UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
Igor Graziano Brito Ferreira

APLICAÇÃO MÓVEL PARA CATEGORIZAÇÃO E MAPEAMENTO DE ESPÉCIES EM UNIDADE DE CONSERVAÇÃO AMBIENTAL

Diamantina
2019

Igor Graziano Brito Ferreira

APLICAÇÃO MÓVEL PARA CATEGORIZAÇÃO E MAPEAMENTO DE ESPÉCIES EM UNIDADE DE CONSERVAÇÃO AMBIENTAL

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas da Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do título de Bacharel.

Orientador: Prof. Dr. Alessandro Vivas Andrade

Diamantina

2019

Igor Graziano Brito Ferreira

APLICAÇÃO MÓVEL PARA CATEGORIZAÇÃO E MAPEAMENTO DE ESPÉCIES EM UNIDADE DE CONSERVAÇÃO AMBIENTAL

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas da Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do título de Bacharel.

Orientador: Prof. Dr. Alessandro Vivas Andrade

Data de aprovação _____/_____/_____.

_____
Prof. Dr. Alessandro Vivas Andrade

_____
Profa Dra Luciana Pereira de Assis

_____
Profa Dra Josiane Teixeira

Diamantina

RESUMO

O presente trabalho apresenta o desenvolvimento de uma aplicação móvel para o sistema operacional Android. Tal aplicação tem como o seu propósito fornecer uma solução tecnológica que venha sanar as dificuldades vivenciadas pelos pesquisadores do Parque Nacional das Sempre-Vivas no exercício das suas atividades de campo. A aplicação faz uso de ferramentas disponíveis na maioria dos dispositivos móveis modernos, como câmera fotográfica digital e GPS, para elevar a produtividade dos pesquisadores do parque. Durante o desenvolvimento deste trabalho ficou claro que o domínio de técnicas de programação que pudessem ser aplicadas em diferentes linguagens era imprescindível.

Palavras-chave: Android; Java; XML; JSON; Dispositivos Móveis; Telecomunicações.

ABSTRACT

This paper comprehends the development of a mobile application for the Android operating system, this application aims to provider a technological solution that could be user to sort the issues faced by the researchers working on Parque Nacional das Sempre-Vivas when carrying out their field activities. The application makes use of features available in most modern mobile devices, such as digital camera and GPS, to increase the productivity of those researchers. During the buildout of this paper it was very clear that the knowledge of programming techniques that could be applied to different languages was crucial.

Keywords: Android; Java; XML; JSON; Mobile Devices; Telecommunications.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

API	Application Programming Interface
ART	Android Runtime
CRUD	Create, Read, Update and Delete
EUA	Estados Unidos da América
GMS	Google Mobile Services
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HAL	Hardware Abstraction Layer
JSON	Javascript Object Notation
PARNA	Parque Nacional
PDF	Portable Document Format
SDK	Software Development Kit
UML	Unified Modeling Language
URSS	União das Repúblicas Socialistas Soviéticas
XML	Extensive Markup Language

SUMÁRIO

INTRODUÇÃO

É notório que as aplicações móveis desempenham um papel importante nas atividades diárias da sociedade moderna. Este segmento vem dando indícios de que o crescimento continua de forma progressiva, alcançando a marca de 30,3 bilhões de novas aquisições de aplicativos ainda no segundo trimestre de 2019, gerando um consumo de 22,6 bilhões de dólares, um crescimento de 20% quando comparado ao mesmo período de 2018 (SYDOW, 2019). Além disso, a indústria de dispositivos móveis exerce um papel importante no desenvolvimento deste mercado. Com a popularização do setor, tem se tornado cada vez mais fácil a aquisição de aparelhos de elevado poder computacional que contam com funcionalidades como GPS (Global Positioning System) e câmera fotográfica (DUDLEY, 2018).

Com a evolução da indústria e da plataforma em si, a aplicação de soluções móveis se mostra não somente conveniente, mas também necessária. De acordo com uma pesquisa realizada pela Samsung, o uso de dispositivos móveis pode gerar uma economia de até uma hora por dia no desempenho de atividades profissionais, alcançando um aumento de até 34% em produtividade (TUREK, 2016).

Embora o setor de aplicações tenha se desenvolvido de maneira expressiva nos últimos anos, o Brasil ainda apresenta problemas de infraestrutura, o que dificulta o investimento e limita sua expansão (KFRAFUNI, 2016). Tal limitação pode ser observada nas atividades de trabalhadores e pesquisadores de áreas rurais e florestais como o Parque Nacional das Sempre-Vivas (PARNA), uma unidade de preservação ambiental localizada no estado de Minas Gerais (BRASIL, 2002).

Nesse contexto, este trabalho busca o desenvolvimento de uma aplicação móvel que auxilie na execução das tarefas realizadas pelos pesquisadores do PARNA, aplicando os benefícios de portabilidade e funcionalidades oferecidos pelos aparelhos digitais modernos para superar as dificuldades impostas pela localização remota e a extensa área do parque.

Justificativa

Criado em dezembro de 2002, o Parque Nacional das Sempre-Vivas é uma área de preservação ambiental que cobre uma área total de aproximadamente cento e vinte e quatro mil hectares (Figura 1) e conta com matas densas que abrigam uma vasta população vegetal. Localizado nos Municípios de Olhos d’Água, Bocaiúva, Buenópolis e Diamantina, no Estado

de Minas Gerais, o PARNA tem como objetivo assegurar a preservação dos recursos naturais e da diversidade biológica, bem como proporcionar a realização de pesquisas científicas e o desenvolvimento de atividades de educação, de recreação e turismo ecológico (BRASIL, 2002). Dentre as atividades exercidas pelos pesquisadores que trabalham no parque, está a realização de diário de campo, catalogando as espécies vegetais existentes no parque. A tecnologia envolvida em diversas aplicações tem o potencial de simplificar e agilizar o trabalho, automatizando e reduzindo o tempo gasto nessas atividades. Entretanto, a falta de cobertura de redes móveis na área do parque, bem como a ausência de aplicações disponíveis no mercado que ofereçam funcionalidades úteis nas atividades no parque, leva os pesquisadores a realizá-las manualmente, com a utilização de câmeras e anotações em papel.

Figura 1: Área do parque

Fonte: Google Maps (https://maps.google.com)

Objetivo

O objetivo deste trabalho é desenvolver uma aplicação para o sistema operacional Android que auxilie os pesquisadores do Parque Nacional das Sempre-Vivas no exercício de suas atividades, automatizando os procedimentos através de uma interface usual.

Estrutura do Trabalho

No Capítulo 1 será feito um levantamento do histórico do sistema operacional Android e da tecnologia por trás da evolução do seguimento, além de um breve detalhamento da sua arquitetura e de como as aplicações operam dentro dele. O Capítulo 2 traz uma introdução das ferramentas utilizadas no desenvolvimento de uma aplicação Android, explicando seu surgimento e qual o papel que cada uma delas desempenha no desenvolvimento. Já os requisitos da aplicação, são demonstrados no Capítulo 3, explicando como estes foram coletados e ilustrando, através de diagramas, como eles foram interpretados em ações e componentes da aplicação. O detalhamento da parte estrutural relevante à aplicação é descrito no Capítulo 4. Já o Capítulo 5, o último deste trabalho, detalha a implementação da aplicação da qual trata este Trabalho de Conclusão de Curso.

CAPÍTULO 1 - TELECOMUNICAÇÕES E SISTEMA OPERACIONAL ANDROID

Esse capítulo apresentará um breve histórico e algumas características chave para a compreensão das tecnologias e plataformas utilizadas para o desenvolvimento do trabalho.

