UNIVERSIDADE FEDERAL DO PARANÁ

ALEXANDRE BELTZAC
RAUL CORREA HESS
SAMUEL YUJI YAMAGUCHI

DNA SHOT - RECONHECIMENTO DE DNA ATRAVÉS DE IMAGENS

CURITIBA
2014

ALEXANDRE BELTZAC
RAUL CORREA HESS
SAMUEL YUJI YAMAGUCHI

DNA SHOT – RECONHECIMENTO DE DNA ATRAVÉS DE IMAGENS

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: Roberto Tadeu Raditz
Co-Orientadora: Jeroniza Marchaukoski

CURITIBA
2014

RESUMO

O BLAST é um conjunto de algoritmos para comparar sequências biológicas por similaridade. O software pode utilizar as sequências de aminoácidos de uma proteína ou uma sequência de nucleotídeos de um DNA, identificando as sequências mais semelhantes do banco com a sequência fornecida dada na consulta. Este trabalho apresenta um aplicativo desenvolvido para o sistema operacional Android que realiza uma pesquisa no BLAST com uma sequência de nucleotídeos de um DNA a partir de uma foto tirada do próprio dispositivo ou de uma imagem qualquer. O aplicativo irá transformar a imagem em uma imagem binária para reconhecer os caracteres com OCR e enviar uma requisição ao BLAST, mostrando os melhores resultados, a imagem utilizada, a imagem após seu processamento e a sequência utilizada na pesquisa.

Palavra Chave: Bioinformática, Alinhamento de sequências, Visualização de dados biológicos, Biologia no Android

ABSTRACT

The BLAST is a set of algorithms to compare biological sequences by similarity. The software can use amino acid sequences of a protein or a nucleotide sequence of a DNA, identifying the most similar sequences of the database to the given query. This work presents an app made for Android operational system that perform a BLAST search with a nucleotide sequence of a DNA from a photo taken from the device or any other image. This app transforms the image into a binary image to recognize the characters with OCR and send a request to BLAST, showing the best results, the used image, the image after processing and the sequence used in the query.

Key-word: Bioinformatics, Sequence alignment, Visualization of biological data, Biology in Android

LISTA DE FIGURAS

LISTA DE SIGLAS

BLAST	-Basic Local Alignment Search Tool
blastn	-Nucleotide BLAST
DDBJ	-DNA Data bank of Japan
DNA	-Ácido desoxirribonucleico
EMBL	-European Molecular Biology Laboratory
FASTA	-Formato utilizado para armazenar sequências de bases e de aminoácidos em arquivo texto
GenBank	-Banco de dados público do National Center for Biology Information
HTML	-HyperText Markup Language
IDE	-Integrated Development Environment
mRNA	-RNA mensageiro
NCBI	-National Center for Biotechnology Information
NR	-Banco de dados não redundante disponibilizado pelo NCBI
OCR	-Optical Character Recognition
PDB	-Protein Research Data Bank
RNA	-Ácido ribonucleico
RUP	-Rational Unified Process
SILA	-Genome Automated Annotation System
UML	-Unified Modeling Language
WBS	-Work Breakdown Structure
XML	-Extensible Markup Language

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Bioinformática é uma ciência que atua na manipulação e extração de conhecimento de dados biológicos. O crescimento da área se deve à necessidade de manipular quantidades enormes de dados genéticos e bioquímicos, utilizando recursos computacionais para catalogá-las, organizá-las e estruturá-las (RASHIDI, 2000).

Na bioinformática, o alinhamento de sequências compara duas ou mais sequências, sejam nucleotídeos ou aminoácidos, com o objetivo de verificar sua similaridade. O BLAST (*Basic Local Alignment Search Tool*) é uma ferramenta para o alinhamento de sequências, executando um algoritmo simples, heurístico e uma base local com apenas uma sequência de entrada. É o programa de alinhamento mais utilizado no mundo (RASHIDI, 2000; PRODOSCIMI, 2003).

A acessibilidade dos dados digitais, independente de local e fios tem evoluído muito nos últimos anos, o que se deve em grande parte pela tecnologia dos dispositivos móveis. Eles fornecem capacidades avançadas que vão além de um celular comum. Com telas maiores e processadores mais avançados, eles utilizam um sistema operacional completo com uma interface padronizada e uma plataforma para desenvolvedores de aplicativos (PHONESCOOP, 2014).

Atualmente existem poucos aplicativos relacionados à bioinformática para smartphones, por isso a equipe propôs o desenvolvimento de um aplicativo que usa as vantagens de um smartphone para realizar o alinhamento de sequências de uma sequência de DNA obtida a partir de uma foto. O alinhamento de sequências será realizado no BLAST via webservice.

1.1 OBJETIVOS

Desenvolver um aplicativo na plataforma Android para reconhecimento de uma sequência de DNA (*Ácido desoxirribonucleico*) com o alinhamento de sequências do blastn (*Nucleotide BLAST*) através de uma foto ou imagem, para que o usuário não necessite dar uma entrada de texto.

1.2 OBJETIVOS ESPECÍFICOS

Este trabalho tem os seguintes objetivos específicos:

- Utilizar a câmera e da galeria de imagens do dispositivo móvel para dar entrada no processamento do aplicativo.
- Aplicar Thresholding para diminuir possíveis sombreados na foto ou na imagem.
- Detectar da inclinação do texto na foto ou imagem e girá-la para o ângulo correto.
- Aplicar de OCR (*Optical Character Recognition*) na imagem processada e aplicação de filtro para eliminar caracteres desnecessários.
- Iniciar uma pesquisa no blastn com a sequência detectada pelo OCR.
- Mostrar os resultados da pesquisa ao usuário.

2 FUNDAMENTAÇÃO TEÓRICA

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

2.1 BIOINFORMÁTICA

Bioinformática envolve a integração de computadores, softwares e banco de dados com o objetivo de tratar questões biológicas (FOX, 2006). Ela organiza dados biológicos relacionados a genomas com o intuito de aplicar essa informação na agricultura, na farmácia e outras aplicações comerciais (MOUNT, 2004).

Duas atividades importantes na Bioinformática são a genômica e a proteômica. A genômica refere-se ao sequenciamento e análise de genomas que abrange o conjunto completo de sequências de DNA que codificam o material hereditário passado de geração para geração. Essas sequências de DNA incluem todos os genes e os seus transcritos. Já a proteômica trata da análise do conjunto completo de proteínas que os genes expressam (FOX, 2006).

2.2 CONCEITOS BIOLÓGICOS

O DNA é uma molécula de cadeia dupla em espiral que contém uma sequência de quatro nucleotídeos—A (adenina), G (guanina), C (citosina), G (timina)—em cada cadeia. O nucleotídeo A sempre está ligado ao T, e o G sempre está ligado ao C. A interação química destes pares de nucletídeos mantêm a cadeia unida (MOUNT, 2004).

Os genes codificam os tipos de proteínas que são traduzidas pela células. Uma cadeia de DNA é copiada para classes de moléculas RNA (*Ácido ribonucleico*) chamada RNA mensageiro (mRNA), que carrega a informação intermédiária para a síntese de proteína. A síntese de proteínas que é traduzida de acordo com as instruções dadas pelos modelos do mRNA é resultado desse processo de transcrição seguido de um processo de tradução (BERG, 2006).

2.3 CONSULTA DE INFORMAÇÕES BIOLÓGICAS

O objetivo desta parte do capítulo é entender um pouco do que é enviado e recebido externamente.

2.3.1 Armazenamento

O formato FASTA é um formato baseado em um texto que representa tanto sequências de nucleotídeos quanto sequências de peptídeos em que cada base de pares e aminoácidos são representados por um código composto por uma letra. A sequência no formato FASTA começa com a descrição na primeira linha, as linhas seguintes contêm os dados da sequência. A linha de descrição é diferenciada dos dados da sequência pelo símbolo “maior que” (>) na primeira coluna (ZHANGLAB, 2014). A FIGURA 1 mostra como é um arquivo FASTA de uma proteína.

```
>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken
ADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTID
FPEFLTMMARKMKDTDSEEEIREAFRVFDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA
DIDGDGQVNYEEFVQMMTAK*
```

FIGURA 1 – Arquivo FASTA

Fonte: Wikipédia (2014).

2.3.2 Banco de Dados

O GenBank é um banco de dados público criado pelo NCBI (*National Center for Biotechnology Information*). O NR (non-redundant) é o maior banco de sequências de nucleotídeos disponível pelo NCBI, juntando dados de todos os bancos do GenBank, do EMBL (*European Molecular Biology Laboratory*), do DDBJ (*DNA Data bank of Japan*), e do PDB *(Protein Research Data Bank*). Este banco deixou de ser não redundante devido ao custo computacional (NCBI, 2014).

2.3.3 BLAST

BLAST (*Basic Local Alignment Search Tool*) é um algoritmo para comparar sequências biológicas primárias, como sequências de aminoácidos de proteínas ou sequências de nucleotídeos de um DNA. O BLAST é um conjunto de programas que depende do tipo da sequência utilizada, do propósito da pesquisa e do banco de dados a ser utilizado (KORF, 2003). A TABELA 1 mostra os tipos de pesquisa que podem ser feitos a partir do BLAST.

TABELA 1 – LISTA DE ALGORITMOS DO BLAST (BAXEVANIS, 2005).

Programa	**Pesquisa**	**Banco de dados**
BLASTN	Nucletídeo	Nucletídeo
BLASTP	Proteína	Proteína
BLASTX	Nucletídeo, tradução de seis quadros	Proteína
TBLASTN	Proteína	Nucletídeo, tradução de seis quadros
TBLASTX	Nucletídeo, tradução de seis quadros	Nucletídeo, tradução de seis quadros

No aplicativo será utilizado o “blastn”, ele utiliza como entrada uma sequência de nucleotídeos e retorna os resultados mais próximos do banco de dados NR.

A sequência de entrada no BLAST deve estar no formato de sequência FASTA e os resultados são retornados em HTML (*HyperText Markup Language*), texto simples ou XML (*Extensible Markup Language*) (KORF, 2003) .

Os dados retornados pelo BLAST são os acertos, chamados de *hits* e dentro deles, temos os emparelhamentos mais próximos da sequência pesquisada, esses emparelhamentos são chamados de HSP (*High-scoring segment pair*) (BAXEVANIS, 2005).

Após a identificação de um HSP é importante verificar se o alinhamento resultante é realmente significante. Usando a contagem cumulativa do alinhamento, junto ao número de vários outros parâmetros, um novo valor chamado E é calculado. Para cada hit, o E dá o número de HSPs tendo a contagem limiar ou maior que o BLAST achou meramente por chance. Por outro lado, o valor de E fornece medidas para verificar se o HSP reportado é um falso positivo. Quanto menor for o valor de E, maior será sua significância biológica. (BAXEVANIS, 2005)

O valor E é resultado de uma fórmula apresentada por Karlin e Altschul. Este valor é obtido através da fórmula $E = kmNe^{-\lambda s}$, onde k é uma constante menor, m é o número de letras na sequência da pesquisa, N é o total de letras no banco de dados selecionado, e ʎ é uma constante usada para normalizar a contagem bruta do HSP, com o valor de ʎ variando dependendo da matriz da contagem utilizada, e S é a contagem do HSP (BAXEVANIS, 2005).

