta subseção é apresentado um panorama da proposta do sistema, possibilitando a definição do escopo do problema a ser resolvido. O Diagrama de Domínio exposto por meio da Figura 2, visa introduzir uma ideia inicial do sistema que será desenvolvido.

A partir do diagrama, pode-se perceber os componentes que fazem parte do escopo do projeto.

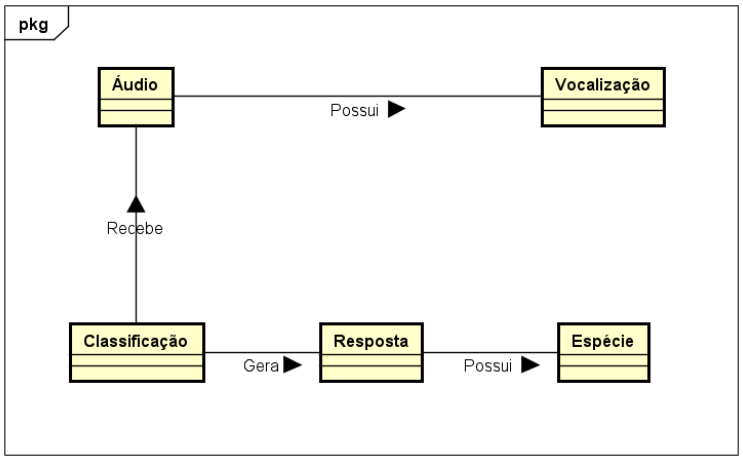

Figura 2. Diagrama de Domínio

4.1.2. Construção da lista de funcionalidades

Dando continuidade aos processos da metodologia *FDD*, esta etapa apresenta a lista de funcionalidades propostas, classificadas como requisitos funcionais e não funcionais. Os requisitos funcionais são as funcionalidades do sistema, aquelas que definirão o comportamento do sistema. Os requisitos não funcionais descrevem as características relacionadas à implementação do sistema, como tecnologias e ferramentas utilizadas no desenvolvimento.

No Apêndice A são apresentados os requisitos funcionais, descrevendo o nome dado a cada funcionalidade, uma breve descrição do seu funcionamento e uma medida a respeito da complexidade de implementação dos requisitos. No Apêndice B são exibidos os requisitos não funcionais.

4.1.3. Planejamento por funcionalidade

Nesta etapa, após ter sido realizado o levantamento das funcionalidades do sistema, são definidos os prazos para implementação de cada uma delas. O Apêndice C exibe esse planejamento trazendo as funcionalidades e seus respectivos prazos estipulados de desenvolvimento.

4.2. Implementação

Nesta subseção é detalhado o processo de construção do sistema, ou seja, a implementação das funcionalidades propriamente ditas.

4.2.1. Detalhamento por funcionalidade

Neste processo, por meio do Diagrama de Atividades exibido na Figura 3, é feito o detalhamento por funcionalidade, no qual é possível observar o fluxo das atividades entre a aplicação cliente e o servidor.

Iniciando no círculo preto, chamado nó inicial, o fluxo do diagrama acontece conforme é indicado pelas setas. Cada retângulo representa uma atividade, funcionalidade do sistema, e os losangos são os nós de decisão, que permitem que o fluxo seja ramificado dada as mensagens “Sim” e “Não” definidas ao redor de cada nó de decisão. O fluxo se inicia com as atividades respectivas à aplicação cliente e, após a execução da atividade “Enviar arquivo”, o fluxo é direcionado às atividades do servidor da aplicação e, posteriormente, retorna ao cliente para realizar a atividade “Exibir resposta”. Se o usuário não desejar realizar uma nova gravação, o fluxo chega ao círculo preto delineado, chamado nó final, encerrando a execução do sistema.