1.1 Histórico das Telecomunicações

É difícil traçar com precisão uma história dos meios de comunicação, estabelecendo-se uma cronologia exata, e mais difícil ainda precisar o surgimento e evolução da comunicação da espécie humana. Contudo, vários estudos foram e continuam sendo realizados nessa área, revelando importantes fatos da história da comunicação humana. Fato é que, ao longo do tempo, a qualidade e a funcionalidade dos veículos de comunicação foram sendo aprimoradas para atender as necessidades crescentes de se comunicar (IPANEMA, 1967). A partir da década de 1960, em um contexto de guerra fria e da corrida armamentista, espacial e tecnológica entre EUA (Estados Unidos da América) e URSS (União das Repúblicas Socialistas Soviéticas), que finalmente existem as condições para o que mais tarde veio a ser chamado de revolução digital, marcada pelo surgimento dos primeiros computadores. Progressivamente, a evolução tecnológica da telecomunicação permitiu a passagem de mídia em massa (Televisão, rádio, imprensa e cinema) para formas individualizadas de produção, armazenamento e distribuição de informação (LEMOS, 2013).

A telecomunicação móvel teve início nos anos 70, sendo, por cerca de duas décadas, capaz de transmitir apenas sinais de voz. Com a implementação da tecnologia GSM (Global System for Mobile Communications), que permitia o envio de sinais digitais, tornaram-se possível funcionalidades como mensagens de texto e até mesmo conexão com a internet. Segundo Bunker (2013), foi a tecnologia GSM que abriu portas para a evolução das redes no que diz respeito a transferência de dados, trazendo a implementação de um padrão internacional de redes móveis que permitia um aumento na largura de banda, o que alterou completamente o mercado de dispositivos móveis. (BUNKER, 2013).

O primeiro aparelho celular comercial do mundo, O DynaTAC 8000X, surgiria apenas em 1983. Após décadas de estudos e pesquisas, o barateamento de microprocessadores e a digitalização das linhas de comunicação das redes de telefonia permitiram, aos poucos, a sua massificação (MANTOVANI, 2005).

Figura 2: Motorola DynaTAC 8000X

Fonte: http://content.time.com/time/specials/packages/article/0,28804,2023689_2023708_023656;00:html

Finalmente, com a difusão da Internet, surge uma nova forma de comunicação interativa caracterizada pela capacidade de enviar mensagens de muitos para muitos, em tempo real ou não.

O mercado de telefonia móvel alcançou um ponto de inflexão na década de 90, quando ambos os dispositivos móveis e as tecnologias de rede convergiram na direção do que seria o futuro do segmento (RANDY, 2008). Desde então, ele vem mostrando uma evolução tecnológica progressiva, produzindo aparelhos que podem oferecer serviços que ultrapassam as capacidades de computadores convencionais (DYROFF, 2018).

As novidades tecnológicas progridem rapidamente e o futuro da computação móvel está se tornando cada vez mais promissor. Os dispositivos móveis estão cada vez mais eficientes, especialmente com a chegada dos Smartphones, que possuem recursos inteligentes integrados. Um dos objetivos é melhorar a mobilidade e usabilidade sem perder o desempenho, pois esses aparelhos vêm adquirindo mais capacidade de processamento e de armazenamento (LEE, 2005).

1.2 O sistema operacional Android

Segundo Tanenbaum (2008), um sistema operacional é um software que tem como tarefa controlar todos os recursos de um computador, oferecendo uma base para outras

aplicações. Um dos sistemas operacionais mais usados no mundo é o Android, sistema operacional móvel da Google.

A origem do Android ocorreu devido ao crescimento da demanda no mercado de dispositivos móveis. Várias empresas de telefonia formaram uma aliança com o objetivo de estabelecer uma plataforma aberta para o desenvolvimento de aplicações coorporativas, o que deu origem ao Android (LECHETA, 2010).

Android, Inc. foi fundada em outubro de 2003 e adquirida pela Google em 2005. Em 2007, o Android foi lançado: uma plataforma móvel implementada com base na versão 2.6 do *Kernel* Linux, um software de código livre responsável por gerenciar os recursos físicos de um computador (LECHETA, 2010). Ele detecta os componentes de *hardware* de um computador e prepara os módulos necessários para o seu uso (NEGUS, 2015).

O primeiro smartphone disponível comercialmente rodando o Android foi o HTC Dream, lançado em 22 de outubro de 2008 (Figura 3).

Figura 3: HTC Dream o primeiro smartphone Android

Fonte: https://www.notebookcheck.net/Time-to-celebrate-Google-Android-is-now-10-years-old.334276.0.html

A arquitetura do sistema operacional Android é formada por conjunto de componentes de *software*, que são dispostos em cinco camadas: *kernel Linux*, *HAL (Hardware Abstraction Layer)*, *ART (Android Runtime)*, *Application Framework* e *Applications* (ANDROID, 201-?), descritas abaixo:

Kernel Linux é a fundação da plataforma Android. Esta camada é responsável por estabelecer comunicação e gerenciar componentes de *hardware*, além de oferecer uma série de dispositivos de segurança (ANDROID, 201-?).

A camada *HAL* define uma interface de comunicação padrão entre *hardware*, APIs (Application Programming Interface) e *Frameworks* Java, oferecendo várias bibliotecas que

implementam a abstração necessária para lidar com componentes de *hardware* específicos (ANDROID, 201-?).

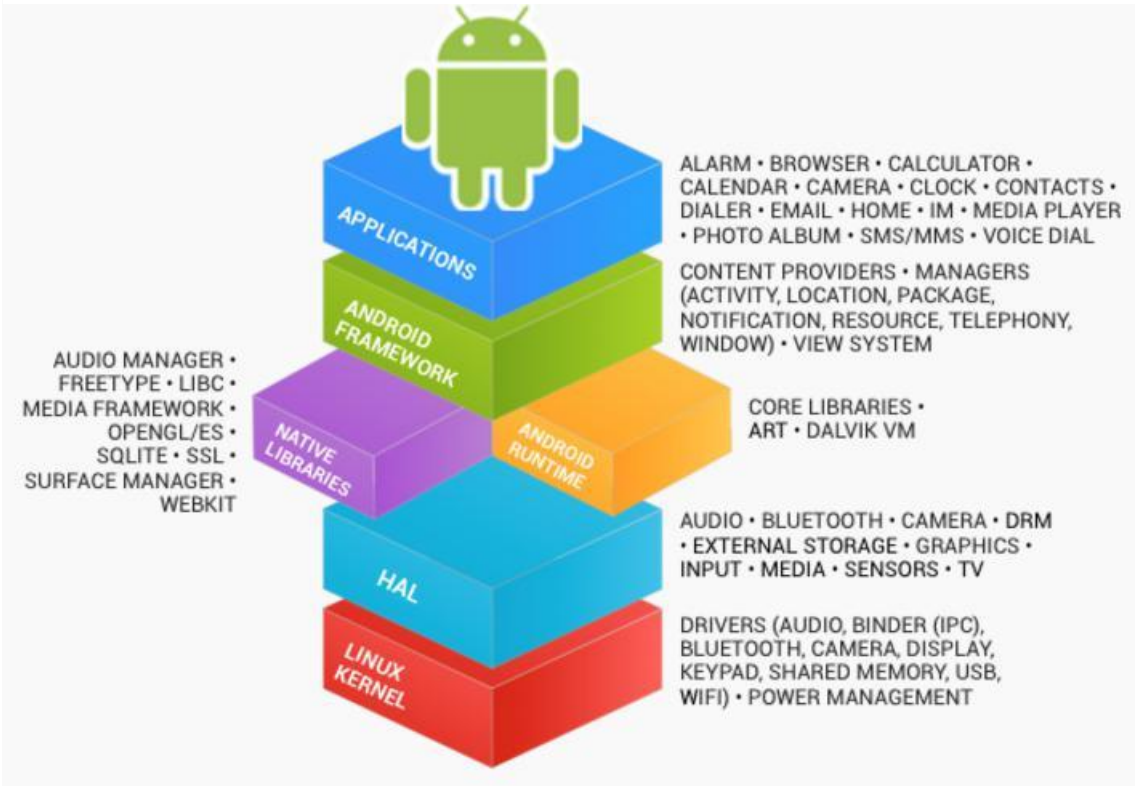
O componente ART é a camada responsável por executar múltiplas máquinas virtuais sob quais as aplicações serão executadas dentro do sistema operacional (ANDROID, 201-?). Segundo VMWARE (2011), “Uma máquina virtual é um computador emulado em software, tal como um computador físico, executando um sistema operacional e aplicações”. A ART introduziu o conceito compilação *ahead-of-time*, que de acordo com a Oracle (2012) é a execução de métodos pré-compilados pela máquina virtual reduzindo o tempo de inicialização das aplicações. Essa camada também fornece uma série de bibliotecas nativas que implementam a maioria das funcionalidades oferecidas pelo Java.