2.4 OCR

O OCR (*Optical Character Recognition*) é uma tecnologia para reconhecer caracteres a partir de um arquivo de imagem ou mapa de bits. A partir deste reconhecimento é possível adquirir um texto editável. Atualmente existem várias bibliotecas e programas para utilizar o OCR (ABBYY, 2014).

2.5 PROCESSAMENTO DE IMAGENS

Processamento de imagens é uma forma de processamento de sinais onde a entrada é uma imagem, uma foto ou um quadro de vídeo. A saída pode ser uma imagem ou uma relação de características ou parâmetros relacionados à imagem. A maioria das técnicas de processamento de imagens envolve o tratamento da imagem como um sinal bidimensional e aplicando técnicas comuns de processamento de sinais. O processamento digital de imagens usa algoritmos para realizar esse mesmo processamento em imagens digitais (ACHARYA, 2005).

O *thresholding* é o método mais simples de segmentação de imagens. de uma imagem em escala de tons de cinza pode-se obter uma imagem binária. O *thresholding* pode ser usado também em imagens coloridas (MATHWORKS, 2014). Ele é usado para separar caracteres do fundo da imagem, algoritmos mais avançados, como o método Sauvola (1999) utilizado, tem tolerância a desuniformidade de luz sobre o objeto, melhorando a binarização e consequentemente evitando possíveis erros no OCR (SAUVOLA, 1999).

Existe um algoritmo na biblioteca de processamento de imagens Leptonica que detecta o ângulo de inclinação do texto através da soma de pixels em uma linha, esse ângulo é utilizado na rotação da imagem, preparando-a para a aplicação do OCR (LEPTONICA, 2014).

2.6 SQLITE

SQlite é uma biblioteca desenvolvida na linguagem C que implementa um banco de dados SQL. Aplicações que utilizam a biblioteca do SQLite tem acesso a um banco de dados SQL sem a necessidade de um SGBD externo (SQLITE, 2014).

Ao invés de conectar-se a um servidor de banco de dados, o SQLite executa o papel de servidor, grava e consulta os dados diretamente para um arquivo em disco com um limite de 2 terabytes (SQLITE, 2014).

O SQLite é um software livre de domínio público e multiplataforma, além de não possuir dependências externas. Esta biblioteca foi escolhida pela equipe por

estar embutida ao Android e por sua simplicidade, já que não há necessidade de instalação, configuração ou administração.

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.

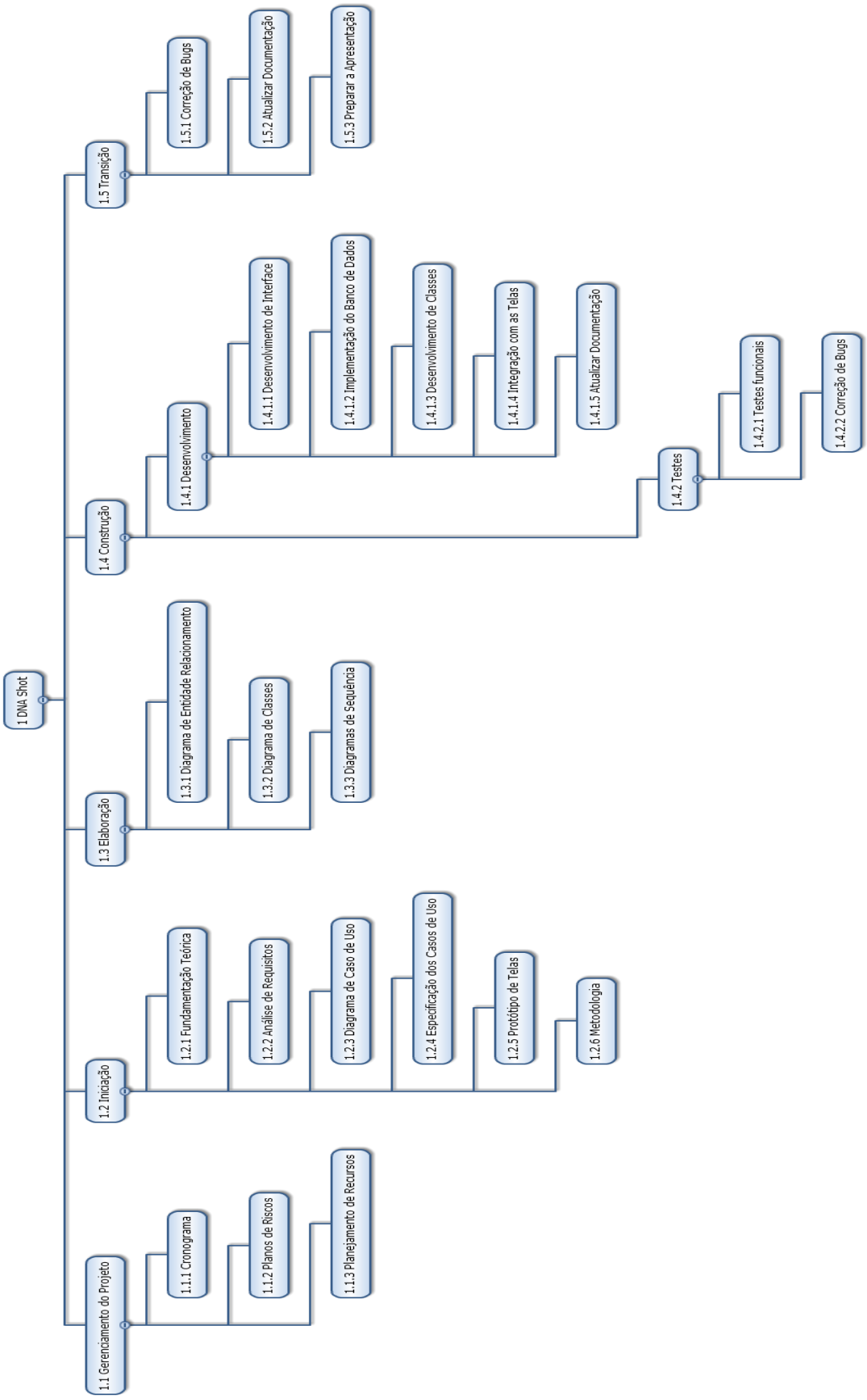

FIGURA 2 – DIAGRAMA WBS

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.

		Nome	Duração	Início	Fim	Antecessores	Nomes dos Recursos
1		**DNA Shot**	**73,875 dias**	**26/08/14 09:00**	**05/12/14 17:00**		
2		**Gerenciamento do Projeto**	**4 dias**	**26/08/14 09:00**	**01/09/14 09:00**		
3		Planejamento de Recursos	3 dias	26/08/14 09:00	29/08/14 09:00		Alexandre
4		Plano de Riscos	3 dias	26/08/14 09:00	29/08/14 09:00		Samuel
5		Cronograma	4 dias	26/08/14 09:00	01/09/14 09:00		Raul
6		**Iniciação**	**21 dias**	**26/08/14 09:00**	**24/09/14 09:00**		
7		Fundamentação Teórica	6 dias	02/09/14 09:00	10/09/14 09:00		Alexandre;Raul;Samuel
8		Análise de Requisitos	2 dias	26/08/14 09:00	28/08/14 09:00		Alexandre;Raul;Samuel
9		Diagrama de Casos de Uso	2 dias	28/08/14 09:00	01/09/14 09:00	8	Alexandre
10		Especificação dos Casos de Uso	2,5 dias	01/09/14 09:00	03/09/14 14:00	9	Alexandre;Samuel
11		Protótipo de Telas	4 dias	01/09/14 09:00	05/09/14 09:00	9	Raul
12		Metodologia	21 dias	26/08/14 09:00	24/09/14 09:00		Alexandre;Samuel
13		**Elaboração**	**11 dias**	**09/09/14 09:00**	**24/09/14 09:00**		
14		Diagrama Entidade Relacionamento	3 dias	09/09/14 09:00	12/09/14 09:00		Alexandre;Raul
15		Diagrama de Classes	3 dias	12/09/14 09:00	17/09/14 09:00	9	Raul;Samuel
16		Diagrama de Sequência	5 dias	17/09/14 09:00	24/09/14 09:00	11	Alexandre;Samuel
17		**Construção**	**42 dias**	**24/09/14 09:00**	**21/11/14 09:00**		
18		**Desenvolvimento**	**35,875 dias**	**24/09/14 09:00**	**12/11/14 17:00**		
19		Desenvolvimento de Interfaces	7 dias	24/09/14 09:00	03/10/14 09:00		Alexandre;Raul
20		Implementação do Banco de Dados	5 dias	03/10/14 09:00	10/10/14 09:00		Samuel
21		Desenvolvimento de Classes	14 dias	10/10/14 09:00	30/10/14 09:00	15	Alexandre;Raul;Samuel
22		Integração com as Telas	7 dias	30/10/14 09:00	10/11/14 09:00	21	Alexandre;Raul
23		Atualizar Documentação	3 dias	10/11/14 08:00	12/11/14 17:00		Samuel
24		**Testes**	**7 dias**	**12/11/14 09:00**	**21/11/14 09:00**		
25		Testes Funcionais	3 dias	12/11/14 09:00	17/11/14 09:00	22	Alexandre;Raul;Samuel
26		Correção de Bugs	4 dias	17/11/14 09:00	21/11/14 09:00	25	Alexandre;Raul
27		**Transição**	**11 dias**	**21/11/14 08:00**	**05/12/14 17:00**		
28		Correção de Bugs	4 dias	21/11/14 08:00	26/11/14 17:00		Alexandre;Raul
29		Atualizar Documentação	4 dias	26/11/14 08:00	01/12/14 17:00		Samuel
30		Preparar a Apresentação	4 dias	02/12/14 08:00	05/12/14 17:00		Alexandre;Raul;Samuel

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.

