UNIVERSIDADE FEDERAL DO PARANÁ

ANDERSON FRANÇA MARCONCIN

NAIANA FÁTIMA OLISCOVICZ

NUTRINFO - APLICATIVO DE INFORMAÇÕES NUTRICIONAIS

CURITIBA

2017

ANDERSON FRANÇA MARCONCIN

NAIANA FÁTIMA OLISCOVICZ

NUTRINFO - APLICATIVO DE INFORMAÇÕES NUTRICIONAIS

Trabalho de conclusão de curso apresentado como requisito à conclusão do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnologia, Universidade Federal do Paraná.

Orientador: Prof. João Eugenio Marynowski

CURITIBA

2017

**RESUMO**

Texto.

Palavra Chave:

**ABSTRACT**

Text.

Key-word:

**LISTA DE FIGURAS**

FIGURA 1 – XXXX.................................................................................0

**LISTA DE SIGLAS**

BLA -Blabla bla

**LISTA DE TABELAS**

TABELA 1 – PLANO DE RISCOS...................................................0

SUMÁRIO

[1 INTRODUÇÃO 11](#_Toc405835694)

[1.1 OBJETIVOS 12](#_Toc405835695)

[1.2 OBJETIVOS ESPECÍFICOS 12](#_Toc405835696)

[2 FUNDAMENTAÇÃO TEÓRICA 14](#_Toc405835697)

[2.1 BIOINFORMÁTICA 14](#_Toc405835698)

[2.2 CONCEITOS BIOLÓGICOS 14](#_Toc405835699)

[2.3 CONSULTA DE INFORMAÇÕES BIOLÓGICAS 15](#_Toc405835700)

[2.3.1 Armazenamento 15](#_Toc405835701)

[2.3.2 Banco de Dados 15](#_Toc405835702)

[2.3.3 BLAST 16](#_Toc405835703)

[2.4 OCR 17](#_Toc405835704)

[2.5 PROCESSAMENTO DE IMAGENS 18](#_Toc405835705)

[2.6 SQLITE 18](#_Toc405835706)

[2.8 UML 19](#_Toc405835707)

[3 METODOLOGIA 20](#_Toc405835708)

[3.1 MODELO DE PROCESSO DE ENGENHARIA DE SOFTWARE 20](#_Toc405835709)

[3.2 PLANO DE ATIVIDADES 20](#_Toc405835710)

[3.2.1 Diagrama WBS 20](#_Toc405835711)

[3.2.2 Diagrama de Gantt 23](#_Toc405835712)

[3.3 PLANO DE RISCOS 25](#_Toc405835713)

[3.4 MATERIAIS 26](#_Toc405835714)

[3.4.1 Ambiente de Desenvolvimento 26](#_Toc405835715)

[3.4.2 Linguagem de Programação 26](#_Toc405835716)

[3.4.3 Banco de Dados 27](#_Toc405835717)

[3.4.4 Ambiente Integrado de Desenvolvimento 27](#_Toc405835718)

[3.4.5 Ferramenta para Criação do Diagrama WBS 27](#_Toc405835719)

[3.4.6 Ferramenta para Criação do Diagrama de Gantt 27](#_Toc405835720)

[3.4.7 Ferramenta para Modelagem UML 28](#_Toc405835721)

[3.4.8 Ferramenta para Criação do Diagrama Entidade Relacionamento 28](#_Toc405835722)

[3.5 FERRAMENTA PARA CONTROLE DE VERSÃO 28](#_Toc405835723)

[3.6 BIBLIOTECAS 28](#_Toc405835724)

[3.6.1 BioJava 29](#_Toc405835725)

[3.6.2 Tess-two 29](#_Toc405835726)

[4 DESENVOLVIMENTO DO PROJETO 30](#_Toc405835727)

[4.1 VALIDAÇÃO DO SOFTWARE 31](#_Toc405835728)

[4.1.1 Testes de OCR 31](#_Toc405835729)

[4.1.2 Testes de Compressão 33](#_Toc405835730)

[4.1.3 Testes de Iluminação 36](#_Toc405835731)

[5 APRESENTAÇÃO DO SOFTWARE 39](#_Toc405835732)

[5.1 INSTALAÇÃO 50](#_Toc405835733)

[6 CONSIDERAÇÕES FINAIS 51](#_Toc405835734)

[REFERÊNCIAS 52](#_Toc405835735)

[APÊNDICE A - DIAGRAMA DE CASOS DE USO 56](#_Toc405835736)

[APÊNDICE B - ESPECIFICAÇÃO DOS CASOS DE USO 57](#_Toc405835737)

[APÊNDICE C - DIAGRAMA ENTIDADE RELACIONAMENTO 64](#_Toc405835738)

[APÊNDICE D- DIAGRAMA DE CLASSES 65](#_Toc405835739)

[APÊNDICE F - DIAGRAMA DE COMPONENTES 71](#_Toc405835740)

# INTRODUÇÃO

A obesidade vem crescendo exponencialmente durante os últimos anos. Doenças como diabetes adquirida e anemia ocorrem com frequência. Devido a esses fatores as pessoas necessitam saber o que estão consumindo, para ingerir os nutrientes necessários de uma dieta saudável. A reeducação alimentar assunto tão importante nos dias atuais tem como base o conhecimento do que está se ingerindo.

Em uma dieta, é necessário ter um valor aproximado dos valores nutricionais das refeições, para que não se caia no erro de consumir alimentos em quantidades controladas, porém pobres em nutrientes, tendo como resultado ao invés do peso ideal, moléstias causadas pela má alimentação.

Existem hoje alguns aplicativos que auxiliam nesse processo. Porém, esses aplicativos normalmente mostram valores específicos de produtos industrializados. Em uma receita, podemos ter diversos produtos, que juntos formam a informação nutricional correta da receita. Cuja qual pode ser particionada, e com isso essas informações também devem ser recalculadas.

Atualmente muitas pessoas fazem comida caseira para complementar a renda informalmente. Porém, o consumidor desse produto, muitas vezes não consegue saber suas informações nutricionais, devido a essa informalidade.

O aplicativo com esse foco (MyFitnessPal), com melhor nota na google play, permite cadastrar a receita para calcular os valores, porém ficam armazenados somente o cálculo nutricional final da receita. Não apresentando a discriminação dos ingredientes para que outros usuários tenham a oportunidade de reproduzir a receita.

# OBJETIVOS

Desenvolver um aplicativo android, que permitirá simular uma tabela nutricional de receitas caseiras. O usuário deverá inserir as quantidades e os ingredientes de sua receita, e o aplicativo realizará o cálculo das informações nutricionais da receita, o cálculo será de acordo com o regulamento da ANVISA (Agência Nacional de Vigilância Sanitária).

O aplicativo não deve ser usado como instrumento oficial para rotulagem de alimentos. Porém, os valores serão aproximados, podendo ser usado para fins de comparação com a recomendação diária.

Os valores nutricionais fornecidos serão de acordo com uma tabela de composição de alimentos. A tabela usada como referência abrangerá o ingrediente de uma maneira geral e não de marcas especificas, os quais podem sofrer variações. Para valores mais exatos é necessário analisar os ingredientes e a composição do produto da marca específica.

* 1. OBJETIVOS ESPECÍFICOS

* + Utilizar a câmera do smartphone para capturar o código de barras de produtos consumidos ou a ser inseridos na receita.

# FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é apresentar conceitos de ..... e um pouco dos recursos que foram utilizados para o desenvolvimento do projeto.

# INFORMAÇÕES NUTRICIONAIS

Texto.

# BLA BLA

Texto.

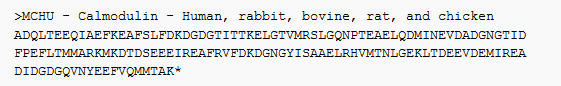


FIGURA 1 – Arquivo FASTA

Fonte: Wikipédia (2014).

# 2.3.2 Banco de Dados

Texto.

# 2.4 SCANNER

Texto.

# 2.6 NOSQL

Texto.

2.7 ANDROID

Android é um sistema operacional móvel baseado em kernel Linux e desenvolvido atualmente pela Google. Com a interface do usuário baseada em manipulação direta, o Android foi desenvolvido primeiramente para dispositivos móveis com tela sensível ao toque como smartphones e tablets (ENGINEERSGARAGE, 2014)

O Android fornece um framework de aplicações que permite o desenvolvimento de aplicativos e jogos para dispositivos móveis em Java, além de ser compatível com diversos dispositivos de configurações diferentes (ANDROID, 2014).

  O Android foi a plataforma escolhida para o desenvolvimento do projeto, pois ela é um dos sistemas operacionais móveis mais utilizados no mundo, além da facilidade de instalação e publicação de aplicativos pela Play Store.

# 2.8 UML

A UML (*Unified Modeling Language*) é uma linguagem visual utilizada para modelar software baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação. Essa linguagem tornou-se, nos últimos anos, a linguagem-padrão de modelagem adotada internacionalmente pela indústria de engenharia de software (GUEDES, 2011).

Neste trabalho utilizamos o diagrama de caso de uso, o diagramas de classe, o diagrama de atividades e o diagrama de componentes. O diagrama de caso de uso permite organizar os requisitos do sistema dando uma visão geral. O diagrama de classes representa a estrutura do sistema orientado a objetos. O diagrama de atividades apresenta o fluxo das atividades do aplicativo e o diagrama de componentes mostra a componentes e suas dependências.

# 3 METODOLOGIA

A função deste capítulo é apresentar as metodologias e ferramentas que foram utilizadas para o desenvolvimento deste aplicativo. Também serão mostradas as responsabilidades dadas a cada membro.

# 3.1 MODELO DE PROCESSO DE ENGENHARIA DE SOFTWARE

Neste projeto foi utilizado o modelo RUP (*Rational Unified Process*)  que fornece diretrizes para definir as tarefas e atribuir responsabilidades aos membros do projeto. O RUP possui quatro fases: iniciação, elaboração, construção e transição. Na iniciação define-se o escopo do projeto. Na elaboração é obtida uma visão abrangente do sistema, através da construção de protótipos e também é definida a arquitetura do sistema. Na construção, o foco está no desenvolvimento do sistema. Por fim, na transição, o produto é transferido ao usuário e o projeto é avaliado e pode ser concluído. Em todas as fases há o gerenciamento dos requisitos e dos recursos do projeto. Por isso o RUP é baseado no desenvolvimento iterativo, que por sua vez é mais flexível quanto às mudanças de escopo durante o desenvolvimento do projeto (IBM, 2014).

# 3.2 PLANO DE ATIVIDADES

O plano de atividades irá mostrar o fluxo de trabalho de acordo com as metodologias escolhidas pela equipe. Esse plano é apresentado através do diagrama WBS (*Work Breakdown Structure*) e do diagrama de Gantt, que serão mostrados a seguir.

# 3.2.1 Diagrama WBS

O diagrama WBS auxilia na divisão do trabalho, formando uma estrutura hierárquica que fornece uma visão geral das etapas deste trabalho (WBSTOOL, 2014).

O projeto foi dividido em quatro fases, de acordo com o RUP e possui uma quinta fase que corresponde ao gerenciamento de projeto. Essa divisão pode ser observada na FIGURA 2, que corresponde a WBS do projeto.