O Android Framework é um conjunto de APIs que fornecem meios de se fazer uso do conjunto de funcionalidades oferecidas pelo sistema operacional, criando blocos que simplificam a reutilização de serviços quando se desenvolve uma aplicação Android (ANDROID, 201-?).

Finalmente, Applications é a camada de mais alto nível dentro da estrutura do sistema, e é ela que engloba os aplicativos instalados nele, sejam estes fornecidos pelo pacote de aplicações nativas do Android ou não (ANDROID, 201-?).

Na Figura 4 é possível observar a disposição das camadas que representam a arquitetura do sistema operacional Android, além de alguns dos componentes contidos em cada camada.

Figura 4: Arquitetura do Android



Fonte: https://info448-s17.github.io/lecture-notes/introduction.html

No Android, aplicações são executadas num paradigma denominado *Sandbox*, que atribui um identificador de usuário Linux para cada aplicativo. Desta forma o sistema pode isolar o acesso aos arquivos de cada aplicativo, atribuindo permissões de acordo com seu identificador (Android, 201-?a).

De acordo com Mitchell, Oldham e Samuel (2001, p. 77) *threads* são unidades de execução criadas por processos para realizar uma tarefa, seja de forma concorrente ou sequencial. No Android, quando aplicações são executadas, uma *thread* principal é criada com o propósito de executar a aplicação em si. Existe também a possibilidade de se criar *threads* secundárias que executam tarefas em segundo plano (ANDROID, 201-?b).

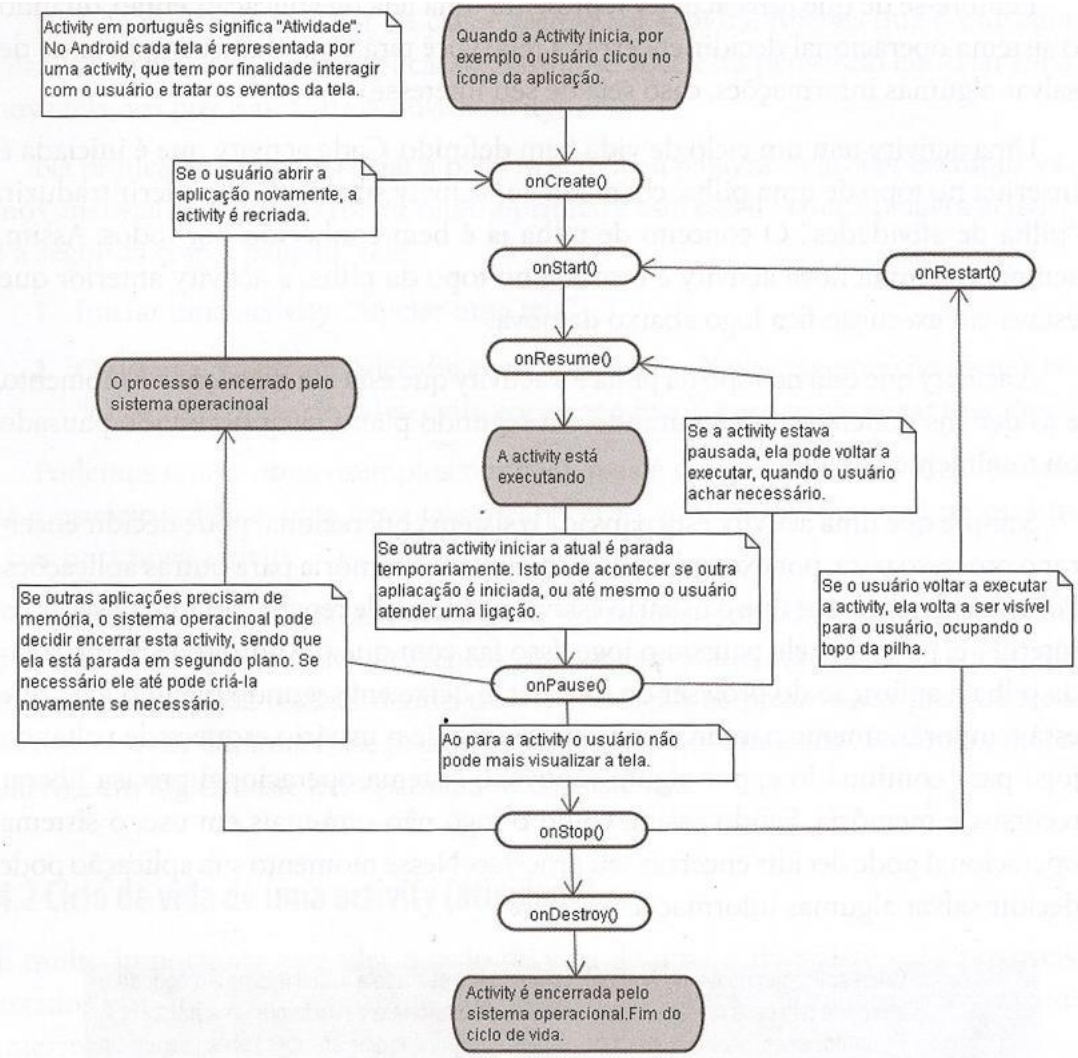
Uma *activity* representa uma tela de uma determinada aplicação, e trata todos os eventos ou mudanças de estados provenientes dela, como por exemplo um toque em um elemento disponível na tela. Ao considerar o Ciclo de vida de uma *activity*, primeiramente é importante saber que o sistema operacional mantém uma pilha de *activities*, gerenciando os estados assumidos por todas as *activities* inseridas na pilha. O Android Framework oferece métodos com o propósito de controlar o comportamento de uma *activity* durante o seu ciclo de vida

(LECHETA, 2010). Ainda segundo Lecheta (2010), os métodos que controlam os estados do ciclo de vida de uma *activity* são os seguintes:

- ***onCreate()***: método obrigatório que deve ser invocado somente uma vez, ele é responsável por inicializar uma aplicação e inserir uma v*iew* na tela. Uma v*iew* representa o layout definido para uma determinada tela;
- ***onStart()***: é invocado assim que uma *activity* está pronta para ser visualizada pelo usuário;
- ***onRestar*t()**: este método é invocado sempre que uma *activity* foi parada temporariamente e está sendo reiniciada;
- ***onResume()***: é chamado somente quando a *activity* se encontra no topo da pilha, ou seja, no momento em que ela está sendo executada e visível para o usuário;
- ***onPause()***: este método é executado caso ocorra algum evento que venha a causar a interrupção da aplicação que está sendo executada, parando a *activity* temporariamente e salvando o seu estado atual para que possa ser recuperado posteriormente;
- ***onStop()***: este método é invocado sempre que a *activity* está sendo encerrada para ceder o topo da pilha para uma outra aplicação. Depois de parada, caso necessário, uma *activity* pode ser reiniciada. Caso a *activity* permaneça em segundo plano por muito tempo o sistema operacional pode decidir por encerrá-la;
- ***onDestroy()***: sempre o último método dentro do ciclo de vida, ele encerra uma a*ctivity*, ou seja, a remove completamente da pilha.