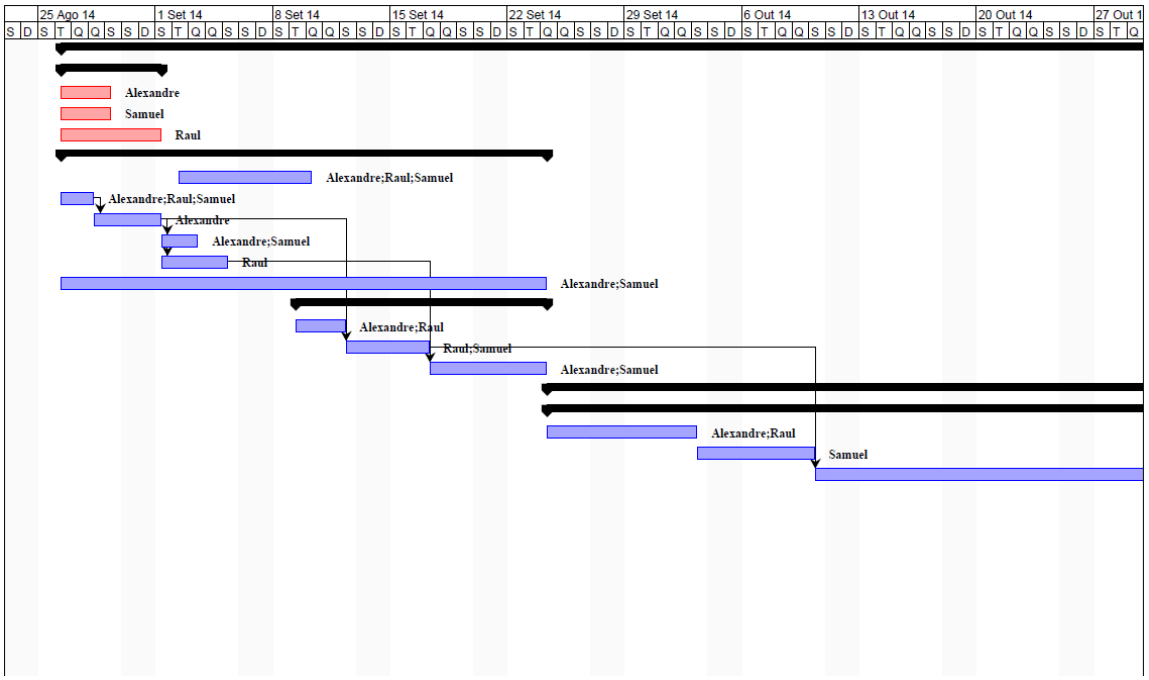

FIGURA 4 – LINHA DO TEMPO DO DIAGRAMADE GANTT

Fonte: Os autores (2014).

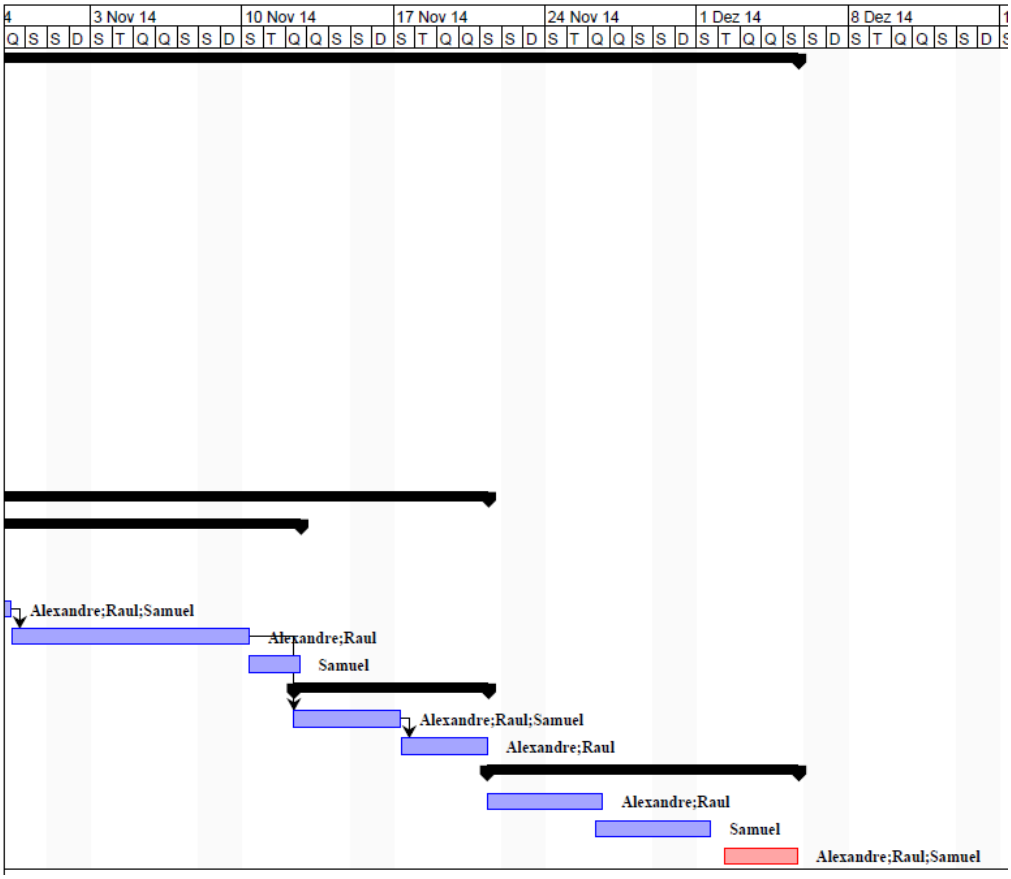

FIGURA 5 – LINHA DO TEMPO DO DIAGRAMADE GANTT

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.

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.

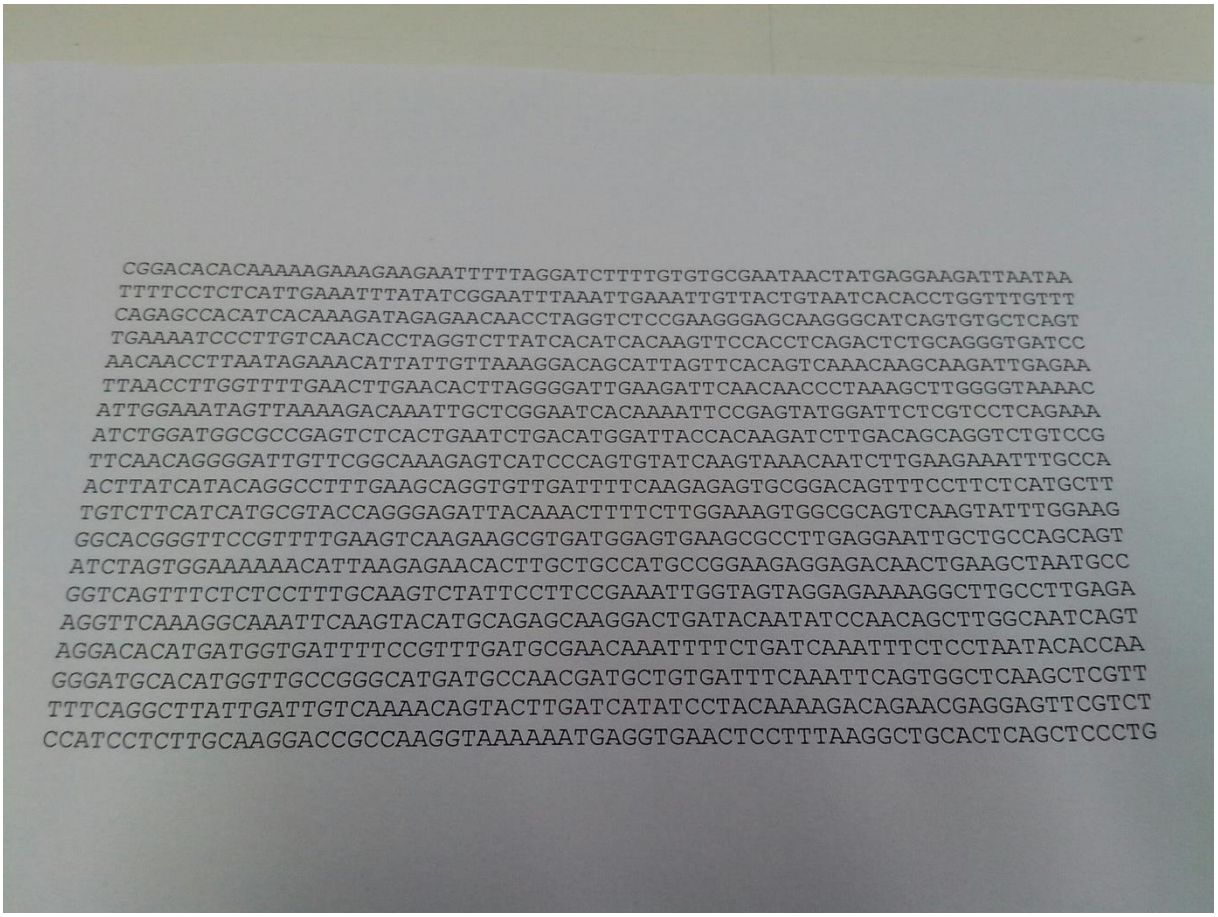

FIGURA 6 – FOTO TIRADA NO NÍVEL 1 DE COMPRESSÃO

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.

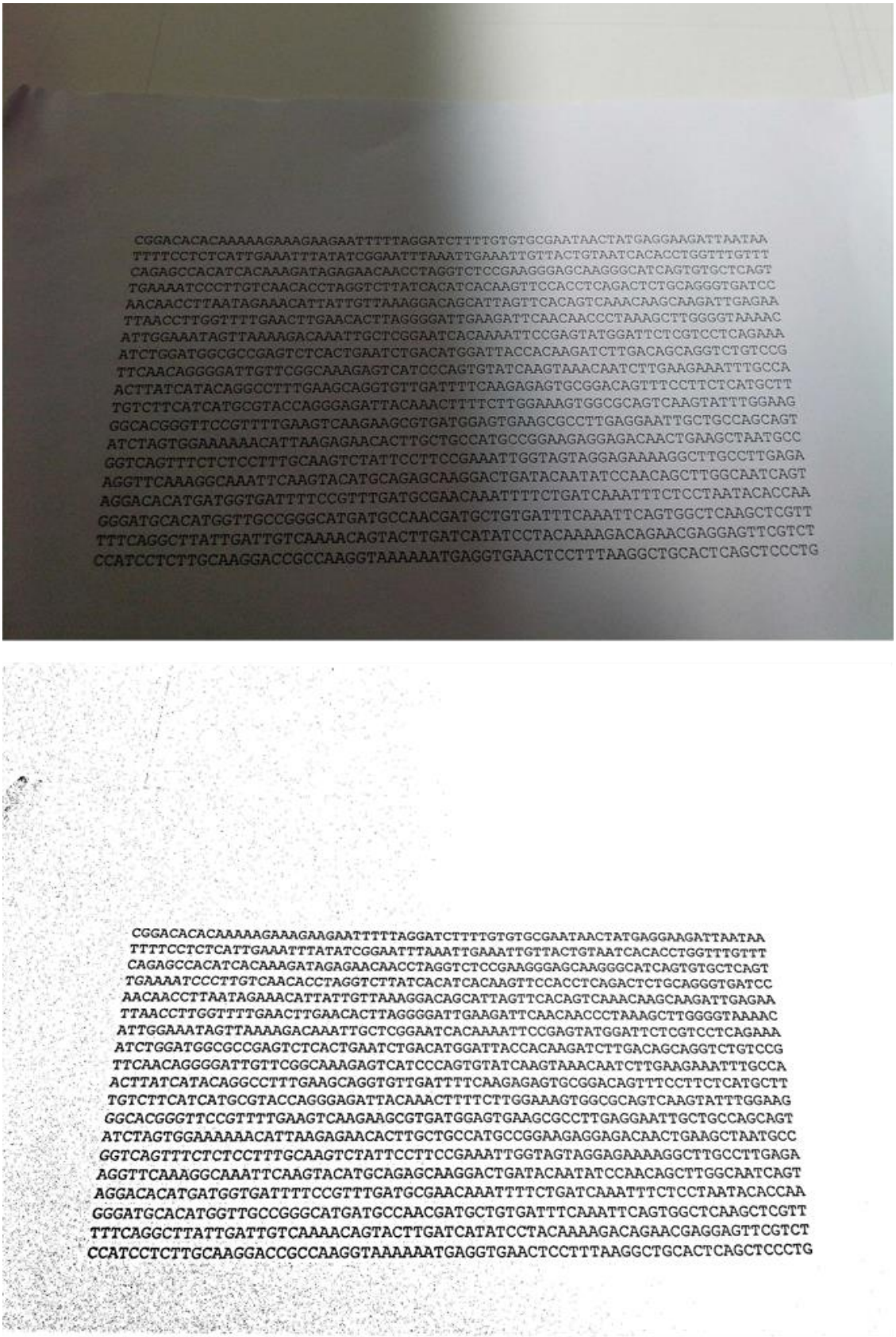

FIGURA 8 – COMPARAÇÃO ENTRE A FOTO ORIGINAL E A IMAGEM APÓS O PROCESSAMENTO DA IMAGEM

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.