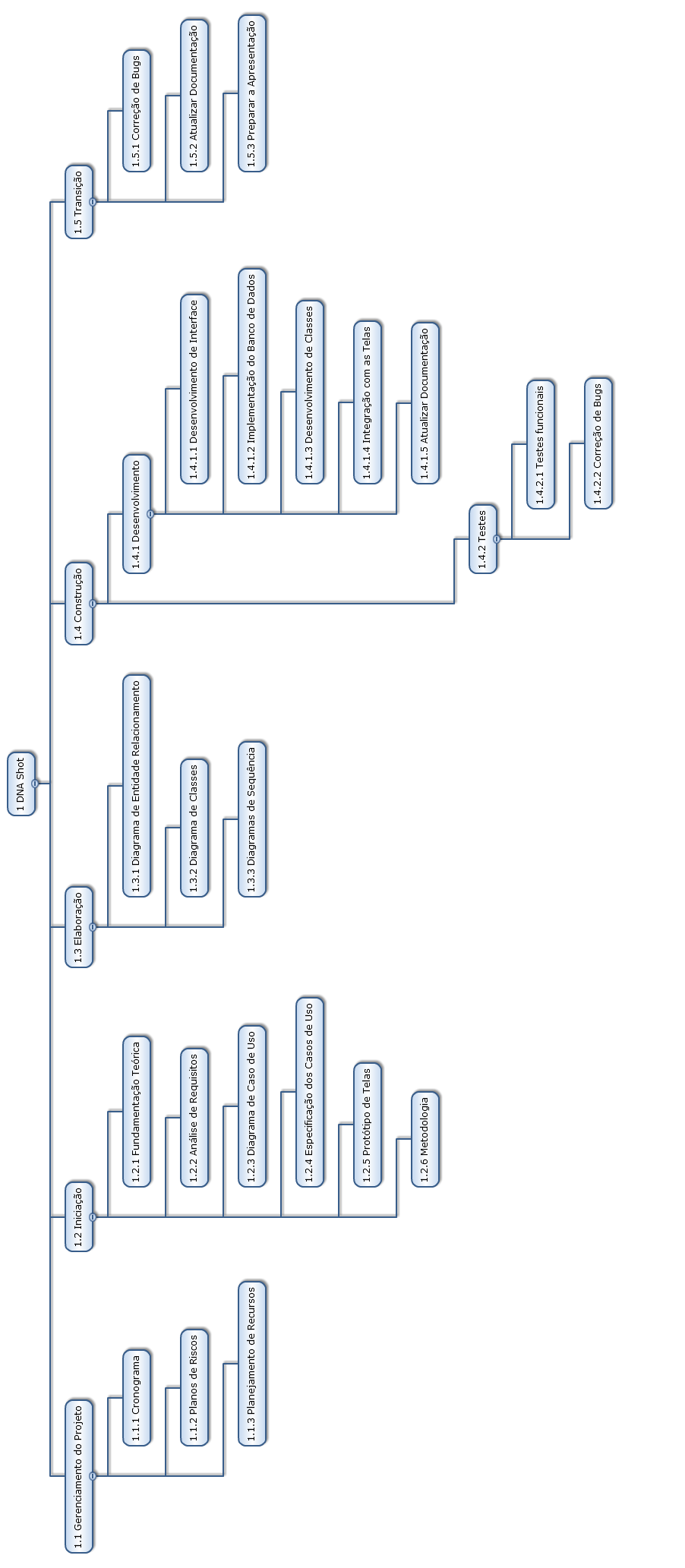


FIGURA 2 – DIAGRAMA WBS

Fonte: Os autores (2014).

# 3.2.2 Diagrama de Gantt

O diagrama de Gantt mostra as atividades de um projeto, quando cada uma deve ocorrer e quem irá executá-las (GANTT, 2014). O diagrama de Gantt foi elaborado de acordo com a WBS, e auxilia no gerenciamento das tarefas e dos recursos do projeto. No diagrama de Gantt (FIGURA 3) foram definidas as atividades no projeto, o prazo de entrega de cada uma e as pessoas responsáveis por elas.

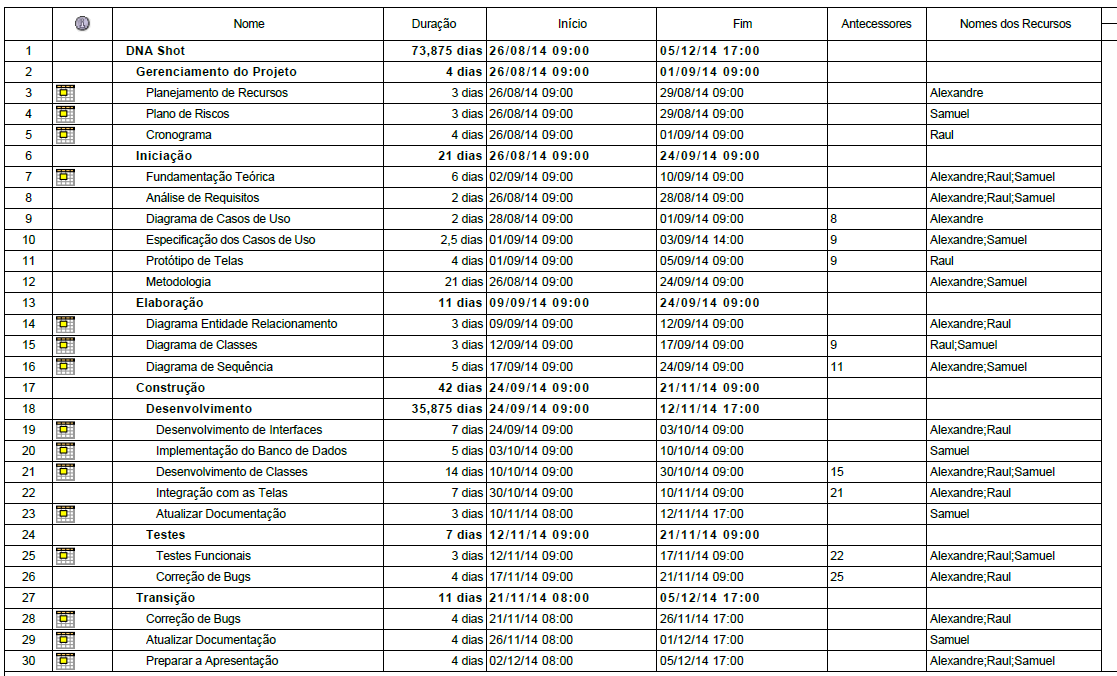


FIGURA 3 – DIAGRAMA DE GANTT

Fonte: Os autores (2014).

A FIGURA 4 e a FIGURA 5 mostram a linha do tempo de acordo com as tarefas da FIGURA 3.

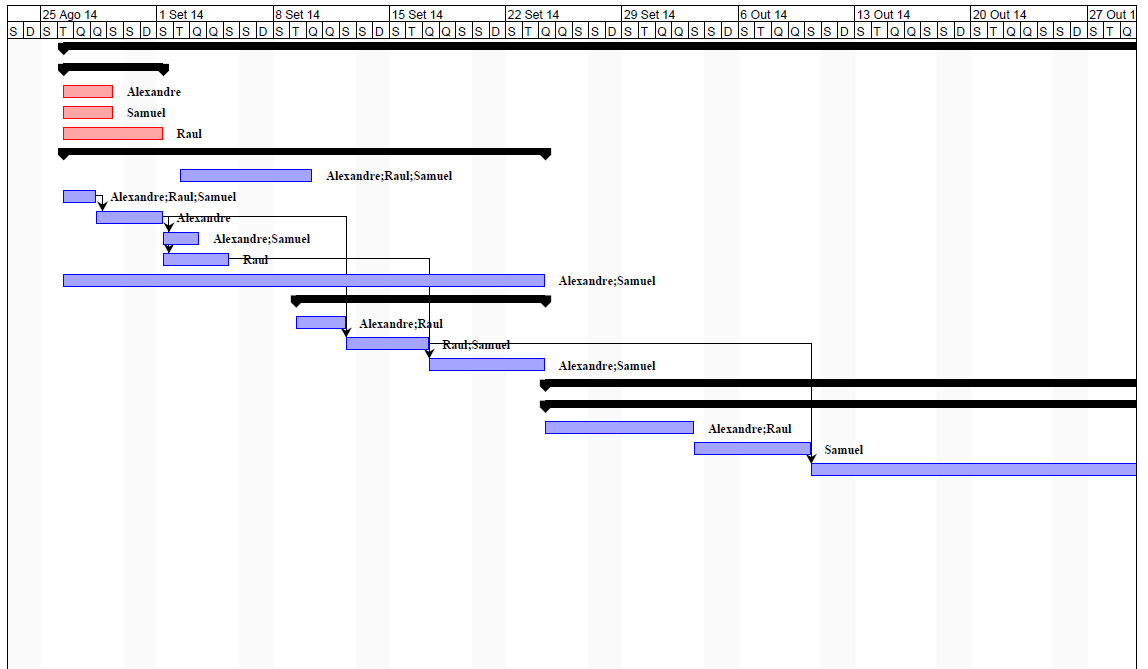


FIGURA 4 – LINHA DO TEMPO DO DIAGRAMADE GANTT

Fonte: Os autores (2014).



FIGURA 5 – LINHA DO TEMPO DO DIAGRAMADE GANTT

Fonte: Os autores (2014).

# **3.3** PLANO DE RISCOS

No plano de riscos foram definidos os impactos e a probabilidade de determinados eventos acontecerem, bem como as possíveis ações que poderiam ser tomadas. O plano de riscos do projeto pode ser observado na TABELA 2.

# 3.3 PLANO DE RISCOS

No plano de riscos foram definidos os impactos e a probabilidade de determinados eventos acontecerem, bem como as possíveis ações que poderiam ser tomadas. O plano de riscos do projeto pode ser observado na TABELA 2.

TABELA 2 – PLANO DE RISCOS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nº** | **Risco** | **Consequência** | **Ação** | **Probabilidade** | **Impacto** | **Classificação** |
| 1 | Mudança de requisitos do aplicativo. | Necessidade de mudar o direcionamento do projeto. | Mudanças na estrutura do projeto. | Alto | Alto | 7 |
| 2 | Falta de tempo para o desenvolvimento do aplicativo. | Impossibilidade de entregar ou atraso na entrega. | Divisão de tarefas e monitoramento do andamento. | Alto | Alto | 7 |
| 3 | Conhecimento limitado em bioinformática. | Dificuldade no desenvolvimento do projeto. | Estudo e pesquisa na internet e em livros. | Alto | Moderado | 6 |
| 4 | Perda de dados. | Impossibilidade de continuar o projeto. | Fazer uma rotina de backup dos arquivos importantes. | Moderado | Alto | 6 |
| 5 | Infraestrutura insuficiente. | Falta de equipamentos para desenvolvimento e testes. | Utilizar emuladores para as necessidades específicas. | Moderado | Moderado | 4 |
| 6 | Desistência de um membro da equipe. | Atrasos no andamento e necessidade de refazer a divisão de tarefas. | Dividir mais uma vez as tarefas. | Moderado | Moderado | 4 |

# 3.4 MATERIAIS

O objetivo deste capítulo é descrever as ferramentas utilizadas para o desenvolvimento do projeto.

# 3.4.1 Ambiente de Desenvolvimento

Para o desenvolvimento do aplicativo, foram utilizados três computadores:

* Desktop, processador Intel Core i3 540 3.07GHz, memória RAM 6GB, placa de vídeo Nvidia GTX 460 1GB, sistema operacional Ubuntu 14.10;
* Notebook Asus S46C, processador Intel Core i7 3517U 1.90GHz, memória RAM 8GB, placa de vídeo Nvidia GT 635M 2GB, sistema operacional Ubuntu 14.04;
* Notebook LG P410, processador Intel Core i5 2410M 2.30GHz, memória RAM 4GB, placa de vídeo Nvidia GT 520M 1GB, sistema operacional Windows 7 Home Premium;

    Para os testes do aplicativo foram utilizados três celulares:

* Motorola Moto G, processador Qualcomm 1.2GHz, memória RAM 1GB, sistema operacional Android 4.4;
* Motorola Moto G, processador Qualcomm 1.2GHz, memória RAM 1GB, sistema operacional Android 4.4;
* Motorola Razr D1, processador ARM 1.0GHz, memória RAM 768MB, sistema operacional Android 4.4;

# 3.4.2 Linguagem de Programação

A linguagem de programação escolhida pelos usuários foi Java. É uma linguagem baseada na orientação a objetos, utilizada em diversas plataformas, inclusive o Android. Atualmente essa linguagem pertence à Oracle (JAVA, 2014).