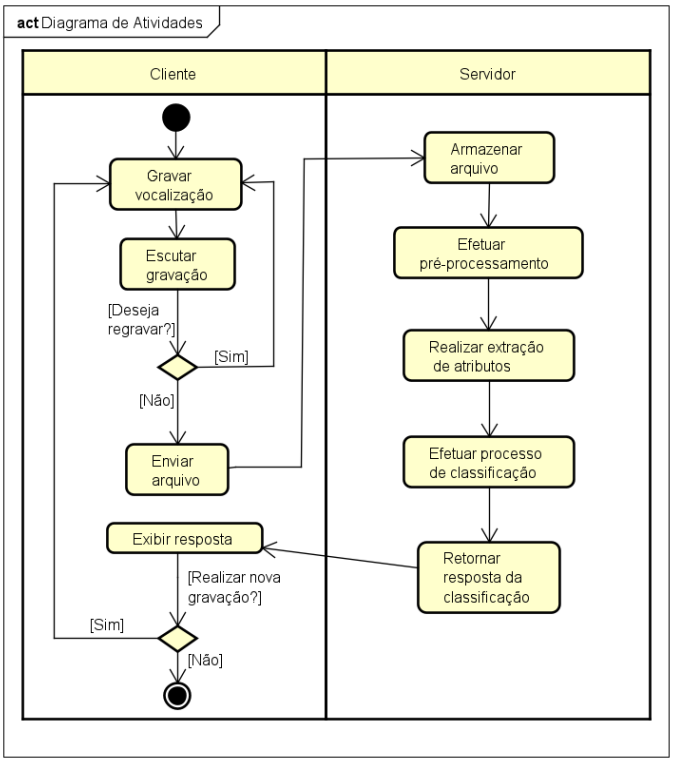

Figura 3. Diagrama de Atividades

4.2.2. Construção por funcionalidade

O último processo da metodologia tem a finalidade de apresentar as funcionalidades propriamente ditas. São elas:

- Gravar vocalização;
- Enviar áudio da vocalização;
- Armazenar arquivo;
- Efetuar pré-processamento;
- Extrair atributos;
- Realizar a classificação;
- Retornar resposta;
- Exibir resposta.

Cada uma das subseções a seguir detalha uma funcionalidade, bem como sua implementação.

4.2.2.1. Gravar vocalização

Para a implementação dessa funcionalidade, foi instanciado o objeto *recorder* a partir da classe *MediaRecorder*. Utilizando os métodos *set* do objeto, são definidas a fonte de gravação, nesse caso o microfone do dispositivo móvel, o formato do arquivo e a codificação do áudio, assim como o nome do arquivo a ser gerado após o término da gravação.

Além disso, foi implementado um bloco *try/catch* a fim de ter um controle sobre o processo de gravação em caso de erros. Essa implementação foi inserida no método descrito na Figura 4, que é executado ao clicar no botão Gravar na tela da aplicação móvel.

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

    @Override
    public void onClick(View view) {

        if (player != null)
            PararAudio();
        progressBar.setVisibility(View.VISIBLE);
        textView.setText("Gravando");
        recorder = new MediaRecorder();
        recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
        recorder.setOutputFile(PATH_NAME);
        try {
            recorder.prepare();
            recorder.start();
        } catch (IOException e) {
            e.printStackTrace();
            progressBar.setVisibility(View.GONE);
            Toast.makeText(getApplicationContext(), "ERRO:\n\n" + e, Toast.LENGTH_SHORT).show();
        }
        gravando = true;
        Toast.makeText(getApplicationContext(), "Gravando", Toast.LENGTH_SHORT).show();

    }
});
```

Figura 4. Método *onClick* vinculado ao botão Gravar

4.2.2.2. Enviar áudio da vocalização

Após o registro do áudio ter sido realizado, o usuário poderá clicar no botão Enviar, o qual levará à execução do código apresentado na Figura 5.