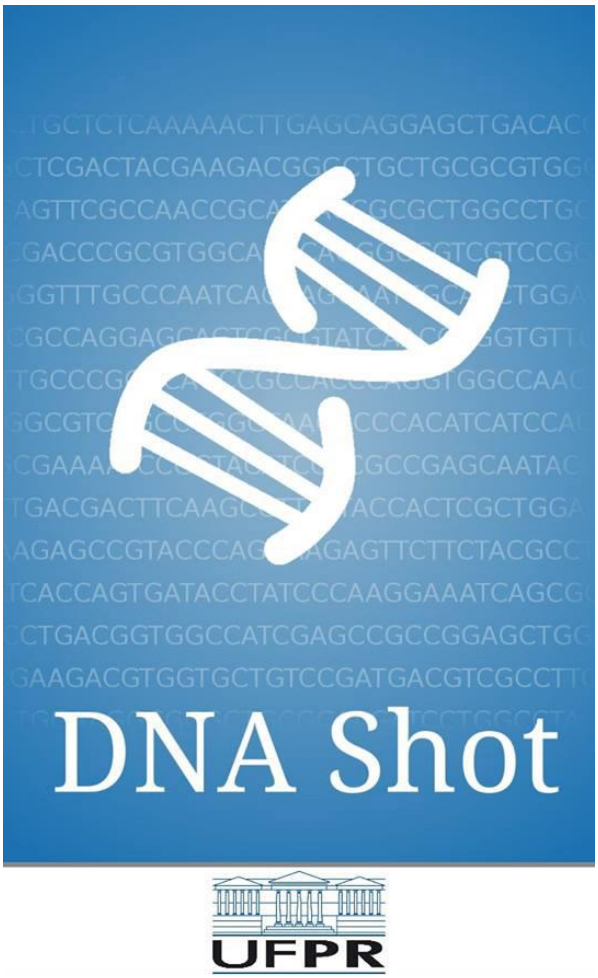

FIGURA 9 - TELA DE ABERTURA

Fonte: Os autores (2014).

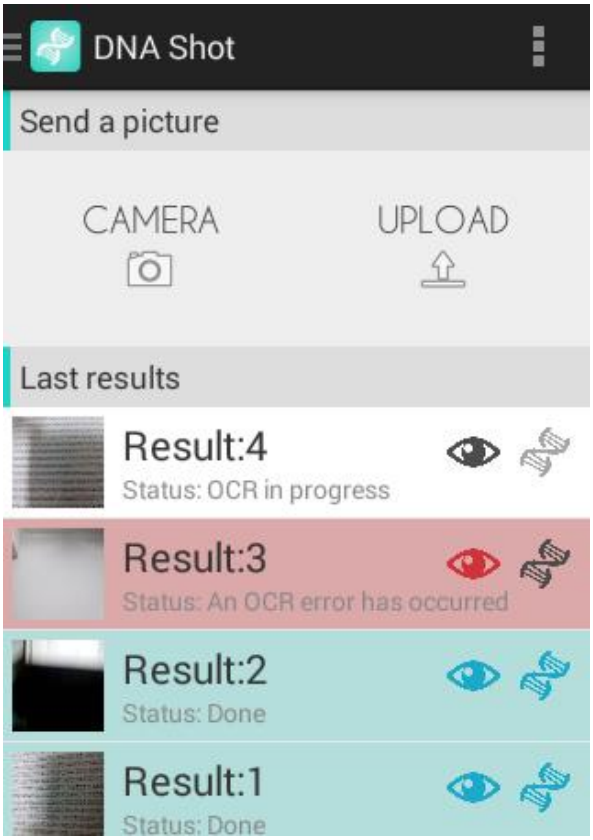

FIGURA 10 - TELA INICIAL DO APLICATIVO

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.

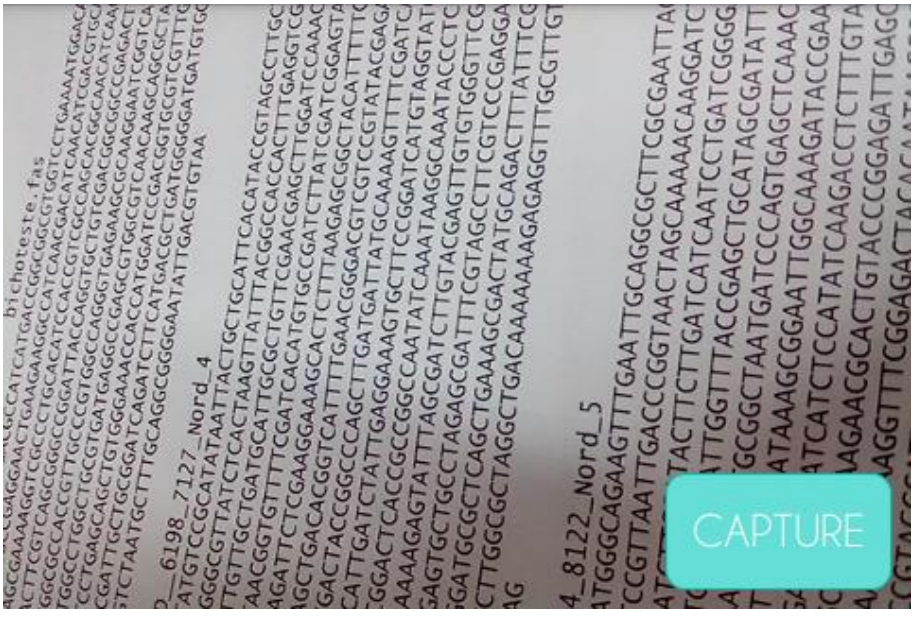

FIGURA 11 - TELA DA CÂMERA

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.

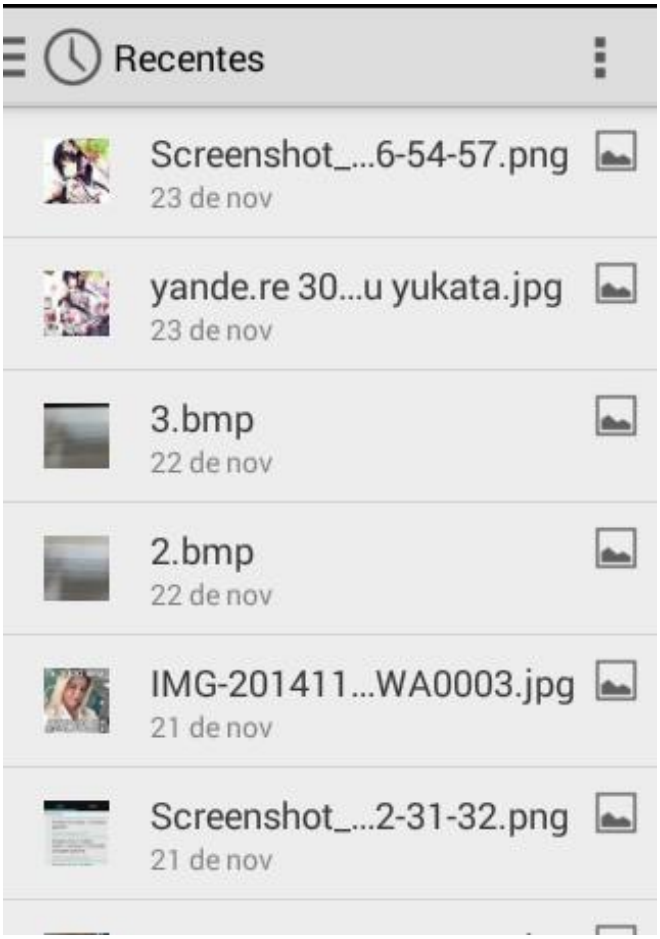

FIGURA 12 - TELA PARA SELECIONAR UMA IMAGEM

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.

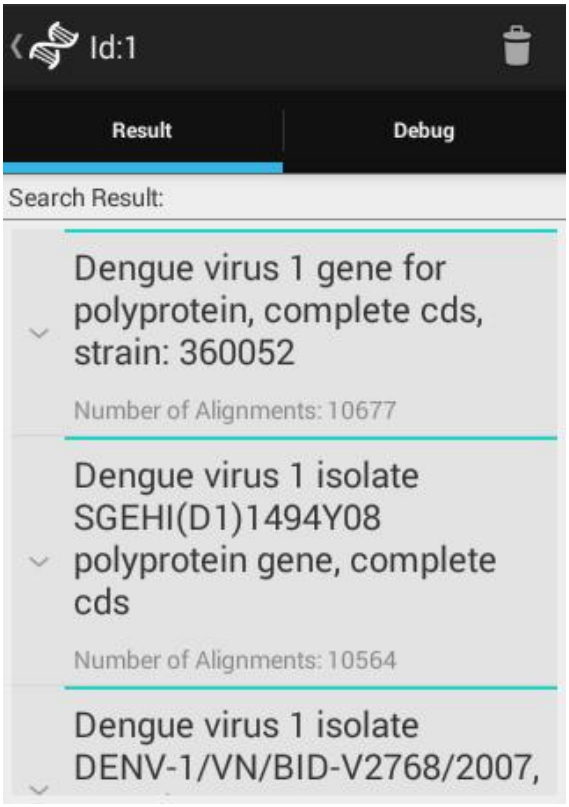

FIGURA 13 - TELA DE RESULTADOS

Fonte: Os autores (2014).