Ela foi escolhida pelos membros, pois é a única opção para criação de um aplicativo em Android. Mesmo sendo a única opção da equipe devido à plataforma escolhida, o Java possui uma quantidade grande de material sobre ela.

# 3.4.3 Banco de Dados

Para implementar o banco de dados foi escolhido o SQlite, pois é uma ferramenta que já vem instalada no Android e como os dados ficam num arquivo local,  não há necessidade de servidores ou sistemas gerenciadores.

# 3.4.4 Ambiente Integrado de Desenvolvimento

A ferramenta Eclipse foi utilizada neste projeto como IDE (*Integrated Development Environment*). O Eclipse possui licença gratuita e suporta diversas linguagens como Java, além de ser uma das poucas opções de IDE para se trabalhar com Android.

A IDE Eclipse foi escolhida porque os membros da equipe já possuem experiência com a ferramenta, além dela estar disponível em vários sistemas operacionais.

# 3.4.5 Ferramenta para Criação do Diagrama WBS

A ferramenta utilizada para criar o diagrama WBS foi a WBS Tool. Essa ferramenta é um software online e gratuito que disponibiliza as funcionalidades necessárias para criar diagramas WBS e organogramas.

# 3.4.6 Ferramenta para Criação do Diagrama de Gantt

A ferramenta utilizada para elaborar o Gantt do projeto foi o ProjectLibre, um software livre e de código aberto para auxiliar no gerenciamento de projetos que roda em vários sistemas operacionais.

# 3.4.7 Ferramenta para Modelagem UML

A ferramenta utilizada para criação dos diagramas UML foi o Software Ideas Modeler e o Violet UML Editor. Apesar de serem ferramentas simples, gratuitas e exclusivas do Windows, elas atenderam bem as necessidades do projeto.

# 3.4.8 Ferramenta para Criação do Diagrama Entidade Relacionamento

A ferramenta utilizada para criação do diagrama entidade relacionamento foi DBDesigner4, software de código aberto e desenvolvido pela fabFORCE. Está disponível para Windows e Linux.

# 3.5 FERRAMENTA PARA CONTROLE DE VERSÃO

O git é um sistema de controle de versão de softwares desenvolvidos por um ou mais desenvolvedores. Ele também é um sistema de gerenciamento de código fonte, ou seja, todos os membros da equipe podem enviar alterações. Todos os arquivos e todo histórico é armazenado num repositório (GIT, 2014).

Neste projeto foi usado o GitHub, que além de gerenciar as versões e as alterações no código, possui um repositório online, facilitando as alterações para os membros da equipe.

# 3.6 BIBLIOTECAS

Bibliotecas são um conjunto de funções pré-escritas para resolver diversos tipos de problema. Os desenvolvedores utilizam bibliotecas para não precisarem criar uma solução já existente.

  Através de pesquisas foi decidido pelos membros da equipe que 2 bibliotecas externas seriam usadas no projeto, o BioJava para o uso do BLAST, e o tess-two para o processamento digital da imagem e para aplicação do OCR.

# 3.6.1 BioJava

O BioJava é um framework desenvolvido para processar dados biológicos em Java. Contêm rotinas analíticas e estáticas, leitura dos formatos comuns de arquivo, permite a manipulação de sequências e de estruturas 3D e acesso aos webservices (BIOJAVA, 2014).

# 3.6.2 Tess-two

O tess-two é uma biblioteca desenvolvida para juntar duas ferramentas para o Android. Estas ferramentas são o Tesseract OCR e o Leptonica (RMTHEIS, 2014).

O Tesseract OCR é provavelmente uma das ferramentas de OCR mais precisas dentre as disponíveis, consegue ler vários tipos de imagem e converte para texto em mais de sessenta línguas. Estava entre as três melhores no teste de precisão da UNLV (*University of Nevada, Las Vegas*) (TESSERACT, 2014).

Entre 1995 e 2006 foi pouco trabalhada, mas desde então foi consideravelmente melhorada pela Google. Ela foi lançada pela Apache (TESSERACT, 2014). Essa ferramenta foi escolhida por sua precisão, velocidade e compatibilidade.

Leptonica é uma biblioteca de código aberto para processamento e análise de imagens orientada para pedagogos (LEPTONICA, 2014).

**4 DESENVOLVIMENTO DO PROJETO**

A ideia inicial do projeto era criar um aplicativo em Android que se comunicaria com um webservice, que também seria desenvolvido, para realizar consultas do BLAST por dispositivos móveis. Foram feitas diversas mudanças de escopo e na escolha de linguagens e bibliotecas ao decorrer do projeto, que causaram alguns problemas com o cronograma.

Como já dito, seria desenvolvido um webservice para comunicar-se com o BLAST e o dispositivo móvel. Inicialmente feito no Octave, o processamento da imagem e o OCR estavam funcionando de forma instável. Mesmo assim, foi desenvolvido um servidor em Python para ligar os processos do Octave à web.

  Após alguns testes, a equipe concluiu que seria inviável continuar com o Octave e decidiu implementar o processamento da imagem e o OCR em Python, assim o webservice ficaria em apenas uma linguagem. O Tesseract-OCR foi escolhido como biblioteca para o OCR, era de uso fácil e rápido. Para consulta ao BLAST foi usada a biblioteca BioPython, que estava funcionando sem problemas, mas o processamento da imagem ainda estava instável e havia a necessidade de um servidor.

A equipe decidiu testar todos esses processos no próprio dispositivo móvel para verificar se não haveria limitações de hardware. Os testes com a biblioteca Tess-two foram satisfatórios, pois obteve bons resultados tanto no processamento da imagem quanto no OCR.

Os resultados do BioJava também foram satisfatórios, necessitando apenas da leitura do XML devolvido pelo BLAST que foi facilmente suprida pela linguagem. Assim o projeto migrou-se totalmente para o Java. Posteriormente houve a necessidade de um banco de dados simples, logo a equipe optou pelo SQLite pela praticidade da ferramenta.

A divisão de tarefas foi mudando de acordo com as mudanças de linguagem, graças ao conhecimento da linguagem dos membros da equipe. Apesar das diversas mudanças, a equipe conseguiu se adaptar a elas.

A equipe decidiu tirar os diagramas de sequência e colocar diagramas de atividade, pois são diagramas mais fáceis de serem lidos e ilustram o que ocorre de forma mais simples.

# 4.1 VALIDAÇÃO DO SOFTWARE

A equipe realizou algumas validações após o processamento da imagem, o OCR e o BLAST estarem estáveis e funcionais. Foi decidido que os testes principais envolveriam a compressão e a iluminação da foto, que são fatores que podem afetar o OCR mesmo com o processamento da imagem.

Para testes do OCR a equipe decidiu utilizar apenas uma sequência onde foram testadas algumas fontes mais comuns, tanto em maiúsculo quanto em minúsculo para verificar se o OCR errou em algum momento.

As sequências selecionadas para os testes foram:

* + Vírus *Zaire ebolavirus* (NCBI, 2014);
  + Bactéria *Herbaspirillum seropedicae* (NCBI, 2014);
  + Vírus da Dengue (NCBI, 2014);
  + Planta *Hevea brasiliensis* (NCBI, 2014);

# 4.1.1 Testes de OCR

    Para os testes de OCR, foi utilizada uma sequência de 100 caracteres da planta Hevea *brasiliensis*, as fotos foram tiradas sob uma iluminação boa, compressão média e cinco fontes diferentes em tamanho 12. As fontes escolhidas foram:

* + Arial;
  + Times New Roman;
  + Lucida Console;
  + Calibri;
  + Lucida Handwriting.

Além da aplicação do OCR, o aplicativo tira os caracteres diferentes de “A”, “C”, “T” e “G”, logo os erros foram verificados pelo tamanho da sequência gerada no OCR e uma comparação caractere a caractere.

Os resultados do OCR foram satisfatórios, 8 dos 10 testes realizados chegaram no resultado esperado, e 2 testes conseguiram reconhecer todos os caracteres sem nenhum erro. A TABELA 3 mostra os resultados encontrados.

TABELA 3 - RESULTADOS DOS TESTES DE OCR

|  |  |  |
| --- | --- | --- |
| **Fonte** | **Tamanho da Sequência (Em caracteres)** | **Valor E** |
| Arial, maiúsculo | 94 | 1.4991e-34 |
| Arial, minúsculo | 98 | 7.1406e-39 |
| Times New Roman, maiúsculo | 95 | 7.89282e-32 |
| Times New Roman, minúsculo | 100 | 1.16687e-42 |
| Lucida Console, maiúsculo | 94 | 9.46033e-31 |
| Lucida Console, minúsculo | 100 | 1.16687e-42 |
| Calibri, maiúsculo | 94 | 1.68243e-27 |
| Calibri, minúsculo | 46 | - |
| Lucida Handwriting, maiúsculo | 92 | 3.19372e-30 |
| Lucida Handwriting, minúsculo | - | - |

    O teste realizado na fonte Lucida Handwriting em minúsculo não detectou nenhum caractere, pois se tratava de uma letra corrida. Já o resultado do OCR no teste feito na fonte Calibri em minúsculo obteve uma sequência muito menor, que causou o erro do resultado.

    Também houve troca do caractere “G” pelo “C” em 3 testes aumentando o valor E dos resultados. Os seguintes testes tiveram troca de caractere:

* + Times New Roman em maiúsculo, duas ocorrências;
  + Lucida Console em maiúsculo, uma ocorrência;
  + Calibri em maiúsculo, duas ocorrências.

# 4.1.2 Testes de Compressão

    Para os testes de compressão, foram utilizadas duas sequências diferentes, o vírus *Zaire ebolavirus* e a bactéria *Herbaspirillum seropedicae*, para passar nos cinco níveis de compressão definidos no aplicativo. As fotos foram tiradas de uma câmera de 5 megapixels, pegando sempre os mesmos caracteres para que as fotos sejam mais parecidas.

    Neste teste foram considerados o tamanho e a qualidade da imagem, a imagem pré-processada, além da qualidade do resultado obtido através do valor E. A TABELA 4 (*Zaire ebolavirus*) e a TABELA 5 (*Herbaspirillum seropedicae*) mostram a relação entre o nível de compressão da imagem, o tamanho da imagem e o valor E para cada uma das sequências.

TABELA 4 - RESULTADOS DOS TESTES DE COMPRESSÃO DO VÍRUS EBOLA.

|  |  |  |
| --- | --- | --- |
| **Nível de Compressão** | **Tamanho da Imagem** | **Valor E** |
| 1 | 2285 KB | 0 |
| 2 | 691 KB | 0 |
| 3 | 382 KB | 0 |
| 4 | 220 KB | 0 |
| 5 | 106 KB | 2.22733e-20 |

TABELA 5 - RESULTADOS DOS TESTES DE COMPRESSÃO DA BACTÉRIA HERBBASPIRILLUM.

|  |  |  |
| --- | --- | --- |
| **Nível de Compressão** | **Tamanho da Imagem** | **Valor E** |
| 1 | 3173 KB | 0 |
| 2 | 660 KB | 0 |
| 3 | 397 KB | 0 |
| 4 | 312 KB | 5.51797e-106 |
| 5 | 108 KB | 4.44747e-11 |

Os resultados mostraram que a compressão influencia diretamente nos resultados da pesquisa no BLAST. Apesar dos ganhos no tamanho da imagem e no tempo de processamento no nível de compressão máximo, ele claramente afeta no OCR fazendo com que a sequência fique com ruído e incompleta, obtendo resultados menos precisos.