```
try {
    socket = new Socket(HOST, 8888);

    DataInputStream din = new DataInputStream(socket.getInputStream());
    DataOutputStream dout = new DataOutputStream(socket.getOutputStream());
    File file = new File(params[0]);
    FileInputStream fin = new FileInputStream(file);

    byte b[] = new byte[1024];
    int read;
    while ((read = fin.read(b)) != -1) {
        dout.write(b, 0, read);
        dout.flush();
    }
    fin.close();

    socket.shutdownOutput();
```

Figura 5. Conexão e envio do áudio ao servidor

No bloco *try* é criado o *socket* para a conexão com o servidor. Em seguida, são definidos os canais de entrada e saída de dados, especificados pelas variáveis *din* e *dout* e também são atribuídas às variáveis *file* e *fin* os dados respectivos do arquivo de áudio que foi gravado anteriormente.

Posteriormente, é realizada a leitura e o envio dos dados do arquivo por meio das instruções definidas no escopo do laço de repetição *while*.

Por último, ao terminar o envio do áudio, a variável *fin* é fechada e o *socket* pausa o envio de dados para, posteriormente, tratar o recebimento da resposta.

4.2.2.3. Armazenar arquivo

Nesta subseção iniciam-se os procedimentos respectivos ao servidor da aplicação. Primeiramente, é definido o nome do arquivo que será salvo no servidor utilizando o

endereço *IP* do cliente e a data em que o arquivo foi recebido. Ao final da função *receive_file* o nome do arquivo é retornado ao *script* principal.

Aqui também é empregado o método *open* que possibilita a criação do arquivo com o nome definido anteriormente. Em seguida, por meio do comando *while* são recebidos os *bytes* do arquivo enviado pela aplicação cliente, que são escritos no arquivo recém criado por meio do método *write*. O comando *with* possibilita um controle sobre a manipulação de arquivos, incluindo tratamento de erros e o fechamento do arquivo. A Figura 6 ilustra os processos aqui descritos.

```
def receive_file(connection, client_address):
    timestamp = datetime.now()
    date_time = timestamp.strftime("%Y-%m-%d-%H-%M-%S")
    file_path = f"{client_address[0]}_{date_time}.3gp"

    with open(file_path, 'wb') as file_received:
        while True:
            bytes_read = connection.recv(4096)
            if not bytes_read:
                break
            print('Receiving data from client...')
            file_received.write(bytes_read)

    print('\nFile received')
    return file_path
```

Figura 6. Definição do nome do arquivo e armazenamento do áudio no servidor

4.2.2.4. Efetuar pré-processamento

O áudio recebido é convertido para o formato *wav*, utilizando a biblioteca multimídia *ffmpeg*, para que possa ser processado pela biblioteca *pyAudioAnalysis*. A Figura 7 exibe o código dessa funcionalidade.

```
def convert_audio(input_file_path):
    output_file_path = f"{input_file_path[:-4]}.wav"
    ffmpegString = f"ffmpeg -loglevel error -i {input_file_path} {output_file_path}"
    print('\nConverting audio file to wav format...')
    os.system(ffmpegString)
    print('Done converting')
    return output_file_path
```

Figura 7. Preparação do arquivo para as etapas posteriores

4.2.2.5. Extrair atributos

A partir desta etapa dá-se início ao processo de classificação da gravação feita pelo usuário. Na Figura 8 é mostrada a função *features_and_train* que define os diretórios com os dados de treinamento. Com a chamada da função *extract_features_and_train* da biblioteca *pyAudioAnalysis*, é possível extrair as características dos áudios já armazenados no servidor e, posteriormente, treinar um modelo de classificação utilizando o tipo de classificador e nome do modelo recebidos pelos parâmetros da função. A partir dos trabalhos relacionados apresentados no presente trabalho, o algoritmo de classificação escolhido foi o *Support Vector Machines (SVM)*.