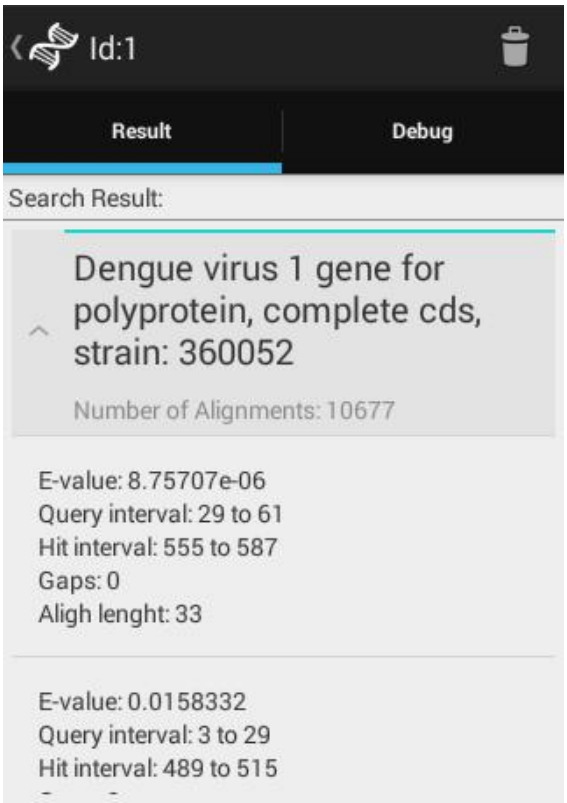

FIGURA 14 - TELA DE RESULTADOS COM HSPS

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.

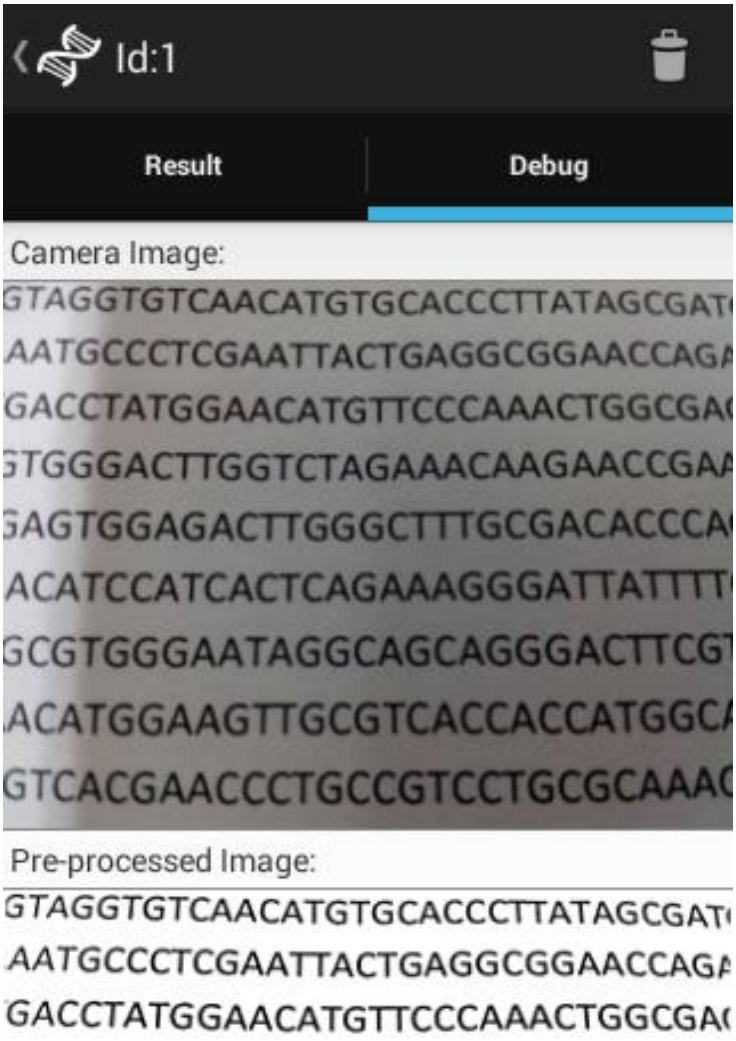

FIGURA 15 - IMAGEM ORIGINAL NA ABA “DEBUG”

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.

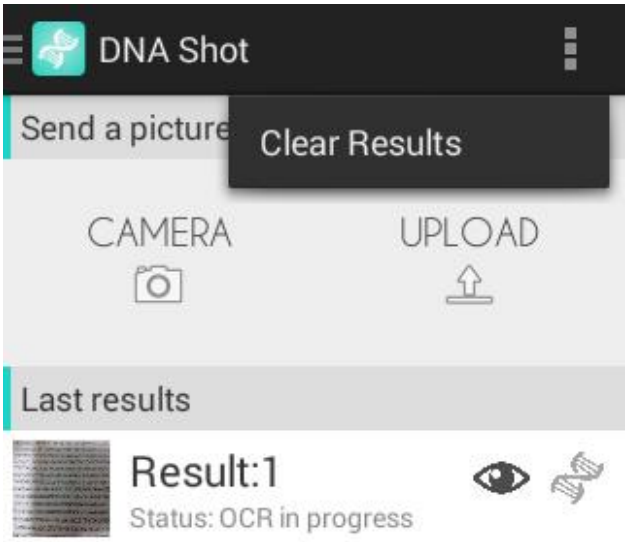

FIGURA 18 - BOTÃO PARA LIMPAR OS RESULTADOS

Fonte: Os autores (2014).

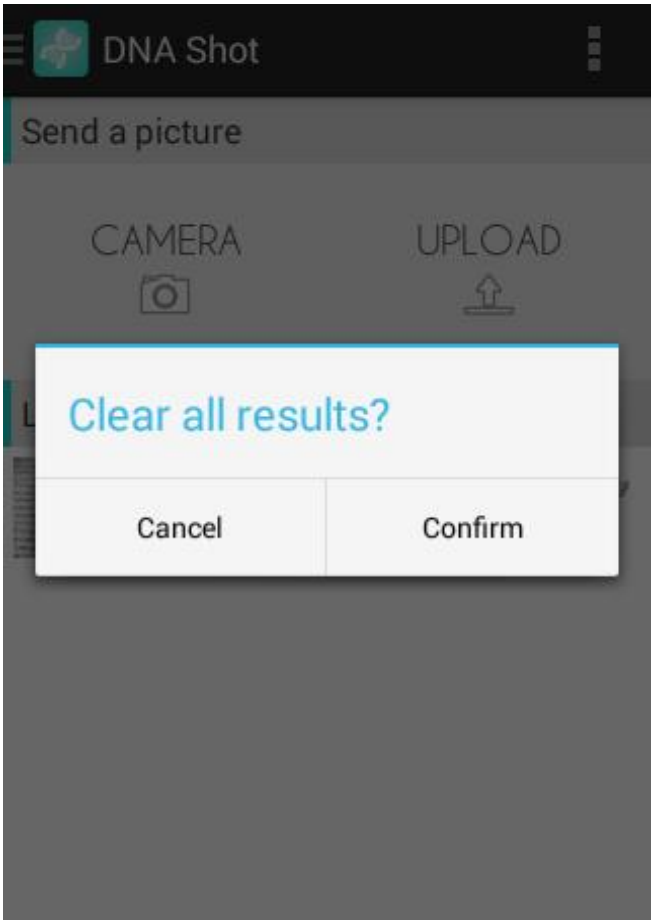

FIGURA 19 - ALERTA DE CONFIRMAÇÃO

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.

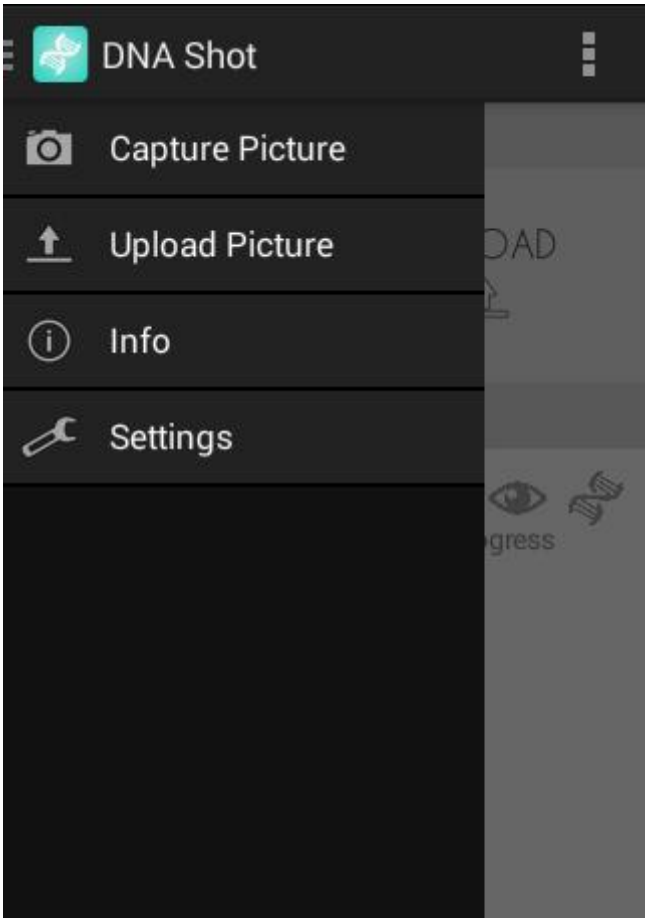

FIGURA 20 - MENU NA TELA INICIAL DO APLICATIVO

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).

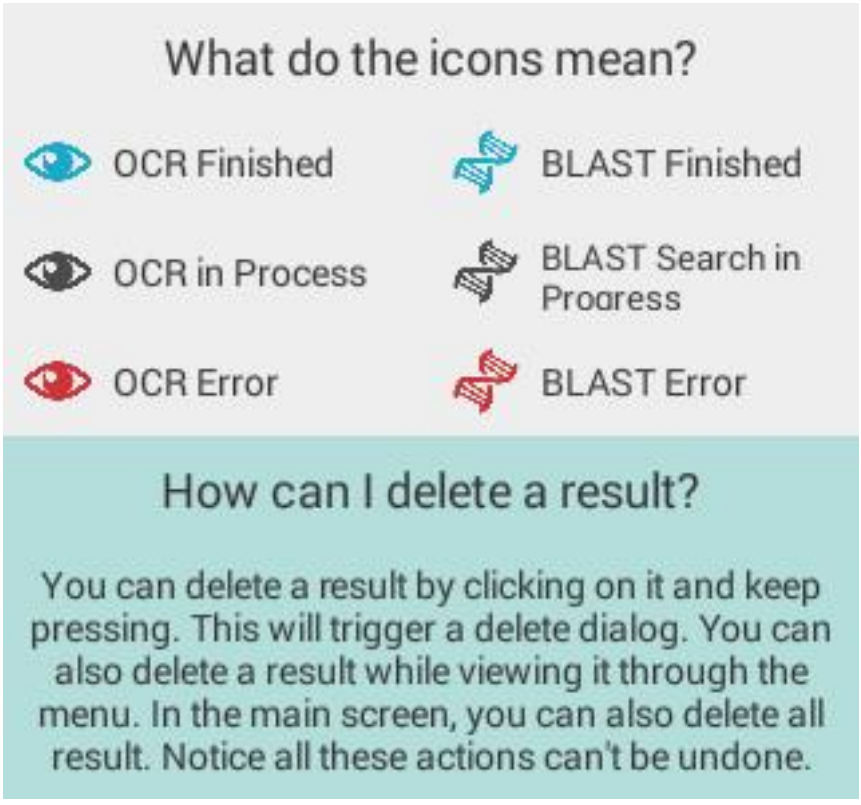

FIGURA 22 - TELA DE INFORMAÇÕES GERAIS

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.