A FIGURA 6 e a FIGURA 7 mostram respectivamente uma foto tirada no nível 1 de compressão e outra foto no nível 5 de compressão. Como o nível 3 obteve bons resultados, além do tamanho razoável da imagem, a equipe decidiu torná-la como padrão no aplicativo. O nível de compressão da imagem ainda pode ser trocado nas configurações do aplicativo.

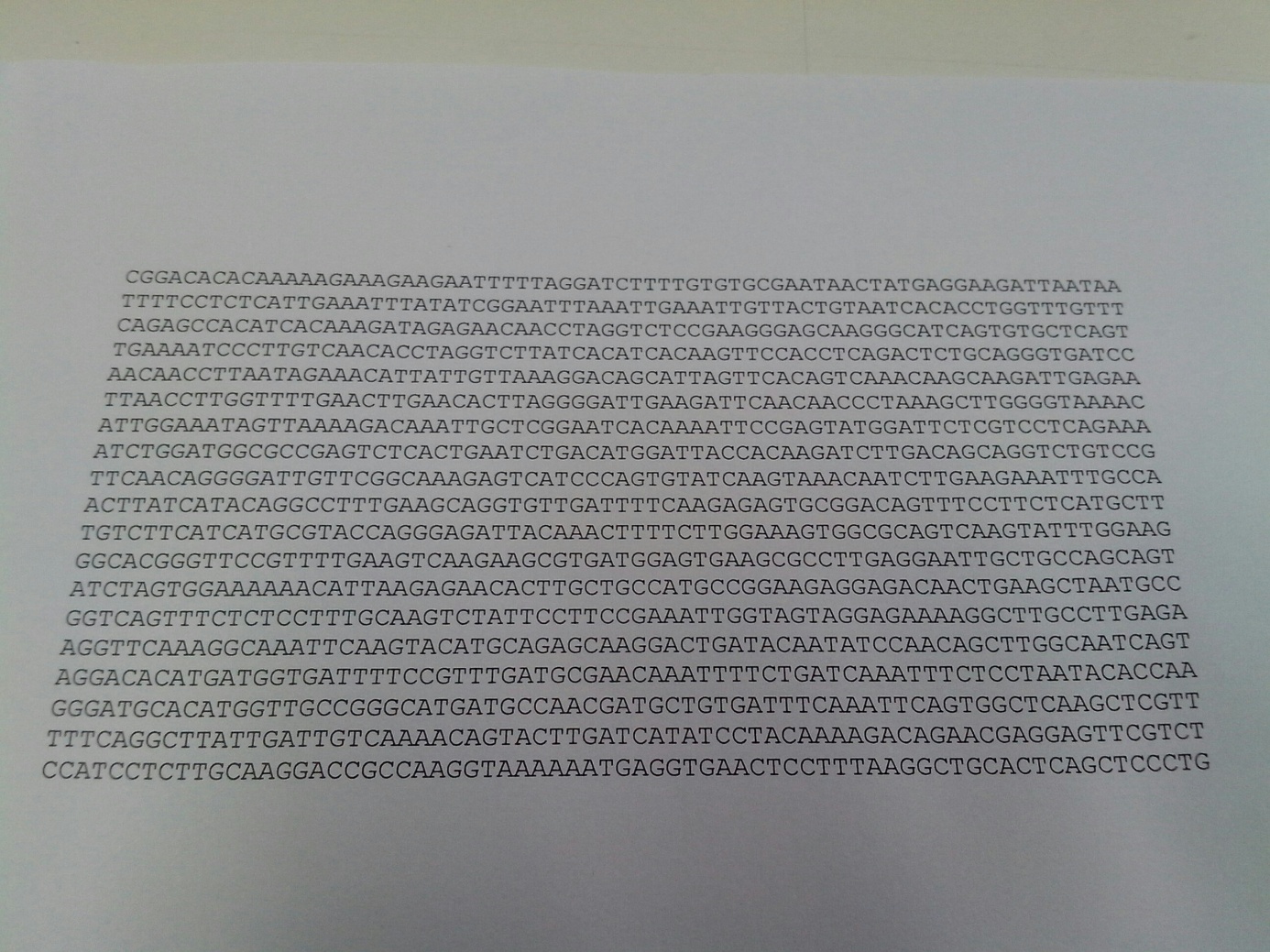


FIGURA 6 – FOTO TIRADA NO NÍVEL 1 DE COMPRESSÃO

Fonte: Os autores (2014).

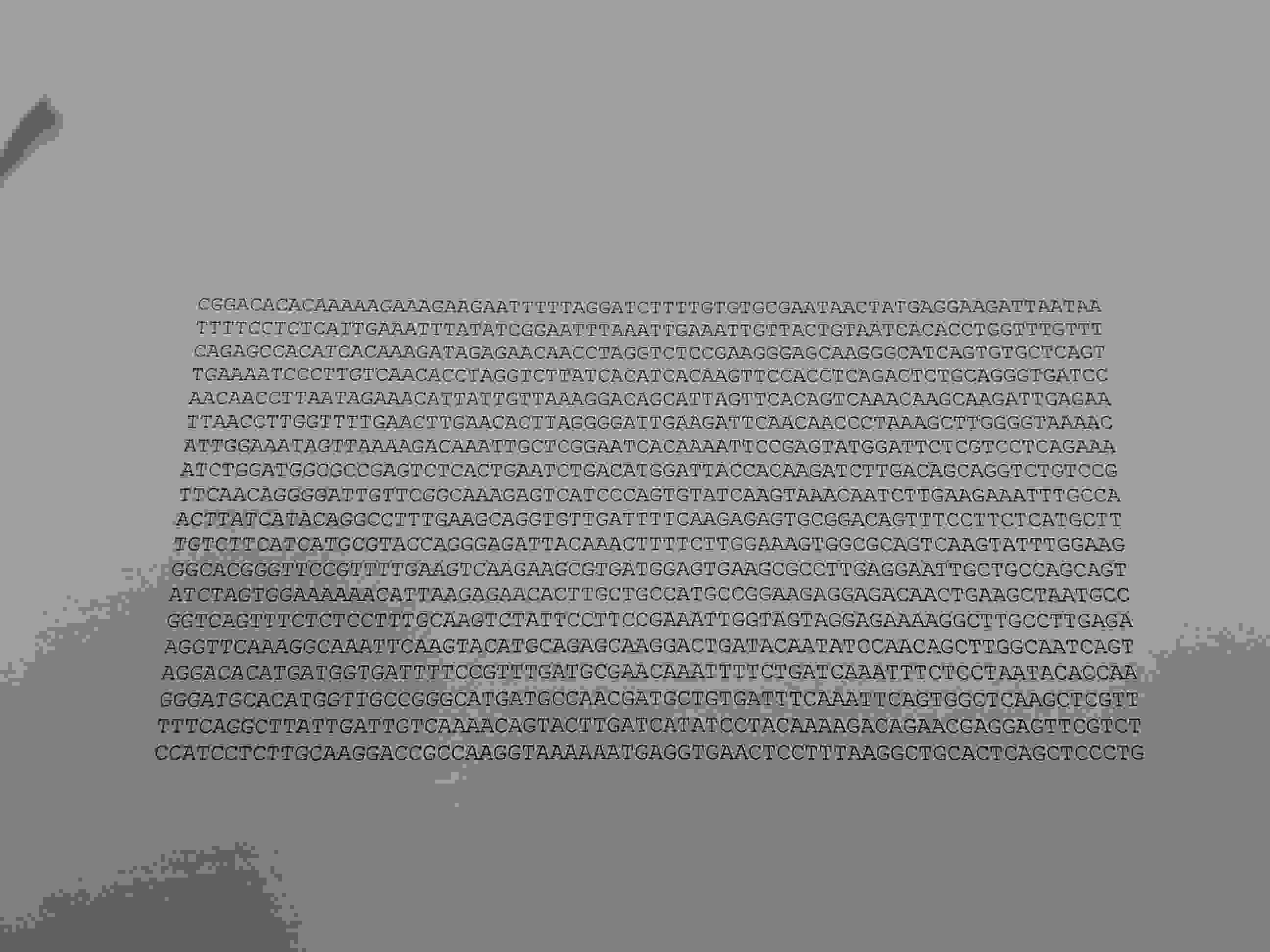


FIGURA 7 – FOTO TIRADA NO NÍVEL 5 DE COMPRESSÃO

Fonte: Os autores (2014).

# 4.1.3 Testes de Iluminação

O teste de iluminação foi feito para verificar se o thresholding está funcionando de forma adequada. Com o intuito de comparar os resultados, a equipe realizou os testes numa sequência de *Hevea brasiliensis,* numa sequência de um vírus da Dengue, numa sequência do vírus *Zaire ebolavirus* e numa sequência da  bactéria *Herbaspirillum seropedicae.* Os testes realizados foram:

* + Tirar foto sob iluminação boa;
  + Tirar foto com uma metade da foto sombreada, outra bem iluminada;
  + Tirar foto com toda a foto sombreada.

    Todos os testes foram a realizados com a compressão média e sempre pegando os mesmos caracteres. O foco deste teste foi a qualidade da imagem pré-processada e o valor E, para comparação de resultados.

Nos testes de iluminação boa e nos testes onde metade da foto teria sombra, todos conseguiram um valor E igual a 0. No teste com a foto sombreada, apenas a bactéria *Herbaspirillum seropedicae* adquiriu um valor E maior que 0, obtendo 3.61693e-28.

    A FIGURA 8 mostra uma comparação entre a foto original que está na parte de cima, e a imagem após o processamento, que está na parte de baixo. A sequência é do vírus *Zaire ebolavirus.* A partir da imagem nota-se que pequenos pontos pretos aparecem onde havia sombra após o processamento da imagem, que pode causar erros no OCR.

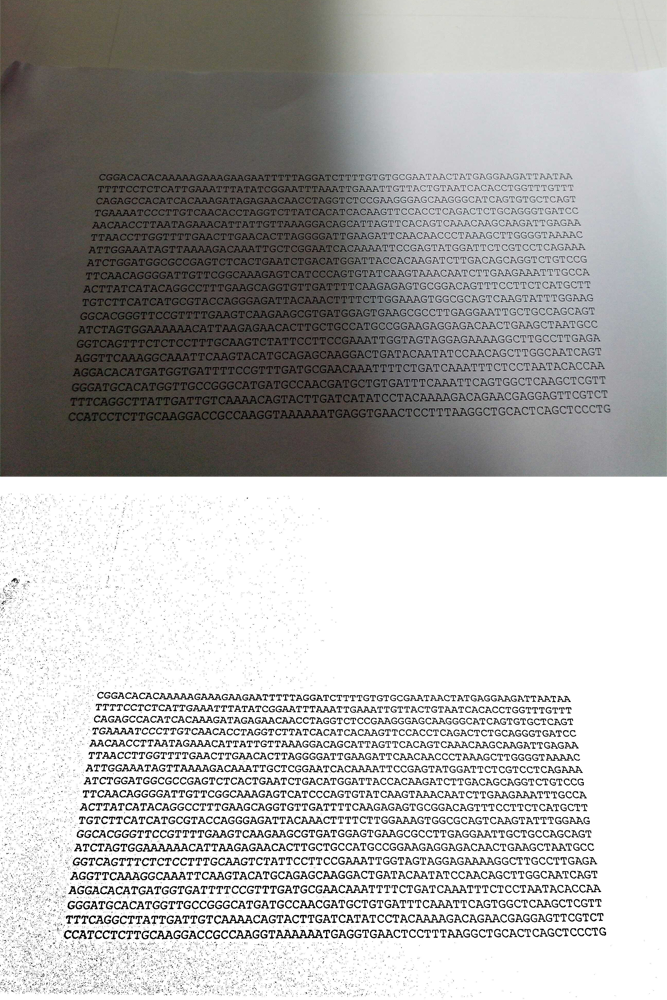


FIGURA 8 – COMPARAÇÃO ENTRE A FOTO ORIGINAL E A IMAGEM APÓS O PROCESSAMENTO DA IMAGEM

Fonte: Os autores (2014).