```
def features_and_train(classifier_type, model_name):

    directory_1 = f"trainingData{os.sep}pitangus-sulphuratus"
    directory_2 = f"trainingData{os.sep}turdus-rufiventris"

    print('\nInitializing the analysis process...\n')
    aT.extract_features_and_train([directory_1, directory_2], 1.0, 1.0,
    aT.shortTermWindow, aT.shortTermStep, classifier_type, model_name)
```

Figura 8. Função responsável pela extração das características e treinamento do modelo de classificação

4.2.2.6. Realizar a classificação

Logo após ter sido criado o modelo de treinamento, o arquivo contendo a gravação a ser classificada é entregue à função *classify*, juntamente com o tipo de classificador e o nome do modelo. Essas variáveis são então passadas à função *file_classification*, onde é feita a comparação do áudio a ser classificado com o modelo já treinado. Essa função retorna as duas espécies (*Pitangus Sulphuratus* e *Turdus Rufiventris*) utilizadas na comparação e a probabilidade em que cada uma se assemelha com o áudio classificado.

Posteriormente, é feita uma comparação, como se pode observar no bloco *if/else*, onde é selecionada a espécie que tem a maior probabilidade de estar classificada corretamente. O nome da espécie, juntamente com a probabilidade de acerto são retornados ao *script* principal da aplicação. Antes do sistema seguir para os próximos passos, o arquivo de áudio analisado e já classificado é movido para o diretório da espécie respectiva, a fim de ser utilizado como dado de treino para futuras execuções da aplicação. Esse passo a passo é demonstrado na Figura 9.

```
def classify(file_path, classifier_type, model_name):
    class_id, probabilities, classes = aT.file_classification
    (file_path, model_name, classifier_type)

    class_1 = classes[0]
    class_2 = classes[1]

    probability_class_1 = format(probabilities[0] * 100, '.2f')
    probability_class_2 = format(probabilities[1] * 100, '.2f')

    print(f"\n\n{class_1} = {probability_class_1}%")
    print(f"{class_2} = {probability_class_2}%")

    if probability_class_1 >= probability_class_2:
        return class_1, probability_class_1
    else:
        return class_2, probability_class_2
```

Figura 9. Classificação do áudio com base no modelo criado na etapa anterior

4.2.2.7. Retornar resposta

Antes de retornar o resultado, é chamada a função *common_name* passando como parâmetro o nome científico da ave classificada e obtendo como retorno o seu nome

comum. Assim sendo possível enviar a resposta à aplicação móvel, como ilustrado na Figura 10. Nota-se a utilização de uma expressão para separar o nome da espécie da probabilidade de acerto, para que a mensagem possa ser interpretada corretamente na aplicação cliente.

```
def common_name(scientific_name):

    species_names = {
        "pitangus-sulphuratus" : "Bem-te-vi",
        "turdus-rufiventris" : "Sabia-laranjeira"
    }