Na Figura 5 é possível observar, por meio de um fluxograma, quando cada método está disponível e quais são seus comportamentos dentro do ciclo de vida de uma *activity*.

Figura 5: Ciclo de vida uma Activity



Fonte: LECHETA, 2010, p.96.

CAPÍTULO 2 - FERRAMENTAS

Neste capítulo serão descritas as linguagens e aplicações que dão suporte ao desenvolvimento de aplicações Android, descrevendo suas características e explicando qual o papel desempenhado por cada uma delas no escopo de uma aplicação.

2.1 Linguagens

De forma oficial, a Google indica que, os aplicativos Android podem ser escritos com as linguagens Kotlin, Java e C++ usando o Android SDK (Software Development Kit), enquanto o uso de outras linguagens também é possível. Esta seção descreve as linguagens utilizadas no desenvolvimento da aplicação proposta neste trabalho.

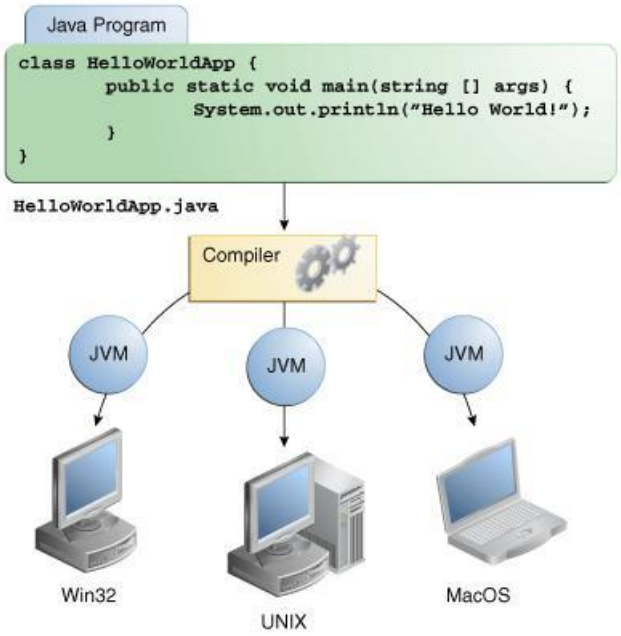
2.1.1 Java

De acordo com Deitel e Deitel (2002) o Java é uma linguagem de programação multiplataforma orientada a objetos. Sua capacidade multiplataforma é alcançada através do uso de máquinas virtuais. O código fonte Java é compilado em um *bytecode*, que por sua vez é interpretado em linguagem de máquina por uma máquina virtual.

> Linguagem de máquina é a ‘linguagem natural’ de um computador e é definida pelo seu projeto de hardware. As linguagens de máquina consistem geralmente em *strings* de números (em última instância reduzidas a 1s e 0s) que instruem os computadores a realizar suas operações mais elementares uma de cada vez. As linguagens de máquina são dependentes da máquina (**isto é, uma linguagem particular de máquina pode ser utilizada apenas em um tipo de computador**). (DEITEL; DEITEL, 2005, p. 5. Grifos nossos).

Cada máquina virtual tem uma implementação distinta para compilar o *bytecode* em linguagem de máquina, satisfazendo as especificidades de cada sistema operacional. A Figura 6 ilustra o processo de compilação do código Java em *bytecode* e a interpretação deste em linguagem de máquina.

Figura 6: Interpretação do código Java



Fonte: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

Na plataforma Android o Java é utilizado para realizar processamento de dados, interpretação de eventos e para interagir com os componentes do aparelho. A seguir há um exemplo de código Java utilizado no Android para carregar um layout.

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_item_detail);
}
```

2.1.2 XML

O XML (Extensive Markup Language) é uma linguagem de *markup* que permite a estruturação de dados arbitrários, tendo sido desenvolvida como um padrão de troca de informações entre dispositivos através da internet. Documentos XML são compostos por unidades de armazenamento, denominadas entidades, as quais podem conter um valor definitivo ou a uma subestrutura de entidades (BOTHA *et al*, 2012).

No Android o XML é utilizado para definir as estruturas que compõem as interfaces gráficas. A seguir é possível ver um trecho de código em XML utilizado na construção de uma interface gráfica para aplicação Android.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
        <TextView android:id="@+id/some_ext"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Random text" />
        <Button android:id="@+id/some_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Click me!" />
</LinearLayout>
```

2.1.3 JSON

> O JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é um formato leve de troca de dados. A escrita e leitura são muito fáceis para humanos, e a interpretação e geração triviais para máquinas. Ele é baseado em um subconjunto da linguagem de programação JavaScript. (JSON, ca. 2000).

O JSON é constituído por duas estruturas base, o objeto e vetor. O primeiro é denotado por um par de formato chave-valor, já o último representa um conjunto ordenado de valores (JSON, ca. 2000).

Assim como o XML o JSON permite a criação de estruturas de dados de forma flexível, no entanto, se tornou mais popular que ele devido ao seu tamanho reduzido e proximidade que tem com a sintaxe da linguagem JavaScript, que fazem com que ele seja facilmente interpretado por navegadores (FREEMAN, 2017).

É possível observar uma estrutura de dados em JSON no trecho de código a seguir.

```
{
        id : 10,
        first_name : "John",
        last_name : "Doe"
}
```

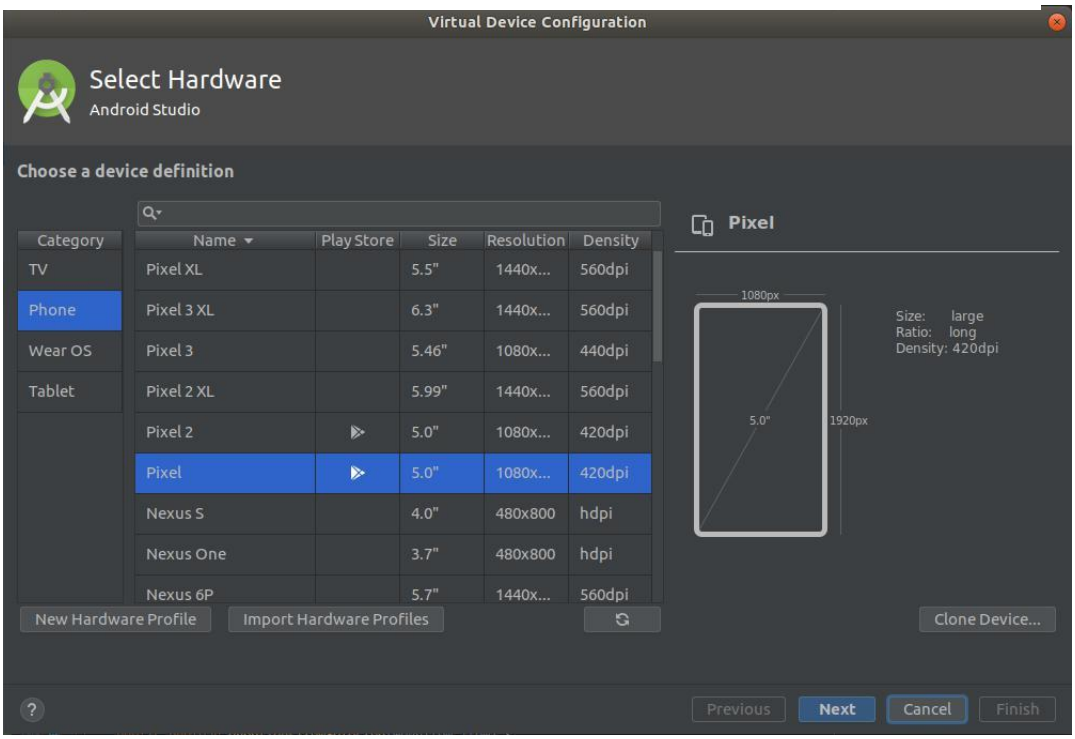
2.2 Android SDK

"O Android SDK é o software utilizado para desenvolver aplicações no Android, que tem um emulador para simular o celular, ferramentas utilitárias e uma API completa para a linguagem Java" (LECHETA, 2010, p.29).