**5 APRESENTAÇÃO DO SOFTWARE**

Neste capítulo será apresentada uma visão geral do aplicativo desenvolvido, mostrando suas telas e funcionalidades.

Antes de entrar no aplicativo, uma tela de abertura é mostrada na FIGURA 9 enquanto o aplicativo carrega seus componentes. A tela inicial mostrada na FIGURA 10 lista as consultas, pode-se verificar o status do processamento ou caso esteja finalizado, verificar os resultados e dá as opções da câmera e de tirar foto.

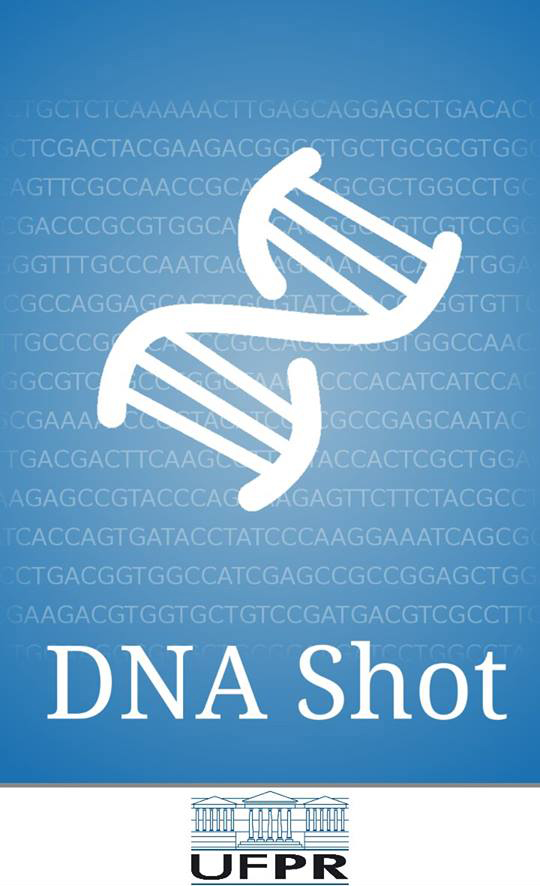


FIGURA 9 - TELA DE ABERTURA

Fonte: Os autores (2014).

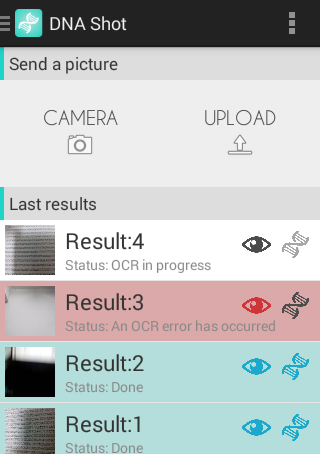


FIGURA 10 - TELA INICIAL DO APLICATIVO

Fonte: Os autores (2014).

    Ao clicar em “CAMERA”, o aplicativo direciona para uma câmera, mostrado na FIGURA 11, onde são tiradas as fotos para a consulta.

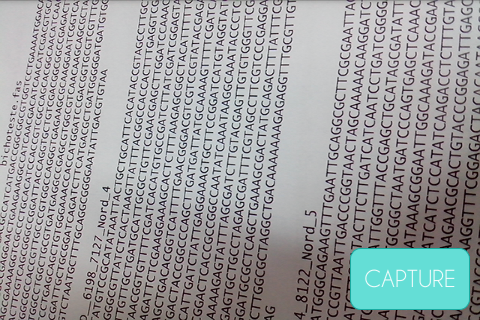


FIGURA 11 - TELA DA CÂMERA

Fonte: Os autores (2014).

Outra forma de realizar a consulta seria clicar em “UPLOAD”, que direciona o usuário para selecionar uma imagem na galeria, mostrado na FIGURA 12.

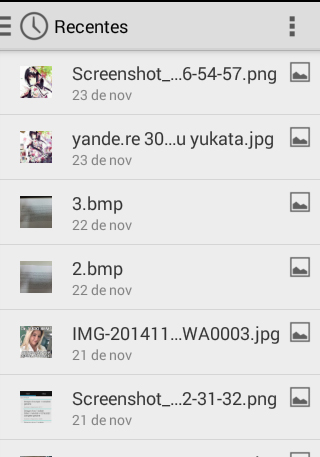


FIGURA 12 - TELA PARA SELECIONAR UMA IMAGEM

Fonte: Os autores (2014).

    Após selecionar um resultado pronto na tela inicial, o usuário é direcionado a tela de resultados, na aba “Result”, mostrada na FIGURA 13. Tocando nos resultados, é possível verificar os HSPs encontrados, assim como na FIGURA 14. Também é possível excluir o resultado selecionado tocando no ícone presente no canto superior direito.

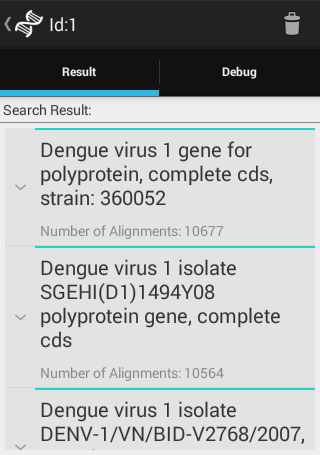


FIGURA 13 - TELA DE RESULTADOS

Fonte: Os autores (2014).

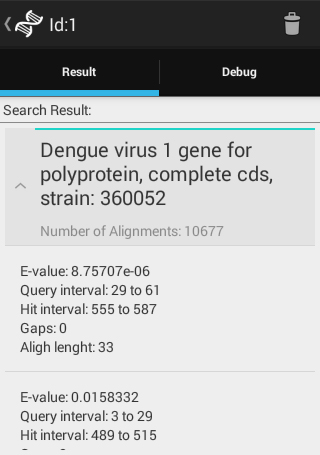


FIGURA 14 - TELA DE RESULTADOS COM HSPS

Fonte: Os autores (2014).

Há outra aba na tela de resultados chamada “Debug”. Nesta tela temos a imagem ou foto original, mostrada na FIGURA 15, a imagem pré-processada, mostrada na FIGURA 16 e a sequência que o OCR achou a partir da imagem pré-processada, mostrado na FIGURA 17.

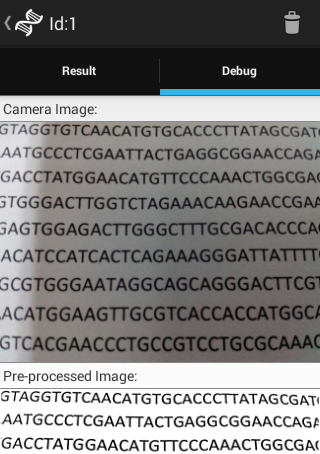


FIGURA 15 - IMAGEM ORIGINAL NA ABA “DEBUG”

Fonte: Os autores (2014).

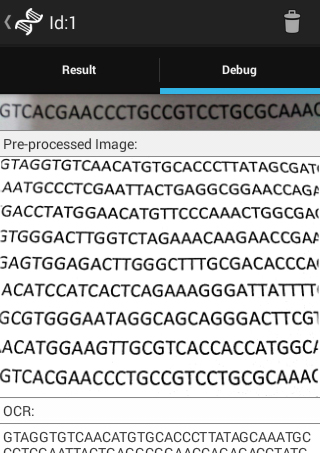


FIGURA 16 - IMAGEM ORIGINAL PRÉ-PROCESSADA NA ABA “DEBUG”

Fonte: Os autores (2014).



FIGURA 17 - SEQUÊNCIA ACHADA PELO OCR NA ABA “DEBUG”

Fonte: Os autores (2014).

Voltando a tela inicial, podem-se excluir todos os itens da lista, tocando no botão “Clear Results”, mostrado na FIGURA 18. Após o toque, o aplicativo irá abrir um alerta (mostrado na FIGURA 19), perguntando se o usuário realmente deseja limpar os resultados.

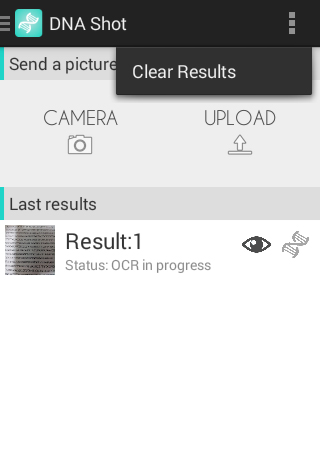


FIGURA 18 - BOTÃO PARA LIMPAR OS RESULTADOS

Fonte: Os autores (2014).



FIGURA 19 - ALERTA DE CONFIRMAÇÃO

Fonte: Os autores (2014).

Além disso, há um menu de navegação lateral à esquerda, mostrado na FIGURA 20, que dá acesso às configurações e informações do aplicativo.

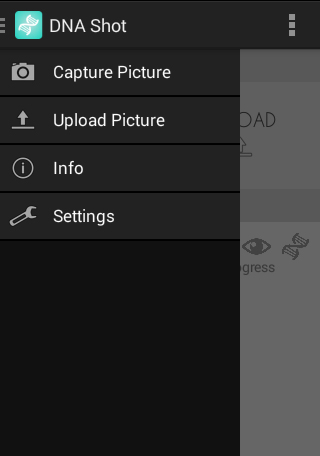


FIGURA 20 - MENU NA TELA INICIAL DO APLICATIVO

Fonte: Os autores (2014).

    Na FIGURA 21 e FIGURA 22 são mostradas as informações gerais do aplicativo a partir do item “Info” do menu lateral.

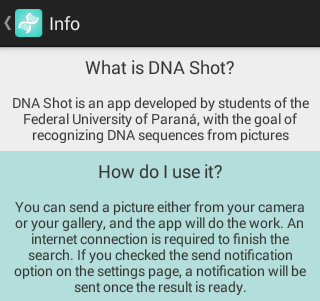


FIGURA 21 - TELA DE INFORMAÇÕES GERAIS

Fonte: Os autores (2014).

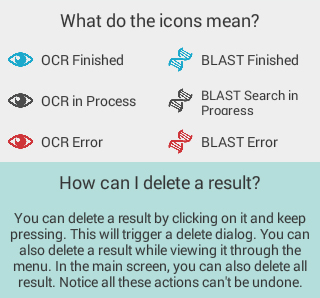


FIGURA 22 - TELA DE INFORMAÇÕES GERAIS

Fonte: Os autores (2014).

O usuário também pode mudar as configurações do aplicativo na tela de configurações, mostrada na FIGURA 19 e acessado pelo item “Settings” do menu lateral. O usuário pode ativar ou desativar as notificações, alterar a língua e selecionar o nível de compressão da imagem.

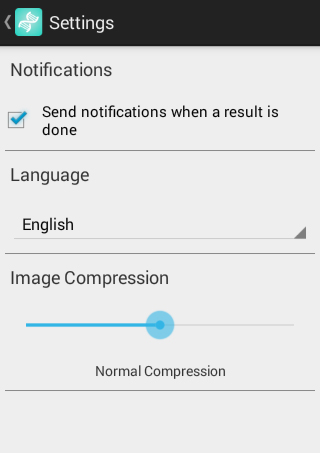


FIGURA 23 - TELA DE CONFIGURAÇÕES

Fonte: Os autores (2014).