    return species_names[scientific_name]
```

```
def send_result(connection, common_name, probability):
    connection.send(common_name.encode('utf-8'))
    connection.send('<SEPARATOR>'.encode('utf-8'))
    connection.send(probability.encode('utf-8'))
```

Figura 10. A resposta é enviada à aplicação por meio da função *send_result*

4.2.2.8. Exibir resposta

Por fim, esta subseção descreve a etapa final do sistema, onde a mensagem com a resposta é recebida. Primeiramente, a mensagem é separada adequadamente com o emprego do método *split*, juntamente com a expressão que foi utilizada pelo servidor. Logo após, é montada a mensagem de resposta e armazenada na variável *aux*. São fechados os *streams* de entrada(*din*) e saída(*dout*) de dados, juntamente com o *socket* de conexão com o servidor.

Se não ocorrer nenhum erro nesta etapa, o que resultaria na execução do bloco *catch*, é executado o comando *return aux*. A variável *aux*, que contém a mensagem de resposta, é recebida como parâmetro no método *onPostExecute* e, no comando *textView.setText(result)*, o resultado é, finalmente, exibido ao usuário.

A Figura 11 (Apêndice D) ilustra a última etapa do sistema, a recepção e exibição da mensagem contendo o resultado. No Apêndice E é apresentada a tela da aplicação móvel, exibindo o resultado de uma classificação feita pelo sistema.

5. Resultados

Nesta seção é apresentado um panorama acerca dos resultados obtidos com a realização do trabalho em questão.

Os arquivos de áudio enviados pela aplicação móvel foram submetidos à classificação perante duas classes, ou seja, duas espécies de aves. O resultado da classificação é apresentado em porcentagem que, durante o trabalho foi descrita como a probabilidade de um determinado áudio pertencer a uma espécie de ave. Na Tabela 1 são apresentados os resultados da análise de dez vocalizações.

Tabela 1. Resultado da aplicação para a análise de dez áudios

Classificada como Espécie testada	Pitangus sulphuratus	Turdus rufiventris
Pitangus sulphuratus	**50,63%**	49,37%
Pitangus sulphuratus	26,36%	73,64%
Turdus rufiventris	56,11%	43,89%
Pitangus sulphuratus	39,70%	60,30%
Turdus rufiventris	51,41%	48,59%
Turdus rufiventris	83,43%	16,57%
Pitangus sulphuratus	37,99%	62,01%
Turdus rufiventris	42,77%	**57,23%**
Turdus rufiventris	59,43%	40,57%
Pitangus sulphuratus	**80,96%**	19,04%

Pitangus sulphuratus e *Turdus rufiventris* são, respectivamente, os nomes científicos das espécies bem-te-vi e sabiá-laranjeira. Pode-se constatar que apenas três resultados, aqueles que estão em destaque, foram classificados adequadamente. Aproximadamente, a metade das verificações apresentou probabilidades equilibradas. Nas classificações que culminaram no resultado esperado, apenas uma apresentou uma taxa de acerto elevada.

O desenvolvimento deste trabalho ocorreu durante um período de pandemia e isolamento social. O que impossibilitou a realização de uma pesquisa de campo, onde haveria a oportunidade de captação da vocalização das aves em seu habitat, criando um cenário mais aproximado de uma aplicação real deste projeto. Devido a isso, foram utilizados arquivos de áudio baixados da *Internet* e reproduzidos por meio de um sistema de som.

Acredita-se que este cenário adverso possa ter contribuído para a obtenção de resultados inconsistentes. Ademais, uma classificação errônea, pode acarretar em mais resultados inesperados, já que os arquivos de áudio classificados servem como dados de treinamento para futuras análises.

6. Conclusão