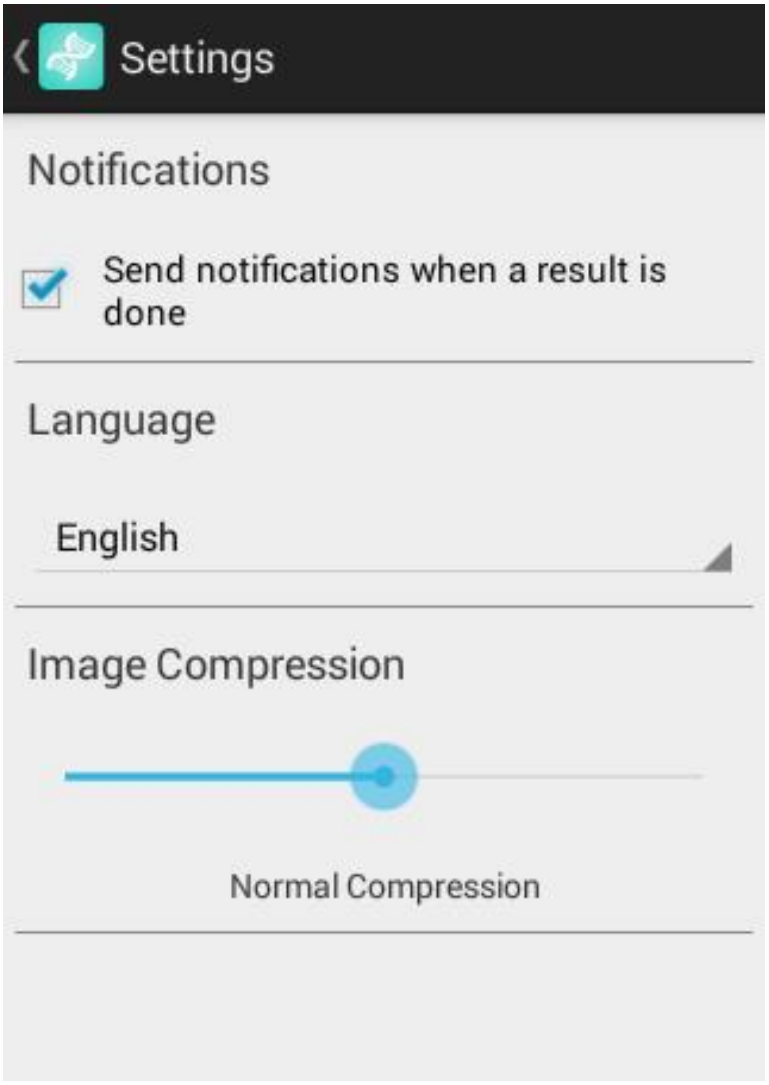

FIGURA 23 - TELA DE CONFIGURAÇÕES

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

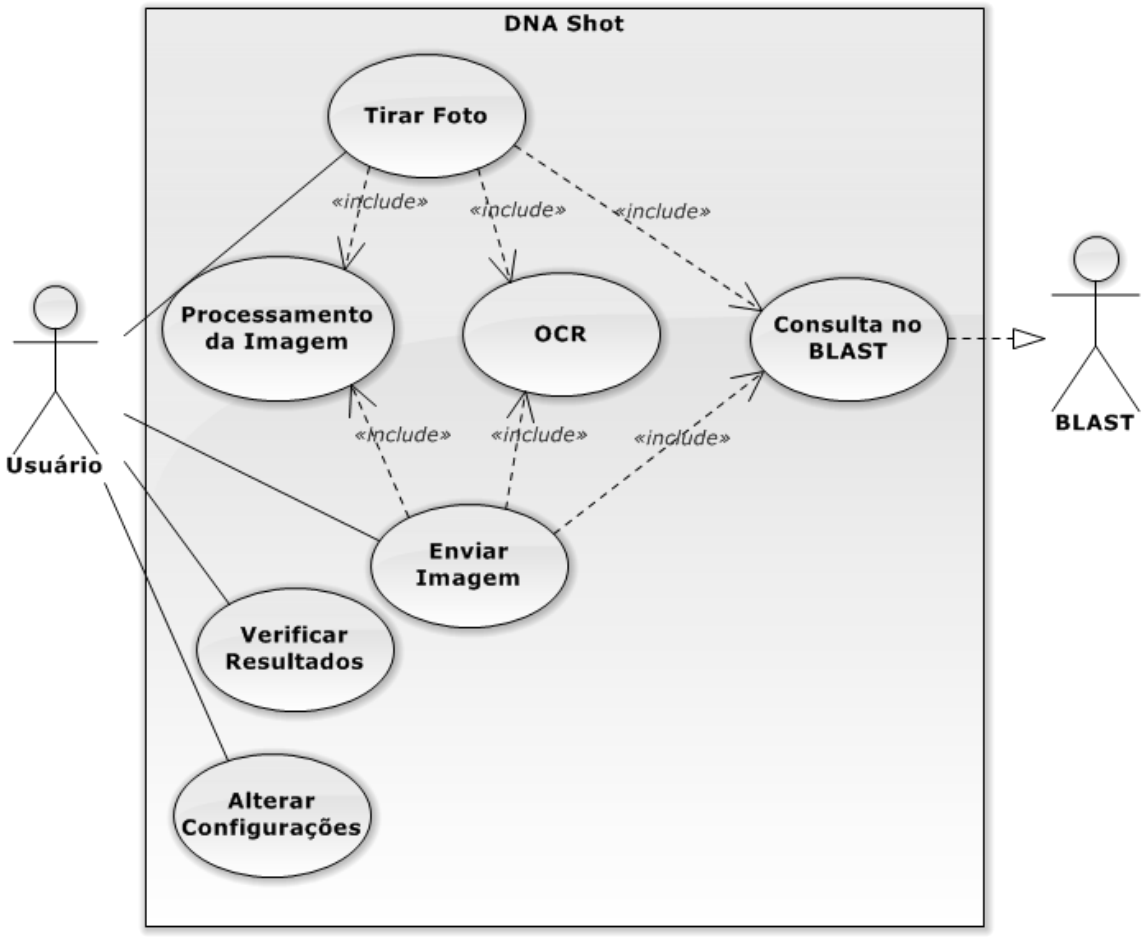

FIGURA 25 – DIAGRAMA DE CASOS DE USO

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

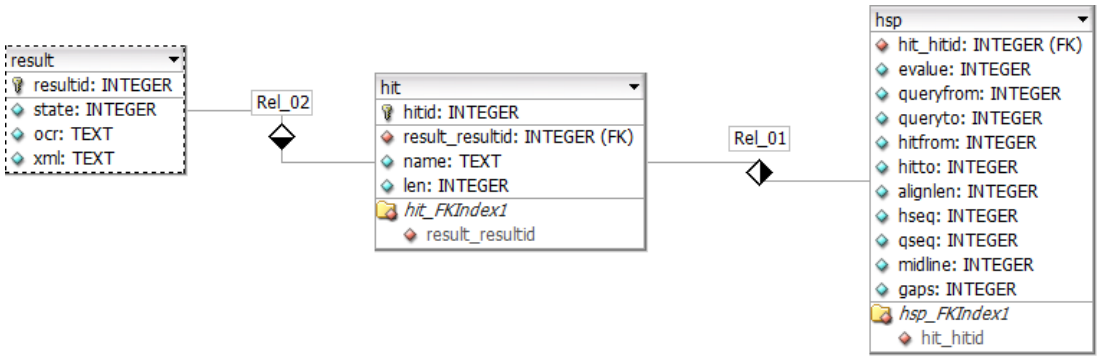

FIGURA 26 – DIAGRAMA ENTIDADE RELACIONAMENTO

Fonte: Os autores (2014).

APÊNDICE D- DIAGRAMA DE CLASSES

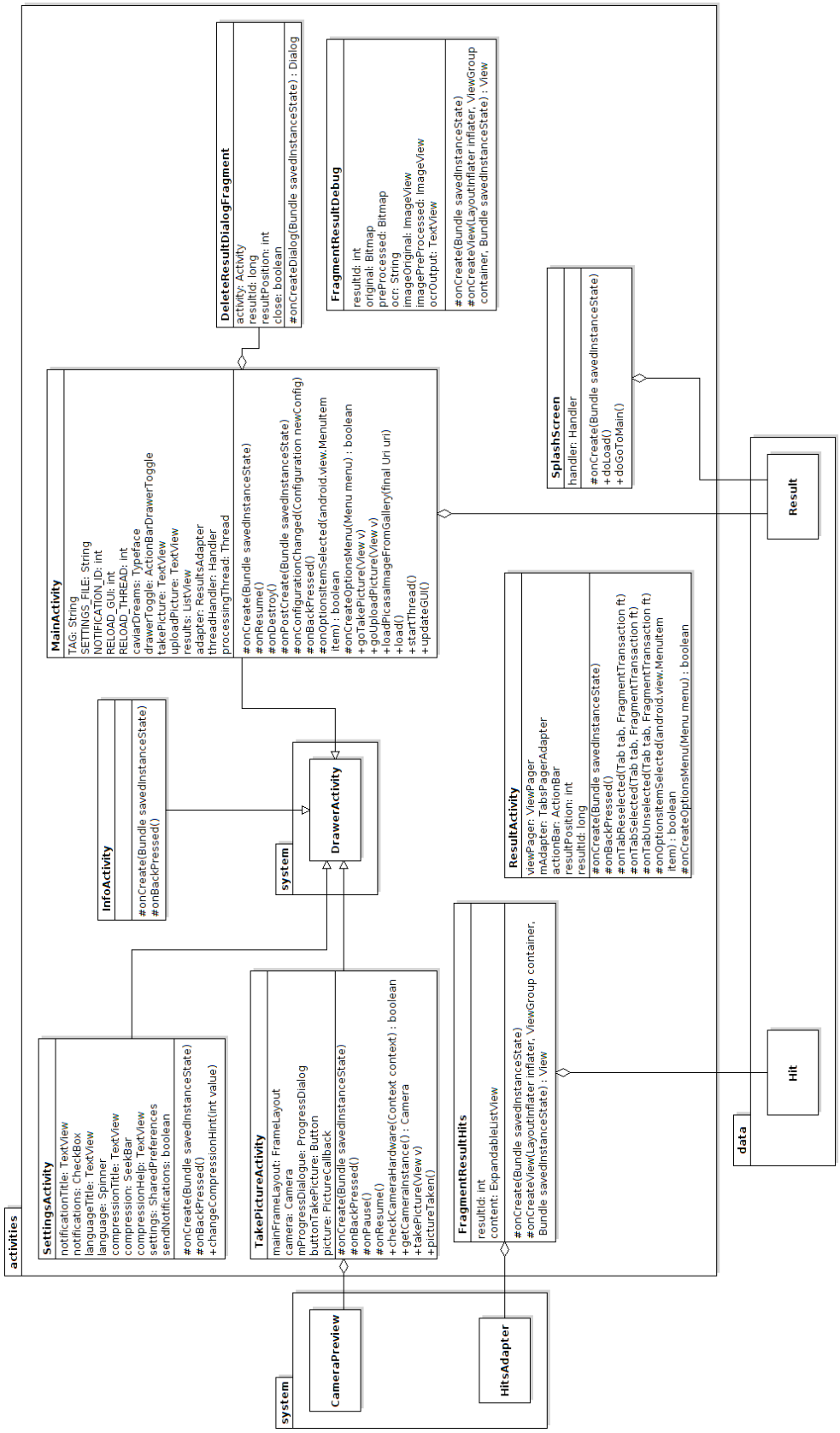

FIGURA 27 – DIAGRAMA DE CLASSES – PACOTE ACTIVITIES

Fonte: Os autores (2014).