# 5.1 INSTALAÇÃO

Para instalar o aplicativo, deve-se entrar no link do aplicativo na Play Store (https://play.google.com/store/apps/details?id=com.dnashot.v0), mostrado na FIGURA X1, toque em “Instalar” e depois em “Aceitar”. Após o download e a instalação do aplicativo, ele já estará pronto para uso. A equipe recomenda o uso deste aplicativo em celulares com processador de 1 GHz ou mais, além de 1GB de memória RAM.



FIGURA 24 - TELA DO APLICATIVO NA PLAY STORE

Fonte: Os autores (2014).

**6 CONSIDERAÇÕES FINAIS**

O desenvolvimento do aplicativo trouxe um contato inicial com a bioinformática para os usuários. Ao trabalhar com apenas um dos programas ofertados pelo BLAST, foi possível perceber que ainda podem-se integrar muitos algoritmos da bioinformática para os dispositivos móveis.

O aplicativo consegue cumprir suas funções de processar sua imagem, aplicar o OCR e realizar uma pesquisa no blastn num dispositivo móvel. A interface gráfica do aplicativo foi feita para que o usuário também possa ver o que aconteceu durante o processo.

  A maioria das funções do aplicativo tem um processamento custoso à nível de dispositivos móveis, logo é possível que celulares mais antigos não funcionem perfeitamente ou tenham a performance afetada.

Durante o desenvolvimento, várias ideias de como o aplicativo poderia ser melhorado e ampliado surgiram, porém não foi possível implementá-las devido a complexidade e tempo. Algumas destas ideias para uma futura versão estão listadas abaixo:

* Melhorias na câmera: É possível melhorar o foco da câmera para que toda vez que haja alguma mudança brusca, ela foque automaticamente ao invés de esperar um segundo.
* Integração ao SILA (*Genome Automated Annotation System*): Através de um webservice é possível fazer uma integração a este sistema de anotação automática de genomas. Como é uma ferramenta rápida, os resultados viriam mais rápido que os resultados do BLAST.
* Integração com o blastp: É necessário um filtro diferente para a sequência de proteínas e também um tempo maior de espera do usuário.

**REFERÊNCIAS**

ABBYY. **What is OCR and OCR Tecnology.** Disponível em: <http://finereader.abbyy.com/about\_ocr/whatis\_ocr/>. Último acesso: 17/11/2014.

ACHARYA, T.; RAY, A. **Image Processing: Principles and Applications**. New Jersey: John Wiley & Sons, 2005.

ANDROID. **Introduction to Android | Android Developers.** Disponível em: < https://developer.android.com/guide/index.html>. Último acesso: 22/11/2014.

BAXEVANIS, A.; OULLETTE, B. **Bioinformatics: A pratical Guide to the Analysis of Genes**. New Jersey: John Wiley & Sons, 2005.

BERG, Jeremy Mark. **Biochemistry**. Nova York: Freeman, 2006.

BIOJAVA. **BioJava.** Disponível em: <http://biojava.org/>. Último acesso: 17/11/2014.

ENGINEERSGARAGE. **What is Android: Introduction, Features & Applications.** Disponível em: <http://www.engineersgarage.com/articles/what-is-android-introduction>. Último acesso: 17/11/2014.

FOX, J. **WHAT IS BIOINFORMATICS?** Disponível em: <<http://www.scq.ubc.ca/what-is-bioinformatics/>>. Último acesso: 24/10/2014.

GANTT. **What is a Gantt Chart? Gantt Chart Information, history and Software.** Disponível em: <http://www.gantt.com/>. Último acesso: 18/11/2014.

GIT. **About - Git.** Disponível em: <http://git-scm.com/about/>. Último acesso: 18/11/2014.

GUEDES, G. **UML 2: uma abordagem prática**. São Paulo: Novatec, 2011.

IBM. **IBM – Definição de RUP.** Disponível em: <http://www-01.ibm.com/software/rational/rup/>. Último acesso: 29/10/2014.

JAVA. **Obtenha informações sobre a Tecnologia Java.** Disponível em: <https://www.java.com/pt\_BR/about/>. Último acesso: 18/11/2014.

KORF, I.; YANDELL, M.; BEDELL, J. **BLAST: An Essential Guide to the Basic Local Alignment Search Tool**. Sebastopol: O’Reilly, 2003

LEPTONICA. **Leptonica.** Disponível em: <http://www.leptonica.com/>. Último acesso: 17/11/2014.

LEPTONICA. **Leptonica: src/skew.c File Reference.** Disponível em: <https://tpgit.github.io/Leptonica/skew\_8c.html>. Último acesso: 17/11/2014.

MATHWORKS. **Image Thresholding.** Disponível em: <http://www.mathworks.com/discovery/image-thresholding.html>. Último acesso: 17/11/2014.

MOUNT, David W. **Bioinformatics: Sequence and Genome Analysis**. Nova York: CSHL Press, 2004.

NCBI. **BLAST Program Selection Guide.** Disponível em: <http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE\_TYPE=BlastDocs&DOC\_TYPE=ProgSelectionGuide>. Último acesso: 14/11/2014.

NCBI. **Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-May - Nucleotide - NCBI.** Disponível em: <http://www.ncbi.nlm.nih.gov/nucleotide/10313991>. Último acesso: 20/11/2014

NCBI. **Herbaspirillum seropedicae SmR1, complete genome - Nucleotide - NCBI.** Disponível em: <http://www.ncbi.nlm.nih.gov/nucleotide/300072131>. Último acesso: 20/11/2014

NCBI. **DNA sequence coding dengue fever virus I(DEN1)-S275/90 - Nucleotide - NCBI.** Disponível em: <http://www.ncbi.nlm.nih.gov/nucleotide/2175014>. Último acesso: 20/11/2014

NCBI. **Hevea brasiliensis hydroxynitrile lyase (hnl) mRNA, complete cds - Nucleotide - NCBI.** Disponível em: <http://www.ncbi.nlm.nih.gov/nucleotide/1223883>. Último acesso: 20/11/2014

PHONESCOOP. **Smartphone definition (Phone Scoop).** Disponível em: <http://www.phonescoop.com/glossary/term.php?gid=131>. Último acesso: 19/11/2014.

PROSDOCIMI, F. **Bioinformática: Manual do Usuário.** Disponível em: <http://www2.bioqmed.ufrj.br/prosdocimi/divulgacao/bioinfo\_manual.pdf Último acesso: 14/11/2014.

RASHIDI, H.; BUEHLER, L. **Bioinformatic Basics – Applications in Biological Science and Medicine.** Londres: CRC Press, 2000.

RMTHEIS. **rmtheis/tess-two.** Disponível em: <https://github.com/rmtheis/tess-two>. Último acesso: 17/11/2014.

SAUVOLA. J.; PIETIKAINEN M. **Adaptive document image binarization**. Disponível em: <http://www.mediateam.oulu.fi/publications/pdf/24.pdf>. Último acesso: 18/11/2014.

SQLITE. **About SQLite.** Disponível em: <http://www.sqlite.org/about.html>. Último acesso: 09/11/2014.

TESSERACT. **tesseract-ocr.** Disponível em: <https://code.google.com/p/tesseract-ocr/>. Último acesso: 17/11/2014.

VIALLE, R. A. et al. **SISTEMA INTEGRADO PARA ANOTAÇÃO AUTOMÁTICA DE GENOMAS**. [ S. I.]    Universidade Federal do Paraná, 2013.

WBSTOOL. **WBS Tool.com - Software web gratuito para WBS (EAP), Organogramas e Hierarquias.** Disponível em: <http://www.wbstool.com/whatIsWBS.php>. Último acesso: 18/11/2014.

ZHANGLAB. **FASTA format.** Disponível em: <http://zhanglab.ccmb.med.umich.edu/FASTA/>. Último acesso: 24/10/2014.

DBDesigner4

Disponível em: < http://www.fabforce.net/dbdesigner4/>. Último acesso: 20/11/2014.

Eclipse

Disponível em: <https://www.eclipse.org/home/index.php>. Último acesso: 10/11/2014

ProjectLibre

Disponível em: <http://www.projectlibre.org/>. Último acesso: 29/10/2014.

Software Ideas Modeler

Disponível em: <http://www.softwareideas.net/>. Último acesso: 30/10/2014.

WBS Tool

Disponível em: <http://www.wbstool.com/>. Último acesso: 29/10/2014.

Violet UML Editor

Disponível em: < http://alexdp.free.fr/violetumleditor/page.php>. Último acesso: 24/11/2014.

**APÊNDICE A - DIAGRAMA DE CASOS DE USO**

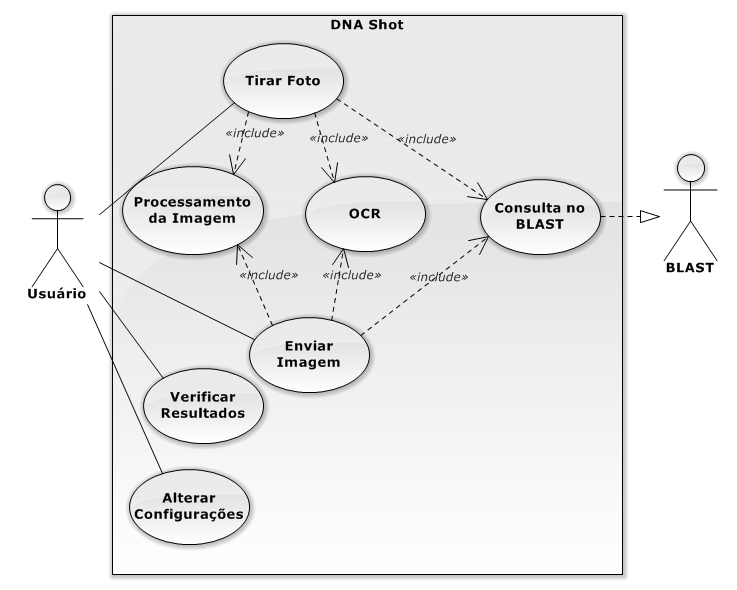


FIGURA 25 – DIAGRAMA DE CASOS DE USO

Fonte: Os autores (2014).

**APÊNDICE B - ESPECIFICAÇÃO DOS CASOS DE USO**

**Caso de uso 001 - Tirar foto**

**Descrição**

Este caso de uso descreve como tirar foto no aplicativo.

**Pré-condições**

Não há.

**Pós-condições**

O processamento da imagem inicia.

**Ator Primário**

Usuário

**Fluxo Principal de Eventos**

1. Na tela inicial, o usuário seleciona a “CAMERA”;
2. O aplicativo irá abrir a tela da câmera;
3. O usuário foca no texto desejado e toca em tirar.
4. O aplicativo irá salvar a imagem para processamento;
5. O aplicativo volta à tela da câmera;
6. O caso de uso é finalizado.

**Fluxo de Exceção**

Não há.

**Regras de Negócio**

Não há.

**Caso de uso 002 - Enviar Imagem**

**Descrição**

Este caso de uso descreve como enviar imagem no aplicativo.

**Pré-condições**

Não há.

**Pós-condições**

O processamento da imagem inicia.

**Ator Primário**

Usuário

**Fluxo Principal de Eventos**

1. Na tela inicial, o usuário seleciona a “LOAD”;
2. O aplicativo irá abrir a galeria do dispositvo;
3. O usuário seleciona a imagem desejada;
4. O aplicativo irá salvar a imagem para processamento;
5. O aplicativo volta à tela inicial;
6. O caso de uso é finalizado.

**Fluxo de Exceção**

Não há.