Este trabalho foi desenvolvido com o intuito de implementar uma ferramenta capaz de auxiliar o processo de reconhecimento de espécies de aves por meio da análise de suas vocalizações. A utilização do sistema traz, antes de tudo, uma alternativa para obtenção de resultados em tempo de execução.

As pesquisas do referencial teórico e dos trabalhos relacionados forneceram recursos importantíssimos para que o projeto e, posteriormente, a implementação deste trabalho pudessem ser concretizados.

No que se refere à metodologia *FDD*, a qual foi empregada como base para a concepção e implementação do sistema, pode-se acrescentar que foi uma escolha adequada devido as suas características corresponderem com os requisitos da aplicação.

Guiado por uma perspectiva crítica, pode-se salientar que o presente trabalho desenvolveu ferramentas de grande valia para a pesquisa relacionada a análise de sons. Principalmente, no que tange a arquitetura cliente-servidor que foi implementada, o que possibilita manter uma base para que se possa avaliar o que prejudicou a obtenção dos resultados esperados.

Os dados obtidos como resultado mostram, claramente, que os procedimentos de análise e classificação devem ser revisitados, a fim de buscar uma solução apropriada para esse tipo de problema.

Como trabalhos futuros, pode-se levar em consideração uma abordagem empregando outros tipos de classificadores mencionados durante o projeto deste trabalho, como por exemplo: *Random Forest* e *k-Nearest Neighbors(kNN)*. Esse método possibilitaria realizar comparações entre os resultados obtidos com a utilização de cada classificador. Um outro campo a ser explorado trata-se da utilização de mais espécies nos testes e treinamentos, possibilitando uma gama maior de dados a serem analisados.

Referências

Batista, F. M. M. (2015). Classificação de sons urbanos usando motifs e MFCC. *Dissertação de Mestrado, Instituto Superior de Engenharia do Porto, Portugal.*

Borges, L. E. (2014). *Python para Desenvolvedores.* Novatec Editora Ltda.

Carvalho, F. A. e Machado, P. C. M. (2011). Pré-processamento de sons para reconhecimento automático de pássaros. *UFG, Goiânia, Escola de Engenharia Elétrica e de Computação.*

Catchpole, C. K. e Slater, P. J. B. (2008). *Bird song: Biological themes and variations.* Cambridge University Press.

Cruz, D. d. C. V. (2016). Plataforma para recolha e identificação automática de espécies de pássaros utilizando registos áudio. *Dissertação de Mestrado, Instituto Superior de Engenharia do Porto, Portugal.*

Dawkins, M. S. (1989). *Explicando o comportamento animal.* Manole Ltda. São Paulo.

Deitel, H., Deitel, P. e Deitel, A. (2015). *Android: Como Programar.* Bookman Editora.

Giannakopoulos, T. (2015). pyAudioAnalysis: An open-source python library for audio signal analysis. *PloS one*, 10(12):1–17.

Giannoulis, D., Benetos, E., Stowell, D., Rossignol, M., Lagrange, M. e Plumbley, M. D. (2013). Detection and classification of acoustic scenes and events: An IEEE AASP Challenge. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE.

Gunasekaran, S. e Revathy, K. (2011). Automatic recognition and retrieval of wild animal vocalizations. *International Journal of Computer Theory and Engineering*, 3(1):136.

Han, J., Pei, J. e Kamber, M. (2011). *Data Mining: Concepts and Techniques.* Elsevier.

Highsmith, J. (2002). *Agile software development ecosystems.* Addison-Wesley Professional.

Márquez, R., de la Riva, I., Gil, D., Sueur, J., Marques, P. A., Genique, D. L., Eekhout, X., Valle, M. P., Ortiz, L. G., Solís, G., et al. (2011). Los sonidos de los animales: uma firma de su identidad. *Quercus.*