O SDK é distribuído pela empresa Google e possui uma série de ferramentas que auxiliam no desenvolvimento de aplicações Android, desta forma todas as atividades necessárias podem ser realizadas dentro do escopo desta aplicação.

A Figura 7 ilustra a interface do emulador disponibilizado pelo SDK.

Figura 7: Interface gráfica do AVD Manager



Fonte: *Screenshot* feita pelo autor.

CAPÍTULO 3 – REQUISITOS DA APLICAÇÃO

Este capítulo especifica os requisitos do sistema e demonstra, por meio de diagramas UML (Unified Modeling Language), o planejamento realizado para dar apoio à fase de desenvolvimento da aplicação.

Para que fosse feito o levantamento destes requisitos foram realizados encontros periódicos com os *stakeholders*[1].

3.1 Levantamento de Requisitos

- O aplicativo deve possibilitar a catalogação de espécies encontradas no Parque Nacional das Sempre-Vivas;
- A catalogação deve conter:
    - foto do item catalogado;
    - título para o item;
    - descrição sobre o item;
    - geolocalização do item;
    - o exato momento que o item foi inserido;
- Após a catalogação o item deve ser adicionado à lista que representa o catálogo do trabalho de campo realizado pelo pesquisador;
- O pesquisador deve ter acesso aos itens já catalogados, podendo visualizar os dados que fazem referência a ele;
- O aplicativo deve permitir que o pesquisador edite os dados de descrição e título de um item catalogado;
- Ao fim do trabalho de campo, caso deseje, o pesquisador pode exportar o histórico de seu trabalho de campo para um formato portável e que não dependa do aplicativo em questão para que seja visualizado.

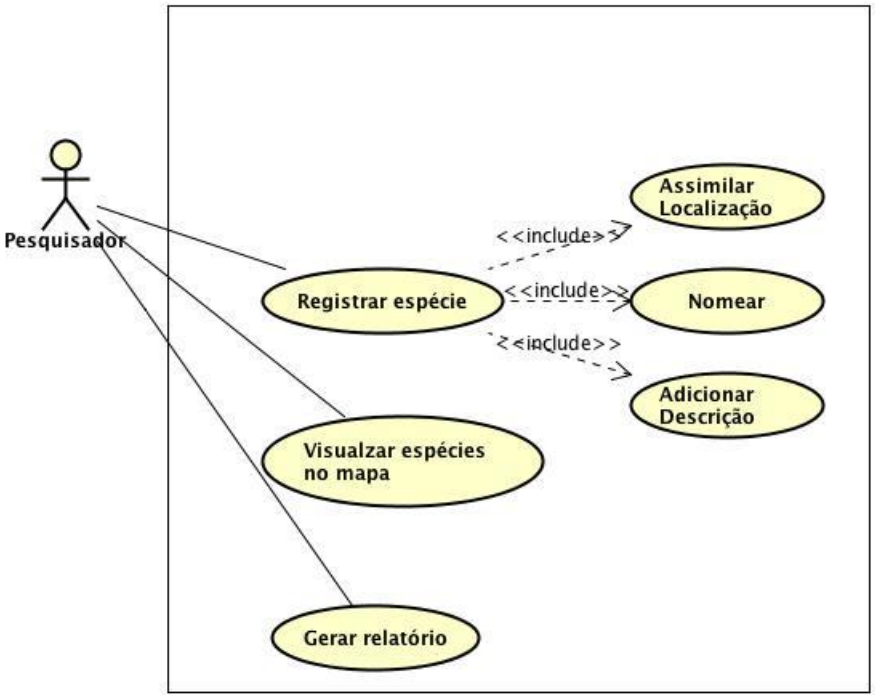
3.2 Diagrama de casos de uso

[1] Pessoa, grupo, ou organização diretamente envolvida em um projeto, e que pode tanto influenciar ou ser afetado pelo resultado deste (WIEGERS, 2003).

Harvey e Paul Deitel (2006) apontam que um diagrama de caso de uso precisa capturar o que um sistema deve fazer, identificando cada um dos casos de uso como uma funcionalidade diferente que o sistema oferece ao seu usuário final.

No caso da aplicação em questão, existe apenas um ator, que pode efetuar todos os casos de uso do sistema. A Figura 8 traz a representação da aplicação através do diagrama.

Figura 8: Diagrama de casos de uso



Fonte: Diagrama elaborado pelo autor

CAPÍTULO 4 - ESTRUTURA

Qualquer projeto de aplicação Android deve obedecer a uma estrutura com organização predefinida. Os componentes mais relevantes para o desenvolvimento dentro dessa estrutura estão listados a seguir.

4.1 Diretórios

4.1.1 Manifests

É o diretório onde se encontra o arquivo *AndroidManisfest.xml,* que segundo Lecheta (2010) é o componente base de uma aplicação Android, contendo a descrição de todas as suas configurações, pacotes e classes.

4.1.2 Java

É neste diretório que ficarão armazenadas todas as classes criadas num projeto de aplicativo Android. Como o próprio nome sugere todas as classes contidas nesta pasta são escritas em Java, dessa forma é natural que seja o diretório do projeto a ser mais utilizado.

4.1.3 Layout

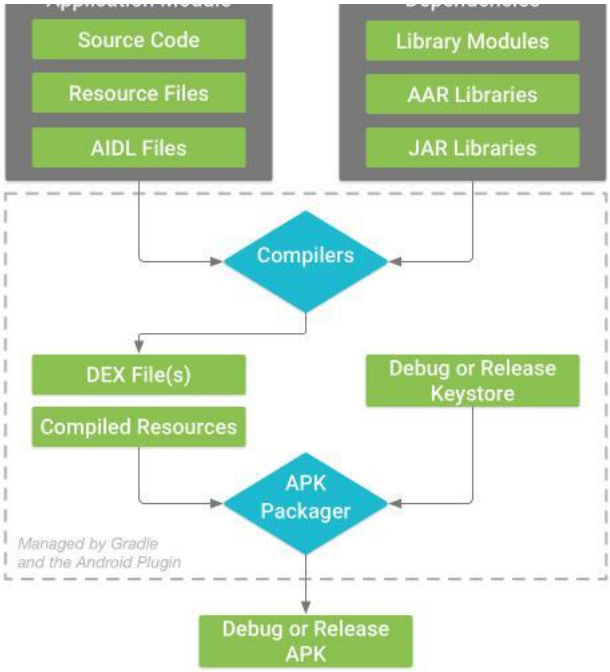
É onde estão localizados todos os *layouts* da aplicação. *Layouts* são arquivos XML que definem a estrutura da interface gráfica de uma *activity.*

4.2 Gradle

O *Gradle* é um gerenciador de dependências responsável por montar todos os módulos de uma aplicação Android. Ele auxilia na construção dos módulos e injeta as bibliotecas necessárias para cada um, compilando assim um pacote que será utilizado na construção do instalador do aplicativo (ANDROID, 201-?c).

A figura a seguir ilustra um fluxograma das operações de montagem realizadas pelo *Gradle*.

Figura 9: O processo de construção de um módulo Android



Fonte: https://developer.android.com/studio/build

CAPÍTULO 5 - IMPLEMENTAÇÃO

Este capítulo tem o propósito de descrever a funcionalidade do código implementado, o papel que cada elemento desempenha dentro da aplicação e como esses elementos interagem entre si no escopo da mesma.

5.1 Modelo de Dados

A aplicação faz uso da flexibilidade da notação JSON para emular a estrutura de uma tabela em um banco de dados. Cada objeto contém pares de dados que se assimilam aos atributos de uma entidade em um modelo de banco de dados relacional comum. Os atributos do modelo em questão são: *title*, *description*, *photoPath*, *lat*, *lng* e *created_at*.