**Regras de Negócio**

Não há.

**Caso de uso 003 - Processamento da Imagem**

**Descrição**

Este caso de uso descreve como ocorre o processamento da imagem no aplicativo.

**Pré-condições**

O aplicativo deve ter salvado uma imagem para processamento, seja pela câmera ou pela galeria.

**Pós-condições**

O OCR inicia.

**Ator Primário**

Usuário

**Fluxo Principal de Eventos**

1. O aplicativo carrega a imagem;
2. Thresholding é aplicado na imagem;
3. O ângulo de inclinação do texto na imagem é detectado;
4. A imagem será girada de acordo com o ângulo encontrado;
5. O caso de uso é finalizado.

**Fluxo de Exceção**

Não há.

**Regras de Negócio**

Não há.

**Caso de uso 004 - OCR**

**Descrição**

Este caso de uso descreve como ocorre o OCR é aplicado na imagem.

**Pré-condições**

O aplicativo deve ter processado a imagem devidamente.

**Pós-condições**

A consulta ao BLAST inicia.

**Ator Primário**

Usuário

**Fluxo Principal de Eventos**

1. O aplicativo carrega a imagem processada;
2. O OCR é aplicado na imagem;
3. O caso de uso é finalizado.

**Fluxo de Exceção**

E1 O texto reconhecido pelo OCR é menor que 20.

        E1.1 Será colocado no status “An OCR error has ocurred” na lista de resultados na tela inicial.

E2 O OCR não reconheceu nenhum caractere.

        E2.1 Será colocado no status “An OCR error has ocurred” na lista de resultados na tela inicial.

**Regras de Negócio**

Não há.

**Caso de uso 005 - Consulta no BLAST**

**Descrição**

Este caso de uso descreve como ocorre a consulta no BLAST.

**Pré-condições**

O aplicativo deve ter terminado o OCR e achado uma sequência com mais de 20 caracteres e o dispositivo deve estar conectado à internet.

**Pós-condições**

O resultado ficará disponível na tela inicial.

**Ator Primário**

Usuário

**Fluxo Principal de Eventos**

1. O aplicativo inicia uma consulta no BLAST com a sequência achada;
2. O aplicativo espera 8 segundos;
3. O aplicativo verifica se o BLAST já finalizou sua consulta;
4. O aplicativo repete os passos 2 e 3 até a consulta finalizar;
5. O aplicativo interpreta o XML da consulta e salva no banco de dados;
6. O aplicativo coloca “Pronto” no status do aplicativo;
7. O caso de uso é finalizado,

**Fluxo de Exceção**

E1 O dispositivo não está conectado à internet.

        E1.1 Ele irá tentar realizar a consulta 6 segundos depois.

    E2 O BLAST não encontrou nenhum resultado.

        E2.1  Será colocado no status “An BLAST error has ocurred” na lista de resultados na tela inicial.

E3 O BLAST não estava com a resposta pronta após 50 tentativas.

E3.1  Será colocado no status “An BLAST error has ocurred” na lista de resultados na tela inicial.

**Regras de Negócio**

Não há.

**Caso de uso 006 - Verificar Resultados**

**Descrição**

Este caso de uso descreve como se verifica os resultados da pesquisa a partir da imagem enviada.

**Pré-condições**

O aplicativo deve ter encerrado a consulta no BLAST e salvo os dados no banco de dados e seu status na tela inicial deve estar “Pronto”.

**Pós-condições**

Não há.

**Ator Primário**

Usuário

**Fluxo Principal de Eventos**

1. O usuário deve tocar no resultado com status “Pronto” na lista da tela inicial;
2. O aplicativo irá abrir a tela de resultados;
3. O caso de uso é finalizado.

**Fluxo de Exceção**

Não há.

**Regras de Negócio**

Não há.

**Caso de uso 007 - Alterar Configurações**

**Descrição**

Este caso de uso descreve como ocorre a alteração de configurações no aplicativo.

**Pré-condições**

Não há.

**Pós-condições**

Não há.

**Ator Primário**

Usuário

**Fluxo Principal de Eventos**

1. O usuário toca em Configurações;
2. O aplicativo abre a tela de configurações;
3. O usuário seleciona as configurações desejadas;
4. O caso de uso é finalizado.

**Fluxo de Exceção**

Não há.

**Regras de Negócio**

Não há.

**APÊNDICE C - DIAGRAMA ENTIDADE RELACIONAMENTO**

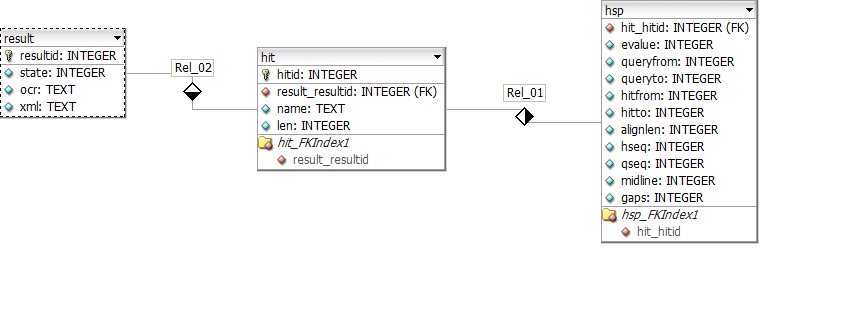


FIGURA 26 – DIAGRAMA ENTIDADE RELACIONAMENTO

Fonte: Os autores (2014).

**APÊNDICE D- DIAGRAMA DE CLASSES**

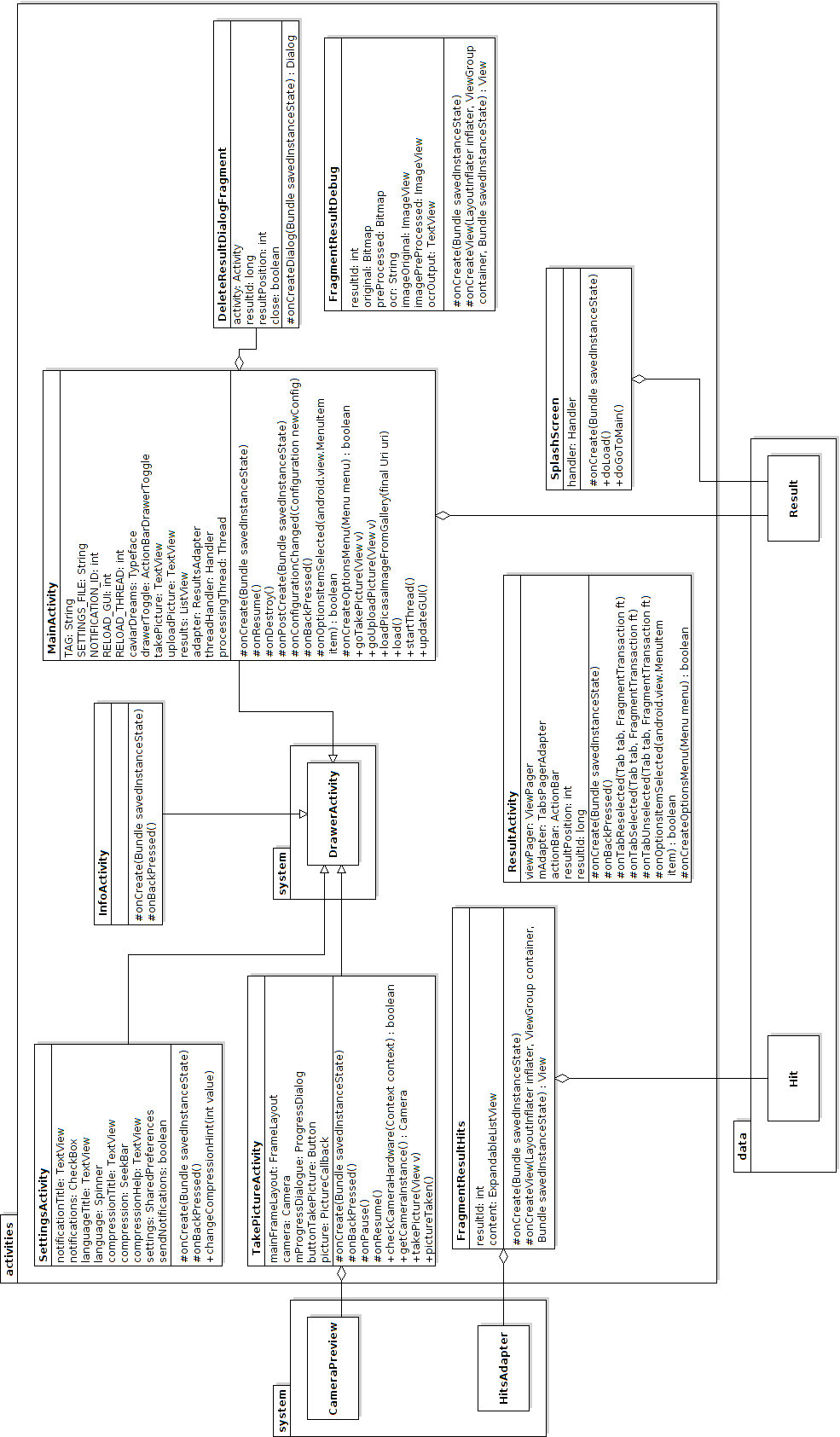
****

FIGURA 27 – DIAGRAMA DE CLASSES – PACOTE ACTIVITIES

Fonte: Os autores (2014).

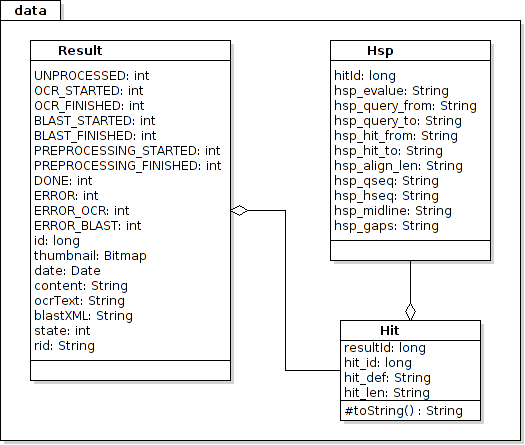


FIGURA 28 – DIAGRAMA DE CLASSES – PACOTE DATA

Fonte: Os autores (2014).

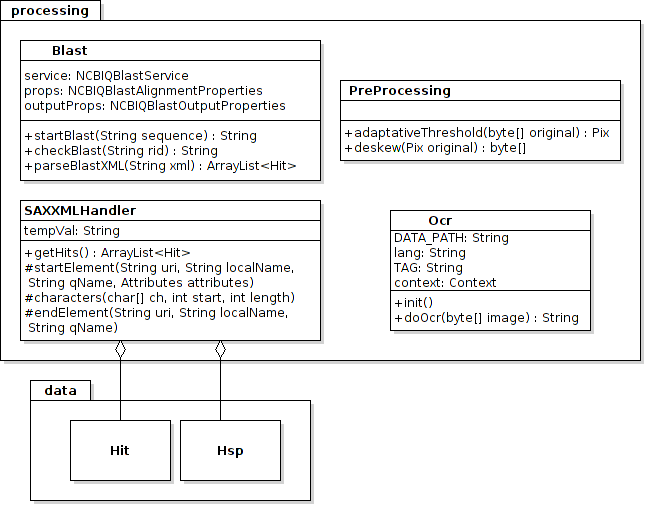


FIGURA 29 – DIAGRAMA DE CLASSES – PACOTE PROCESSING

Fonte: Os autores (2014).