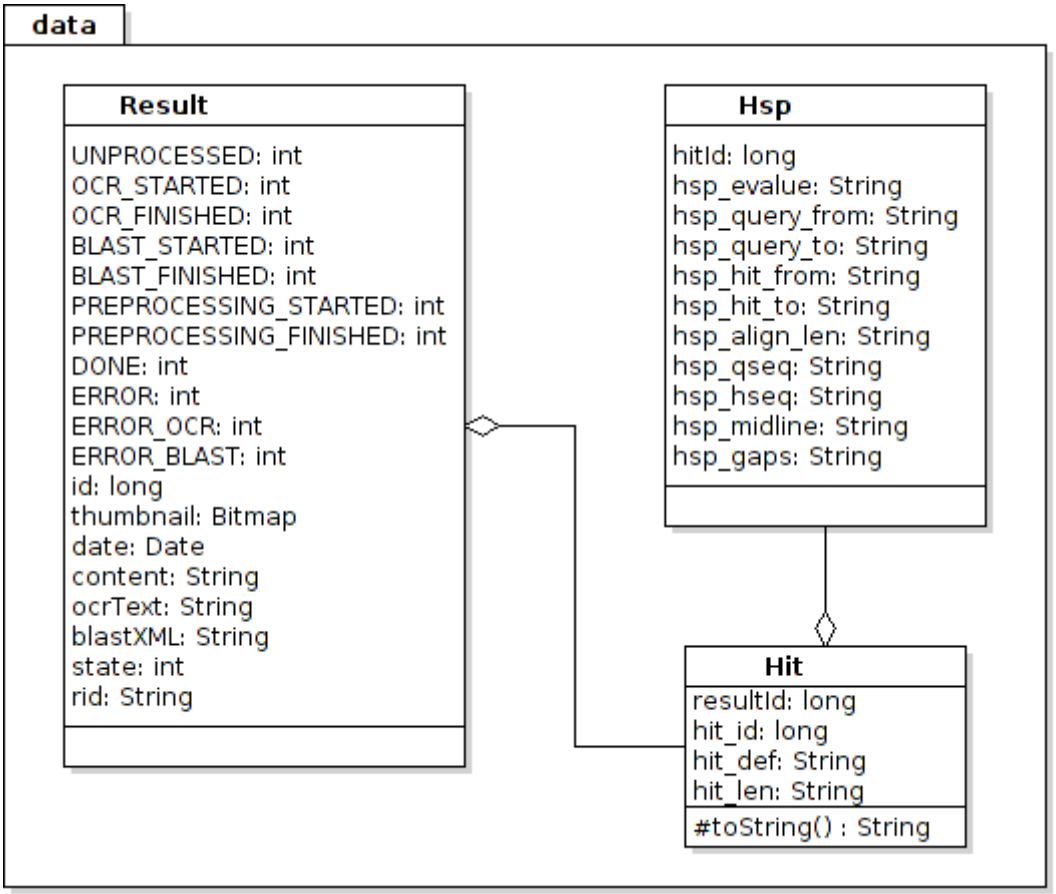

FIGURA 28 – DIAGRAMA DE CLASSES – PACOTE DATA

Fonte: Os autores (2014).

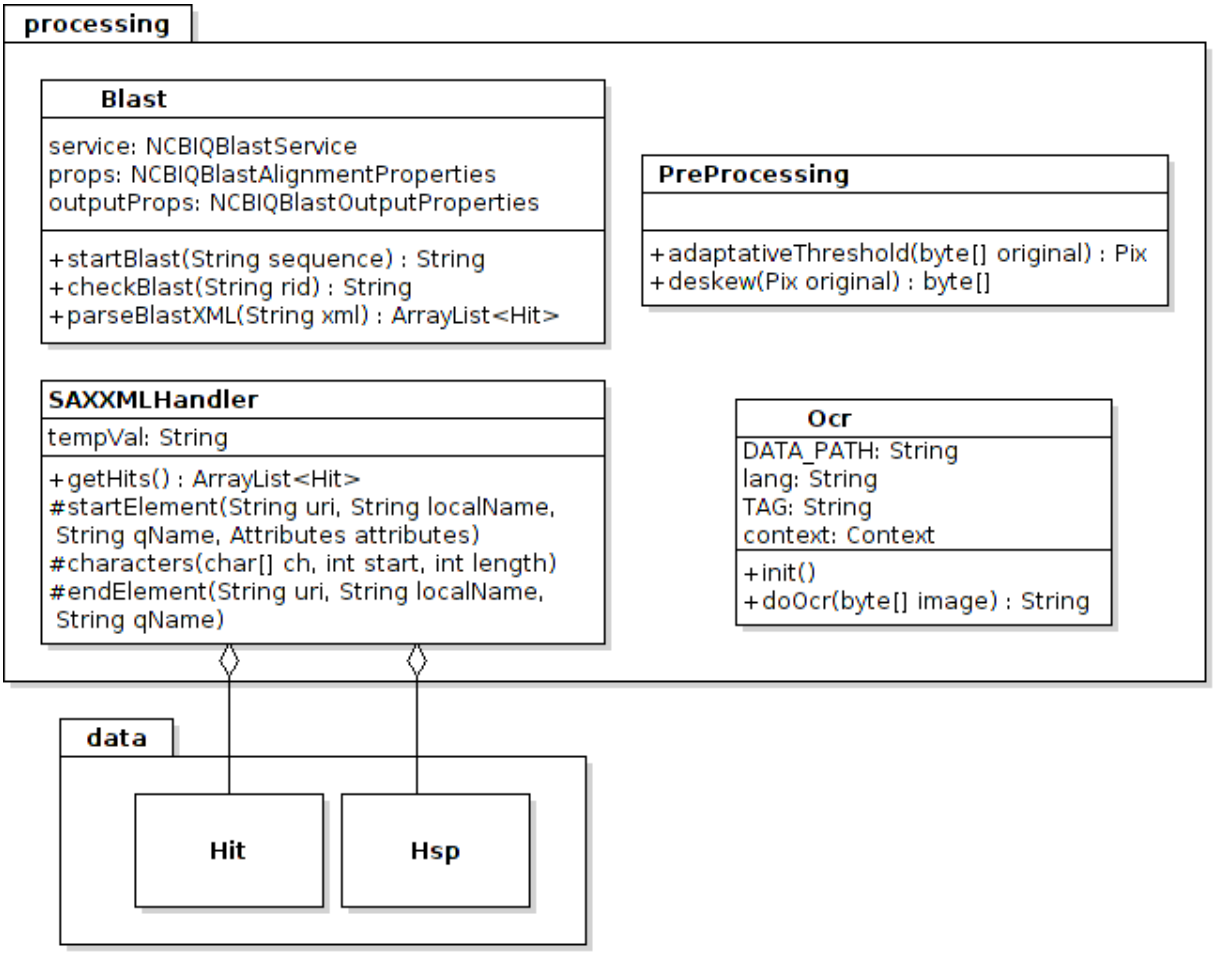

FIGURA 29 – DIAGRAMA DE CLASSES – PACOTE PROCESSING

Fonte: Os autores (2014).

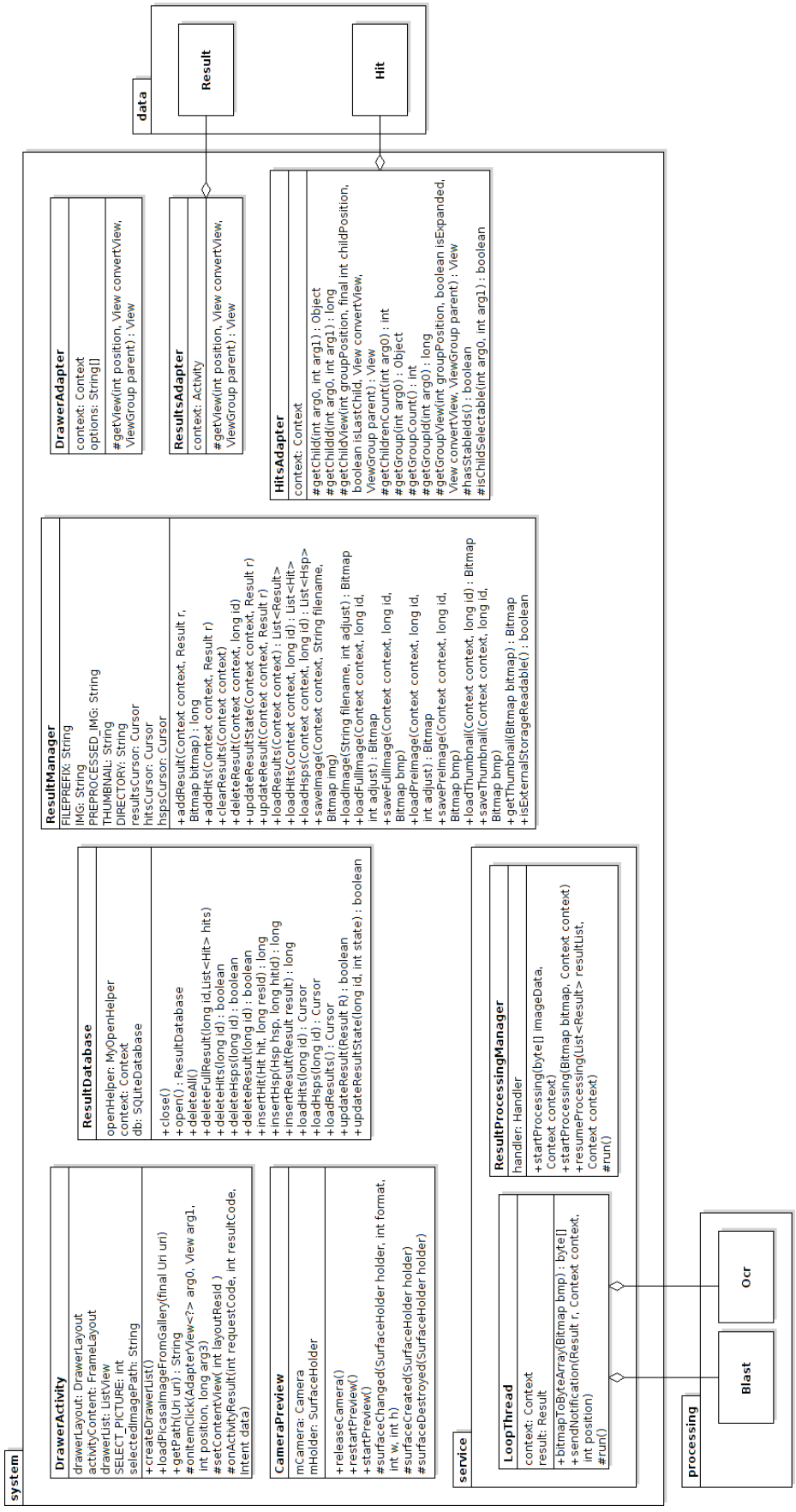

FIGURA 30 – DIAGRAMA DE CLASSES – PACOTE SYSTEM

Fonte: Os autores (2014).