Quando coletados os objetos são armazenados em um vetor, cada item do vetor representa uma linha em uma tabela de um banco de dados relacional comum. O trecho de código a seguir demonstra a estrutura do modelo de dados descrito.

```
[
        {
                title: "Cactus",
                description: "Lorem ipsum.",
                photoPath: "Pictures\/JPEG_20190702.jpg",
                lat: -18.2321,
                Lng :-43.6117,
                created_at: "02\/07\/2019 17:18"
        },
        {
                title: "Paper Leaf",
                description: "Just a leaf.",
                photoPath: "Pictures\/JPEG_20190702.jpg",
                lat: -18.2320,
                lng: -43.6116,
                created_at: "02\/07\/2019 17:18"
        }
]
```

5.2 Classes

Todas as classes implementadas neste projeto serão explicadas nesta seção, começando pela classe que representa a modelo de dados, passando pelas classes que auxiliam em operações comuns dentro da implementação e finalizando com as classes que herdam da classe *activity*.

5.2.1 Item

Representa a abstração do modelo de dados, fornecendo acesso rápido às informações referentes ao objeto em questão por meio de métodos *get*, que segundo Harvey e Paul Deitel (2006) são métodos de acesso a dados.

5.2.2 UIHelper

Esta classe implementa métodos práticos para acessar e injetar conteúdo em elementos contidos nas *views* do aplicativo.

5.2.3 Data

A classe Data fornece funções que facilitam a execução de operações práticas para manipulação dos dados armazenados no aplicativo, permitindo um acesso fácil às operações CRUD (create, read, update and delete). Os métodos que implementam tais operações são os seguintes:

- ***fetchAll***: retorna uma coleção que contém todos itens já catalogados no sistema tal coleção no entanto não possui os dados de cada item em sua integridade, somente imagem e título, o que permite que cada item sejam visualmente identificável para os usuários, e o identificador único deste item, este identificador serve como referência para outras *activities* que executam operações sob um item específico;
- ***fetchFromId***: este método espera o ID de um item como parâmetro, o ID fornecido permite uma consulta que retornará todas as informações a respeito do item em questão;
- ***save***: salva as modificações feitas em um item da coleção, armazenando a coleção atualizada em seguida;
- ***remove***: deleta um item da coleção e armazena a coleção atualizada em seguida.
- ***clear*:** deleta a coleção por inteiro.

5.2.4 ItemAdapter

Essa classe auxilia na implementação de um fragmento visual customizado que representa cada item na coleção existente. Ela é derivada da classe *ArrayAdapter*, disponibilizada pelo Android SDK com o propósito de retornar uma *view* para cada um dos itens pertencentes a uma coleção de objetos.

5.2.5 MainActivity

Toda e qualquer aplicação Android possui uma classe *MainActivity*, que é a *activity* principal do aplicativo, essa *activity* controla a *view* inicial do aplicativo e por via de regra será sempre a primeira da pilha.

Na aplicação em questão a *MainActivity* disponibiliza para o usuário uma lista de todos os itens catalogados em memória, e também oferece a possibilidade de adicionar um novo item à coleção existente.

Em seu método *onCreate*, executado assim que a *activity* inicia o seu ciclo de vida na aplicação, ela faz uso da classe *Data* para consultar a lista de itens existentes em memória. Havendo um ou mais itens, eles serão inseridos em uma lista de objetos da classe *Item*.

Tendo sido devidamente populada, a lista então será usada como parâmetro ao se instanciar outro membro da *MainActivity*, um objeto do tipo *ItemAdapter*, classe descrita anteriormente e que como dito tem por função gerar um fragmento gráfico customizado.

Ainda no método *onCreate* também são inicializados todos os elementos contidos na *view*. A lista de elementos é inicializada em um objeto da classe *ListView* que faz uso do *ItemAdapter* descrito no parágrafo anterior para construir a interface gráfica dos items. Cada item dessa lista possui um *listener*[2], que permite que se crie um *intent*[3] quando um item é tocado, levando o usuário para outra área do aplicativo.

O botão disponível nesse layout também recebe um método *listener*, este lança um *intent* para a utilização da câmera do dispositivo que executa a aplicação. Essa operação é disponibilizada pelo Android SDK de forma nativa.

A *activity* ainda possui um menu embutido em seu cabeçalho, este menu por sua vez contém uma única opção que tem por função exportar todo o conteúdo até então catalogado para um arquivo PDF(Portable Document Format).

O método responsável por tal execução é denominado *generatePdf*, ele faz uso de uma biblioteca de terceiros que foi integrada à aplicação e que será detalhada adiante. Quando

---

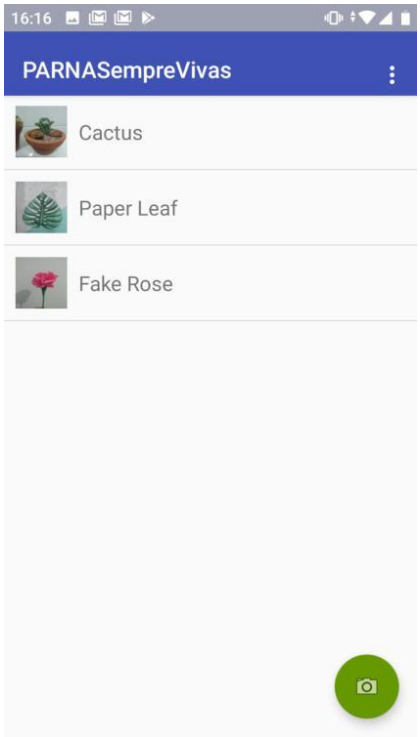
[2] Segundo Mozilla (2019a), *listeners* observam um elemento, identificando eventos que o afetam.
[3] Segundo Lecheta (2010, p. 135), “uma intent representa a intenção da aplicação de realizar determinada tarefa”.

invocado, este método executa um laço de repetição sobre a coleção existente adicionando as informações de cada item ao documento PDF, ao fim de cada iteração do laço uma nova página é criada e se dá início a outro item. Ao completar-se o laço o método *clear* da classe *Data* é então executado e o layout atual é atualizado.

O último método de relevância dentro desta classe é o que trata dos resultados retornados a ela por outras *activities*. Este método é o *onActivityResult* e é herdado da classe *Activity*.

Existem três casos a serem tratados dentro dele: o primeiro trata do retorno da operação de utilização da câmera, que caso tenha sido positiva gera um novo *intent* direcionando a aplicação para a *AddItemActivity*, que será descrita posteriormente. O segundo e terceiro casos são tratados separadamente devida à natureza de suas operações, adição e edição respectivamente, no entanto o resultado prático final é bastante similar, ambos modificam a lista que é mostrada ao usuário.

Figura 10: Interface gráfica da classe MainActivity



Fonte: *Screenshot* feita pelo autor

Figura 11: Amostra do relatório gerado após exportação dos dados

Fonte: *Screenshot* feita pelo autor

01/07/2019 21:56

Cactus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Sed adipiscing diam donec adipiscing tristique risus nec feugiat in. Malesuada fames ac turpis egestas maecenas. Cursus eget nunc scelerisque viverra mauris in.

Lat: -18.2320438
Lng: -43.6117132

02/07/2019 16:15

Paper Leaf

Just a leaf printed on paper. It looks quite real though!