Mendes, D. R. (2011). Reconhecimento de orador em dois segundos. *Dissertação de Mestrado, Faculdade de Engenharia da Universidade de Porto, Portugal.*

Petry, A., Zanuz, A. e Barone, D. A. C. (1999). Utilização de técnicas de processamento digital de sinais para a identificação automática de pessoas pela voz. *Simpósio sobre Segurança em Informática, São José dos Campos*, SP.

Sick, H. (2001). *Ornitologia Brasileira.* Nova Fronteira, Rio de Janeiro.

Sick, H. e Barruel, P. (1984). *Ornitologia Brasileira*, volume 1. Editora Universidade de Brasília.

Silva, M. L. (2001). Estrutura e organização de sinais de comunicação complexos: o caso do sabiá-laranjeira turdus rufiventris (aves, passeriformes, turdinae). *Tese de Doutorado*, USP, São Paulo.

Silva, M. L. e Vielliard, J. (2011). *A aprendizagem vocal em aves: evidências comportamentais e neurobiológicas.* Editora da UFPA, Belém.

Stastny, J., Munk, M. e Juranek, L. (2018). Automatic bird species recognition based on birds vocalization. *EURASIP Journal on Audio, Speech, and Music Processing*, 2018(1):1–7.

Tolentino, V. C. d. M. (2015). Repertório vocal e variações no canto de aves em diferentes áreas florestais no cerrado sensu lato. Dissertação de Mestrado, UFU, Uberlândia.

Tsai, W. H. e Xue, Y. Z. (2014). On the use of speech recognition techniques to identify bird species. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 19, Number 1, March 2014.*

Vielliard, J. M. (1987). O uso da bioacústica na observação de aves. *II Encontro nac. Anilhad. Aves.*

Vielliard, J. M. (2004). A diversidade de sinais e sistemas de comunicação sonora na fauna brasileira. *I Seminário Música Ciência Tecnologia.*

Apêndice A

Tabela 2. Requisitos Funcionais do sistema

Requisitos Funcionais (RF)		
Funcionalidade	**Descrição**	**Complexidade**
RF01: Gravar vocalização	A aplicação móvel permitirá a gravação do áudio da vocalização de uma ave.	Baixa
RF02: Enviar áudio da vocalização	A aplicação móvel possibilitará o envio do arquivo ao servidor *web*.	Média
RF03: Armazenar arquivo	O servidor, ao receber a conexão da aplicação móvel, deverá salvar o arquivo de áudio.	Baixa
RF04: Efetuar pré-processamento	Realizar as tarefas de pré-processamento utilizando os recursos da biblioteca *pyAudioAnalysis*.	Média
RF05: Extrair atributos	Obter as características pertinentes a partir do arquivo de áudio pré-processado.	Média
RF06: Realizar classificação	Realizar os procedimentos de classificação, a fim de classificar o áudio como uma possível espécie de ave.	Média
RF07: Retornar resposta	Enviar à aplicação móvel, os resultados obtidos na classificação da espécie.	Baixa
RF08: Exibir resposta	A informação recebida do servidor será exibida como resposta ao usuário da aplicação móvel.	Baixa

Apêndice B

Tabela 3. Requisitos Não Funcionais do sistema

Requisitos Não Funcionais (RNF)
RNF01: Implementação do servidor com a linguagem de programação Python.
RNF02: Desenvolvimento da aplicação móvel por meio da plataforma *Android* e a linguagem de programação Java.
RNF03: Utilização da biblioteca *pyAudioAnalysis* nas tarefas referentes à manipulação de áudio.

Apêndice C

Tabela 4. Planejamento por funcionalidade

Requisitos Funcionais (RF)	
Funcionalidade	**Duração**
RF01: Gravar vocalização	05 dias
RF02: Enviar áudio da vocalização	10 dias
RF03: Armazenar arquivo	05 dias
RF04: Efetuar pré-processamento	15 dias
RF05: Extrair atributos	15 dias
RF06: Realizar a classificação	20 dias
RF07: Retornar resposta	05 dias
RF08: Exibir resposta	05 dias

Apêndice D