APÊNDICE E - DIAGRAMA DE ATIVIDADES

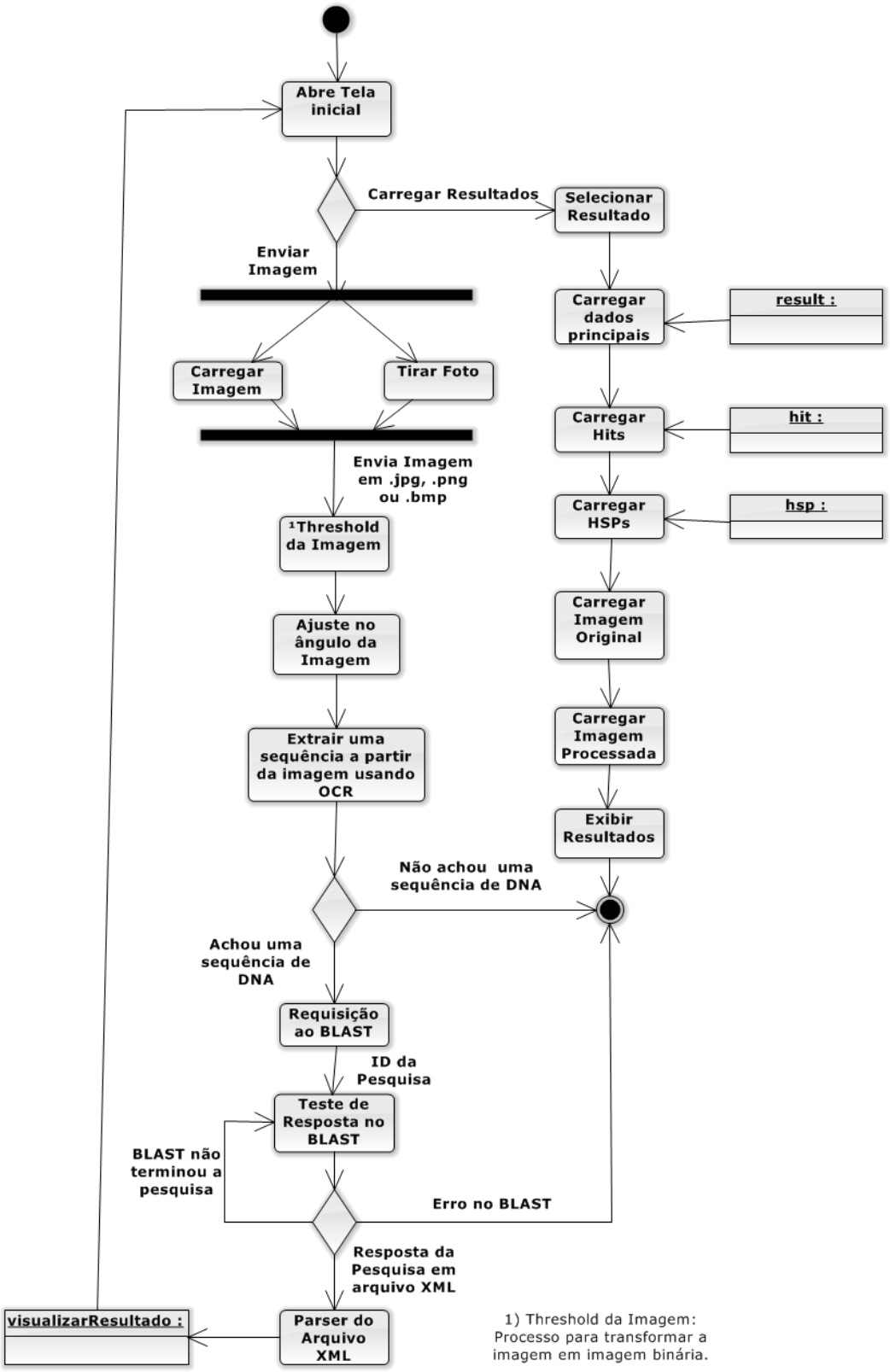

FIGURA 31 – DIAGRAMA DE ATIVIDADES – USO GERAL

Fonte: Os autores (2014).

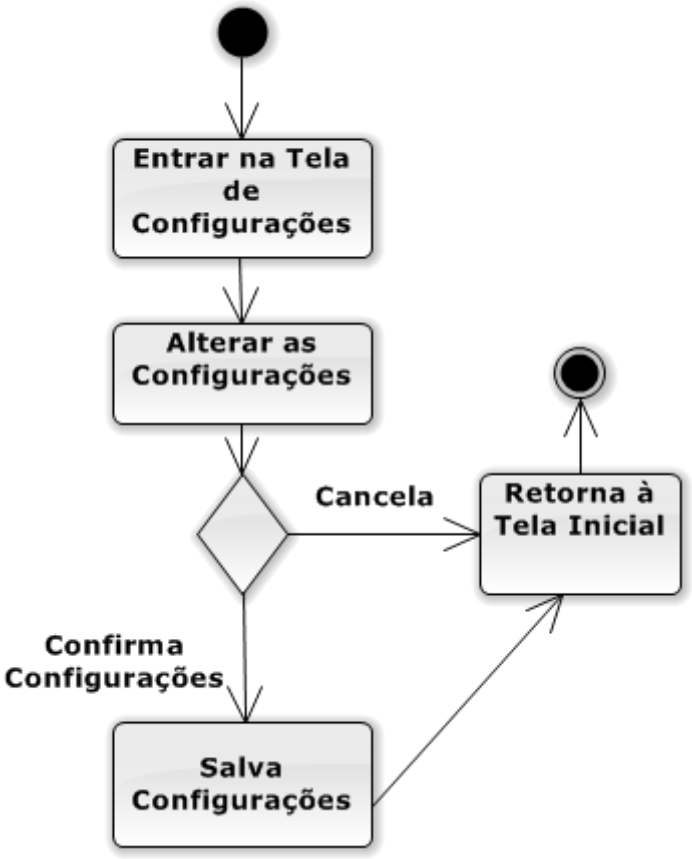

FIGURA 32 – DIAGRAMA DE ATIVIDADES – ALTERAR CONFIGURAÇÕES

Fonte: Os autores (2014).

APÊNDICE F - DIAGRAMA DE COMPONENTES

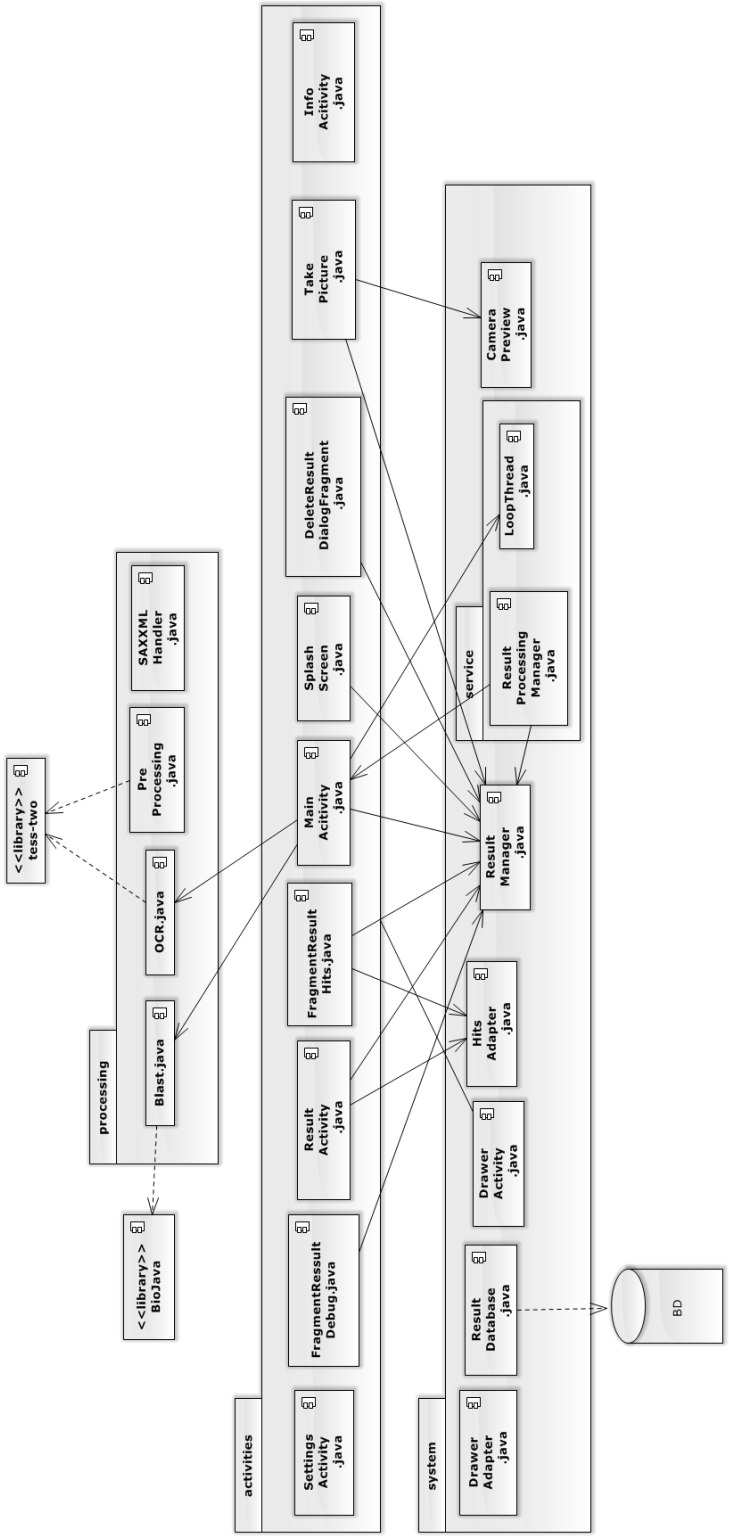

FIGURA 33 – DIAGRAMA DE COMPONENTES

Fonte: Os autores (2014).