Lat: -18.2321539
Lng: -43.6117611

5.2.6 AddItemActivity

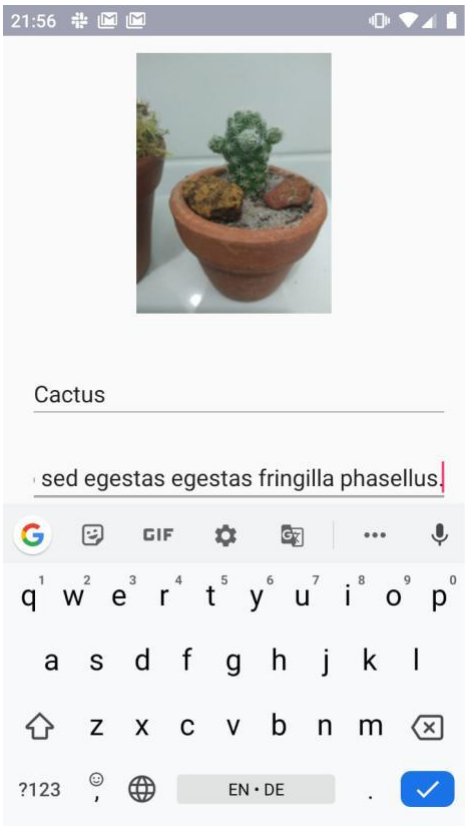
Essa classe é responsável por coletar todas as informações necessárias para a criação e adição de um novo item à coleção. A inicialização dela está condicionada ao resultado positivo do *intent* que implementa a funcionalidade da câmera fotográfica.

Ao se iniciar ao ciclo de vida dessa *activity* a primeira operação realizada é a validação e captura do resultado retornado pelo *intent* da câmera. Após essa verificação faz-se a checagem de permissão do uso dos componentes de GPS, caso a permissão não tenha sido dada ainda o usuário será interpelado à fazê-lo, sendo satisfeita esta condição o método então faz uso dos serviços oferecidos pelo Android SDK para geolocalização, e invoca outro método membro para armazenar a foto recebida através do *intent* da câmera. Esse método se chama *savePhoto*, ele armazena a foto processada por outro método da classe chamado *createImageFile*, responsável por tratar e comprimir o arquivo retornado pelo *intent* que lida com a câmera.

Ainda no primeiro estágio do seu ciclo de vida a classe implementa um método *listener* para o botão que visa finalizar a adição do novo item, esse método faz uso dos serviços de geolocalização para coletar a latitude e longitude referentes ao posicionamento geográfico do dispositivo. Este mesmo método utiliza métodos abstratos da classe *UIHelper* para assisti-lo na captação dos dados inseridos pelo usuário, e chama um outro método membro responsável por executar a adição deste item à coleção existente.

O método responsável por finalizar a adição de um novo item é denominado *storeItemInfo*, este método espera como parâmetros os dados inseridos pelo usuário e uma referência à foto capturada anteriormente. Tendo sido satisfeitos todos os parâmetros necessários um objeto da classe Item é criado e adicionado à coleção, a coleção atualizada é então armazenada em memória. Após seu armazenamento o resultado da *activity* é registrado para que seja retornado para a *MainActivity* assim que seu ciclo de vida chegar ao fim.

Figura 12: Interface gráfica da classe AddItemActivity



Fonte: *Screenshot* feita pelo autor

5.2.7 ItemDetailActivity

É esta classe que permite ao usuário visualizar, ainda no aplicativo, os itens até então catalogados, e é chamada sempre que um item presente na *MainActivity* é tocado. No início do seu ciclo de vida essa classe faz uso do ID a ela fornecido para consultar as informações a respeito do item em sua totalidade, essas informações então são aplicadas aos elementos presentes no layout controlado pela classe. Para cada item existente no catálogo do aplicativo o usuário poderá ver: seu título, sua descrição, sua foto, a data em que foi catalogado e a sua localização em um mapa interativo.

O mapa utilizado para ilustrar a geolocalização do item é um fragmento que implementa a interface do Google Maps, este fragmento é disponibilizado pela Google no pacote GMS (Google Mobile Services). É necessário que a classe implemente a interface *OnMapReadyCallback* que por sua vez requer a sobreposição do método *onMapReady*, um *callback*[4] que será executado assim que todas dependências necessárias para a execução do mapa estiverem a pronto emprego.

Na implementação em questão esse método faz uso dos pontos de latitude e longitude disponibilizados pela classe que implementa o modelo de dados para estabelecer a localização de um marcador que será etiquetado com o título dado ao item pelo usuário. Configurações gerais a respeito do mapa também são estabelecidas neste *callback*, como por exemplo o zoom mínimo ao qual o mapa pode ser submetido.

O *layout* dessa classe também contém um botão, este cumpre a função de permitir que o usuário edite ou delete o item, e que ao ser acionado traz para o contexto a classe *EditItemActivity*, responsável por editar um item e que será tratada adiante. Ainda em referência à classe *EditItemActivity*, a classe aqui descrita aguarda um resultado vindo da mesma, este resultado confirma se houve de fato alguma alteração no item, atualizando seu conteúdo atual caso ele tenha sido editado ou retornando para o contexto corrente a classe *MainActivity* caso o item tenha sido deletado.

[4] Uma função passada como argumento para outra função, este argumento será invocado pela função externa quando esta executar alguma ação (MOZILLA, 2019).

Figura 13: Interface gráfica da classe DetailItemActivity

Fonte: *Screenshot* feita pelo autor

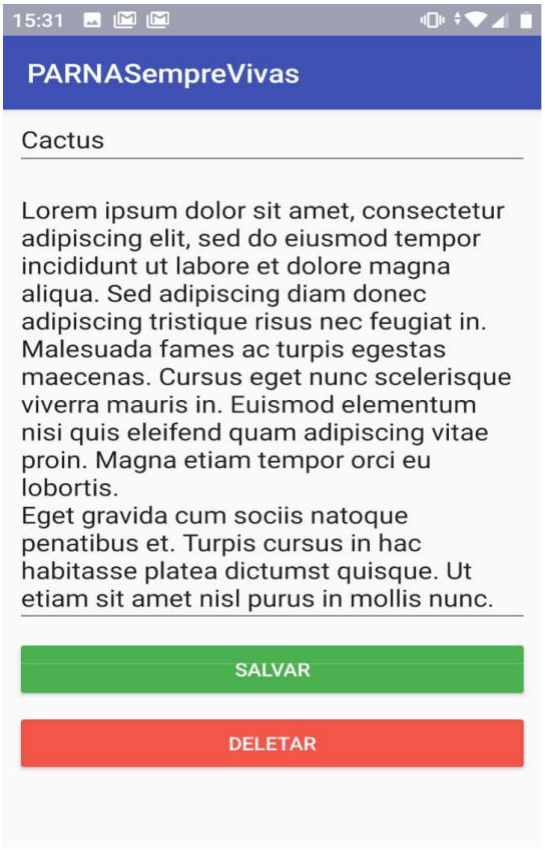
5.2.8 EditItemAcitivity

Tal qual a classe *ItemDetailActivity*, responsável por trazê-la ao escopo de uso, a classe *EditItemAcitivty* depende do identificador do item sobre qual executará suas operações.

O comportamento regular desta classe é o de popular os campos editáveis do item com suas informações atuais, permitindo que estas sejam alteradas. Caso as alterações sejam salvas essa classe tem seu ciclo de vida encerrado, retornando o item alterado para a *activity* que a precede na pilha, que dentro do contexto do aplicativo sempre será a *ItemDetailActivity*.

Esta classe também oferece a opção de remoção do item. Caso a remoção seja executada o item será prontamente deletado da coleção existente, direcionando o usuário para a *MainActivity* em seguida.

Figura 14: Interface gráfica da classe EditItemActivity



Fonte: *Screenshot* feita pelo autor

5.3 Bibliotecas e Pacotes

Esta seção visa explicar de forma mais detalhada como operam as bibliotecas e pacotes de terceiros que foram utilizados no projeto, estes são o pacote GMS, disponibilizado pela empresa Google, e uma biblioteca chamada iText, implementada pela empresa a qual ela também dá nome.

5.3.1 iText

O iText é um conjunto de ferramentas que oferece várias operações com arquivos PDF, desde a criação e edição de arquivos até a conversão de outros formatos. É uma das bibliotecas mais completas para este propósito. Sua instalação é inteiramente gerenciada através do *Gradle*, o que possibilita que com a simples adição do trecho de código a seguir a biblioteca esteja prontamente disponível para uso:

```
dependencies {
        compile 'com.itextpdf:itextg:5.5.10'
}
```

Esta biblioteca oferece métodos prontos que permitem que operações relativamente complexas como formatações de documento, imagens, texto e codificação de caracteres sejam feitas de maneira extremamente simples.