```
            response = din.readLine();

            responseSplit = response.split("<SEPARATOR>");
            specie = responseSplit[0];
            probability = responseSplit[1];

            aux = "Espécie: " + specie + "\nProbabilidade: " + probability + "%";

            if (din != null) din.close();
            if (dout != null) dout.close();
            if (socket != null) socket.close();

        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Erro: " + e.getMessage());
        }
        return aux;
    }

    @Override
    protected void onPostExecute(String result) {
        Toast.makeText(getApplicationContext(), "\n\n  Finalizado  \n\n", Toast.LENGTH_LONG).show();
        progressBar.setVisibility(View.GONE);
        textView.setText(result);
    }
```

Figura 11. A aplicação móvel recebe a mensagem de resposta e a exibe na tela.

Apêndice E

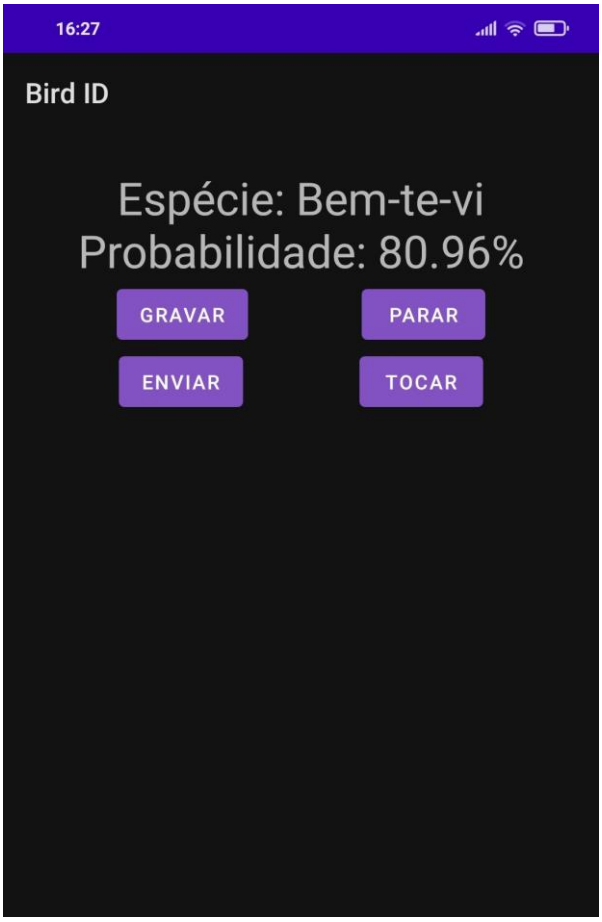

Figura 12. Interface da aplicação móvel exibindo o resultado de uma classificação.

Anexo A

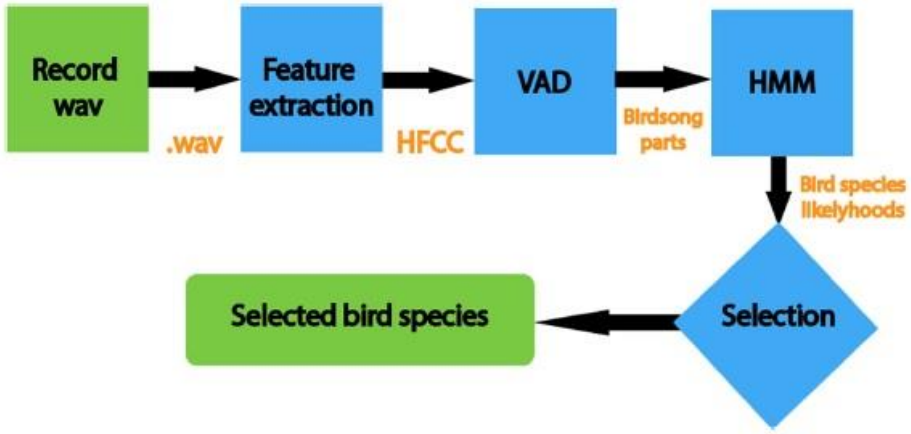

Figura 13. Módulos para o reconhecimento de espécies de aves [Stastny et al. 2018].

Anexo B

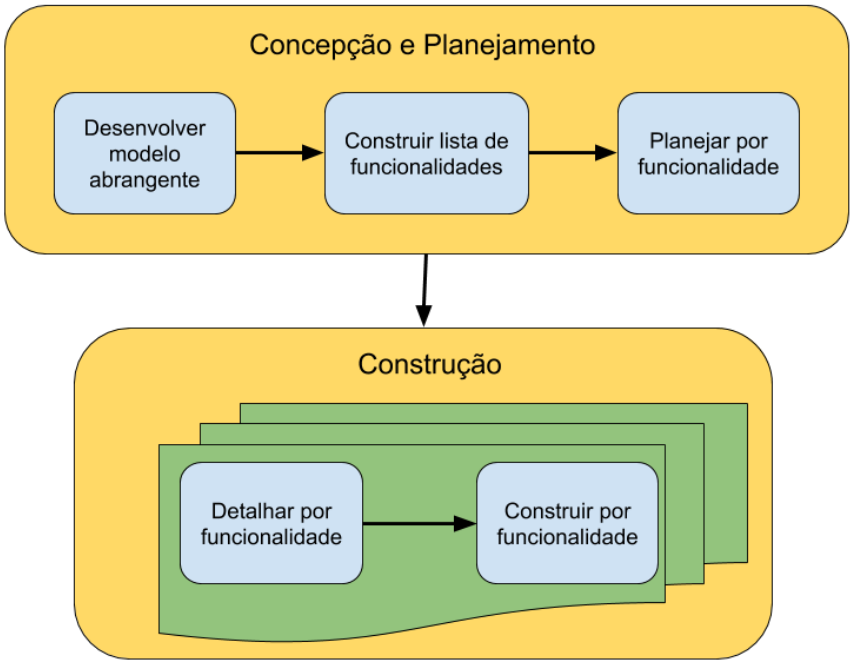

Figura 14. Fases da metodologia *FDD*.