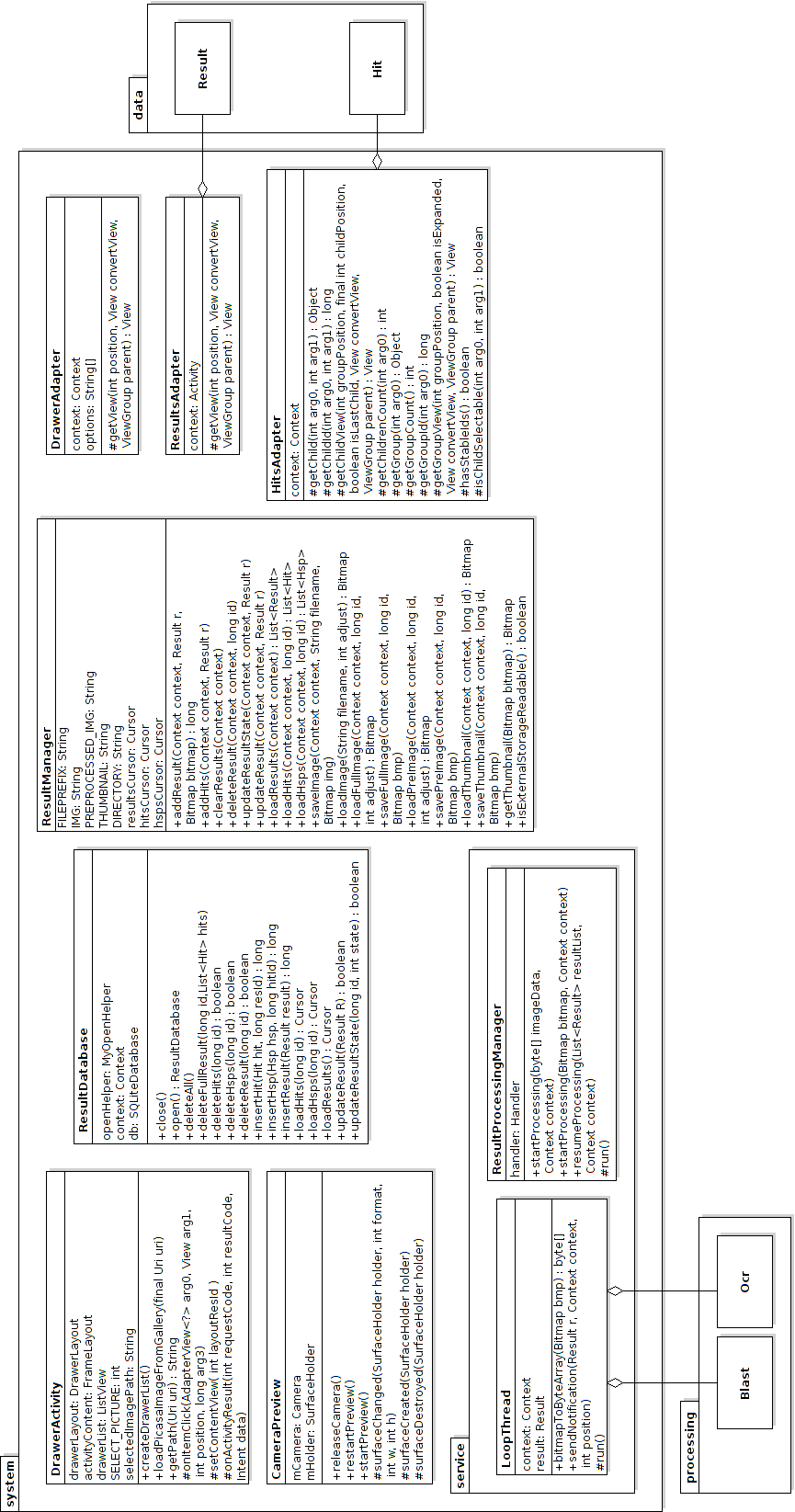


FIGURA 30 – DIAGRAMA DE CLASSES – PACOTE SYSTEM

Fonte: Os autores (2014).

**APÊNDICE E - DIAGRAMA DE ATIVIDADES**

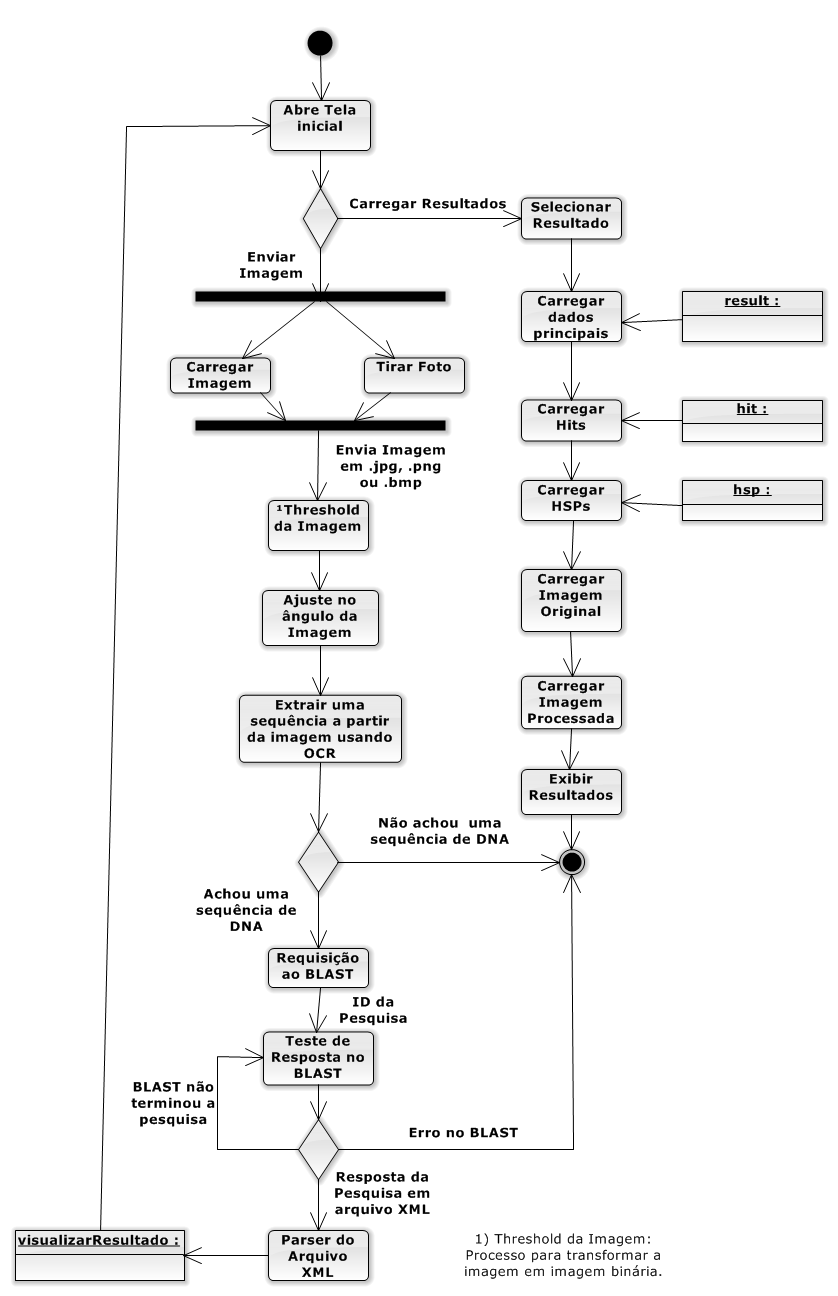
****

FIGURA 31 – DIAGRAMA DE ATIVIDADES – USO GERAL

Fonte: Os autores (2014).



FIGURA 32 – DIAGRAMA DE ATIVIDADES – ALTERAR CONFIGURAÇÕES

Fonte: Os autores (2014).

# APÊNDICE F - DIAGRAMA DE COMPONENTES

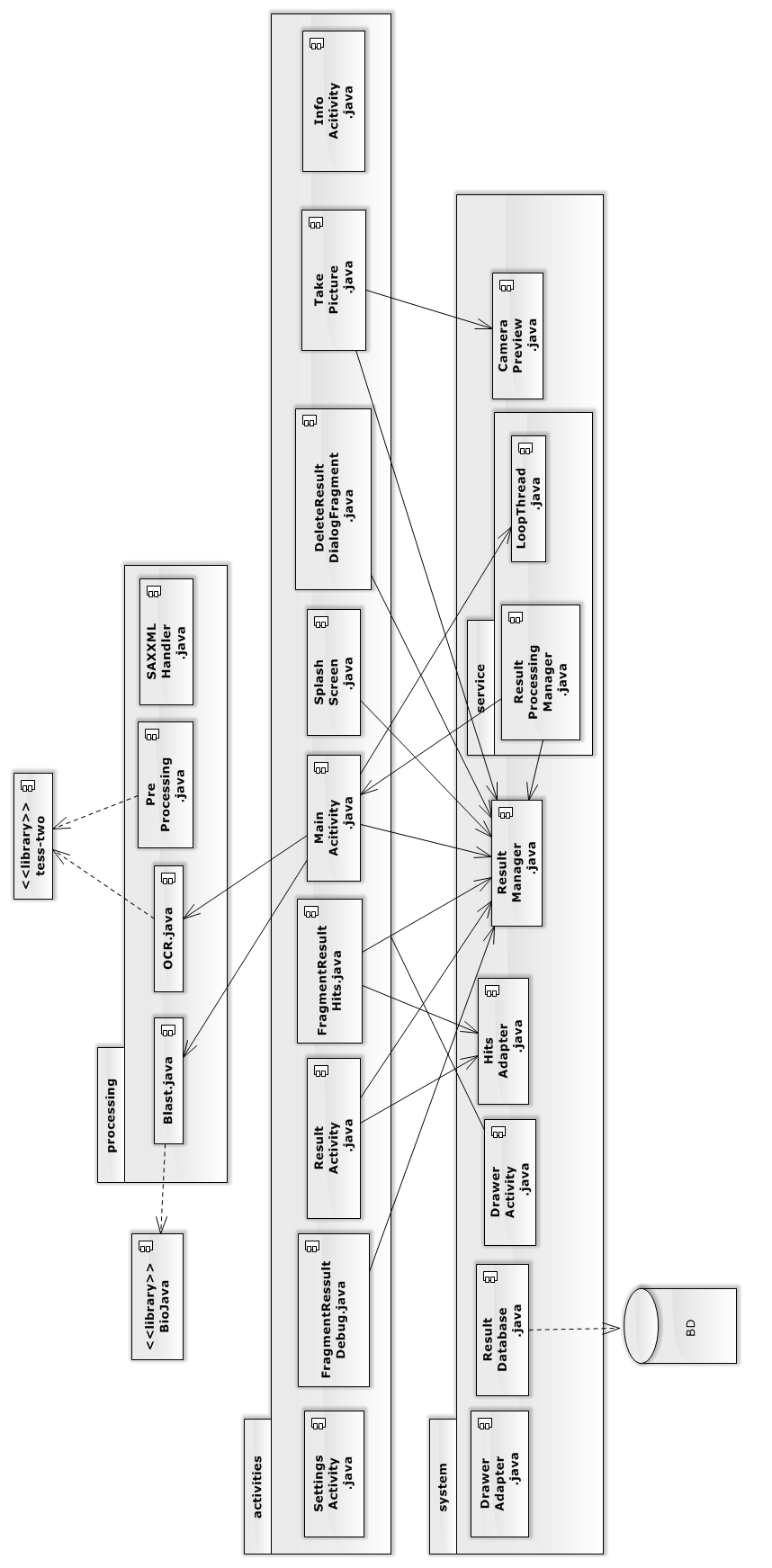


FIGURA 33 – DIAGRAMA DE COMPONENTES

Fonte: Os autores (2014).