5.3.2 GMS

É uma coleção de aplicações e APIs disponibilizadas pela Google, e tem o propósito de dar suporte à adição dos serviços oferecidos por ela de maneira homogênea entre diferentes dispositivos. Um dos serviços disponibilizados pelo pacote GMS é o fragmento que integra o Google Maps ao layout de uma *activity*, ele foi utilizado no contexto da classe *ItemDetailAcitivity* para demonstrar a localização do item em um mapa interativo.

CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho proporcionou uma análise clara do que é necessário pra se implementar uma aplicação móvel visando automatizar tarefas manuais, demonstrando que as tecnologias e ferramentas disponíveis atualmente oferecem um nível elevado de flexibilidade.

Apesar disso houve alguns obstáculos, como a dificuldade em encontrar uma biblioteca de renderização de mapas adequada, no entanto, eles se deram por conta de restrições relacionadas à disposição organizacional da instituição que a proposta deste trabalho visava beneficiar, e também à corrente situação da infraestrutura de redes no país. Fatores que não tem relação alguma com a tecnologia disponível no mercado.

Não obstante, foi possível alcançar um resultado satisfatório, especialmente quando se considera a situação prévia nas quais o trabalho era realizado, de maneira inteiramente manual. Possibilitando a execução das mesmas tarefas de forma ágil, com a utilização de um único dispositivo como ferramenta central durante todo o processo.

A solução criada por este trabalho não só substitui as ferramentas anteriores com primazia como também traz consigo novas funcionalidades que irão aprimorar o resultado dos trabalhos, como a capacidade de catalogar a geolocalização de espécies junto aos dados coletados.

A flexibilidade da plataforma e da própria aplicação também permite que este trabalho seja usado como base para trabalhos futuros, seja por meio de incrementar as funcionalidades aqui desenvolvidas ou mesmo buscar a integração do aplicativo com outras plataformas.

REFERÊNCIAS

ANDROID. **Application Fundamentals.** [201-?]a. Disponível em: <https://developer.android.com/guide/components/fundamentals>. Acesso em 04 de Julho 2019.

ANDROID. **Configure your build.** [201-?]c Disponível em: <https://developer.android.com/studio/build>. Acesso em 04 Julho de 2019.

ANDROID. **Platform Architecture.** [201-?]. Disponível em: <https://developer.android.com/guide/platform>. Acesso em 04 de Julho de 2019.

ANDROID. **Processes and threads overview.** [201-?]b. Disponível em: <https://developer.android.com/guide/components/processes-and-threads>. Acesso em 12 de Julho 2019.

BOTHA, I.; BÂRA, A.; OPREA S., JORDÃO, T.C. W. M. **Integrating XML techology in object-relational databases into decision support systems**. 2011. Academy of Economic Studies, Bucharest; Transelectrica Company, Bucharest; University of Pardubice, Czech Republic. 2011.

BRASIL. Ministério da Casa Civil. Subchefia para Assuntos Jurídicos. Decreto de 13 de dezembro de 2002. **Diário Oficial da União**, Brasília, DF, 16 dez. 2002. Seção 1, p. 7.

BUNKER, A. **A brief history of mobile data: The road to the next generation.** 2013. Disponível em: <https://blog.vodafone.co.uk/2013/01/21/a-brief-history-of-mobile-data/>. Acesso em 12 Julho de 2019.

DEITEL, H. M.; DEITEL, P. J. **C++:** como programar. 5ª Edição. São Paulo: Pearson Prentice Hall, 2006.

DEITEL, H. M.; DEITEL, P. J. **Java How to Program**. 4th Edition. New Jersey: Pearson Prentice Hall, 2002.

DEITEL, H. M.; DEITEL, P. J. **Java:** Como Programar. 6ª Edição São Paulo: Pearson Prentice Hall, 2005.

DUDLEY, D. **The Evolution of Mobile Phones: 1973 to 2019.** 2018. Disponível em: <https://flauntdigital.com/blog/evolution-mobile-phones/>. Acesso em 11 de Julho de 2019.

DYROFF, C. **Here's how much cellphones have actually changed over years.** 2018. Disponível em: <https://www.wired.com/2008/04/dayintech-0403>. Acesso em 12 de Julho de 2019.

FREEMAN, J. **What is JSON? Javascript Object Notation Explained.** 2017. Disponível em: <https://www.infoworld.com/article/3222851/what-is-json-javascript-object-notation-explained.html>. Acesso em 12 de Julho de 2019.

IPANEMA, Marcello. História da Comunicação. Brasília: Editora Universidade de Brasília, 1967.

JSON. **Introdução ao JSON**. [ca. 2000]. Disponível em: <http://www.json.org/json-pt.html>. Acesso em 07 de Março de 2016.

KRAFUNI, S. **Rede e infraestrutura são os maiores desafios do Brasil.** Disponível em: <https://www.correiobraziliense.com.br/app/noticia/tecnologia/2016/10/18/interna_tecnologia,553694/rede-e-infraestrutura-sao-os-maiores-desafios-do-brasil.shtml>. 2016. Acesso em 11 de Julho de 2019.

LECHETA, R. R. **Google Android**: aprenda a criar aplicações para dispositivos móveis. 2ª Edição. São Paulo: Novatec, 2010.

LEE, Valentino; SCHNEIDER, Heather; SCHELL, Robbie. Aplicações móveis: arquitetura, projeto e desenvolvimento. São Paulo: Pearson Education do Brasil, 2005.

LEMOS, André. Cibercultura: tecnologia e vida social na cultura contemporânea. Porto Alegre: Sulina, 2013.

MITCHELL, Mark; OLDHAM, J.; SAMUEL, A. **Programação linux avançada**. Indianapolis: New Riders Pub, 2001. Disponível em: <http://www.advancedlinuxprogramming.com/>. Acesso em 04 de Julho de 2019.

MOZILLA. **Callback function.** 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Glossary/Callback_function>. Acesso em 12 Julho de 2019.

MOZILLA. **Introduction to events.** 2019a. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events>. Acesso em 13 Julho de 2019.

NEGUS, C. **The Linux Bible:** The comprehensive, tutorial resource. 9ª Edição. Indianapolis: Wiley, 2015.

ORACLE. Ahead-of-time compilation. 2012. Disponível em: <https://docs.oracle.com/javame/config/cdc/cdc-opt-impl/ojmeec/1.1/custom/html/aot.htm>. Acesso em 07 de Março de 2016.

RANDY, A. **April 3, 1973: Motorola calls AT&T... by cell.** 2008. Disponível em: <https://www.insider.com/the-history-of-the-cellphone-2018-7>. Acesso em 12 de Julho de 2019.
SYDOW, L. **Q2 2019 Was the Biggest Quarter for mobile to Date.** 2019. Disponível em: <https://www.appannie.com/en/insights/market-data/q2-2019-mobile-market-index/>. Acesso em 11 de Julho de 2019.

TANENBAUM, A. S. **Sistemas Operacionais, Projeto e Implementação**. 3ª Edição. Porto Alegre: Bookman, 2008.

TUREK, M. **Employees Say Smartphones Boost Productivity by 34 Percent: Frost & Sullivan Research.** 2016. Disponível em: <https://insights.samsung.com/2016/08/03/employees-say-smartphones-boost-productivity-by-34-percent-frost-sullivan-research/>. Acesso em 11 de Julho de 2019.

VMWARE. **VSPHERE DOCUMENTATION Center.** 2011. Disponível em: <https://pubs.vmware.com/vsphere-50/index.jsp>. Acesso em 07 de Março de 2016.

WIEGERS, K. **Software Requirements**. 2nd Edition. Redmond: Microsoft Press, 2003.